Proceedings of the 3rd Ph.D. Retreat of the HPI Research School on Service-oriented Systems Engineering

hrsg. von
Christoph Meinel, Hasso Plattner, Jürgen Döllner, Mathias Weske, Andreas Polze, Robert Hirschfeld, Felix Naumann, Holger Giese

Technische Berichte Nr. 27

des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam

Universität Potsdam

HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam | 27

Proceedings of the 3rd Ph.D. Retreat of the HPI Research School on Service-oriented Systems Engineering

herausgegeben von

Christoph Meinel
Hasso Plattner
Jürgen Döllner
Mathias Weske
Andreas Polze
Robert Hirschfeld
Felix Naumann
Holger Giese

Universitätsverlag Potsdam

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der
Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über
http://dnb.d-nb.de/ abrufbar.

Universitätsverlag Potsdam 2009
http://info.ub.uni-potsdam.de/verlag.htm

Am Neuen Palais 10, 14469 Potsdam
Tel.: +49 (0)331 977 4623 / Fax: 4625
E-Mail: verlag@uni-potsdam.de

Die Schriftenreihe **Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam** wird herausgegeben von den Professoren des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam.

Druck: docupoint GmbH Magdeburg

ISSN 1613-5652
ISBN 978-3-940793-81-2

Zugleich online veröffentlicht auf dem Publikationsserver der Universität Potsdam:
URL http://pub.ub.uni-potsdam.de/volltexte/2009/2914/
URN urn:nbn:de:kobv:517-opus-29148
[http://nbn-resolving.de/urn:nbn:de:kobv:517-opus-29148]

Contents

1. **Extending the WPVS Visualization and Interaction Capabilities**
   Benjamin Hagedorn
2. **Optimizing Virtualization Concepts in (Guest-) Operating Systems**
   Michael Schöbel
3. **Implementation of a Service Platform to Evaluate Virtual Team Communication**
   Matthias Uflacker
4. **Modelling Security Configurations for Service-oriented Architectures**
   Michael Menzel
5. **A Flexible Live Inspection Framework**
   Alexander Schmidt
6. **Active Information Graphs**
   Hagen Overdick
7. **FMC-QE – Hierarchies, Transformations and Rules**
   Stephan Kluth
8. **Identity Management for Cross-Organizational SOA**
   Ivonne Thomas

9. **Taking Trust Management to the next level: Analysis and Formalization**
Rehab AlNemr

10. **Automated Service Composition for Minimal Goals**
Harald Meyer

11. **Business Process Model Abstraction and Flexible Process Graph**
Artem Polyvyanyy

12. **ContextJ – Context-oriented Programming for Java**
Malte Appeltauer

13. **Modelling and Verification of Self-adaptive Service-oriented Systems**
Basil Becker

14. **On a Model for a Service Database**
Mohammed AbuJarour

15. **Towards the Automatic Generation of Effective, Map-Like Visual Representations from Heterogeneous Geodata in a Service-oriented Infrastructure**
Dieter Hildebrandt

Extending the WPVS Visualization and Interaction Capabilities

Benjamin Hagedorn
Benjamin.Hagedorn@hpi.uni-potsdam.de

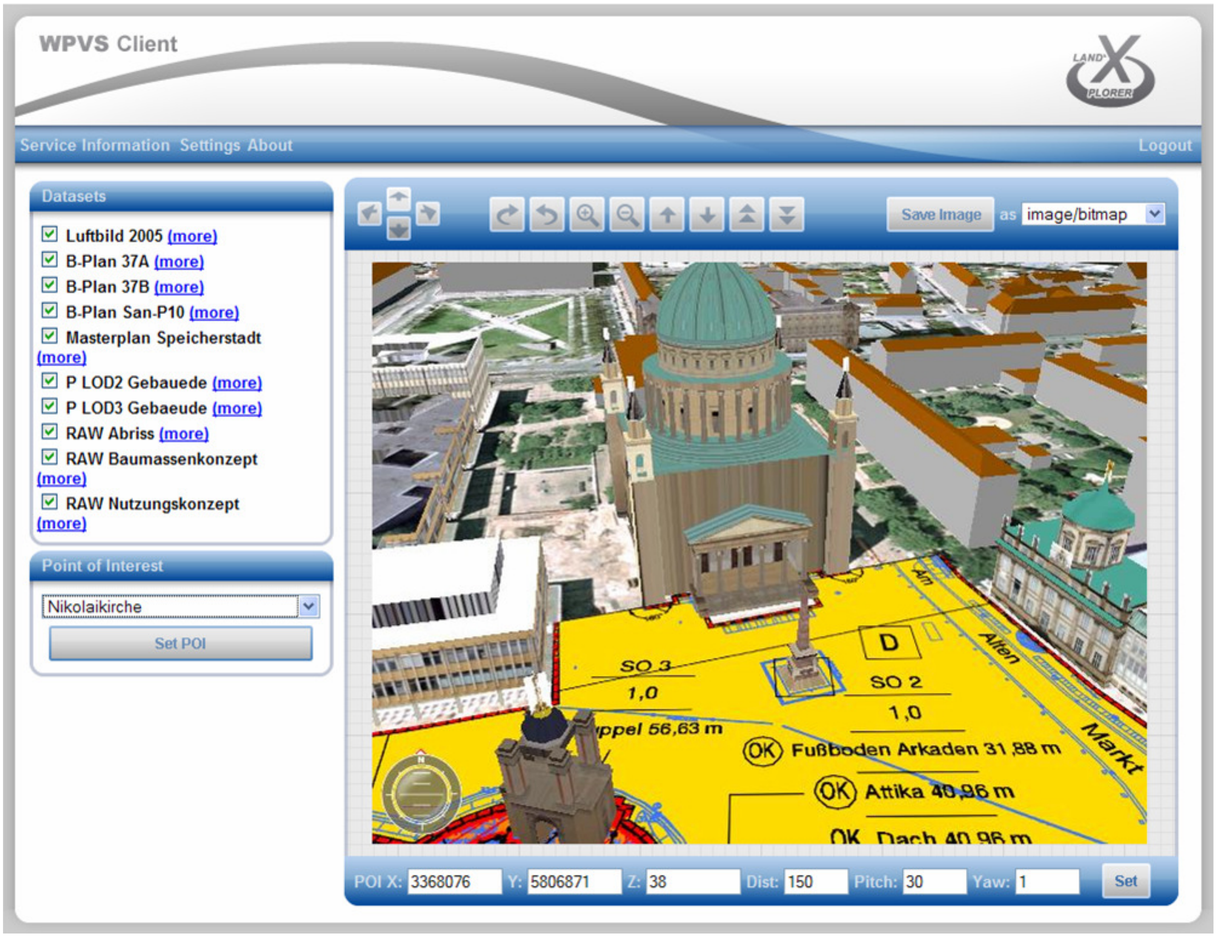
A Web Perspective View Service (WPVS) provides images of perspective views of 3D geovirtual environments (3DGeoVE) which can be displayed by the service consumer without client-based 3D rendering. Such visualizations can be easily accessed by simple clients such as mobile devices and can be integrated into processes, e.g., for city planning or ad-hoc threat response applications. So far, the WPVS does not provide any interaction or navigation capabilities, which restricts its degree of usability. To increase the spectrum of the WPVS functionality and, thus, its usability and applicability, we present the concept of an extended, smart WPVS. The proposed extensions provide meta information for generated views and more effective 3D interaction and 3D navigation for service-based complex 3D geovirtual environments.

1 Introduction

The amount of collected geodata, it's complexity, and it's usages are constantly growing. Geodata comprises spatial, thematic, and temporal information and can be applied in various domains, applications, and systems. For these systems and for the sustainability of the collected geodata, interoperability is a crucial issue. This includes aspects such as a common understanding of geoinformation content and quality, geodata description, and geodata access. The Open Geospatial Consortium represents a jointly international effort targeting at geoinformation interoperability. In varous initiatives, the members of the OGC reached consensus on the format of data and several web services for accessing, processing, and visualizing geodata.

Geovisualization plays an important role in the chain of geoinformation usage. In the field of two-dimensional geoinformation, visualization is mainly provided by the *Web Map Service (WMS)* [13]. It allows for retrieving map-like views of various geodata. By the help of *Styled Layer Descriptors (SLD)* and *Symbology Encoding (SE)* [14] this visualization can be styled and adjusted according to the service consumers' requirements. So, from the same data basis, e.g., a road map and a map of trails can be generated, which contain and emphasize different elements and are specific for different user groups and usage scenarios.

The growing importance of three-dimensional geoinformation has been reflected by the revision of geodata description standards (GML3), but also by the development of 3D portrayal services. The key 3D portrayal services in the OGC's geoservice family are the *Web 3D Service (W3DS)* and the *Web Perspective View Service (WPVS)* [10,



- ☑ Luftbild 2005 (more)
- ☑ B-Plan 37A (more)
- ☑ B-Plan 37B (more)
- ☑ B-Plan San-P10 (more)
- ☑ Masterplan Speicherstadt (more)
- ☑ P LOD2 Gebauede (more)
- ☑ P LOD3 Gebaeude (more)
- ☑ RAW Abriss (more)
- ☑ RAW Baumassenkonzept (more)
- ☑ RAW Nutzungskonzept (more)

Point of Interest: Nikolaikirche

Save Image as image/bitmap

POI X: 3368076 Y: 5806871 Z: 38 Dist: 150 Pitch: 30 Yaw: 1

Figure 1: Web Perspective View service visualizing city planning alternatives in Potsdam.

12]. Both services have not become OGC standards yet. The W3DS has the status of an OGC discussion paper and the WPVS is still an internal draft. The current OGC web services initiative (OWS-6) brings these 3D portrayal services back into focus.

We are participating in a 3D portrayal initiative within the OGC which resumes the work at these services, reviews the existing specification efforts and aims at creating a 3D portrayal services family. As part of this initiative, we want to extend the WPVS by visualization and interaction functionality based on state-of-the-art computer graphics architectures. This report describes the conceptual foundations for extending the WPVS and derives several new functionality from investigating typical application scenarios for 3D portrayal.

2 3D Portrayal Services

2.1 Relevance of 3D Portrayal

For 2D portrayal, the WMS represents the ”work horse” for interoperable visualization of 2D maps. The service consumer chooses the information layers to include in the map-like visualization and may adjust the visual appearance of the 2D features. Additionally, the WMS allows for requesting type descriptions of the represented features and for retrieving additional information of a feature at a specific image position.

Several activities within the OGC aim at the interoperable description of 3D geoinformation. Those are, e.g., GML3 (which includes three-dimensional features) and CityGML (which is a GML3 profile for describing regional landscape models and city models). For the portrayal of this 3D geoinformation, W3DS and WTS (which will be replaced by the WPVS) represent primary specifications. Complementing the interoperable 2D map generation and access by WMS, 3D portrayal becomes more and more important. Compared to 2D maps, perspective views can improve the perception and interpretation of complex spatial information: They show the height of geoobjects, included facades hint on the usage of buildings, etc. Perspective views become a generally known and accepted medium for spatial information. Nevertheless, there are no standard approaches widely used for 3D geovisualization.

2.2 The OGC Portrayal Model

The OGC defines portrayal as information presentation to the human – portrayal elements may be either images or display elements. Corresponding to this, the OGC defines providing portrayal services and consuming application services as part of their OWS service framework [11]. Application services may be either application servers, or application clients. The application of geoservices for geovisualization raises the question of the separation of rendering concerns between service provider (portrayal service) and service consumer (application service), i.e., between server and clients.

The OGC portrayal model (Fig. 2) can also be applied to 3D portrayal. Components of a 3D portrayal services family are arranged along that pipeline. Major aspects of these services and specifications aim at the generation of display elements (e.g., a scene graph), rendering images, and giving the possibility to influence the visual appearance of features by styling. While generating and serving display elements leads to client-site rendering, generating images represents fully server-side rendering. The OGC Portrayal model allows for three segmentations which are described in [2]:

- *Thick Client / Thin Server:* The client request selects data from the server and performs the remaining steps of the geovisualization pipeline. This requires appropriate computer graphics capabilities of the client. A possible geoservice participating in this scenario is a WFS.

- *Medium Client / Medium Server:* The service provides computer graphical representations (e.g., a VRML scene graph) to the client which has to synthesize images. Again the client needs computer graphics capabilities for rendering images. In this scenario the style of the resulting visualization is more in concern of the server. The W3DS is a possible participant in this scenario.

- *Thin Client / Thick Server:* All the visualization steps are performed by the server. The server defines the final visualization and the client must only provide capabilities for displaying the visualization to the end-user.

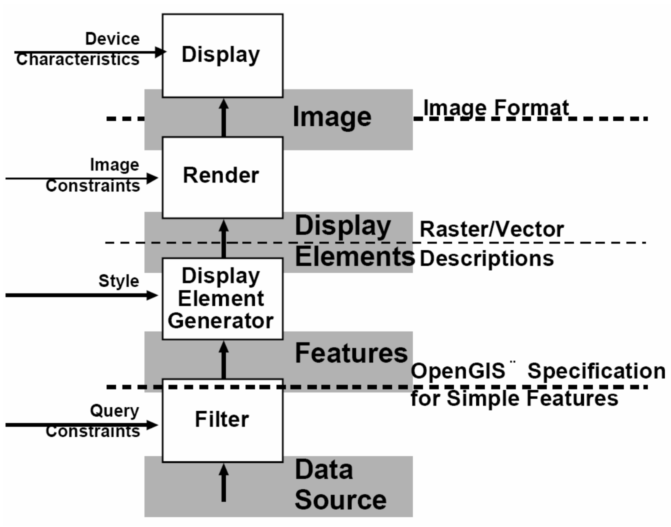


Figure 2: The OGC portrayal model. [11]

2.3 The Web 3D Service

The W3DS [12] generates computer graphical representations of 3D landscape and city models, which are streamed to the service consumer in 3D graphics formats (e.g., VRML, X3D, COLLADA). This graphics data can include appearance information but is mostly not capable of transferring thematic information. The computer graphical representations, which could be arranged in a 3D scene graph, have to be processed and rendered by a W3DS client using the rendering techniques of its choice. The W3DS client is a rich client allowing for an interactive 3DGeoVE including real-time navigation.

A full W3DS and a web-based client have been implemented at the University of Bonn. [1] It provides exploration capabilities for the city of Heidelberg. The client is implemented as a Java Webstart application which installs and retrieves the computer graphical representations from the server. For rendering high detailed and high-quality visualizations of the presented 3DGeoVE, modern 3D graphics acceleration hardware and high bandwidth are required.

2.4 The Web Perspective View Service

The WPVS aims generates images of perspective view of a 3D scene and sends this to the service consumer. With this approach, we can set up a dedicated server with appropriate 3D hardware, i.e., we don't have to deal with incompatible, diverse 3D hardware configurations on the client-side. Users get access to arbitrary complex geovirtual environments with high-quality graphics output and without having to install specialized 3D applications or streaming complex 3D data – only images are transferred. This a) omits incompatibility problems of 3D renderer systems and hardware, b) reduces administration and maintenance costs, and c) allows for a simple integration of 3D geovisualizations into complex workflows. The WPVS defines four operations (see Ta-

Table 1: WPVS operations (from [10]).

Operation	Description
GetCapabilities	The mandatory GetCapabilities operation allows clients to retrieve service metadata from a server. The response to a GetCapabilities request is an XML document containing service metadata about the server, including metadata about the data available from that server.
GetView	The WPVS GetView operation allows clients to obtain a specified subset of identified layers from a WPVS server. In addition, the GetView operation allows the client to select the perspective view of the retrieved layers and the format.
GetDescription *(optional)*	The GetDescription operation allows clients to retrieve the descriptions of one or more identified datasets or styles.
GetLegendGraphic *(optional)*	The GetLegendGraphic operation allows a client to retrieve a graphic containing a map legend for an identified dataset and style. Implementation of this operation is optional by WPVS servers.

ble 1); the GetView and GetCapabilities options are mandatory. Fig. 3 contains an example of a WPVS GetView request.

2.5 Comparison of WPVS and W3DS

Table 2 briefly compares the W3DS and WPVS. From these characteristics different usages of both services follow. Using the W3DS together with a powerful rendering client allows for real-time navigation and interaction in the 3D scene. For the provision of complex scenes by W3DS a high bandwidth is needed.

The WPVS can be used by more simple clients and in situations where available bandwidth is low. Navigation in a base WPVS is step-by-step and far from real-time interaction. We believe that this drawback can be attenuated by intelligent loading and display strategies (e.g., image preloading or cube map generation).

The WPVS encapsulates the hole rendering process and provides simple portrayal functionality by generating and delivering images; the W3DS provides 3D scene graphs dedicated for consumer-side rendering. Thus, in a spatial data infrastructure providing

```
http://myserver.org/wpvs?SERVICE=WPVS&VERSION=1.0&REQUEST=GetView&
CRS=EPSG:26912&POI=424994.4,4513359.9,1550.0&PITCH=30&YAW=30&ROLL=0&
DISTANCE=1000&AOV=60&BOUNDINGBOX=424585.3,4512973.8,425403.5,4513746.0&
ELEVATIONMODEL=Default&WIDTH=640&HEIGHT=480&OUTPUTFORMAT=image/jpeg&
LAYERS=City,River&STYLES=default&EXCEPTIONFORMAT=INIMAGE
```

Figure 3: Example of a WPVS v.1.0.0 URL-encoded GetView request.

Table 2: Comparison of the WPVS and W3DS.

	WPVS	W3DS
Transferred data	Imagery	General scene graph
Transmission load	Depends on type, quality, and size of imagery	Depends on the size of the 3D scene
Server-side complexity	High	High
Client-side complexity	Low	High
Resulting visual quality	Determined by the service provider	Depends on the service consumer rendering capabilities

3D portrayal, the WPVS could act as a consumer of the W3DS. Acclaiming that, there should be a large overlapping in the WPVS and W3DS service interfaces.

A major difference between WPVS and W3DS is in the application of optimization strategies for rendering. The W3DS rendering client can implement strategies such as culling, caching, and level-of-detail mechanisms and can take advantage of the navigation and interaction state. In contrast, the WPVS is stateless and does not provide any session management functionality. Thus, the WPVS rendering system can not make any assumptions about the data to load and render. In worst case, an underlying (naive) rendering system would have to load and render different data for each single WPVS GetView request.

3 WPVS Extension Requirements

3.1 Relevant 3D Portrayal Functionalities

The functionalities of a 3D portrayal system (WPVS, W3DS) can be grouped according to the following functional dimensions:

- *Information*, which is included within and represented by the generated visualization. An essential part of this information are, e.g., images of the 3DGeoVE. The WPVS arranges information content by layers, which must be chosen by the WPVS consumer for being included in the finally rendered image. Beyond the portrayal of real world geometry, additional spatial and non-spatial (but georeferenced) geodata could be included in the visualization. Fig. 1 shows an example of a 3DGeoVE including plat information. Even non-visual visualizations could be integrated, e.g., sounds emphasizing the characteristics of a scene.

- *Styling* comprises the information aiming at influencing the visual appearance of the rendered images. On a technical level, it describes the mapping of geoinformation to computer graphical elements, regarding visual variables such as shape, size, orientation, color, and texture. For example, the OGC defines the SE styling specification, which can be applied, e.g., with a WMS request. For 3D portrayal

Table 3: WPVS operations grouped by their main type of 3D portrayal functionality.

Type	Functionality	Scenarios[a]
Information	Show the surrounding of a site	2.1
	Visualize underground geodata	1.6
	Generate a video tour passing specific locations	5.4
	Create a set of views along a path	6.1
	Retrieve additional information for a scene object.	1.1, 1.4
	Enrich the scene by further scene objects	4.2, 4.3
	Calculate and display the visibility of objects	3.5
	Inspect the inside of objects	3.4, 4.3
	Measure distances and size	2.3
Styling	Color scene objects according to attribute values	2.2, 3.2
	Use a styling tool for defining the scene appearance	5.2
Interaction	Select a scene object by picking	1.1, 1.2, 1.4, 1.5
	Annotate scene objects by text or sketch	1.2, 1.5
	Use specific interactions for information retrieval	3.3
	Use 3D portrayal on mobile devices	4.1
	Use 3D portrayal in a web browser	5.1

[a]Numbering according to the scenarios in appendix A.

no styling specifications exist, yet. Recently, Neubauer and Zipf [9] developed a 3D extension for SLD and introduced this to the OGC specification process.

- *Interaction* is the user-centered process of operating the information system, e.g., changing the general information content, investigating the scene by navigation, and investing scene objects by retrieving specific information.

A specific portrayal functionality may not only regard a single dimension, but is positioned in the functional space set up by these dimensions. E.g., retrieving information about a specific building means to interact with the scene (or it's image, e.g., by picking a building and choosing the interesting information) and to visualize the retrieved information according to a specified styling description.

3D portrayal plays a role in various application domains and within those in various application scenarios. As a basis for specifying 3D portrayal capabilities within the OGC standardization process, [8] lists selected scenarios in several application domains, e.g., in the field of civil service, business development, and tourism. Appendix A lists these application scenarios in more detail. The list is not complete but gives an idea of what might be possible by the help of 3D portrayal services.

Table 3 groups the functionalities, which can be derived from these application scenarios. The functionalities are often relevant in more than one application domain and slightly overlap.

3.2 General Requirements for WPVS-Based Portrayal

According to the 3D portrayal model, the separation of concerns is important when extending the WPVS. The following general requirements for 3D portrayal basing on server-side rendering could be formulated.

- *Support simple clients:* The main benefit of server-side rendering is to omit client-side rendering capabilities, e.g., high-end graphics hardware. The client must be able to easily request a 3D portrayal service and to use the generated visualization.

- *Provide high quality visualizations:* Even with simple clients, the visualizations should be of high quality. This could mean to retrieve aesthetic images of appropriate resolution from the portrayal service and to interact in a smart and effective manner for exploring the 3DGeoVE.

- *Keep service interface simple:* For allowing for the easy use of the portrayal service, the service interface should be kept simple. So it should contain a little number of parameters or, at least, provide default values. This would lead to an easy integration within applications and systems for supporting workflows.

- *Keep transmission load low:* For retrieving a specific functionality, the number of service requests should be small. Additionally, the used bandwidth should be kept low. E.g., rendered images should be provided in appropriate resolutions and size.

- *Support user assistance:* Smart user interfaces allow for easy use of the 3D portrayal. User interface functionality should be simple, obvious, familiar, safe. Intelligent clients and smart client-server interaction strategies could reduce the time needed for achieving a specific goal, and also reduce rendering load and bandwidth usage.

3.3 Limitations of the WPVS

The interaction capabilities of the WPVS are restricted to the generation of perspective views. In particular, it does not provide advanced interaction capabilities or information retrieval beyond RGB images.

Further on, the WPVS does not address efficient and goal-oriented navigation (including wayfinding and motion) and supports only a point of interest (POI)-based camera specification: the camera position is described by polar coordinates relative to a geospatial position, the POI. Consumer-side service proxies (e.g., the WPVS application client) are in charge of providing appropriate navigation functionalities and map those to the camera configuration.

The WPVS GetView request defines a SLD parameter which is intended to reference or contain a styled layer description. As far as we know, there is no WPVS implementation which takes advantage of this parameter. Furthermore, the current SLD/SE specification does not allow for styling three-dimensional features.

4 Suggestions for Extending the WPVS

4.1 Enhanced Styling for 3D Portrayal and WPVS

A specific visual appearance of the portrayal output is achieved by styling functionality. As described above, the OGC styling specifications, SLD and SE, do not consider the styling of three-dimensional features. Neubauer et al. [9] extended the SE for 3D styling and implemented it for a W3DS. These additions extend the existing SE symbolizers (point, line, polygon), add symbolizers for solids and surfaces, include external 3D objects, support billboards, lines as cylinders, introduce material descriptions and different local shading models, and support rotations. A W3DS user can edit a styling description and use this with requesting a scene from the W3DS. This styling information is used for generating the scene graph representation which is returned to the requester (e.g., as VRML or X3D structure). The complex rendering client uses this information for synthesizing the images.

Beyond these extended SE styling capabilities, it is up to the rendering client to implement and provide additional rendering techniques and effects. Table 4 lists several rendering techniques, which are relevant for 3DGeoVEs and would have to be implemented and provided by the WPVS. For requesting these appearances, additional and more abstract styling definitions than those in the planned SE extension are required (e.g., for describing solar altitude or cloudiness).

4.2 Cartographic Geovisualizations

Cartography deals with the mapping of real world features onto graphical representations considering specific users, specific purposes, a required map scale, or limited space of the presentation media. It mainly aims at the production of highly legible and effective maps.

Generalization represent the main cartographic method for reaching this aim. It comprises several generalization operations such as selection, simplification, combination, enhancement. Graphically, these techniques, e.g., influence the shape and geometric complexity of a geographic feature representation, its position, its size, as

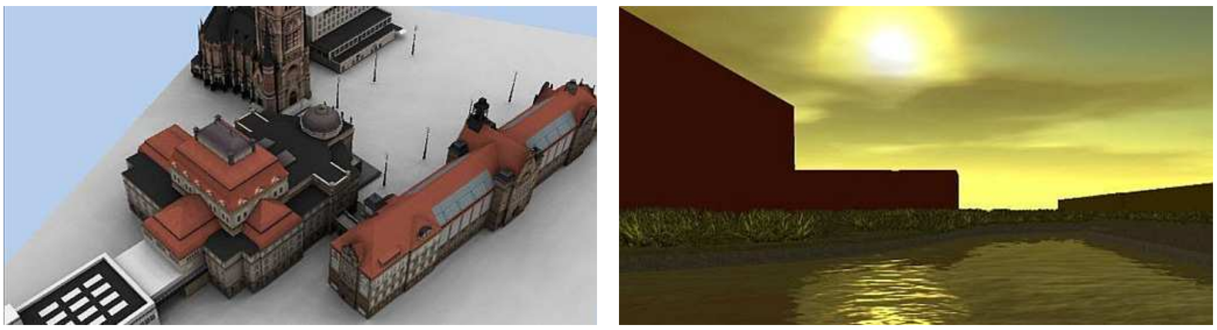


Figure 4: Examples for rendering styles and effects relevant for 3DGeoVE: ambient occlusion simulating daylight (left) and sun, clouds, atmospheric effects, and water shading (right).

Table 4: Advanced rendering techniques for 3D geovirtual environments.

Rendering Styles & Techniques	Description
Global illumination	Rendering techniques which produce a more natural lighting regarding scene objects interferences. Additionally, it helps to interpret and understand the image.
Water rendering	Using environment mapping, water surfaces mirror the surrounding features or sky. Waves on the water can make the image more authentic.
Sun and cloud rendering	Sun and cloud do not only support the authenticity of a generated image but can also transfer additional relevant information such as weather conditions or day time.
Depth of field	Before and behind a specific depth range, the scene objects get blurred. The sharp scene objects get more focused.
Focus & context lenses	Within a virtual lense area (the focus area), specific information are displayed, e.g., geo-referenced thematic information such as geological data.
NPR rendering	Non-photorealistic rendering is the stylization of the appearance of scene objects within the 3DGeoVE. It can include, e.g., color quantization and edge enhancement. So, NPR can provide facade generalization and can support information communication.

well as it color or texture.

In general, cartographic principles and methods also apply for 3D perspective visualizations. In particular, 3D visualizations mostly contain different scale levels, which is different from 2D maps: in a perspective view, scene part close to the virtual camera have a larger scale than those that are far away. Without an appropriate generalization techniques, these far away scene parts appear as pixel mush and, thus, lack legibility.

Further on, in 3D perspective views, vertical surfaces become visible and provide essential information, which is not the case in 2D maps. For example, a building facade could hint on the type, usage, age, or state of the building; windows and doors could serve as scale defining elements indicating the building size. In 3D virtual environments, these surface information are often given by textures, recorded, e.g., by aerial photography. Non-photorealistic rendering could be applied for the generalization of these textures, as it could reduce the overall texture information, but emphasize the relevant features of a texture, and improve the legibility. Fig. 5 illustrates the examples of geometric generalization and texture simplification for 3D city models.

For the WPVS the consideration of cartographic visualization would mean to allow the service requester to directly specify the visual appearance of the generated image of the perspective view or to indirectly specify the appearance by describing the re-

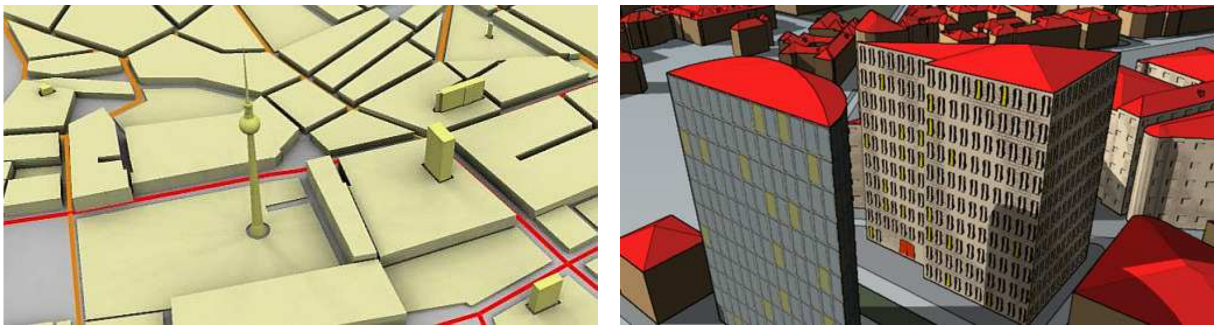


Figure 5: Geometrical generalization (left [4]) and texture generalization (right [3]) for a virtual 3D city model.

quirements of the generated map, which then has to be evaluated by the WPVS for deriving appropriate scale levels and accessing or generating appropriate generalized data.

4.3 3D Annotations

The annotation of a 3DGeoVE by text or symbols is an important aspect. It allows, e.g., for the integration of meta information or comments. To ensure high legibility, annotations have to be embedded such that they avoid occlusions among themselves and with geospatial objects of the scene. As described in [7] annotations can be integrated within the rendering process by providing a so-called depth image by WPVS additionally to the RGB image. Together with the camera information it serves as input for a processing service producing annotated images by synthesizing an annotation image and blending this with the WPVS produced RGB image.

4.4 Object Information

The WPVS could also support emphasizing and accessing object information that can not be directly descried from the common object visualization (in photorealistic or non-photorealistic style). This additional information could include general information (e.g., object categories, identifiers, or names) as well as domain-specific information (e.g., building height, floor space ration, or other BIM-related data). The representation of this information could be either dedicated to the user or could be evaluated and facilitated by a client application. This functionality corresponds to the *GetGeature* operation of the WMS, which allows for retrieving additional object information of features visible in a 2D map.

4.5 Analysis Functionality

The WPVS could provide simple *measuring functionalities*, which could be, e.g., measuring an area or measuring distances of a point to camera or between several points in the image. This requires interaction capabilities of the WPVS, which had to process

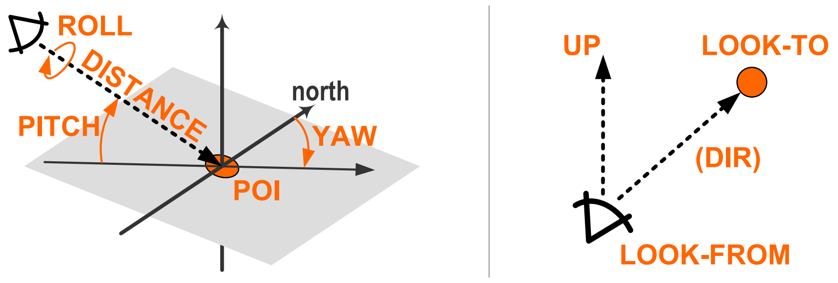


Figure 6: WPVS camera definition. Current POI-based description (left) and convenient camera-centered version (right).

picked points or sketched paths by projecting them into the 3D scene ans use this as input for analysis functionalities. *Information lenses* represent an other category of analysis functionality which permits the integration and combination of different information layers; the user would, e.g., move the mouse and retrieve thematic data for a specified region within the image. Information lenses could be implemented by client-side blending of different information layers; considering depth information would allow for perspective lenses.

4.6 Convenient Camera Specifications

Positioning the virtual camera is an essential part of the WPVS GetView operation. A WPVS implementation has to transform this information into the internally used camera specification and derive orientation and projection matrices for synthesizing the images. The WPVS supports a POI-based camera definition: a POI defines the camera look-to and the camera position is given relative to the POI by polar coordinates, distance of the camera, role angle, and angle of view (AOV) centered at the POI – see Fig. 6 left. The POI is defined in a specific coordinate reference system. This POI-based camera definition is useful for a POI-based navigation, i.e., to look to some specific locations.

Several use cases conceptually deal with a camera-based view definition. They require, e.g., to define the camera position, the gaze direction or look-to position, and the camera up vector – see Fig. 6 right. Complementing the camera definition by a camera-oriented variant would make the WPVS usage more convenient for ad hoc usage and integration: E.g., a user retrieved geocoordinates of two specific addresses and could use them as input for a WPVS request without additional calculation. On the one side, this conversion could be encapsulated by the WPVS client, on the other side, the WPVS itself could provide an additional camera definition method.

4.7 Smart Navigation

The general navigation capabilities of the WPVS allow consumers to explore the 3DGeoVE and to retrieve the contained information. So far the WPVS does not explicitly address navigation support. The only way to explore the 3DGeoVE is the modification of the camera parameters. On top of this, a client has to install more high-level navigation functionalities, e.g., moving forward, backward, up down, etc.

Including smart navigation techniques as described in [6] could be a promising approach for using the image itself as a navigation interface. Curves and points sketched on the image are sent to the WPVS, which projects these sketches into the 3D scene for determining navigation intentions and either generates a video or a set of images representing the navigation. This navigation technique is user-oriented and is applicable to all touch-sensitive devices, such as PDAs or smartboards. For the integration of sketch-based navigation, the WPVS would have to be extended by a navigation subsystem and operations for requesting a sequence of views are required.

4.8 Single Object Inspection

Besides the visualization of regional 3D landscape and city models, a WPVS profile could facilitate the visualization and inspection of specific features. [5] For example, for building models a specific WPVS instance could integrate the calculation of appealing camera views including filtering and styling capabilities for allowing for gaining insight into the building, e.g., by using techniques such as 3D cut-away-views.

5 Conclusion and Future Work

For enabling interoperable 3D portrayal within 3D spatial data infrastructures, we participate in an OGC initiative for specifying a 3D portrayal standards family. One of the service candidates is the WPVS, which is an OGC draft specification. As a contribution to the OGC 3D portrayal initiative, we revise and suggest additions to the current WPVS specification. The WPVS is limited in visualization, interaction, and navigation capabilities and could be extended in these aspects. This technical report represents a first step towards an extended WPVS. It describes the functional and non-functional context for extending the WPVS and suggests a next version WPVS. These suggestions need to be further formalized; the WPVS service interface must be extended. These extensions could be implemented by introducing new service operations (e.g., for retrieving depth images) or by add parameters to the existing operations(e.g., extending the camera description). Furthermore, for evaluation purposes, a prototypic implementation of the proposed WPVS extensions is planned.

References

[1] Heidelberg 3D. URL: http://www.heidelberg-3d.de/ (called at 19/10/2008).

[2] Angela Altmaier and Thomas Kolbe. Applications and Solutions for Interoperable 3D Geo-Visualization. In Dieter Fritsch, editor, *Proceedings of the Photogrammetric Week*, Stuttgart, 2003. Wiechmann Verlag.

[3] Jürgen Döllner, Henrik Buchholz, Marc Nienhaus, and Florian Kirsch. Illustrative Visualization of 3D City Models. In *Proceedings of Visualization and Data Analysis 2005 (Electronic Imaging 2005, SPIE Proceedings)*, pages 42–51, 2005.

[4] Tassilo Glander and Jürgen Döllner. Techniques for Generalizing Building Geometry of Complex Virtual 3D City Models. In *2nd International Workshop on 3D Geo-Information: Requirements, Acquisition, Modelling, Analysis, Visualisation*, Delft, Netherlands, December 2007.

[5] Benjamin Hagedorn and Jürgen Döllner. High-Level Web Service for 3D Building Information Visualization and Analysis. In *ACM 15th International Symposium on Advances in Geographic Information Systems (ACM GIS)*, Seattle, WA, November 2007.

[6] Benjamin Hagedorn and Jürgen Döllner. Sketch-Based Navigation in 3D Virtual Environments. In *8th Int. Symposium on Smart Graphics 2008*, August 2008.

[7] Benjamin Hagedorn, Stefan Maass, and Jürgen Döllner. Chaining Geoinformation Services for the Visualization and Annotation of 3D Geovirtual Environments. In *4th International Symposium on LBS and Telecartography*, November 2007.

[8] Benjamin Hagedorn, Alexander Zipf, Arne Schilling, and Steffan Neubauer. 3D Portrayal Services - Use Cases. OGC Discussion Paper, August 2008. Version 0.0.6, OGC 08-140.

[9] Steffen Neubauer and Alexander Zipf. Suggestions for Extending the OGC Styled Layer Descriptor (SLD) Specification into the third Dimension - An Analysis of possible Visualization Rules for 3D City Models. In *Urban Data Management Symposium*, Stuttgart, Germany, October 2007.

[10] Open Geospatial Consortium. *OpenGIS Web Perspective View Service (WPVS) Implementation Specification*, October 2005. Draft Version.

[11] Open Geospatial Consortium Inc. *OGC Reference Model, Version 0.1.3*, September 2003.

[12] Open Geospatial Consortium Inc. *Web 3D Service, Version 0.3.0*, February 2005.

[13] Open Geospatial Consortium Inc. *OpenGIS Web Map Server Implementation Specification, Version 1.3.0*, March 2006.

[14] Open Geospatial Consortium Inc. *Symbology Encoding Implementation Specification, Version 1.1.0*, July 2006.

A Application Scenarios for 3D Portrayal

Table 5: Scenarios for the application of 3D Portrayal Services (from [8]).

1.	**Civil Service**
1.1	**Use a 3D view for processing a building application** A civil service agent can generate a perspective view for a specific address for getting an idea of the current building situation around that address. By picking buildings in the image further building information are displayed.
1.2	**Use perspective views as interface for notifications of claim** A citizen can generate a perspective view for a specific address, can navigate for defining a specific view, can pick a position in that image, and can leave a notification of claim, e.g., about an out-of-order fire hydrant.
1.3	**Show a city planning to the public** For public participation, the municipality of a city can make a city planning available to the public by 3D portrayal services. Via web portal, citizens can view this planning, select and investigate planning alternatives. Either interesting views are offered, or the users can navigate freely.
1.4	**Access city planning data easily** By picking at interesting parts of the perspective view, citizens can request relevant city planning data, e.g., planned size of buildings, planned usage, etc.
1.5	**Allow for commenting a city planning** Citizens can comment a city planning by annotating a specific perspective view by text or even drawing or by selecting and annotating a specific position or object in the view.
1.6	**Inspect underground infrastructure** Engineers and department heads responsible for the maintenance of telecommunication or power cables, sewers, gas pipes, or fresh water pipes can inspect the current situation underground. When a new subways line is planned, the underground infrastructure must be analyzed and checked for conflicts.
2.	**Business Development**
2.1	**Present an urban space to an investor** For attracting an investor, a 3D city model can help to investigate interesting urban spaces. It shows the general surrounding, transportation infrastructure, etc. This reduces the number of on-site inspections and, thus, saves time and money.
2.2	**Color buildings according to their usage** For judging the surrounding of an object of interest, the agent defines to color the buildings in the perspective view according to their usage type, e.g., living space in green, public buildings in blue, and industrial buildings in red.
2.3	**Measure distances and areas** The perspective view can support simple analysis functionality. E.g., distances to nearby building areas or the overall size of a specific area could be calculated and displayed.
3.	**Real Estate Business**
3.1	**Offering interesting buildings to a customer** An estate agent can use the 3D portrayal services for showing buildings to customers. The perspective views show the appearance of the building, including detailed building structures (doors, windows, balconies, etc.) and real facades. Furthermore the agent can show, describe, and investigate the urban surrounding of these buildings to the potential buyer by help of specific navigation techniques supported by the client application.
3.2	**Highlight buildings to sell** Within an interesting building area, the estate agent can influence the visual appearance of buildings that can be rent or bought. He selects some buildings by their address and others by their owner and defines which color to use for their representation. Furthermore he highlights all nearby public buildings in a different color.
3.3	**Access BIM information easily** For all the buildings in the view, the agent can request additional BIM information such as the age of the building, usage, number of rooms, room sizes, its renovation state, land parcel information, etc. This is done by easily picking the object of interest in the image.
3.4	**Allow for looking inside buildings** Within a city model the agent can select a single building of interest and request a building-specific visualization which allows, for looking into the building and, e.g., see the structure of the rooms at each floor.
3.5	**Calculate and display the visibility of objects**

	For installing a hotel, an investor looks for a building with a good view, i.e., from where a set of relevant points of interests (e.g., church, market place, monuments, palace garden, etc.) have a high visibility. The system allows the agent to define the points of interest and the area in which to search for suitable buildings. Then, the system calculates the visibility of these POIs for each building, maps this information to the building faades, and generates new perspective views.
4.	**Security and Safety**
4.1	**Introduce an operational area to a mobile action force** In preparation for safeguarding a demonstration, action forces need to get an overview of the operation area, including size and topology of street, danger spots, etc. The presentation of the city by 3D portrayal can replace on-site inspections, and thereby reduce cost. If necessary, perspective views can be accessed by mobile devices.
4.2	**Generate views containing escape routes** For a mass event, escape routes have to be defined and must be visualized to the security personnel as well as to the visitors. By help of a styling tool, the safety officer can enrich a perspective view of that area by additional path objects representing these escape routes. This styling description is then published and used by end users.
4.3	**Support Fire Fighters** The fire department requests a detailed indoor building model in case of a fire incident. The 3D indoor model is analyzed in the command center before the fire fighters reach the site. Possibly the visibility in the building is limited due to the smoke. Possible access and escape routes are determined to help the fire fighter evacuating the building.
5.	**Tourism**
5.1	**Investigate a vacation spot** A tourist can use web-based 3D portrayal for investigating a vacation spot already from home. For example, this could support the decision for a specific accommodation according to the appearance of the surrounding.
5.2	**Generate a specific touristic 3D city map** For a city tour, a tourist needs specific information such as the location of bus stops, railway stations, hotels, museums, theaters, hotels, bars, restaurants, the tourist information, etc. By an authoring tool, a municipality clerk can create a specific touristic 3D city map containing this information. A styling tool allows him to define how the integrated information shall be represented.
5.3	**Show the current environmental situation** For making a virtual city model more appealing and more realistic, they can include sun and clouds, which could be in accordance with the real weather conditions.
5.4	**Generate a video tour negotiating selected points of interest** For tourists a virtual city tour can give an impression about important points of interest, shopping facilities, etc. By an authoring tool a municipality clerk can define such points of interests, which are used as way points for the virtual tour. The tourist can select some or all of these way points for generating an individual video tour, e.g., a flight or a city walk.
6.	**Car and Pedestrian Navigation**
6.1	**Illustrate a track by sequences of perspective views** For illustrating a path description, perspective views along that path can be generated. Even the individual path can be embedded into the visualization. The path illustrations can be derived automatically from the defined path.
6.2	**Guide vehicles and pedestrians on the road.** Mobile users are supported by providing 3D animations or interactive visualizations of the course of the route, which has been calculated by an OpenLS route Service. 3D Visualizations are especially useful for pedestrians using their PDAs, since they can get a better idea of where to go if the system contains also landmarks.

Optimizing Virtualization Concepts in (Guest-) Operating Systems

Michael Schöbel
michael.schoebel@hpi.uni-potsdam.de

In the past few years, system virtualization technologies received a lot of attention: virtualization promises to solve data center problems such as server under-utilization and high energy consumption.

From an operating system point of view, virtualization invalidates some basic assumptions about the execution environment: an address space with fixed size, or CPUs with the same speed in SMP systems are two basic examples. Different optimization concepts deal with these invalidated assumptions and try to optimize the execution of operating systems in virtual machines.

In this paper a short survey on different approaches of optimizing systems which utilize virtualization is given. Furthermore, some initial thoughts on how monitoring on different levels of the virtualization stack can help to develop self-optimizing virtualized system environments are presented.

1 Introduction

In general, *virtualization* can be defined as a technology which presents a unique view of an interface to different computation entities. There a five different basic levels of virtualizable interfaces [9]:

- Instruction set level
- Hardware abstraction layer level
- Operating system level
- Library level
- Application level

Depending on the level, different type of applications can use the virtualized interfaces. Hardware support for virtualization is available especially at the instruction set level and at the hardware abstraction layer level (e.g. *Vanderpool* technology from Intel or *Pacifica* from AMD).

To virtualize at the hardware abstraction layer level a *virtual machine monitor* (VMM) is introduced between hardware and operating system level. The VMM provides a virtualized view of the hardware which is used by operating systems installed in virtual machines (e.g. VMWare ESX or Xen [2]). Another approach provides an emulated hardware interface on top of a host operating system without a VMM (e.g. VMWare Workstation or VirtualPC).

Virtualization at operating system level is achieved by providing unique views on the OS system service call interface and the kernel state in a way that each virtual system runs in isolation (e.g. Solaris Zones [12], FreeBSD Jails [6]).

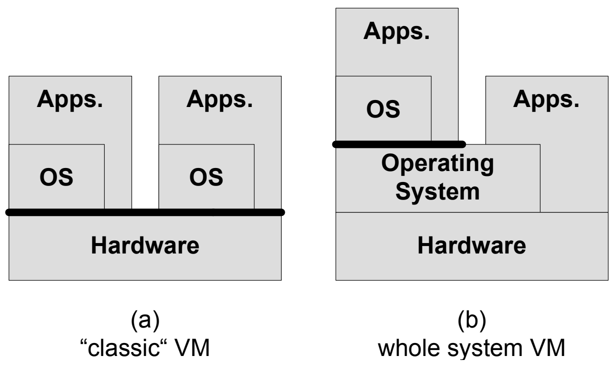


Figure 1: System Virtual Machines

Both approaches support *system virtual machines* [14] - they allow instantiating virtual machines which provide a complete system environment and support multiple processes. This paper focuses on the kind of system virtual machines on which the virtualized interfaces are at the hardware or the operating system level as depicted in figure 1.

A “classic” virtual machine allows the parallel execution of multiple operating systems on the same hardware platform. A VMM (or *hypervisor*) is executed and provides each VM with a unique view on the hardware. Whole system VMs use the operating system services of a host system to build the hardware abstraction for the guest operating system.

Both approaches are used in current systems. There are different implementations available for such virtualization scenarios.

Different implementations can be compared by considering the operating system which can be executed in the virtual machines: Some virtualization technologies require *paravirtualized* guest operating systems. Paravirtualization makes a guest operating system aware of the fact that it is executed in a virtual machine. In the past, some virtualization approaches could only execute paravirtualized guest operating systems [2]. Today, paravirtualization is used primarily for performance reasons: optimized device driver enable guest operating systems to access specific hardware components in an efficient way.

In this paper, some approaches for optimizing virtualized system environments by paravirtualization are described. Furthermore, optimizations of the underlying hypervisor are surveyed as well. Finally, optimization criteria and control possibilities which can be used to build self-optimizing virtualized systems are identified.

2 Optimizing virtualized systems

To support system virtual machines, the virtualization technology must provide a view of a specific hardware platform to the virtual machines. The guest operating system utilizes this virtual hardware interface.

In [11] formal requirements for a virtualizable hardware architecture were defined. Two classes of machine instructions were distinguished: (1) *privileged instructions* are instructions which "trap" into a specific trap handler if executed in (unprivileged) user mode, (2) *control sensitive instructions* which either change the memory configuration or the CPU mode, and (3) *behavior sensitive instructions* whose behavior depends on either the current memory configuration or the CPU mode.

It was proven that a virtual machine monitor can be constructed if the set of sensitive instructions (either control sensitive or behavior sensitive) is a subset of the set of privileged instructions. On such architectures, the VMM can intercept all instructions executed by a virtual machine which could destroy the illusion of the unique view on the hardware.

If these requirements are not fulfilled completely (e.g. the IA-32/x86 architecture is not virtualizable in this sense), the required behavior can be enforced by techniques such as binary rewriting (i.e. replacing sensitive instructions by privileged instructions; after the trap, the original sensitive instructions are emulated in an adequate way).

By considering the formal requirements for virtualization, the area of interaction between the guest operating system inside the virtual machine and the virtual machine monitor can be identified: the execution of privileged instructions is intercepted by the hypervisor. Privileged instructions are used to manage the (1) virtual memory subsystem, (2) IO related activities, (3) interrupt handling in general, and (4) CPU specific features such as the performance counters. Furthermore, the virtualized hardware platform may show a different timing behavior than real hardware.

All of these issues lead to the observation that the operating system is no longer the main resource manager in a virtualized environment. The hypervisor between the virtual machine and the hardware takes this role. The guest operating system still *thinks* that it is the only resource manager, but it is managing a virtual platform with completely different features than real hardware.

Design decisions made during the development of the operating systems assume certain properties of the target hardware platform. Possibly, these assumptions are not longer true if the operating system is used in virtual environments.

For an optimized virtual system, the hypervisor and the virtual machines have to coordinate their activities. This coordination can be achieved by different approaches: (1) the hypervisor may adapt the behavior of the virtual hardware to the assumptions of the executed operating system, (2) the operating system may adapt its behavior to the new environment (paravirtualization), or (3) hybrid approaches may use concepts from both areas.

Recent research has shown that virtualized systems suffer from the mismatch between virtual hardware and operating system assumptions. Different approaches can lead to improved performance of such systems. In the remaining part of the chapter such approaches will be described in more detail.

2.1 Memory management

From the perspective of the virtual machine monitor the (physical) available main memory must be assigned to concurrently executed virtual machines. This assignment problem has similarities to the assignment of physical memory to concurrent processes in traditional operating systems.

In general, a (hardware) memory management unit (MMU) translates virtual addresses into physical addresses with the help of page tables and translation look-aside buffers (TLB). If the size of the actual available physical memory is smaller than the size of assigned virtual memory, paging concepts have to be established. Virtual memory pages are written to secondary storage (*page-out*) if the physical memory page is required from another process. They are loaded back to physical memory when they are required again.

The performance of virtual memory management systems depends on the page replacement algorithm (decides which pages are written to disk if more space is required) and on different optimization heuristics. Such heuristics are e.g. marking pages as *copy-on-write* or to detect “hot” pages which are used frequently and to prevent that such pages are written to disk.

To summarize, a traditional operating system makes the following assumptions about the physical memory: (1) the size of the available memory can be determined and is fix, (2) the physical address space is zero-based and contiguous, (3) process isolation is realized by switching between different address spaces, and (4) paging has to be used if more virtual memory is allocated than physical memory is available.

From the perspective of the hypervisor, these assumptions must be ensured for each guest operating system: (1) the operating system must be able to determine a fixed main memory size, (2) this “physical” memory must be addressable in a zero-based manner, (3) the hypervisor must emulate address space switches in the required way, and (4) the hypervisor must consider that the guest operating system implements its own paging algorithm.

Each of these four points can lead to virtualization overhead, i.e. to calculation or memory consumption on the hypervisor level which are not necessary if the guest operating system is executed directly on the hardware.

(1) If each virtual machine gets a fixed portion of the physical main memory, memory over-provisioning is hard to achieve. Memory over-provisioning allows to execute virtual machines with memory requirements whose sum is higher than the actual available memory. Such an over-provisioning makes sense in server consolidation scenarios.

(2) The requirement of the “physical” address space seen by a virtual machine to be zero-based and contiguous leads to another level of indirection for address translation: the hypervisor can distinguish between virtual addresses (used by the applications running in a virtual machine), virtual physical addresses (the pseudo physical addresses used by the guest operating system in a virtual machine), and real physical addresses. The mapping from virtual physical addresses to real physical addresses must be managed by the hypervisor.

(3) In general, an operating system switches address spaces by enabling a new set of page tables and by flushing the translation look-aside buffer. The TLB optimizes

the address translation between virtual addresses and real physical addresses. These operations require privileged instructions. Therefore, the hypervisor has to emulate such address space switches.

(4) If memory over-provisioning is used in a specific virtualized system, the hypervisor has to implement some kind of paging mechanism. This can lead to subtle performance problems, such as double-paging[1]. Furthermore, the hypervisor requires its own page file which consumes additional resources from the host machine.

To sum up, a virtual machine monitor leads to some space and computational overhead during virtual memory management. Some of these drawbacks were addressed by recent research work.

The memory management unit in VMware ESX server [16] is capable of supporting virtual machines with overcommitted memory. If memory is scarce (for one virtual machine), the hypervisor adapts the assigned memory for all virtual machines. To implement this approach, two problems have to be solved: (1) How can the hypervisor *reclaim* memory pages from a virtual machine? (2) How can the hypervisor *optimize* the memory assignment?

In VMware ESX server, reclaiming is implemented with paravirtualization: a “balloon” driver is installed in the guest operating system. This balloon can be “inflated” (memory is allocated by the driver) or “deflated” (memory is freed). The balloon driver marks this allocated memory as unused and the hypervisor can rearrange the memory assignment. This indirect approach for memory reclaiming has one important advantage: if memory is scarce and the balloon driver allocates memory, the guest operating system may decide which memory pages can be removed from physical memory. This decision can not be done by the hypervisor in an informed way: the guest operating system employs some kind of page replacement heuristics for the executed applications.

A second technique used for memory page reclaiming is transparent page sharing: VMware ESX server detects memory pages with identical content across different virtual machines. The page tables are adapted in a way that such pages are only stored once in physical memory. The implementation is based on page content hashing.

For optimally assigning memory to concurrently executed virtual machines, the hypervisor has to consider the goal memory shares, which were configured by the administrator. In these limits, VMware ESX server optimizes the memory allocation based on a heuristic called “idle memory tax”: the hypervisor tries to determine the number of active memory pages which are really required by the guest operating system. A sampling approach is used: (1) a randomly selected set of memory pages is marked, (2) a page access marks the specific page as “active”, (3) after the sampling period, the fraction of active pages is calculated, and finally (4) the memory share for a specific virtual machine is adapted to reflect its actual required memory size. The basic idea behind this heuristic is that the memory transfer from a machine with fewer active pages to a machine with more active pages leads to fewer page faults.

In [5] an approach for improving virtual memory management in virtual machines

[1]If the hypervisor writes a page to its page file and a guest operating system also wants to replace this page, then a page fault occurs, because the guest OS has to read the page. But the virtual machine does not really need the memory page and only writes it back to disk.

is described which exploits knowledge about the content of specific memory pages: most IO operations use DMA and assigned memory pages as buffers. Each memory page is marked with an associated disk location (ADL) by intercepting IO operations. In this way the hypervisor can monitor memory operations (allocation and eviction) of memory pages used as buffer. The ADL information is used to (1) estimate the working set size (by simulating the cache behavior for the case in which more memory would be available), and (2) implement a secondary (hypervisor level) cache for evicted buffer pages from guest operating system memory. An important property of this approach is that the hypervisor infers the required information passively, i.e. no paravirtualized operating system is required.

In [8] the authors argue that it is important to predict the page miss rate for a specific virtual machine for different memory sizes. With this information, the hypervisor can calculate the optimal memory assignment which leads to a (global) minimum of page faults. To achieve reliable predictions, the assigned physical memory of a virtual machine is subdivided into a regular (guest operating system managed) area and in a (hypervisor exclusive) cache area. This subdivision is achieved by using ballooning as described above. Now, every eviction of a memory page from the memory of the virtual machine is inserted into the cache and page faults are (if possible) satisfied with pages from the cache. If the cache is full, pages are written to the page device as usual. By observing the caching activities over a certain time span, the hypervisor can derive the effects of more (or less) memory to the observed virtual machine.

2.2 IO / device management

To access hardware components, device specific drivers are necessary. These drivers are developed for a specific operating system platform. Therefore, a virtual machine monitor has two possibilities for granting device access to its virtual machines: (1) let the guest operating system use its own device drivers, or (2) provide hypervisor specific virtual devices in each virtual machine.

The first option is hard to realize because the concurrent access of virtual machines to the same device requires synchronization between guest operation systems. Therefore, this option can only be used if such devices are exclusively used by a single virtual machine or the specific device is capable of doing the synchronization on its own without hypervisor support.

The second approach has another problem: How can the hypervisor communicate with the physical device? Or: Who implements the hypervisor version of the device driver? The *frontend driver* can be implemented in a straight-forward manner. A guest operating system specific virtual device driver delivers IO requests to the virtual machine monitor. The synchronization is done on the hypervisor level, the actual device request is handled by a *backend driver* and can be made by (1) a device driver which is integrated into the hypervisor, or (2) by forwarding the request to a privileged virtual machine which accesses all devices exclusively.

Integrating device drivers into the hypervisor leads to more complex hypervisors. Furthermore, additional development effort is necessary to implement the device driver for a specific hypervisor. This can be alleviated if the hypervisor implementation is based on a standard operation system such as Linux.

A privileged virtual machine for handling device access leads to another type of problem: the privileged virtual machine competes with all other virtual machines for resources, especially for CPU time. This leads to accounting problems (how can CPU time, used by the privileged machine, be assigned to the original IO requesting machine) and can lead to priority inversion problems (e.g. if a low priority machine sends IO requests which block more important requests).

Besides the outlined implementation problems, the described IO model again leads to virtualization overhead compared to native execution of the guest operating system and its application: there are “more steps” necessary for IO processing. Some of these issues were addressed in recent research work.

One approach to improve the overall performance of the IO system was presented in [4]: the IO system and the scheduling algorithm work together to schedule such virtual machines that are likely to issue a IO system “near” the current position of an IO device (e.g. a harddisk). They use an implementation of *anticipatory scheduling* which is available in Linux. To work correctly, such an IO scheduler needs information about the currently active concurrent processes which initiate IO operations. Therefore, in a virtualized environment a technique is required to detect which process is running in a specific virtual machine. This problem was solved by observing address space changes for virtual machines on the hypervisor level and by associating these address spaces with the executed processes.

Some InfiniBand devices support concurrent access of multiple applications. In virtualized environments, this advantage is lost. In [7] an approach is presented which allows using such hardware in virtual machines: (1) the hypervisor must allow specific privileged operations for the initialization and configuration of such devices, and (2) paravirtualized device drivers must be loaded in the guest operating systems. This approach shows near native performance for InfiniBand devices in virtual environments.

An important class of IO operations in virtualized environments is network IO. If network IO is not implemented in an efficient way, server consolidation can lead to performance degradation of critical applications. Therefore, especially network IO behavior of virtual machines were investigated.

In [10] the Xen system was investigated: different types of workload (processor-intensive, bandwidth-intensive, latency-sensitive applications) were executed on different configurations of the hypervisor scheduler. In this way, the impact of virtual machine scheduling on IO performance should be determined. Their main results are (1) preempting the driver domain[2] can have adverse effects on IO performance, and (2) mixing latency-sensitive workload with processor-intensive workload leads to bad response behavior of the latency-sensitive application. These effects are caused by the hypervisor scheduler which tries to balance CPU usage but can not react fast enough to IO events.

An implementation of a hypervisor scheduler which is able to react “fast enough” to network IO events was proposed in [3]. To achieve this goal, short periods of unfair CPU assignments are used to decrease scheduling-induced delay while processing network IO. The scheduler prefers virtual machines which are likely to receive or send

[2] *Driver domain* is a Xen term for “privileged virtual machine which accesses devices”

a network packet. The probabilities are determined by monitoring IO activities of virtual machines at the driver domain.

2.3 CPU time management

The virtual machine monitor schedules virtual machines. The applied algorithms are similar to classic algorithms used in traditional operating systems for process or thread scheduling. The main focus is on fairness: each virtual machine should get a fair share of the available computing power. In many cases, the VMM allows configuring the CPU share for each VM.

In the single CPU case, the main cause of overhead in the area of CPU management is interrupt delivery: if a hardware interrupt occurs, the currently running virtual machine is preempted and the hypervisor gets active. Now the target VM for the particular interrupt is determined, the interrupt is queued, and gets delivered to the guest operating system if the VM is scheduled the next time. Therefore, the time between interrupt occurrence and interrupt handling can vary depending on the behavior of other VMs and the actual scheduling algorithm.

If the host hardware contains multiple CPUs or CPUs with multiple cores, the hypervisor could provide each virtual machine with multiple virtual CPUs. In this setup, another class of overhead can be caused by the virtualization: assumptions made by operating systems for SMP systems are invalidated if the CPUs are virtual. These assumptions are: (1) all CPUs in an SMP system are equally fast (without considering any power management techniques, such as frequency scaling), (2) efficient interprocessor synchronization can be achieved via spin locks.

The first assumption does not hold if a virtual CPU has to handle multiple virtual machines. In this case, the “speed” detectable by the guest operating system depends on the workload of concurrently executed VMs.

Generally, virtual CPUs for a specific VM must not run at the same time on the real hardware. This can lead to problems when using spin locks: If a particular spin lock can not be acquired by a CPU, this CPU spins until the CPU which actually holds the spin lock does a release operation. The underlying assumption is that all CPUs are running at the same time and that spin locks are held only for very short time periods. In a virtualized setting, the virtual CPU which holds the spin lock may be preempted (*lock holder preemption* problem). All virtual CPUs that are waiting for this lock will waste CPU time by just waiting in a busy wait loop.

Paravirtualization or optimizations on the hypervisor level can be used to solve these problems.

As already described in section 2.2, the hypervisor can use information about the executed workload for improving scheduling behavior. In [4] the hypervisor tries to infer information about the current process executed in a virtual machine and in [3] a communication-aware scheduler was introduced.

In [15] the problem of multiprocessor virtual machines was investigated. Lock holder preemption can be eliminated in two ways: detect lock contention and prevent virtual machines from spinning, or avoid lock holder preemption altogether. The first approach can be implemented non-intrusively on the hypervisor layer by observing the virtual machine. Virtual machines are only preempted in “safe” states, which can be detected by considering elements such as the current interrupt request level or the (virtual) CPU state. The second approach (lock holder preemption avoidance) can be implemented with paravirtualization: the guest operating system gives hints to the hypervisor how long a lock is held and the hypervisor does not preempt this virtual machine for the specified time.

Another proposed concept is time ballooning. The time balloon is used to ensure that the scheduler assumption “all processors have the same processing speed” is valid. To achieve this, a balloon driver creates virtual workload which actually does no “work” but yields the virtual CPU to the hypervisor. The hypervisor calculates the time of virtual load for each virtual CPU. The load generation depends on the guest operating systems scheduling algorithm and load balancing strategy.

3 Towards self-optimizing virtualized systems

On a high level, the last section can be summarized as follows: for improving the runtime behavior of virtualized environments, (1) guest operating systems have to use information provided by the hypervisor, (2) the hypervisor has to use information provided by the guest operating systems, or (3) a combination of both approaches has to be used.

Different approaches from these categories were described in the last section. But most of the described solutions concentrate on one single aspect of virtualized systems, e.g. improving the memory management. The lack of a combined approach can be explained by the diversity of guest operating systems, optimization criteria, and workload characteristics that are executed in a virtual machine.

Another problem is the level of optimization. On the guest operating system level, optimizations can be introduced by paravirtualization. It is relatively easy to exchange information with the hypervisor, because the paravirtualizing application “knows” the virtualization platform and how to call the hypervisor. The disadvantage is that paravirtualization depends on the underlying virtualization platform and can not be implemented in a platform independent way. Therefore, a different approach is, to place the optimization on the hypervisor level. This non-intrusive approach leads to more complex information gathering processes: the hypervisor gets no information directly from the guest operating system kernel and must infer useful state information by observing the virtual machines.

In short, both approaches have different drawbacks and not all imaginable optimizations can be implemented in both ways because of different available information of the execution state.

An *optimization controller* for virtualized systems can be defined as an application which tries to maximize/minimize a specific control value for the host system. Such a controller could for example try to maximize the (weighted) throughput of all virtual machines, or try to minimize the global page fault rate.

The input values for the controller can be derived at different levels of the virtualization stack and with different granularity. On a coarse-grained level, the controller may use metrics such as CPU consumption of the virtual machines, or the current network interface throughput. On a very fine-grained level, the controller may use information about the currently accessed memory pages, or the currently issued IO operations.

The control points in virtualized systems can also be identified at different levels. The configured CPU or memory shares are high-level variables that can be adapted by a controller. At the low-level, the guest operating system may adapt its page replacement algorithm.

In every case, the current system state must be observed. At the guest operating system level different tools and monitoring frameworks are available. Generally, the hypervisor implementations provide similar tools. The main question is, how to combine the gathered information.

One approach for sharing information about the internal state of an operating system kernel was proposed in [1]: the kernel provides an generic interface to different internal data structure. E.g. applications can query the kernel for information about the state of the memory management system. The kernel returns a list of the available memory pages in the order of their (potential) replacement. Now, the application may adapt its behavior to this information. This approach for information sharing between the operating system kernel and external entities may be useful in the context of virtualized system environments.

In an abstract way an optimization controller have to consider three layers of a virtualized system. Each of these layer provide different observation points to gather information about the current execution state. Furthermore, each layer provides different control points for adaptation.

- Application layer. Observable metrics and events at this layer are e.g. the amount of required CPU time, the number of created threads, or the name and location of accessed files. Furthermore, application specific metrics such as throughput, response time, or latency can be measured.

- Virtual machine layer / Guest operating system layer. At this layer, e.g. page faults, context switches and locking behavior can be monitored. Generally, the workload characteristics of the executed applications can be determined, i.e. whether a compute-bound or IO-bound workload is running.

- Hypervisor layer. By observing the hypervisor scheduler and monitoring accesses to the physical hardware, metrics such as the required main memory, or the CPU utilization can be measured. The compliance of the provided virtual platform with certain fairness guarantees or QoS-constraints in general can be monitored.

The development of a model for such systems and the implementation of an optimization controller for virtualized environments is ongoing work.

4 Summary and conclusion

Virtualized system environments promise to solve problems in today's data centers such as server-underutilization or high energy costs (by using server consolidation). But placing different applications onto the same physical hardware leads to new problems: many assumptions made by operating systems are no longer valid if the hardware platform is virtual. To address this mismatch between assumptions and reality, different approaches can be used: paravirtualization makes the guest operating system aware of the virtual hardware, or hypervisor level optimizations try to consider the special needs of the virtualized applications.

Both approaches require different information for the optimization algorithm, such as the currently executed process inside the virtual machine, which virtual CPU holds a specific lock at the moment, or which memory pages can be released. This information can be gathered at different levels of the virtualization stack.

A combined approach using information provided by the guest operating systems and information measured at the hypervisor level, seems a promising general approach for optimizing virtualized system environments.

Concepts implemented in the Windows Monitoring Kernel [13] can be used as a platform for gathering information at the guest operating system level. How to aggregate and transfer this information to the hypervisor level, and how to implement the optimization controller is ongoing work.

References

[1] Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, Nathan C. Burnett, Timothy E. Denehy, Thomas J. Engle, Haryadi S. Gunawi, James A. Nugent, and Florentina I. Popovici. Transforming policies into mechanisms with infokernel. *SIGOPS Oper. Syst. Rev.*, 37(5):90–105, 2003.

[2] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM Press.

[3] Sriram Govindan, Arjun R. Nath, Amitayu Das, Bhuvan Urgaonkar, and Anand Sivasubramaniam. Xen and co.: communication-aware cpu scheduling for consolidated xen-based hosting platforms. In *VEE '07: Proceedings of the 3rd international conference on Virtual execution environments*, pages 126–136, New York, NY, USA, 2007. ACM.

[4] Stephen T. Jones, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. Antfarm: tracking processes in a virtual machine environment. In *ATEC '06: Proceedings of the annual conference on USENIX '06 Annual Technical Conference*, pages 1–1, Berkeley, CA, USA, 2006. USENIX Association.

[5] Stephen T. Jones, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. Geiger: monitoring the buffer cache in a virtual machine environment. In *ASPLOS-XII: Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, pages 14–24, New York, NY, USA, 2006. ACM.

[6] Poul-Henning Kamp and Robert Watson. Jails: Confining the omnipotent root. In *Second International System Administration and Networking Conference (SANE 2000)*, May 2000.

[7] Jiuxing Liu, Wei Huang, Bulent Abali, and Dhabaleswar K. Panda. High performance vmm-bypass i/o in virtual machines. In *ATEC '06: Proceedings of the annual conference on USENIX '06 Annual Technical Conference*, pages 3–3, Berkeley, CA, USA, 2006. USENIX Association.

[8] Pin Lu and Kai Shen. Virtual machine memory access tracing with hypervisor exclusive cache. In *ATC'07: 2007 USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference*, pages 1–15, Berkeley, CA, USA, 2007. USENIX Association.

[9] Susanta Nanda and Tzi cker Chiueh. A survey on virtualization technologies. Technical report, Department of Computer Science, SUNY at Stony Brook, February 2005.

[10] Diego Ongaro, Alan L. Cox, and Scott Rixner. Scheduling i/o in virtual machine monitors. In *VEE '08: Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 1–10, New York, NY, USA, 2008. ACM.

[11] Gerald J. Popek and Robert P. Goldberg. Formal requirements for virtualizable third generation architectures. *Commun. ACM*, 17(7):412–421, 1974.

[12] Daniel Price and Andrew Tucker. Solaris zones: Operating system support for consolidating commercial workloads. In *LISA '04: Eighteenth Systems Administration Conference*, pages 241–254. USENIX Association, November 2004.

[13] Michael Schöbel. The Windows Monitoring Kernel. In *Proceedings of the 2nd Ph.D. retreat of the HPI Research School on Service-Oriented Systems Engineering*, 2008.

[14] J. E. Smith and Ravi Nair. *Virtual Machines*. Elsevier Science Inc., 2005.

[15] Volkmar Uhlig, Joshua LeVasseur, Espen Skoglund, and Uwe Dannowski. Towards scalable multiprocessor virtual machines. In *VM'04: Proceedings of the 3rd conference on Virtual Machine Research And Technology Symposium*, pages 4–4, Berkeley, CA, USA, 2004. USENIX Association.

[16] Carl A. Waldspurger. Memory resource management in vmware esx server. *SIGOPS Oper. Syst. Rev.*, 36(SI):181–194, 2002.

Implementation of a Service Platform to Evaluate Virtual Team Communication

Matthias Uflacker
matthias.uflacker@hpi.uni-potsdam.de

This report summarizes the design and implementation of a software platform to capture and analyze the information space of informal, collaborative team processes such as global engineering design. By processing the digital traces of ad-hoc communication and information transfer of distributed teams, the platform formalizes relationships between shared information and involved participants in form of semantic, RDF-based *team communication networks*. A resource-oriented platform interface allows the collaborative editing, exploration, and utilization of these networks. Basic concepts of this work have been introduced in previous reports, but have been partially revised and elaborated during the implementation process to fulfill the requirements of ongoing research activities. In this context, the software is now utilized for storing, inferring, and analyzing the communication characteristics of eleven global engineering teams. Architectural properties of the platform as well as early results from its application are presented in order to demonstrate the role of this tool in the continued investigation of information sharing activities in virtual collaboration groups.

1 Introduction

Interpersonal communication, i.e. the informal sharing and distribution of information represents a pivotal and continuous activity in collaboration groups such as design and engineering teams [12,15]. The ability to trace the proliferation of this information and to assess communication characteristics of distributed collaboration processes presents a growing need for global teams and their management. Having a detailed insight into the *who*, *what* and *when* of a team information space early in the process can expose detrimental group characteristics and facilitate the evaluation and improvement of collaborative activities [1]. However, measuring and appraising the communication characteristics of social communities is a challenging task. Sharing information in a team is an intrinsic ad-hoc procedure, utilizing heterogeneous communication channels for synchronous and asynchronous information transfer between two or more participants. Exploring these communication structures usually requires active third-party involvement, long-term observations, or intruding research methods that impair the work of individuals and the team as a whole.

With the digital footprint of shared information steadily growing, computer-supported observation and analysis of team collaboration becomes increasingly feasible. The Internet, and with it the success of the World Wide Web continues to shape the way collaborating groups communicate and exchange information, especially in geographically distributed teams. Email has become the de-facto standard for informal, digital exchange of messages and documents. Wikis, blogs, community platforms, and online collaboration tools are combining social networking services with functionality that until recently was reserved for stand-alone desktop applications. A new generation of software applications is entering global organizations and communities that set out to alleviate distances, support creative collaboration processes, and connect distributed people and information. Naturally, more and more team-generated knowledge is captured and communicated via online services, resulting in project-relevant information that is *just a mouse-click away*. However, the growing number of services provided on the Web today is creating highly distributed and decentralized fragments of project-relevant information. Required knowledge about the existence and location of information objects is likely to diminish quickly in large and long-running projects. Consequently, information often gets disconnected from other related objects and its context.

While we only have started to understand the long-term implications of virtual collaboration processes (cf., e.g., [7, 8, 11]), the decentralized and heterogeneous nature of digital team information spaces has immediate impact on the way distributed team processes can be observed, assessed, and evaluated. Here, information sharing activities are dispersed across multiple communication channels, systems, and formats, resulting in a nontransparent and hard to trace network of actors and data.

To address this issue and to increase the comprehensibility of team collaboration and information sharing practice, this work suggests a method to leverage the digital communication footprint of distributed virtual engineering teams to compute formal, semantic networks of people and information. To model the emerging relationships of involved actors and entities, the concept of a *team communication network* (TCN) is defined and introduced. With the help of this structure, a project-centric view on digital information spaces of virtual teams can be created, which provides a foundation for the graph-based analyses and formal, semantic comprehension of global collaboration.

Several attributes of team communication networks render this approach distinct from previous works and create new opportunities for the computationally supported assessment of virtual collaboration. First, a single network may cover and consolidate information and relationships from multiple communication channels such as emails and Wiki-based collaboration. This allows for a holistic view on cross-channel concerns and the interplay of heterogeneous information sources in a group. Secondly, team communication networks are time-aware, meaning that the evolution of a network is tracked and recorded over the course of a project, providing the opportunity to jump back in time and inspect previous states of the information space. Thirdly, this research is following a highly automated approach to create team communication networks by processing incidental traces of digital information sharing activities such as document archives and server log files. Building only on data that is readily made available through the communication process itself creates a non-intruding and unobtrusive alternative to real-time team observation.

To validate this approach, *d.store*, a resource-oriented service platform for team communication networks has been designed and implemented to support the collaborative editing and analysis of semantic relationships on the Web. It is used actively in a number of global research activities to explore the characteristics of engineering design processes and to identify communication signatures that are conducive or detrimental for the performance of global teams.

This report presents the underlying concepts and the architectural design of the d.store platform. It is structured into the following sections: Section 2 introduces team communication networks and the ontologies used to define the concepts and relationships of the communication channels under analysis. Section 3 continues with an overview of the platform implementation. Some early results and the application environment of the platform used for the generation of realistic team communication networks are detailed in section 4. Section 5 discusses related research before the work concludes.

2 Resource-oriented Team Communication Networks

This section introduces resource-oriented team communication networks as a computer-processable semantic representation of relationships between information resources generated in a collaboration process. In the context of this work, the focus is set on digital, text-based information that is shared asynchronously within global, virtual teams, i.e. a group of geographically distributed people who collaboratively pursue a common goal and who interact and alleviate distance with the help of information technology [9]. The information networks that are spawned implicitly through virtual, inter-personal communication comprise heterogeneous representations of information entities (documents) and participants in the collaboration process. These objects are mutually connected through relationships of various types, e.g. the author-relationship between a document and a person. Many other types of relationships are relevant in the context of team collaboration: relations between two information entities, such as revisions or references, relations between information and people, as for the sender and receivers of an email, as well as inter-personal relations like organizational relationships and hierarchies.

For this graph structure of nodes and edges, the range and semantic of its entities is provided by a set of domain-specific ontologies, describing concepts and relationships for particular information sharing practices. With these ontologies, a formal conceptual model for the existence and constraints of the network building blocks is created, which allows to infer additional relationships out of asserted network values. With such a structured semantic representation of information objects and their identified interdependencies, we can contemplate the role of individual information and persons in a contextual manner and delve into the communication characteristics of collaboration groups.

The graph structure that describes a team communication network represents meta-information about existing objects. In other words, a node of a team communication network represents enriched contextual information about distributed team resources such as the type of information, relationships, or additional descriptive data. The idea of centralizing context information for heterogeneous and distributed information resources creates several requirements for team communication network system. First and foremost, it has to to provide the functionality to assign meta-information to arbitrary, uniquely identifiable resources and make this meta-information again accessible as a resource. The system must be able to provide concurrent access to the graph structures to remote actors, allowing reading and updating the state of a network independently. At the same time, the communication with such a system should be regulated by an interface that is built on open standards, with a well-defined syntax and semantic in order to ensure a low entry barrier and widespread reusability in a heterogeneous environment. From this point of view, the principles of resource-oriented architectures [6, 14] provide a set of architectural constraints for distributed software applications that are capable to address these requirements and establish the basis for a scalable platform for team communication networks. Hence, resource orientation constitutes the underlying design philosophy for the introduced team communication network system *d.store*.

The remainder of this chapter amplifies the theoretical concepts of resource-oriented team communication networks. First, a formal definition of a team communication network is provided. Then, the structure of underlying ontologies that are applied in this work to define the concepts and associations in distributed information sharing processes are presented. A number of inference rules are defined on top of the ontological concepts, that illustrate the computational derivation of node relationships in the networks. Abstracting away from implementation specifics, the chapter completes with mapping core principles of resource orientation to the concept of team communication networks.

2.1 Network Foundations

Team communication networks are defined by a set of typed nodes that are connected via directed, typed relationships. They are used to capture relations between people and information, expressing heterogeneous associations between individuals in a team and/or information resources that occur in the collaboration process, such as emails, Web resources, or shared documents.

Definition. A team communication network TCN is a directed graph $G_{TCN} := (V, E, A, C_V, C_E, C_A)$, where

- V is a set of typed network nodes. A node $v \in V$ is a 4-tuple $\langle cls, attr, t_s, t_e \rangle$ with cls being a list of assigned node types $c \in C_V$, $attr$ as a set of appending attribute instances with types in C_A, and two timestamps $t_s \in \mathbb{N}$ and $t_e \in \mathbb{N} \cup \infty$ with $t_e > t_s$ to mark the beginning and the expiration of a node's validity period.

- E is a set of typed edges, representing the relationships between nodes. An edge $e \in E$ is defined as a 6-tuple $\langle s, p, o, attr, t_s, t_e \rangle$ where $s \in V$ is the source node, $p \in C_E$ is the relation type, and $o \in V$ is the target node. $attr$ represents a list of edge attributes. t_s and t_e are again two timestamps to define the time period of its existence in a network.

- A is a set of typed attributes. An attribute is a 5-tuple $\langle s, p, val, t_s, t_e \rangle$. s is one of $v \in V$ and $e \in E$, the node or edge to which the attribute is assigned to. $p \in V_A$ is an attribute type and val is a literal or numerical data value of the attribute instance. t_s and t_e are defined as before.
- C_V is a collection of node types to characterize the entities (information, people) that are represented by network nodes.
- C_E represents a set of relationship types, which denote the semantic classes of individual network edges. Relationships are directed and can exist between any two nodes $v_i, v_j \in V$. Every edge is of exactly one relation type $p \in C_E$.
- C_A is a set of attribute types that can be instantiated to assign literal or numerical data values to the nodes and edges of a network.

The individual node types of a team communication network are hierarchically ordered through subclass / superclass relationships. The root node type, and hence superordinate of all other types is the *Resource* type, which is implicit element of C_V. Every node $v \in V$ is at least of type *Resource*, meaning that $Resource \in cls$ for a $v \in V, v = \langle cls, attr, t_s, t_e \rangle$. In addition, every node can have an arbitrary number of subordinate types. The type set of a node characterizes the semantic of the information it represents in the collaboration process (e.g. $\{email, meeting\}$).

Figure 1 shows a simplistic representation of a team communication network with $V := \{a, b\}$, $C_V := \{cls1, cls2\}$, $C_E := \{rel1\}$, and $C_A := \{attr1, attr2\}$. The network features a relationship of node a to node b of type $rel1 \in C_E$. Node a and b have different attributes assigned to them, as the figure demonstrates. The temporal properties t_s and t_e of nodes, relations, and attributes are not represented in this figure.

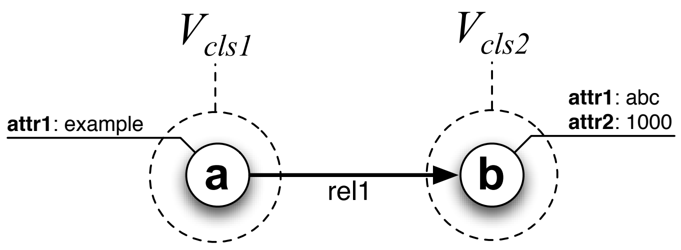
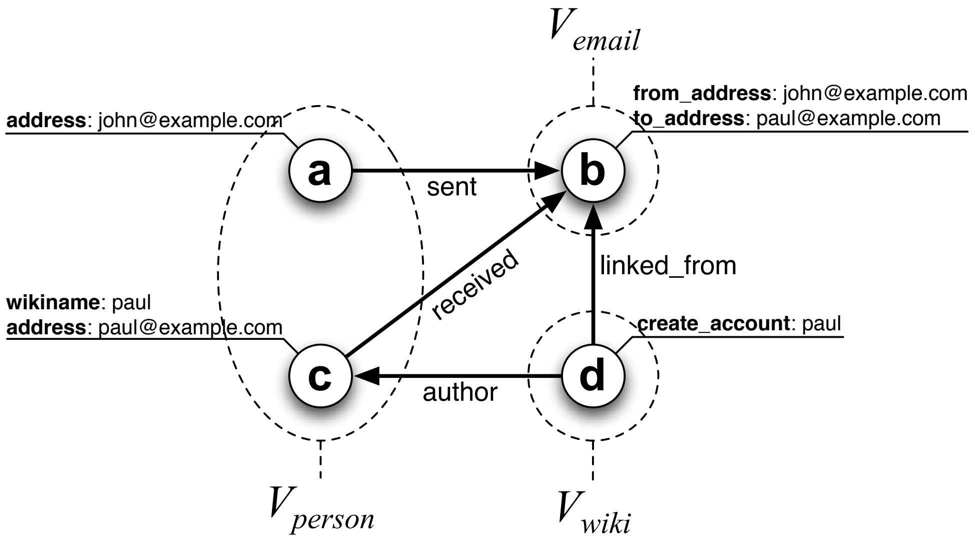


Figure 1: Schematic representation of a team communication network

With the introduction of two timestamps t_s and t_e for the annotation of primary network entities, the point of time at which each individual instance (a node, relation, or attribute) became part of or has been removed from a network is explicitly stored in the model. The value of t_s indicates the point of time at which an entity was created in the network. t_e is set to *infinity* as long as the entity is present in the network. After removal, t_e is set to the date of its deletion, preserving all the information about an entity, but excluding it from subsequent representation. This way, the model cares for the traceability of a communication network's evolution over the course of a project and enables the exploration of previous states of an information space. Having knowledge about when a piece of information has been added to the information space of a team can very likely influence the interpretation of this event. Likewise, being able to reproduce the evolution of relationships between nodes over time can provide additional insights into the communication behavior in collaborating groups.

Figure 2 gives an example for a more realistic network instance. Within the set of node instances $V := \{a, b, c, d\}$ nodes a and c represent two persons, b depicts an email that has been sent by person a, and d presents a wiki page that has been created by person c and is referenced by email b.



V_{email}
from_address: john@example.com
to_address: paul@example.com
address: john@example.com
a
sent
b
linked_from
received
wikiname: paul
address: paul@example.com
create_account: paul
c
author
d
V_{person}
V_{wiki}

Figure 2: Example of a team communication network, showing instances of different types of nodes, relations, and attributes

The set of node types, attributes and relationships that can be assigned to network entities has been defined specifically for the use case of this work. The next section will introduce the ontologies that are used to describe this common set of concepts and relations in the context of virtual team collaboration.

2.2 Network Ontologies

A set of ontologies provide the formal understanding of the concepts, constraints and relationships, which is necessary to describe the state of team communication networks in a computer-processable format. While the ontologies define the conceptual model of a team communication network, the concrete instantiations of ontological concepts represent the different, individual network occurrences on instance level. A team communication network system features a set of ontology definitions and network instances which are defined and encoded in a standardized format.

The Resource Description Framework (RDF) [2,3] and the Web Ontology Language extension OWL [4] provide a suitable framework to encode the ontological elements required for this work. Both standards are used for the specification of ontologies and for the representation of concept and instance models in the platform implementation. The RDF/OWL framework supports a logical separation of concepts and instances through namespaces and the integration of multiple concept models through the import of other namespaces. This way, a decoupling of concept models and instance models across multiple team communication networks can be achieved, which allows independent customization and adaptation of models on individual network level without affecting the models of other networks. The organization of system- and network-specific RDF/OWL graph models for a team communication network system of team communication networks is visualized in figure 3.

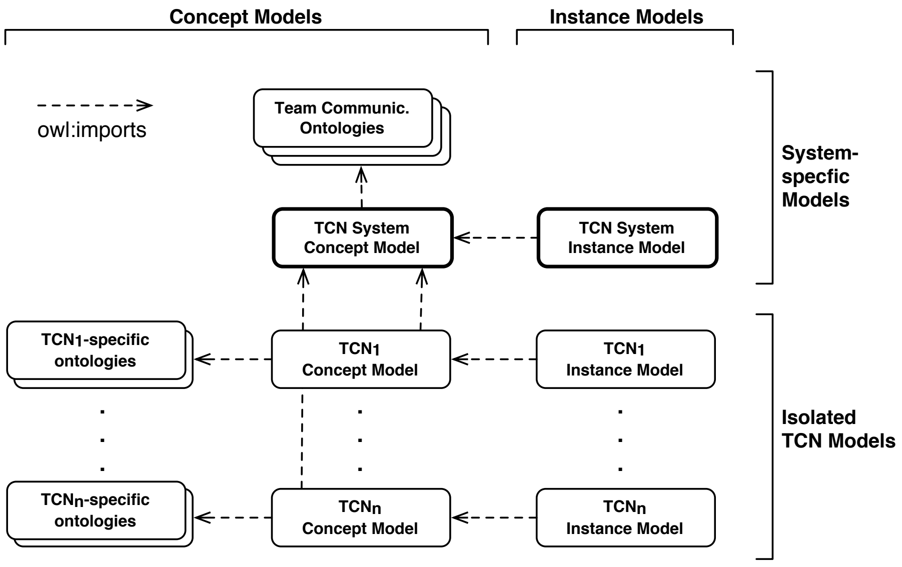


Figure 3: Import hierarchy of RDF/OWL graphs in a TCN system

A global instance model organizes the collection of network instances. The basic concepts that are needed to describe team communication networks are defined in a system-wide concept model, which represents the core ontology of the system. It is presented in section 2.2.2. For every communication channel that the system considers for the analysis of team collaboration, the system model imports a domain-specific ontology model that describes the relevant information and relationship types for that channel. The ontologies that are currently utilized for this work are described in sections 2.2.3 (Web- and Wiki-based information sharing) and 2.2.4 (email communication). Finally, for each network instance in the system, two RDF/OWL models for network-specific concepts and instances specify the individual configuration of a network. The concept models of the network instances import the global type definitions and hence the ontology definitions for team communication provided by the system. Optionally, additional ontologies are imported for reuse into the conceptual model of a network without affecting the state and knowledge base of other networks.

2.2.1 Graphical Representation of RDF/OWL Ontologies

Lacking standards for the graphical notation of RDF/OWL-based ontology models, this report uses a custom presentation language to describe the elements of the platform ontologies. The notation is borrowing concepts from DLG2[1], a graphical presentation

[1] Directed Labeled Graph Two: http://www.charlestoncore.org/dlg2/

language for RDF and OWL, but has been tailored and extended to address the requirements in the context of this work. To give a short introduction to the notation, figure 4 shows a simple ontology model.

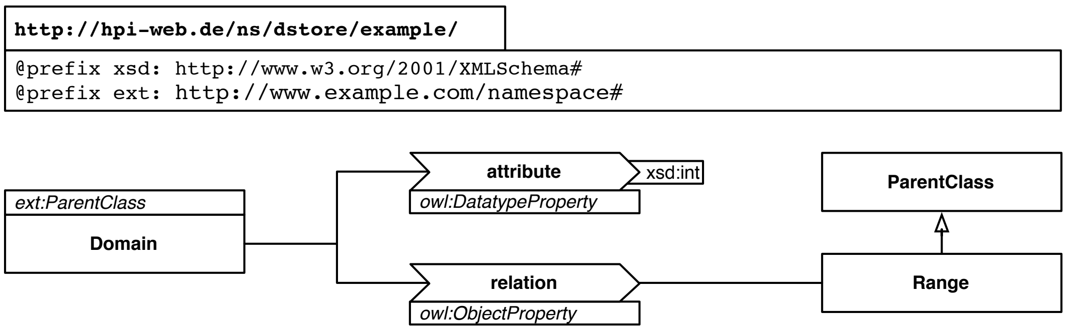

http://hpi-web.de/ns/dstore/example/
@prefix xsd: http://www.w3.org/2001/XMLSchema#
@prefix ext: http://www.example.com/namespace#
ext:ParentClass
Domain
attribute
xsd:int
owl:DatatypeProperty
relation
owl:ObjectProperty
ParentClass
Range

Figure 4: A graphical notation for RDF/OWL-based ontologies.

In this notation, the ontology namespace along with a number of prefix definitions for reused ontologies is stated on top of the graphical representation. Class definitions are represented as boxes and property types are represented as flat, acuminated shapes. Associated superordinates for classes and properties are listed above the particular definition, while additional type allocations are appended below. The example ontology presented in figure 4 features a class $Domain$, which is a subclass of a class $ParentClass$ defined in the *ext* namespace. Note that internal superclass/subclass relationships within a namespace can also be expressed by a connecting arrow as shown for the classes $ParentClass$ and $Range$ in the ontology namespace. The ontology also defines two properties, a data property $attribute$ and an object property $relation$. The latter defines a relationship type between two resource classes. The domain and range types of a property are indicated by inbound and outbound connections to the associated resources. In this example, the ontology specifies that all instances are of type $Domain$ if they are subject of a property $relation$. Likewise, the range of $relation$ asserts that targeted node instances of that relationship are of type $Range$ (and hence of type $ParentClass$). In the case of data-valued properties, the type of the value range is stated on the outbound side of the property (e.g., *xsd:int* for property $attribute$).

With these graphical primitives, an adequate overview of the ontology specifications that form the foundation of team communication networks can be provided. The following section introduces these ontologies and explains their elements.

2.2.2 System-specific Concept Model

This section gives an overview of the basic semantic elements for a system of team communication networks. An excerpt of the ontology, its basic concepts and property definitions is shown in figure 5.

The most fundamental concept of resource-oriented team communication networks is the $Resource$ type. On the most abstract level, a resource represents a node in a network, i.e. an entity of information or individuals for which contextual meta-information is provided. Following the definition of team communication networks in section 2.1,

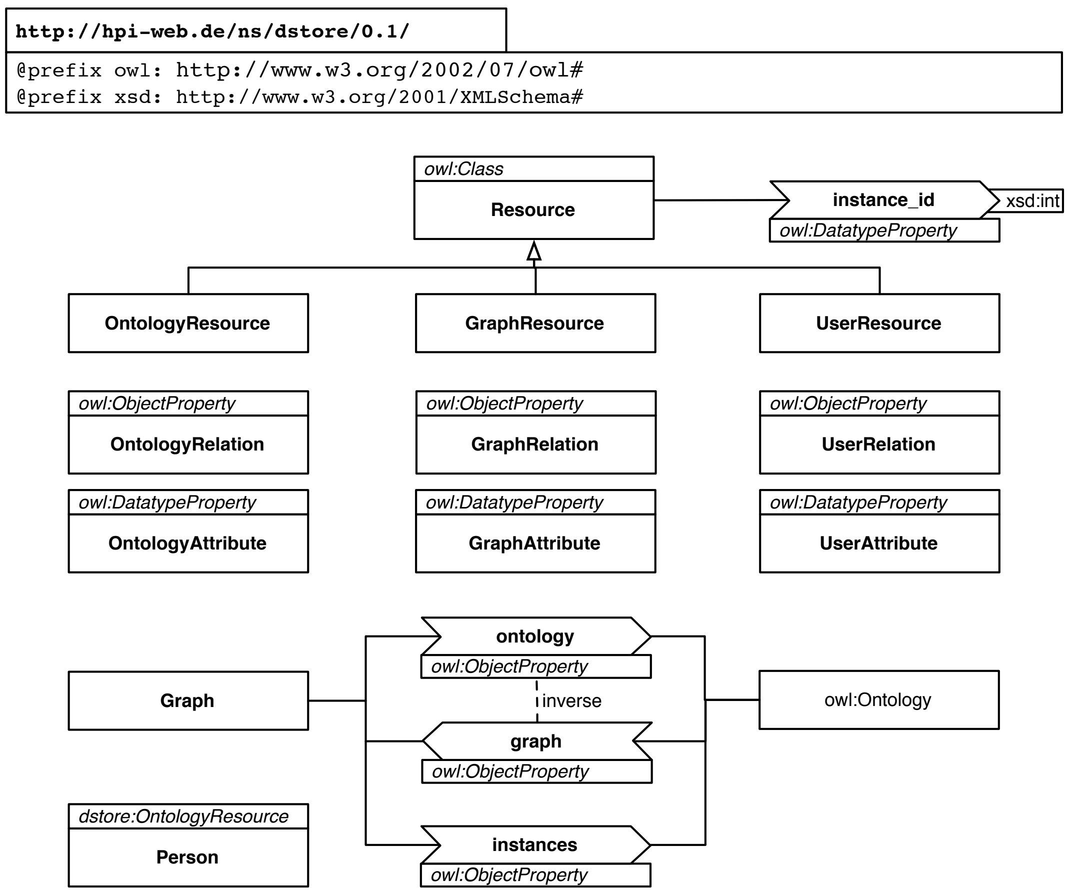


Figure 5: The central d.store ontology defining global concepts and relations of team communication networks.

all network nodes are of type $Resource$. To further concretize the kind of information represented by a node, the ontology defines three subordinate classes of node types. Differentiating between these three classes allows to distinguish between concrete types that are defined in domain-specific ontologies for team communication ($OntologyResource$), in ontologies imported on network-level ($GraphResource$), or in network-specific concept models that provide space for custom, user-defined resource types ($UserResource$).

The second first-class entity of team communication networks are edges, which represent directed relationships between network nodes. To define the type of an edge, and with it the semantic characterization of a relationship, the ontology specifies again three classes that serve as superordinates for concrete relation types. As for the node types, the class of a concrete relation type denotes its origin within the model: $OntologyRelation$ for system-defined relationship types, $GraphRelation$ for graph-specific relations, and $UserRelation$ for custom relation types that are defined ad-hoc. Relation classes are subtypes of the $ObjectProperty$ class of OWL, constraining relation instances to express a relationship between two resources only.

Similar to relation types, the available attribute types of a team communication network are subclasses of one of the abstract concepts $OntologyAttribute$, $GraphAttribute$, and $UserAttribute$. These are a specialization of the $DatatypeProperty$ of OWL, restricting the value of an attribute instance to a literal domain.

The concept of a team communication network itself is represented in the ontology by the $Graph$ type. The dedicated concept and instance models of a network are appointed by the $ontology$ and $instances$ properties, respectively. Finally, the ontology defines a first concrete node type $Person$, representing individual actors in the collaboration process.

The following two sections present ontologies of node, relation, and attribute types that are applied in the creation of team communication networks in order to formalize the contents and associations of digital, asynchronous team information sharing.

2.2.3 Web and Wiki Resources

The Web of today has become a collaboration platform to support team interaction and information sharing activities. Even if no central Web application is used to coordinate the teamwork or the communication between members of a team, the Web is still regularly used for research, benchmarking and support. Information that is gathered in arbitrary resources is shared and distributed among process participants in order to disseminate insights and knowledge. This can be done in many different ways, e.g., verbally or by sending emails with hyperlinks or attachments. Obviously, Web resources do not only reference other hyperlinked Web resources, but are also linked from other resources, which are not necessarily Web resources. This link dependencies between Web resources and the more general concept of a resource presented before is captured by the ontology depicted in figure 6. By defining the two relationship types $hyperlink$ and $linked\_from$ as inverse properties, a back reference from and to hyperlinked resources can be inferred.

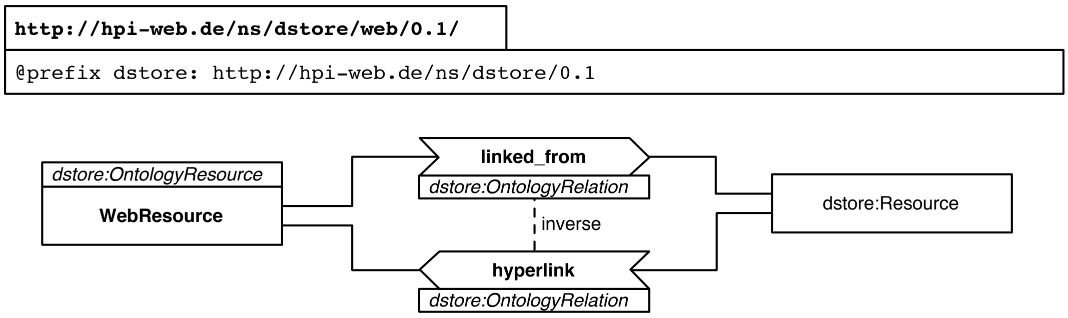


Figure 6: An ontology for hyperlink relations between heterogeneous types of resources.

One of the enablers for the trend towards social Web-based information exchange in virtual communities was the emergence of Wiki applications that allow distributed and collaborative editing of Web pages. Wikis are now widely adopted in professional scenarios and project-based collaboration to support in information management and organizational tasks. The content is generally contributed, accessed, and incrementally revised by an interacting community of registered users, which makes this medium applicable to the analysis of collaboration practices [10]. Wiki applications that are keeping logs of page revisions and changes allow the examination of relationships between created content and contributers. Authors having created or edited a Wiki page can be identified by inspecting account names assigned to the revisions. Additionally, by inspecting the delta of two revisions, the individual contribution of an user becomes evident.

Figure 7 shows the basic association types in conjunction with Wiki-based collaboration. As a specialization of a Web resource, a Wiki page naturally is also domain and range of hyperlink relations. The ontology defines two attribute types $create\_account$ and $edit\_account$ for the resource type $WikiPage$. These attributes are used to appoint the Wiki account name of the creator or of one of potentially many editors of a Wiki page. The ontology further defines an attribute type that is used to assign Wiki names to a person.

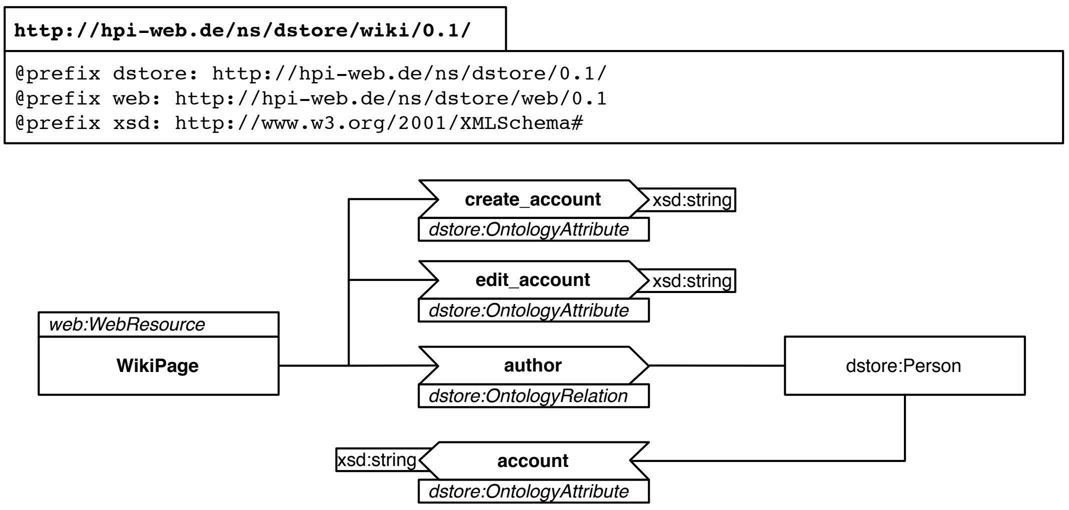


Figure 7: An ontology for wiki-related information sharing concepts.

2.2.4 Email

Email messaging is one of the most prominent technologies used today to digitally exchange everything from simple, text-based information to rich media in an asynchronous and reliable manner. Studies reveal that global, virtual teams largely rely on daily email communication [9]. Several advantages have let to a strong pervasiveness of email usage in virtual collaboration. Today, access to email systems is available almost anytime and anywhere, allowing asynchronous, ad-hoc transmission and retrieval of messages without needing to organize and participate in synchronous interactions. Switching between online and offline scenarios makes email messaging a convenient tool for composing or receiving information on-demand or at a convenient point in time. The personal mailbox simultaneously serves as a message archive and information repository. Email distribution lists provide a convenient way to share information with a larger group of recipients quickly and at once.

These benefits turn email communication into an important carrier of team information, which supports this work's approach of computer-supported diagnosis of team

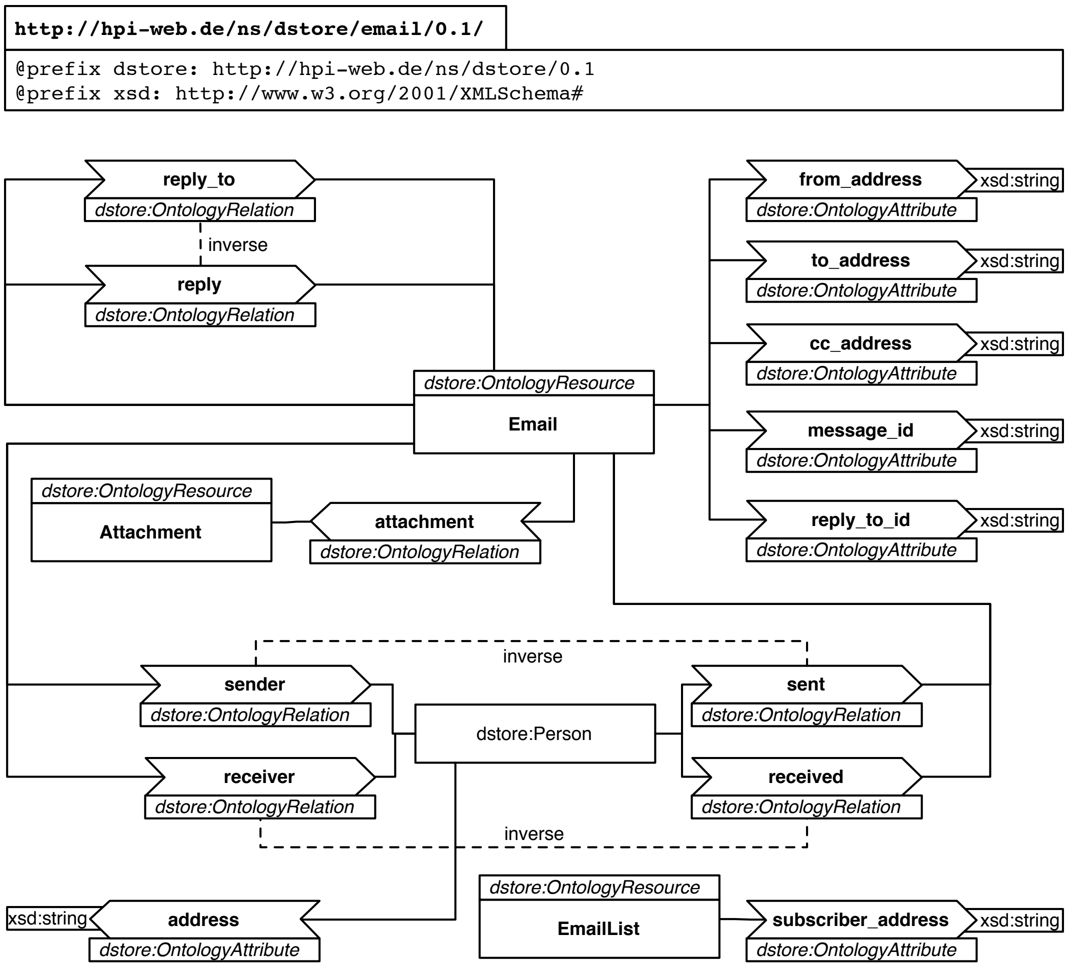


Figure 8: The d.store ontology for email communication.

relationships. Figure 8 shows the ontology that defines the node types, relationships, and attributes used to describe email-related communication activities in team communication networks. The $Email$ class is the central node type for transmitted email documents. Instances of this type have a number of lexical attributes such as mailbox addresses listed in the *to* and *cc* fields of an email, a unique message ID, or the ID of a previous message to which an email replies to. This dependency between two emails is expressed through the $reply$/$reply\_to$ relationship types. Email addresses are assigned to persons via the $address$ attribute. Bi-directional relations between an email and sending or receiving persons are expressed via the $sender$/$sent$ and $receiver$/$received$ relationship types. Additionally, the concept of an $EmailList$ is introduced, which is characterized by having a number of email recipients signed up to the distribution list via the $subscriber\_address$ attribute.

As the different ontologies presented above indicate, multiple types of relationships between one and the same person and different types of information objects can exist. These relations are, however, first of all grounded on interpretation and lexical identifiers associated with information objects and persons, such as email addresses or account names. Only if a mapping between those aliases and concrete persons is provided, many of the defined relation types can be instantiated correctly. Such a mapping is defined through the assignment of $address$ and $wikiname$ attributes to person nodes. With this annotation, inference rules can be applied to directly spawn relations between information objects and the actual persons being involved. The following section specifies the rules that provide this formalization to automatically derive relationships in team communication networks based on matching attributes values.

2.3 Rule-based Inference of Node Relationships

If associations are to be identified in different sources of transmitted information, a mapping between channel-specific literal identities of an actor and an actual person node in a network must be accomplished. To give an example, it can only be inferred through our knowledge or assumptions that an email with the sender address *'john@example.com'* has actually been sent by a person named John. This can only be true if that person exists and if that address is assigned to that person (eventually with several other addresses). Likewise, the authoring of a Wiki resource through an account named *'john'* can only be related to the same person if one has the knowledge that this account name is also associated with him. In order to correctly reflect the communication activities of individuals in a team, the different addresses, aliases, etc., of a person must be attributed to a single person node to which the resulting relationships shall be assigned to.

With a set of node attributes and the help of inference rules, relations between information objects and persons can be derived computationally and instantaneously. This automatic inference of relationships decreases the number of relations that need to be explicitly stated in the graph, leading to increased flexibility and less overhead in the manipulation of networks, nodes and attribute values. By applying these rules with every change of a network's state, a consistent view on the attribute-based relationships is assured.

Based on the ontologies presented above, a number of inference rules have been specified for a team communication network system to dynamically derive relationships between network nodes. These rules are defined as pairs of preconditions and implied postconditions, presented here in the form $antecedent \Rightarrow consequent$. If the left-hand precondition of a rule is met for any combination of nodes in a network, then the right-hand postcondition is inferred to also hold true. Using the ontological concepts defined for email communication, an inference rule to establish $sender$ relationships between emails and people can be specified in the following form:

$$from\_address(x,y) \wedge address(z,y) \Rightarrow sender(x,z)$$

This rule states that for any node x with an attribute $from\_address$ of value y and any node z with an $address$ attribute of the same value y, a $sender$ relationship between x and z is inferred. Note that, due to the inverse specification of the *sender* relationship, z instantly has an inferred *sent* relation to node x whenever this rule is applied in a network. Analogical rules can be constructed to spawn direct relations from an email message to its recipients:

$$to\_address(x,y) \wedge address(z,y) \Rightarrow recipient(x,z)$$
$$cc\_address(x,y) \wedge address(z,y) \Rightarrow recipient(x,z)$$

Accordingly, associations can be inferred that are relating person nodes to wiki resources, which have been edited or created by this person:

$$create\_account(x,y) \wedge wikiname(z,y) \Rightarrow author(x,z)$$
$$edit\_account(x,y) \wedge wikiname(z,y) \Rightarrow author(x,z)$$

In this case, the two rules infer an authorship relation between a node that has been created or edited by a certain account name and a node that has this name assigned as a Wiki name.

Inference rules are also used to instantiate relationships between two information objects that feature logical interdependencies. One example is the $reply\_to$ relationship between an email and a preceding message to which the sender of the email replies. On message data level, this interdependency is encoded in the email header via unique message IDs and the value of a *reply-to* field that features the ID of a foregoing message. The following rule triggers the inference of $reply\_to$ relationships (and the inverse $reply$ relations) with the occurrence of two email nodes and matching ID attributes:

$$reply\_to\_id(x,y) \wedge message\_id(z,y) \Rightarrow reply\_to(x,z)$$

More complex dependencies between nodes can be expressed by adding additional prerequisites to the antecedent of a rule. The following example stipulates that all emails, which have been sent to an $EmailList$ have been received by the persons who are subscribed to that list:

$$received(w,x) \wedge subscriber\_address(w,y) \wedge address(z,y) \Rightarrow received(z,x)$$

This rule defines that any email x that has been received by an email list w has also been received by a node z, if w and z have matching $subscriber\_address$ and $address$ attributes.

Referring back now to the network representation in figure 2, it becomes apparent that only the attributes and the $hyperlink$ relationship need to be modeled explicitly in the graph. All other relationships and node types can be inferred by a team communication network system with the help of the presented rules and the attribute values that are assigned to the four nodes.

2.4 Resource Orientation

Representational State Transfer (REST) [6] defines a set of architectural principles to build highly distributed, highly scalable hypermedia applications on the Internet. One or more uniquely identifiable resources that share a uniform interface provide the state and functionality of a *RESTful application*. A stateless application protocol regulates the transfer and manipulation of the application state between clients and server. Following the REST principles, a resource-oriented application constitutes addressable resources that represent meaningful concepts (*nouns*) for the description of the application's state. In the context of this work, every network instance and every individual node in a network become a uniquely identifiable resource that is provided by the service platform. Additional resources manage the collections of entities, such as the collection of nodes in a network, and the attributes or relationships for a particular node instance. The unified interface of the resources allows independent and distributed retrieval and manipulation of the application state, such as reading a node or creating a new resource in a collection.

Every resource can be represented in different formats, allowing different clients with different capabilities and intents to make use of the functionality provided by the system. Therefore, REST defines a process for negotiating the content type between requesting clients and the server. The resource-oriented platform for team communication networks is utilizing this methodology to provide different representations for one and the same resource, such as a network or one of its nodes.

3 Platform Implementation

To validate the feasibility of our approach, we have implemented *d.store*, a resource-oriented platform for the construction of social information networks. The platform provides a REST-based [6] service interface for accumulating derived information structures from arbitrary collaboration technologies such as email messaging and Wiki applications. Using the ontologies introduced above, a number of pre-defined concepts and relations are readily provided, that form the basis for representing semantic team communication networks in form of OWL/RDF triple statements [2–4].

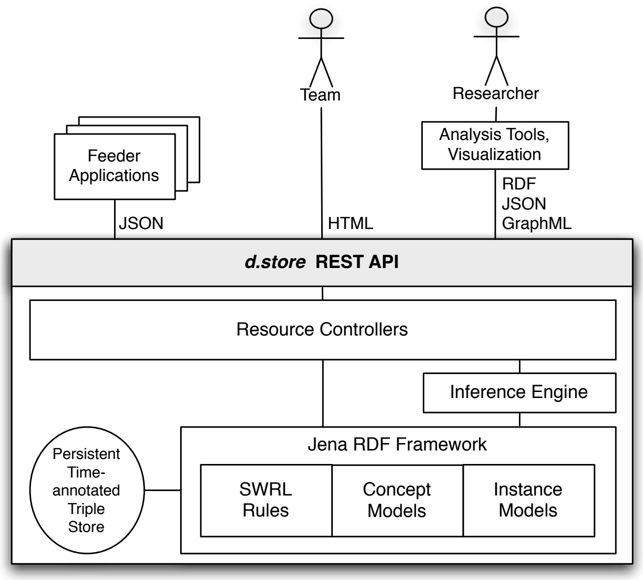


Figure 9: d.store architecture outline

To organize the concept and instance models, the platform is built on top of the

Jena semantic web framework[2]. The framework provides the functionality to store and query RDF/OWL triples, import ontologies, and applying algorithms for the inference of triples out of the set of asserted statements in a model. The application of the inference rules presented in section 2.3 is based on the Pellet[3] reasoner library, a rule engine that interprets rule statements that have been encoded using the Semantic Web Rule Language format SWRL[4]. The following XML representation of SWRL specifies the rule to infer $sender$ relationships between two nodes (cf. section 2.3):

```
<swrl:Variable rdf:about="#x" />
<swrl:Variable rdf:about="#y" />
<swrl:Variable rdf:about="#z" />

<swrl:Imp rdf:about="&email;SenderRule">
  <swrl:head rdf:parseType="Collection">
    <swrl:IndividualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="&email;sender" />
      <swrl:argument1 rdf:resource="#x" />
      <swrl:argument2 rdf:resource="#z" />
    </swrl:IndividualPropertyAtom>
  </swrl:head>
  <swrl:body rdf:parseType="Collection">
    <swrl:DatavaluedPropertyAtom>
      <swrl:propertyPredicate rdf:resource="&email;from_address" />
      <swrl:argument1 rdf:resource="#x" />
      <swrl:argument2 rdf:resource="#y" />
    </swrl:DatavaluedPropertyAtom>
    <swrl:DatavaluedPropertyAtom>
      <swrl:propertyPredicate rdf:resource="&email;address" />
      <swrl:argument1 rdf:resource="#z" />
      <swrl:argument2 rdf:resource="#y" />
    </swrl:DatavaluedPropertyAtom>
  </swrl:body>
</swrl:Imp>
```

Team communication networks are gradually generated through posting identified objects and associations to the resource-oriented service interface provided by the platform. The internal RDF representations for the networks are decomposed into triple statements, which are processed in main memory for performance reasons. A persistent and consistent copy of the asserted statements is stored in relational tables. Adopting the Jena semantic framework to our needs, the notion of an RDF triple has been extended by two additional timestamps which mark the beginning and the end of a statement's validity period. These additional two values are used to implement timestamps t_s and t_e, which are defined for the elements of team communication networks, and which allow the reproduction of the chronological evolution of the network. The

[2]http://jena.sourceforge.net/
[3]http://pellet.owldl.com/
[4]http://www.w3.org/Submission/SWRL/

resulting time-annotated RDF graph provides the basis for reasoning on and analysis of the constructed networks.

A number of helper applications to feed information and relationships identified in email archives or wiki logs have been implemented to automate the data import. Read and write access to the networks is provided via the unified interface of the HTTP/1.1 protocol [5]. d.store supports a number of representation formats for the provided resources of a network, including RDF/XML, the JavaScript object notation JSON, GraphML[5], and HTML. Clients can use the *Accept* header field of HTTP to negotiate the representation type that is returned from the server upon request.

4 Application

Starting to employ the platform in the analysis of multi-modal communication behavior of virtual teams, we used data collected from eleven global engineering projects, which were running for a period of nine month. The projects were placed in a joint academic partnership between Stanford University and six global institutions, with each project team distributed and composed out of two groups of global and local students. Teams were set up in a multi-disciplinary way, involving students with backgrounds in mechanical engineering, software engineering, economics, as well as product and industrial design. The project teams were working independently on prevailing engineering design tasks that were accompanied by global enterprises and corporate liaisons under realistic project conditions, budget and time constraints. The design process involved early need finding activities, user observations, iterative prototyping and evaluation, and finished with a fully functional and documented prototype, which was handed over to the company. The internal organization of individual team processes with regard to decision making, meetings, presentations to users and customers, and the definition of work packages was fully left to the responsibility of the student teams. However, in addition to the similar team structures and budget constraints, all projects were synchronized in terms of start and end dates, major milestones and deadlines, which supports and simplifies the comparison the information sharing activities across different teams.

The teams have been provided with the required IT infrastructure to support synchronous communication (audio/video conferencing, multi-user desktop systems) and asynchronous communication (email lists, project wikis, shared document spaces) with their peer team members off-site. The email archives, Wiki and server log files that have been generated during the nine month of intense collaboration constitute now the data basis for current research into asynchronous information sharing activities via email, project wikis, and document shares. All of these technologies turned out to be highly-adopted tools in the observed projects.

We started with the generation of team communication networks out of approx. 8700 project-related emails (containing more than 2900 hyperlinks and 1700 file attachments), 1200 wiki resources and shared documents in public online folders. The average ratio between emails being sent and the number of direct relationships to other information resources submitted in the form of attachments and hyperlinks was approx.

[5]http://graphml.graphdrawing.org/

1 to 0.6. Obviously, email messaging was commonly used as a tool to share project information that is not only encoded in the message itself, but is provided in files or in external information resources on the Web, supporting this approach of creating a multi-modal, contextual view on networked team communication and information sharing processes.

With the import of data from multiple projects, we have created semantically rich social information networks with each one connecting nodes in the range of 1,000 resources or more. The average amount of asserted, time-annotated RDF triple statements per resource was above seven statements. This number does not include inferred statements, which increases the effective number of associations for a resource considerably. Taking the statements derived by the presented inference rules into account, the total number of RDF statements grew up to more than 20.000 statements per project and team communication network, created only out of the traces of digital communication. First results show that the performance of the platform, and especially the inference engine has proven to fully scale up to the analysis needs. Current activities comprise the visualization and combined analysis of the generated network structures and the content of the information resources in order to reveal hidden characteristics of team communication signatures in global virtual teams.

5 Related Work

The assessment and analysis of communication signatures in collaborating teams has been the topic of several preceding works. This section briefly discusses some of the related research conducted in this area.

Yen [20] has created an information retrieval system to capture, index, and distribute design information. Here, the focus is on direct design conversations that involve face-to-face communication and sketch activities.

Reiner [13] proposes a modeling framework to support collaboration and distributed knowledge management for design teams. His work demonstrates the use of design history as a source of insight into team design processes. It identifies multiple correlations between historic design data and team performance in the domain of software engineering.

Both works implement a software prototype that is used to record design information during a (mostly local) collaboration process. Both tools require direct interaction with the designers, which presents a different, more penetrating approach to design observation than the one followed in this work.

Müller [10] presents a graph-based approach to analyze the evolution of Wiki networks. While using only server log files to create separate networks for the analysis of social relationships and link relationships, the goal of this research work is to support self-organized knowledge management in Wiki communities. Unlike team communication networks, which provide a broader view on team information spaces, the work is dealing with Wiki-related information only.

With the development of a team communication network platform and its application for the analysis of multi-modal communication signatures of global virtual teams, this research relates to and builds on results of these works, aiming to contribute new knowledge to the design research community.

6 Conclusion & Next Steps

This report elaborated on the design and implementation of a resource-oriented platform for team communication networks. A software system that provides team communication network services for the analysis of information sharing practices in collaboration groups has been motivated with the analytical potential it introduces to the field of design research and of global, virtual teams in general. The theoretical principles of team communication networks have been presented and mapped to the REST architectural style for resource-oriented distributed systems. With that design decision, an appropriate basis for the implementation of utilizable and reusable services for team communication networks has been established. Building on a data model that is grounded on a formal framework of ontological concepts, the approach aims for the flexibility and expressiveness needed to specify, interpret and infer the informative value of team communication networks.

A reference implementation, *d.store*, has been developed and is currently applied to demonstrate the feasibility of this approach and to start with the analysis of global, virtual teams. Continuing with the investigative exploration into the digital communication archives of distributed design teams, a number of projects have started to use and extend the services provided by the d.store platform. This involves the visualization and more profound analysis of the network models, as well as the combined analysis of information content and network structures to locate significant objects in team communication.

This novel approach to the analysis of global virtual design teams has been published and presented to the research community at international conferences, workshops and events [16–19]. The next steps in this research work comprise a deeper investigation into multi-modal communication signatures and their impact on team performance. The results of this analysis are expected to be published in early 2009.

References

[1] Liam Bannon and Susanne Bodker. Constructing common information spaces. In *ECSCW'97: Proceedings of the fifth conference on European Conference on Computer-Supported Cooperative Work*, pages 81–96, Norwell, MA, USA, 1997. Kluwer Academic Publishers.

[2] Dave Beckett. RDF/XML syntax specification (revised). W3C recommendation, W3C, February 2004.

[3] Dan Brickley and R.V. Guha. RDF vocabulary description language 1.0: RDF schema. W3C recommendation, W3C, February 2004.

[4] Mike Dean and Guus Schreiber. OWL web ontology language reference. W3C recommendation, W3C, February 2004.

[5] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC, The Internet Engineering Task Force, 1999. `http://www.ietf.org/rfc/rfc2616`.

[6] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000.

[7] E. Hustad. Knowledge networking in global organizations: The transfer of knowledge. In *SIGMIS CPR '04: Proceedings of the 2004 SIGMIS Conference on Computer Personnel Research*, pages 55–64, New York, NY, USA, 2004. ACM Press.

[8] Andreas Larsson. Making sense of collaboration: the challenge of thinking together in global design teams. In *GROUP '03: Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work*, pages 153–160, New York, NY, USA, 2003. ACM Press.

[9] J.S. Lurey and M.S. Raisinghani. An empirical study of best practices in virtual teams. *Information & Management*, 38(8):523–544, 2001.

[10] Claudia Mueller. *Graphentheoretische Analyse der Evolution von Wiki-basierten Netzwerken für selbstorganisiertes Wissensmanagement*. Gito, April 2008.

[11] MJ Perry, R. Fruchter, and D. Rosenberg. Co-ordinating distributed knowledge: A study into the use of an organisational memory. *Cognition, Technology & Work*, 1(3):142–152, 1999.

[12] Steven Poltrock, Jonathan Grudin, Susan Dumais, Raya Fidel, Harry Bruce, and Annelise Mark Pejtersen. Information seeking and sharing in design teams. In *GROUP '03: Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work*, pages 239–247, New York, NY, 2003. ACM.

[13] Kurt A. Reiner. *A Framework for Knowledge Capture and a Study of Development Metrics in Collaborative Engineering Design*. PhD thesis, Stanford University, Stanford, CA, USA, 2006.

[14] Leonard Richardson and Sam Ruby. *RESTful Web Services*. O'Reilly Media, Inc., May 2007.

[15] Diane H. Sonnenwald and Leah A. Lievrouw. Collaboration during the design process: a case study of communication, information behavior, and project performance. In *ISIC '96: Proceedings of an international conference on Information seeking in context*, pages 179–204, London, UK, UK, 1997. Taylor Graham Publishing.

[16] Matthias Uflacker. Resource-oriented knowledge sharing in user-centered design communities. In *Proceedings of the 10th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems*, September 2007.

[17] Matthias Uflacker. A resource-oriented information network platform for global design processes. Proceedings of the 2. Ph.D. retreat of the HPI Research School on Service-oriented Systems Engineering 23, Hasso-Plattner-Institut für Softwaresystemtechnik, 2008.

[18] Matthias Uflacker and Alexander Zeier. d.store: Capturing team information spaces with resource-based information networks. In *IADIS International Conference WWW/Internet 2008*, Freiburg, Germany, October 2008.

[19] Matthias Uflacker and Alexander Zeier. A graph-based approach to assessing multi-modal team communication in global organizations. In *IEEE Symposium on Advanced Management of Information for Globalized Enterprises*, September 2008.

[20] Samuel J. Yen. *Capturing Multimodal Design Activities in Support of Information Retrieval and Process Analysis*. PhD thesis, Stanford University, Stanford, CA, USA, 2000.

Modelling Security Configurations for Service-oriented Architectures

Michael Menzel
Hasso-Plattner-Institute
michael.menzel@hpi.uni-potsdam.de

In this report I will summarize my research work of the past six month that has been focused on the model-driven generation of security policies based on security annotations in business process notations.

Business process modelling represents a cornerstone of process-aware information systems and provides an abstract view on organisational workflows. The specification of such processes is heavily tool-supported and facilitates a mapping to service-based systems that provide a suitable foundation to execute business processes. Service-oriented Architectures emerged as a concept to deliver a flexible infrastructure to allow independently developed software components to communicate in a seamless manner. Along with an increased connectivity, the corresponding security risks escalate exponentially. However, security requirements are usually defined on a technical level, rather than on an organisational level that would provide a comprehensive view on the participants, the assets and their relationships regarding security.

To facilitate a high-level specification of security requirements, I proposed Secure-BPMN as an extension for the Business Process Modelling Notation (BPMN). Secure-BPMN enhances BPMN by security elements which allow to evaluate the trustworthiness of participants based on an rating of enterprise assets and to express security intentions such as confidentiality or integrity on an abstract level. Based on a classification for enterprise security, I will describe properties and artefacts that enhance BPMN with regard to security in this report.

My aim is facilitate the generation of security configurations in any policy language. For this purpose, I foster a model-driven approach: Information at the modelling layer are gathered and translated to a domain-independent security model. Concrete protocols and security mechanisms are resolved based on a security pattern system that is introduced in the course of this report.

1 Introduction

Business Process Modelling gain more and more attention, as it is the foundation to describe, standardize and optimize organizational workflows to enhance the enterprise's competitiveness. A business process model is defined as a set of activities and execution constraints between these activities [29] and can be used to describe complex interactions between business partners and related business requirements on an abstract level.

At the same time, IT-infrastructures evolved into distributed and loosely coupled enterprise system landscapes such as Service-oriented Architectures, which expose a companys assets and resources as business services. The SOA paradigm provides a vast amount of flexibility in the way complex software systems are implemented. The independent nature of the services, with respect to operating systems and system architectures, facilitate a composition of different services. Various business services are orchestrated by business domain experts and modelled as business processes having their own business logic in order to adopt faster to market changes and business demands.

The cooperation with business partners demands the utilization of services across organizational boundaries. In fact, the involvement of independent trust domains constitutes the key aspect regarding security in service-oriented architectures, since the seamless and straightforward integration of cross-organisational services conflicts with the need to secure and control access. A broad range of security protocols and mechanisms has been specified to address this insufficiency in the scope of SOA, emphasising the fact security is considered on an technological level. However, protecting single endpoints on a technological level not sufficient to enforce and guarantee security in an SOA. A comprehensive understanding and evaluation of threats and associated risks is needed, especially with regard to regulations such as Basel II.

Business process modelling offers an appropriate layer to describe security requirements and to evaluate risks, since it identifies participants and assets in an information processing system and its relations on an abstract level. I described an approach to integrate security goals and constraints in business process modelling in [30]. Similar approaches exist, for example defined by Rodríguez [18], that provide extensions for BPMN to express security requirements as well. However, current approaches enhance BPMN to model basic security intentions, but are not feasible to enable a comprehensive verification of security properties. For instance, it is not sufficient to model a lock at the process layer as an intention to ensure confidentiality for a single connection, since it would not consider the flow of information in the process. Information may be passed through multiple intermediaries until it is stored or processed by a service and, therefore, multiple parameters must be assessed such as the information's value, the trustworthiness of participants and the dependencies between modelled entities. I believe that the modelling should not just integrate abstract security intentions – it should include additional security meta information to rate entities facilitating a comprehensive view on security to model, evaluate and verify security requirements.

Based on modelled and verified security intentions, I want to facilitate the generation of concrete security configurations for process-aware information systems. However, in the face of current implementations and the variety of security specifications regarding SOA realized with Web Services, these solutions come along with incompatibilities and multiple dependencies. Current model-driven approaches enhancing business process models with security intentions do not describe the consistent selection of appropriate security concepts.

Therefore, I foster a model-driven approach in which security intentions and ratings are annotated in business processes that can be translated to consistent security polices. My research work has been focused on

- a security extension for business processes that provides properties and annotations to integrate the revealed organisational security concepts.
- an model-driven based approach to generate policies. I introduce a security model to gather security information modelled at the process layer and describe the usage of security pattern to resolve appropriate security protocols.

This report is structured as follows. Section 2 provides an overview about enterprise security concepts and based on this I introduce my approach to enhance BPMN with security properties and annotations in the next Section. Section 4 introduces my domain-independent security model to gather security information from the modelling layer and introduces security patterns to resolve appropriate protocols and security mechanisms to enforce modelled security intentions. As a proof of concept, I present a mapping to the Axis2 security configuration in Section 5. In Section 6 I discuss and conclude my approach and outline some suggestions of future work, such as the integration of cross-organisational services.

2 Enterprise Security

The central aspect of security engineering is the management of risks that result from potential threats referring to business assets (e.g. information, tasks, etc.). To evaluate the impact of threats, assets must be evaluated to determine its overall importance in an enterprise. Based on the evaluation of enterprises assets, appropriate measures can be identified that reduce the threats and minimize the risks. Threats and countermeasures can be classified according to the related security goal [16]. Therefore, threats can be related to the usage of identity information and enforcement of associated rights (*authentication, authorisation, trust*), transferred, processed or stored information (*data confidentiality and data integrity*), functioning of a service (*system integrity and availability*), and the repudiation of malicious behaviour (*auditing*).

2.1 Authentication, Authorisation, Trust

Authentication ensures the credibility of identity information by verifying that a claimed identity is authentic, while *authorisation* is the process of granting rights to participants to perform a task, for instance to access a service. These goals presume a secure management and trustworthy provision of identity information. With regard to a Service-oriented Architecture, the underlying trust relationships must be considered, since the usage and provision of services might not be limited to one trust domain. Services from different organisations might be integrated or services might be exposed to different business partners having their own security requirements and procedures. To evaluate the trustworthiness of the authentication and authorisation process, it is important to analyse the underlying trust relationships that can be classified in *Organisational Trust* and *Identity Trust* as described by Ivonne Thomas [27].

Organisational Trust – Organisational trust rates the trust relationship between organisations and, on a technical level, represents the trust between identity provider. The level of Organisational Trust is based on various parameters such as the reputation of business or the enterprise's minimal requirements for user registration and authentication.

Identity Trust – The identity of a subject is important to hold us service user liable in case anything bad happens. Therefore, the provision trust-related identity information is required to build up trust in the identity of the user and its behaviour. This trust, which is called *identity trust*, is the concept behind all access control models.

A broad range of access control models have been developed in the last decades, defining access control constraints based on particular security information such as the user's role (RBAC [22]) or the user's team affiliation (TBAC [28]). Since all these pieces of information can be considered as attributes of involved objects, the attribute-based access control model (ABAC) can be seen as the most comprehensive access control model, as described in [5].

2.2 Data Confidentiality and Data Integrity

Confidentiality provides protection against the unauthorised notice of transferred, processed, or stored information, while *Data Integrity* ensures the properness (intactness, correctness, and completeness) of information. Transferred, processed, or stored data must not be modified with proper rights and - in economic terms - modifications must correspond to business values and expectations. Since the enforcement of these security goals might involve the application of complex security mechanisms, there is always the trade-off between the desired level of security and performance. The required security level depend on the information's value that influence the implementation (type of protocols, algorithms, etc.) of these goals. Moreover, these goals can refer to transferred, processed, or stored data. Data should be secured if it is transferred over unsecured connections or if it is processed or stored by untrustworthy participants. Compliance requirements might be an additional reason that require the application of data confidentiality, e.g. incoming orders must be signed and stored for compliance reasons in a log-file.

2.3 System Integrity and Availability

System Integrity ensures the correct functioning of a system to guarantee that a system acts in an expected and proper way at each point in time. *Availability* that ensures that data, resources and services, which are needed for the proper functioning of a system, are available at each point in time regarding the requested quality of service. Availability depend on system integrity since availability can not be guaranteed if system integrity is compromised. However, the quality of service requirements for availability might require additional technical solutions, e.g. mechanisms for load balancing, to ensure availability.

To ensure the correct functioning of a system, services must be protected from various threats and attacks that can be classified in two categories [1, 12], as follows:

1. *malicious content based attacks* – The purpose of this class of attacks (e.g. data with viruses, injection attacks, recursive/oversized XML Documents) is to exploit the service by sending data or messages with malicious content. Countermeasures can be used applying filters for content inspection to transferred data.
2. *protocol misuse* – Attacks based on the misuse of protocols (e.g. WSDL Scanning, WSDL parameter tampering/ error interface probing, replay attacks) intend to gain information or to bypass authorisations. Intrusion detection systems can help to recognize this class of attacks.

Again, there is a trade-off between the desired security level and performance. The necessity to scan transferred data in a system depend on the trustworthiness of the involved participants in a communication process and the importance of involved tasks and services. In general, scanning mechanisms are necessary if borders of security domains are passed, or if the sender of a message is less trustworthy. This enables the single verification of data for all group of tasks.

2.4 Auditing

Traceability and Auditing provide verifiability regarding all performed actions in an information processing system. This can be related to simple logging mechanisms, but also to monitoring as real-time auditing. Auditing events can refer to all layers in the underlying architecture and might refer to the communication infrastructure, the messaging layer, the enforcement of security goals or business rules.

3 SecureBPMN – Modelling Security in BPMN

The previous Section provided an overview about organisational security concepts that address various security threats in SOA. I outlined that the required level of security in an business-aware information system depend on metrics that determine the value of enterprise assets and assign trust level to each participant. These values determine the risk that are associated with each asset. Appropriate security measures can be applied for each asset to reduce risks. An process-aware information system can be considered as secure, if the asset's risk comply with its business value. To facilitate a verification of security requirements and risks at the business process layer, I defined SecureBPMN as an enhancement to BPMN. In this Section I will describe my approach to enhance BPMN to describe the revealed aspects.

3.1 Evaluating Assets

Enterprise assets in the scope of BPMN are represented by performed tasks or data that is passed between tasks and participants. To rate these assets I added a property called *AssetRating* to task and data that is assigned a rating as listed in Table 1. This classification is based on an generic approach for asset valuation described by Markus

Table 1: overall asset value scale

Rating	Description
Extreme	Endangering human life or threatening enterprise existence
Very High	Servere financial or security consequences
High	Impact on customer services and reputation
Medium	Affect the enterprises mission
Low	Minor financial damage and little business impact
Negligible	No security relevance

Schumacher et al. [24]. They determine the asset's overall rating based on three partial values: The *security value* represents the importance the organisation places on guaranteeing the assets value, the *financial value* quantifies the monetary value for the enterprise and the *business value* determines the impact on the business. Each partial value is based on a rating with six categories that are defined with a clear semantical meaning.

3.2 Modelling Trust

In the previous Section I identified organisational trust and identity trust as the basic concepts to enable a trustworthy interaction of participants. Organisational trust constitute a prerequisite for interactions by predefining a trust relationship between two participants. To express organisational trust in BPMN, I defined an artifact called *Organisational Trust* that can be connected to two or more pools to express this trust relationship. The organisational trust level quantifying the relationship between two participants is determined by parameters that are added as properties to the artifact. Although various parameters can affect the orgnaisational trust value, I decided to consider three basic parameters for modelling:

The *InitialTrust* parameter represents an initial trust value based on how a trust relationship between organisations has been established. Table 1 list four categories that is based on an overview about business security patterns provided by IBM [9]. Two participants can belong to the same organisation (*operational*) with unlimited trust, the trust relationship can be established by contract (e.g. identity federation), an organisation might be trusted without contract (e.g. an other acquainted organisation serving as OpenId-Provider).

The *MinAuthenticationTrust* and the *MinRegistrationTrust* value represent the trust that at least can be put in the process of user registration and authentication. For instance, if an organisation acts as an OpenId identity provider that can be used by users with an valid email-address and that authenticates users based on an user-definable password, then the corresponding trust values will be quite low. An approach to measure these values has been described in the previous section.

In contrast to organisational trust, identity trust is established dynamically during service access. The required level of identity trust needed to access a service is determined implicitly by the requirements for access control. I will describe the integration of authorisation requirements and constraints alter on. Factors for identity trust that

Table 2: initial trust rating

Type	Trust	Known by
Operational	absolute	same organisation
B2B	high	contract
B2C	low	reputation
WebPresence	none	unknown

are usually not described in access control policies – such as the required level of authentication trust – can be derived from the asset's value.

3.3 Expressing Security Intentions

The security intentions introduced in the previous section, such as confidentiality, identify measures that should be applied to reduce certain risks. These risk are not solely related to a single entity such as an connection, they usually result from complex dependencies and interactions. Therefore, my approach is to specify security goals and associated requirements within the broader scope of a group of activities or a pools instead of assigning intentions to single modelling elements. I added the new artifact *Security Group* to BPMN containing the properties *Confidentiality*, *DataIntegrity* and *SystemIntegrity* representing security intentions. The value of these properties determines a security level. An security intention has to be enforced for an asset, if the security level of the containing group exceed an asset's value. A security group differs from a Group in the type of entities that can be combined. While a Group is limited to a set of activities, a Security Group can additionally contain pools.

SecureBPMN is designed to capture security intentions and ratings providing a foundation to determine which measures have to be applied. The implication of these requirements – e.g. at which location in the process and to which degree will data be filtered to comply to the system integrity requirement – might be visualized by a special security view. However, this is not in the scope of my work yet.

Table 3: SecureBPMN Elements

Element	**new Property**
Task	Asset Rating
Data Object	Asset Rating
Organisational Trust	InitialTrust, MinAuthenticationTrust, MinRegistrationTrust
Security Group	Confidentiality, DataIntegrity, SystemIntegrity
Event	Auditing

3.4 Expressing Auditing Requirements

In Section 2 I described four architectural layers relevant for auditing. I consider requirements for auditing regarding messaging and security, while auditing concerning business rules might be modelled in the process itself. To express requirements for auditing messaging events, I propose to add the property *Auditing* to BPMN events that provide an expression to specify monitoring constraints. Regarding the auditing of security events these requirements are implicitly bound to the usage of security intentions.

4 Translating Security /Security Pattern

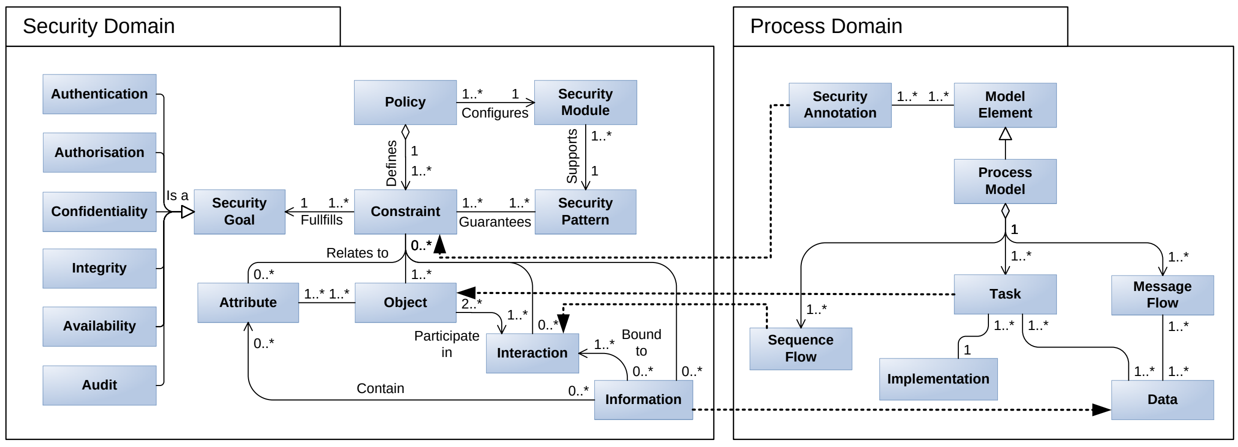


Figure 1: Security Policy Model

As discussed in the previous section, SecureBPMN should enable business process experts and security experts to express security intention on an abstract model. It should facilitate a verification of high level security requirements and trust relationships at the business process layer to detect risks. In addition, this model should enable a generation of security configurations according to the model-driven paradigm. Therefore, I defined an platform-independent security model abstracting from concrete implementation platforms and their provided security features. This model is used to collect and refine security requirements from the business process model and serves as a common model to generate platform specific security configurations in different languages.

4.1 Domain-independent Security Model

To express, compare and verify security requirements in a technically and policy language independent way, an abstract security layer has been introduced in previous work [31]. The conception of this layer is close to the OASIS reference model for SOA [13] and enables a straight mapping to the business process layer. The security layer is designed to reveal all security aspects in an SOA landscape and the relationship among affected entities. Therefore, my model describes basic security goals and outlines the relationship to specific security attributes and mechanisms. The relations among security goals and affected entities are described by *Constraints* that are composed in a security *Policy* as indicated by Figure 1.

These policy constraints always refer to a set of objects (e.g. a service or a participant), which is the basic entity in my security model. I define an object as an entity that is capable of participating in an *Interaction* with other objects. This interaction may involve a set of *Information* that is exchanged. Each object is related to a set of attributes describing its meta information that are derived from the modelling layer, such as the service name, adress, etc.

Furthermore, security constraints are related to a security level that describes the strength of this requirement. This value is assigned during the transformation to the domain-independent security model based on asset evaluation and the context of an security intention at the business layer.

As shown in Figure 1, policies are interpreted and enforced by a *Security Module* that support specific *Security Patterns* to guarantee the defined constraints.

4.2 Security Pattern

The aforementioned security model represents a set of basic information based on the modelled intentions at the process layer. However, these information are not sufficient to generate concrete security configurations since further knowledge is still needed. Expertise knowledge is required to determine an appropriate strategy to secure a service orchestration, since multiple solutions might exists to satisfy a security goal. For example, confidentiality can be implemented by securing a channel using SSL or by securing parts of transferred messages using WS-Security. Which strategy is suitable to satisfy a modelled security intention best, depends on information mapped to my security model (e.g. secure channel is applicable when information in transit must be secured and information is not passed through untrustworthy intermediaries).

To describe these strategies and their preconditions in a standardized way, I foster the usage of security patterns. Security patterns have been introduced by Yoder and Barcalow [32] in 1997 and are based on the idea of design pattern as described by Christopher Alexander *'A pattern describes a problem which occurs over and over again in my environment, and then describes the core of the solution to that pattern'* [2]. In general, security pattern are defined in an informal way, usually in the natural language, to enable programmer and system designer to adapt the solution described by the pattern to their own specific problem in a particular implementation context.

However, my intention is to enable an automated selection of appropriate pattern to gather information for the generation of security configurations. Therefore, a formal pattern specification is needed as described by Markus Schumacher [23] to enable a reasoning on a set of security patterns. Figure 2 illustrates the structure of a pattern and its relationship to my security model. A pattern is composed of

- **problem** – the problems that are addressed in the context of security are threats. As described in Section 2 a group of threats can be related to a security goal.

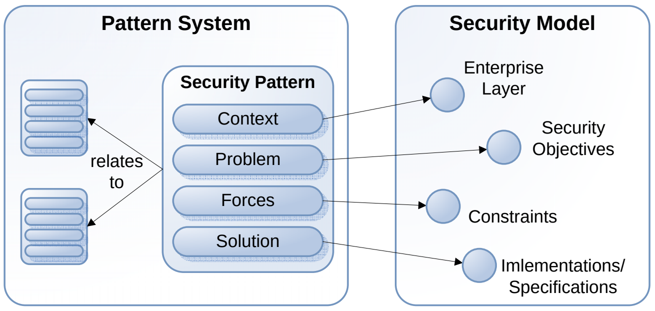


Figure 2: Security Pattern

Since this level of abstraction is adequate for my model-driven approach, the problem refers a security goal.

- **context** – the context describes the hierarchical layer or life-cycle phase a pattern is referring to. Buschmann et al. provided a classification of patterns and identified three main categories: *Architectural Pattern*, *Design Pattern* and *Idioms* [6]. Since I am referring to design patterns solely in this report, this field is less important.

- **forces** – Forces describe the conditions under which a pattern can be applied. These preconditions have to be matched with information provided by my security model to determine appropriate patterns.

- **solution** – A Pattern describes a strategy to solve a problem that is adapted by concrete security mechanisms or security protocols. In my approach, a pattern solution identifies these protocols.

Various pattern might exist that specify different solutions for the same security goal. Moreover, dependencies between patterns might exist. As described by Zimmer [34] there are three basic dependencies that might occur between security pattern: *Usage*, *Refinement* or *Conflict*.

Based on previous work in the field of security pattern [33], I defined a pattern system that describes pattern for each security goal and their relationship. Figure 3 shows an example for the security goals integrity and confidentiality. Two patterns are illustrated: *SecurePipe* to secure data exchanged over an insecure channel, and *MessageConfidentiality* and *MessageIntegrity* to secure data exchanged with messaging. Each pattern refer to particular security goals and identifies appropriate security protocols.

5 Modell-driven Generation of Security Policies

In the previous sections I presented a model to aggregate security requirements in a domain-independent security model. Security pattern has been introduced as an possibility to resolve an appropriate security protocols and mechanisms. Therefore, it

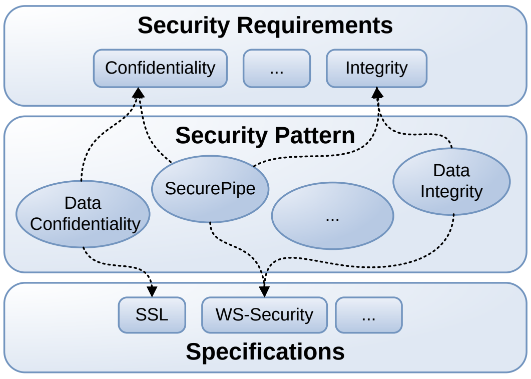


Figure 3: Confidentiality Pattern

is necessary to define a mapping for each defined security pattern to a specific security policy. In this section I will describe the generation of security policy based on the resolved information and present an example to generate polices for a service realized with the Apache Axis 2 Web Service Stack [15].

The Apache Rampart Module [7] is used to apply security to the ingoing and outgoing messages of web service calls. Rampart is configured by the Rampart configuration parameters [7] or the WS-Policy [3] language. The native Rampart configuration uses a flat list of constraints stated in XML with implementation specific information, while WS-Policy provides a grammar to express and group policy assertions describing a broad range of requirements on an more abstract level. Since WS-Policy is not specific to a problem domain, security assertions are defined in the WS-SecurityPolicy specification providing an implementation independent approach to express constraints for WS-Security, WS-Trust and WS-Secure Conversation [20].

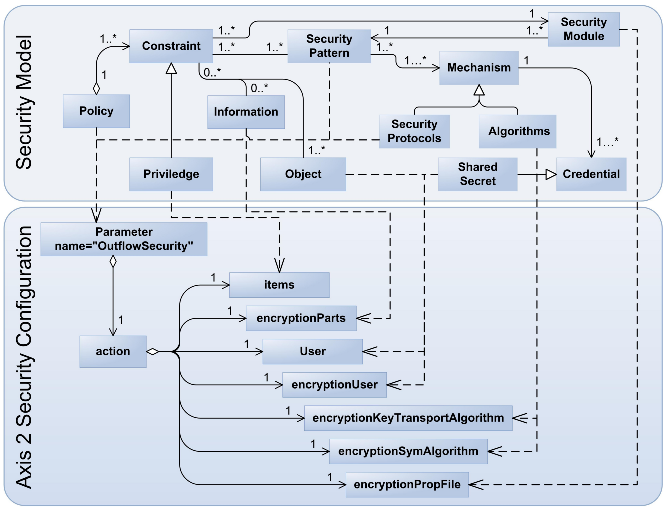


Figure 4: Mapping to the Apache Rampart Security Configuration

In the scope of the native Rampart configuration, a policy is represented by the element *parameter* with the attribute *name='OutflowSecuity'* | *'InflowSecuity'*. The attribute controls whether the security settings are applied to ingoing or outgoing messages. The policy element *parameter* contains the element *action* that encapsulates several elements to configure Rampart. Figure 4 shows the mapping to the Rampart configuration for outgoing messages regarding the security goal confidentiality. Since Rampart operates on the exchanged Web Service messages, this transformation as well as the generated policy are associated with the security pattern *message confidentiality*.

```
<! -- Body Encryption for calling service DebitOrder-->
<parameter name="InflowSecurity">
   <action>
      <items>Encrypt</items>
      <user>Customer</user>
      <encryptionUser>OnlineStore.DebitOrder</encryptionUser>
      <encryptionPropFile>OnlineStore.DebitOrder.properties</encryptionPropFile>
      <encryptionParts>{}{}Body</encryptionParts>
      <encryptionSymAlgorithm>
         http://www.w3.org/2001/04/xmlenc#tripledes-cbc
      </encryptionSymAlgorithm>
      <encryptionKeyTransportAlgorithm>
         http://www.w3.org/2001/04/xmlenc#rsa-1_5
      </encryptionKeyTransportAlgorithm>
   </action>
</parameter>
```

Listing 1: Rampart Inflow Encryption

6 Related Work

The domain of model-driven security in the context of business processes is an emerging research area. The need to support the application scenario and hypothesise the related security policies for the affected services on an abstract level is discussed in [26]. I extended this suggestion by defining security configuration requirements in the context of application scenarios captured on the business process layer.

Recent work done by Nagaratnam *et al.* [14] discusses an approach to overcome this shortage by expressing security requirements in the context of business processes and how to monitor and manage them on the different enterprise architecture levels. This intention, while similar to my concepts regarding the benefits of a modelling approach, does not provide a detailed analysis of security goals, their conceptual models, and their relationship to the business process related entities.

This has been addressed by Rodrguez *et al.* [17, 19] by defining a meta-model that links security requirement stereotypes to activity elements of a business process and proposed graphical annotation elements to visually enrich the process model with related security requirements. Although they support several security intentions, they do not provide a comprehensive security model based on the evaluation of assets considering authentication and trust.

Our security model could be complemented by modelling concepts for compliance rules for business processes [21] as described by Sadiq *et al.*. They propose model annotations with control tags that are mappable to the Formal Contract Language (FCL) focusing on the intended behaviour of the process model in the context of organisational compliance regulations. Their control tags cover order of event, data, and authorisation aspects, but they do not address how to actually derive enforceable compliance rules at runtime.

Enforcing authorisation constraint in workflows is addressed in [11]. SecureFlow implements a Workflow Authorisation Model (WAM). Authorisations can be defined and enforced at runtime for users, roles, and workflow tasks. In contrast to my general security modelling approach they focus on authorisation constraints in centralised workflow management system, without considering other security requirements, such as confidentiality or integrity that are important in a service-oriented environment as well. These authorisation concepts are refined in [8] by a semantic policy-based security framework for business processes identifying two levels of security for business processes. On the task or activity level, security concerns, such as non-repudiation, confidentiality, and data integrity are considered. On the process level, general compliance rules, such as required by Sarbanes-Oxley are defined, but a model connecting security and process aspects is not given.

Model-driven security and the automated generation of security enhanced software artefacts and security configurations has been a topic of interest in recent years. For instance SecureUML [4] is a model-driven security approach for process-oriented systems focusing on access control. Similar to SecureUML, Jrjens presented the UMLSec extension for UML [10] in order to express security relevant information within a system specification diagram. One focus of UMLSec lies on the modelling of communication-based security goals, such as confidentiality, for software artefacts, while SecureUML describes desired state transitions and access control configurations for server-based applications, both do not leap for establishing the link between business processes and model-driven generation of related security requirements.

7 Conclusion

Business process modelling represents a cornerstone of process-aware information systems. Service-oriented Architectures supports the execution of processes by facilitating the mapping between task and services. While business processes provides an abstract view on organisational aspects and related requirements, security requirements are usually specified on a technological level rather at the business level as stated in [26].

Process modelling notations provide a suitable abstract perspective to specific security goals on a more accessible level, as I have shown in [31]. However, previous approaches are limited regarding the description and evaluation of security requirements and do not describe a transformation of these requirements to security configurations. To enable an overall evaluation of security requirements based on the value of enterprise assets and the trustworthiness of participants, I provided a compilation of organisational security aspects and outlined their relationships. Based on this overview, I introduced SecureBPMN as an modelling extension for BPMN to enable business process designer and security experts to define and discuss security requirements on an abstract level.

I presented a model-driven approach addressing the difficulty to generate security configurations for a process-aware information system. The foundation constitutes my generic security model that specifies security goals, policies, and constraints based on a set of basic entities, such as *Objects*, *Attributes*, and *Interactions*. The strength of my model lies in its general description of security goals, and the abstraction from technical details. Security intentions and related requirements defined at the process layer can be mapped to this model. To resolve concrete security protocols and mechanisms, a security pattern system has been described that resolves appropriate security protocols with regard to specific preconditions.

The gathered information can be mapped to an arbitrary application or technical specification. As an example I introduced a mapping to the configuration of the Axis2 Rampart module. As a result, these security configurations would be consistent with the affected business processes and result in a decreased error-proneness.

7.1 Future Work

I stated that my proposed security extension for BPMN is a promising approach to describe, verify and translate security requirements at an abstract level. While I described a model-driven approach to translate security intentions to concrete security configuration, I outlined the possibility to perform the verification. In the next step I will address the process of verification in detail to prove that the verification will guarantee the consistency of modelled security goals. Some concepts have been recently discussed [25] and I will investigate their applicability to my concepts.

Moreover, I will have to continue implementing the described model-driven approach based on the domain-independent security model. A security ontology has to be designed to gather the security information from the modelling layer. Aditionally, a security pattern registry is needed to resolve further information. To visualize revealed risk and verification results, I will integrate a security view into the ORYX business process modelling tool. This view should visualise risk information, trust relationships and additional security information.

Another important aspect is the consumption and provision of services and service compositions across trust domains [Menz08]. To enable a model-driven approach regarding cross-organisational service compositions, it must be considered that federation partners state their own security requirements that must be considered as well as compatibility issues. In addition, it is important to reveal dependencies and contradictions between requirements from different service providers that would prevent a secure or compatible service provisioning. This information can also be used to provide feedback at the modelling layer.

References

[1] Eege project. grid and web service security vulnerabilties and threads analysis and model. (See: https://edms.cern.ch/documents/632020/), 2005.

[2] Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobsen, Ingrid Fiksdahl-King, and Shlomo Angel. *A Pattern Lanuage: Towns - Buildings - Construction*. Oxford University Press, 1977.

[3] Siddharth Bajaj, Don Box, and et. al. Web services policy 1.2 - framework (ws-policy). Public Draft Specification, April 2005.

[4] David Basin, Juergen Doser, and Torsten Lodderstedt. Model Driven Security for Process-Oriented Systems. In *SACMAT '03: Proceedings of the 8th ACM symposium on Access control models and technologies*, pages 100–109, 2003.

[5] Hai bo Shen and Fan Hong. Modelling security goals in business processes. In *Modellierung 2008*, LNI. GI, 2008.

[6] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern-Oriented Software Architecture: A System of Pattern*. John Wiley & Sons, Ltd, 1996.

[7] Saminda Abeyruwan et. al. Apache rampart : Ws-security module for axis2, 2008.

[8] Dong Huang. Semantic policy-based security framework for business processes. In *Proc. of the Semantic Web and Policy Workshop*, 2005.

[9] IBM. Introduction to business security pattern. 2004.

[10] Jan Juerjens. UMLsec: Extending UML for Secure Systems Development. In *UML '02: Proceedings of the 5th International Conference on The Unified Modeling Language*, pages 412–425, 2002.

[11] Wei kuang Huang and Vijayalakshmi Atluri. Secureflow: A secure web-enabled workflow management system. In *ACM Workshop on Role-Based Access Control*, pages 83–94, 1999.

[12] P. Lindstrom. Attacking and defending web services, a spire research report. (See: http://forumsystems.com/papers/Attacking_and_ Defending_WS.pdf), 2004.

[13] Matthew MacKenzie, Ken Laskey, Francis McCabe, Peter Brown, and Rebekah Metz. Reference model for service oriented architecture 1.0. OASIS Committee Specification, February 2006.

[14] N. Nagaratnam, A. Nadalin, M. Hondo, M. McIntosh, and P. Austel. Business-driven application security: From Modeling to Managing Secure Applications. *IBM Systems Journal, Vol 44, No 4*, 2005.

[15] Srinath Perera, Chathura Herath, Jaliya Ekanayake, Eran Chinthaka, Ajith Ranabahu, Deepal Jayasinghe, Sanjiva Weerawarana, and Glen Daniels. Axis2, middleware for next generation web services. In *ICWS*, pages 833–840. IEEE Computer Society, 2006.

[16] Charles P. Pfleeger and Shari Lawrence Pfleeger. *Security in Computing*. Prentice Hall Professional Technical Reference, 2002.

[17] Alfonso Rodríguez, Eduardo Fernández-Medina, and Mario Piattini. Towards a uml 2.0 extension for the modeling of security requirements in business processes. In *TrustBus*, pages 51–61, 2006.

[18] Alfonso Rodríguez, Eduardo Fernández-Medina, and Mario Piattini. A bpmn extension for the modeling of security requirements in business processes. *IEICE Transactions*, 90-D(4):745–752, 2007.

[19] Alfonso Rodríguez, Eduardo Fernández-Medina, and Mario Piattini. Towards cim to pim transformation: From secure business processes defined in bpmn to use-cases. In *BPM*, pages 408–415, 2007.

[20] Jothy Rosenberg and David Remy. *Securing Web Services with WS-Security: Demystifying WS-Security, WS-Policy, SAML, XML Signature, and XML Encryption*. Pearson Higher Education, 2004.

[21] Shazia Wasim Sadiq, Guido Governatori, and Kioumars Namiri. Modeling control objectives for business process compliance. In *BPM*, pages 149–164, 2007.

[22] Ravi S. Sandhu and Edward J. Coyne. Role-based access control models. *IEEE Computer*, 29:38–47, 1996.

[23] Markus Schumacher. *Security Engineering with Patterns - Origins,Theoretical Model, and New Applications*. Number ISBN 3-540-40731-6. Springer, Berlin, 2003.

[24] Markus Schumacher, Eduardo Fernandez-Buglioni, Duane Hybertson, Frank Buschmann, and Peter Sommerlad. *Security Patterns - Integrating Security and System Engineering*. John Wiley & Sons, Ltd, 2006.

[25] Kaijun Tan, Jason Crampton, and Carl A. Gunter. The consistency of task-based authorization constraints in workflow systems. In *CSFW*, pages 155–, 2004.

[26] Michiaki Tatsubori, Takeshi Imamura, and Yuhichi Nakamura. Best-practice patterns and tool support for configuring secure web services messaging. In *ICWS*, pages 244–251, 2004.

[27] Ivonne Thomas. Identity management for cross-organizational soa. Technical report, Hasso-Plattner-Institute, Fall 2008.

[28] Roshan K. Thomas and Ravi S. Sandhu. Task-based authorization controls (tbac): A family of models for active and enterprise-oriented autorization management. In *DBSec*, pages 166–181, 1997.

[29] Mathias Weske. *Business Process Management*. Springer, 2007.

[30] Christian Wolter, Michael Menzel, and Christoph Meinel. Modelling security goals in business processes. In *Proc. GI Modellierung 2008*, number ISBN 978-3-88579-221-5. GI LNI, Berlin, Germany, 1008.

[31] Christian Wolter, Michael Menzel, and Christoph Meinel. Modelling security goals in business processes. In *Proc. GI Modellierung 2008*, number ISBN 978-3-88579-221-5. GI LNI, Berlin, Germany, 1008.

[32] Joseph Yoder and Jeffrey Barcalow. Architectural patterns for enabling application security. In *PLoP*, 1997.

[33] Nobukazu Yoshioka, Hironori Washizaki, and Katsuhisa Maruyama. A survey on security patterns. *Progress in Informatics*, 5:35–47, 2008.

[34] Walter Zimmer. Relationships between design patterns. pages 345–364, 1995.

A Flexible Live Inspection Framework

Alexander Schmidt
alexander.schmidt@hpi.uni-potsdam.de

In this paper I will give an update of my research efforts during the past half year. I will first present a brief review of the KStruct project, a flexible monitoring infrastructure focused on operating system kernel data structures. The KStruct system has been used for teaching operating system courses here at HPI. It turned out that the infrastructure indeed is flexible enough to be used in other contexts as well, which will be the second part of this paper. I spent most of the time with an internship at Microsoft, which although not directly related to my research, was beneficial for presenting the KStruct approach within the company. My experiences and work at Microsoft will be covered in the third part of the report.

1 The KStruct Framework

Service-oriented architectures recently received quite significant attention in both the industry and research. Being building blocks of higher level business processes, failing services may have significant impact on its providers [12]. Thus, services have to be highly available and should provide high performance to meet the business needs. Service-oriented architectures also offer a new kind of business model with open standards allowing for seamlessly exchanging services of different service providers. While several solutions exist that address the issue of how to deal with service service failures at the consumer site [3, 9, 13], they do not address solutions on how to support the service providers.

For example, consider a data center consisting of thousands of nodes and hosting hundreds of individual services. To better utilize single nodes within the data center, nowadays the physical machine, the host, is shared among several virtual machines. Each individual service is hosted exclusively in a virtual machine. Which virtual machine has (exclusive) access to what resource is usually controlled by a meta operating system called the hypervisor, or virtual machine monitor. However, each virtual machine and its hosted service may eventually encounter its peak load. If several virtual machines of the same node encounter the peak load at the same time, it may be beneficial to migrate a virtual machine to another node with less load. This node may be in the same data center or in another data center. Otherwise, as resources get short, the service level agreement of some service on that node is in danger to be violated. Also, in case of a crash failure [5] of a service or its virtual machine, the service may be redeployed on another node within the cluster. Monitoring and inspecting (arbitrary) indicators of these events is thus crucial to enable the described scenario.

The KStruct framework is an approach to selectively inspect certain information within the OS kernel. The inspection may be enabled during the runtime of the OS and does not require rebooting the system, which is essential for productive computing environments. Although different technologies exist to gather performance related data [4, 8, 22] exist, KStruct focuses in particular on data structures of the OS kernel. However, the approach is flexible enough to extend the inspection to data structures that are related to applications running on top of that operating system. The modular design of the KStruct framework can support different kinds of front-ends, such as WMI [22] or HP's System Insight Manager (SIM) [2].

To inspect data structures the following problems have to be solved: (1) The kernel address space has to be made accessible. (2) Data structures need to be described in a way suitable for automation, *i.e.*, without any human intervention, neither by adapting code nor by halting the system or relaunching the application. The latter is especially in production environments where most of the services hosted are inherently long running. (3) Data structures must be addressed in a way to easily locate them on the heap while allowing runtime verification. (4) the inspection shall prevent from any data races regarding the data structure of interest to guarantee liveness properties of the inspected system. That is, a failure, if any, perceived by the user must not be caused by the inspection framework.

The KStruct framework solves (1) by using a driver that inherently has access to the kernel address space. In order to achieve flexibility and extensibility, we introduce the *KStruct Access* domain-specific language (DSL). To ensure (4), KStruct Access provides means to express *locking hierarchies* among data structures. The lock granularity depends on the OS and may range from very fine grained locks for each data structure instance to coarse-grained global kernel locks. Expressing locking hierarchies is a non-trivial task and an approach on how to derive dependencies automatically is part of ongoing research. For solving (3), we introduce *object paths*.

Based on this approach, we implemented a prototype. At the HPI, this prototype has been used for two generations of undergraduate students, including hands-on labs and assignments. The setting for these labs is however different. For the OS classes we provide a pre-configured and fixed KStruct environment based on the Windows Research Kernel (WRK) [11]. Results have been presented at the USENIX Annual Technical Conference [19].

1.1 KStruct Architecture

The KStruct consists of two major components: the *KStruct Framework* and the *KStruct Generator*. Both components are shown in Fig. 1. The KStruct Generator is our tool for generating the source code for both the device driver and the rendering library. To configure the amount of data structures supported by the KStruct Framework, it is necessary to supply a KStruct specification file. Usually, this file is created by a domain expert, *e.g.*, an engineer familiar with kernel details or an operating systems teacher. To ensure type safe access to kernel data structures, the KStruct Generator relies on symbol information files (PDB files) generated by the Microsoft C compiler `cl`. Our generator is capable of coalescing symbol information of several OS kernels starting with Windows XP. However, we concentrate here only on symbol information of the

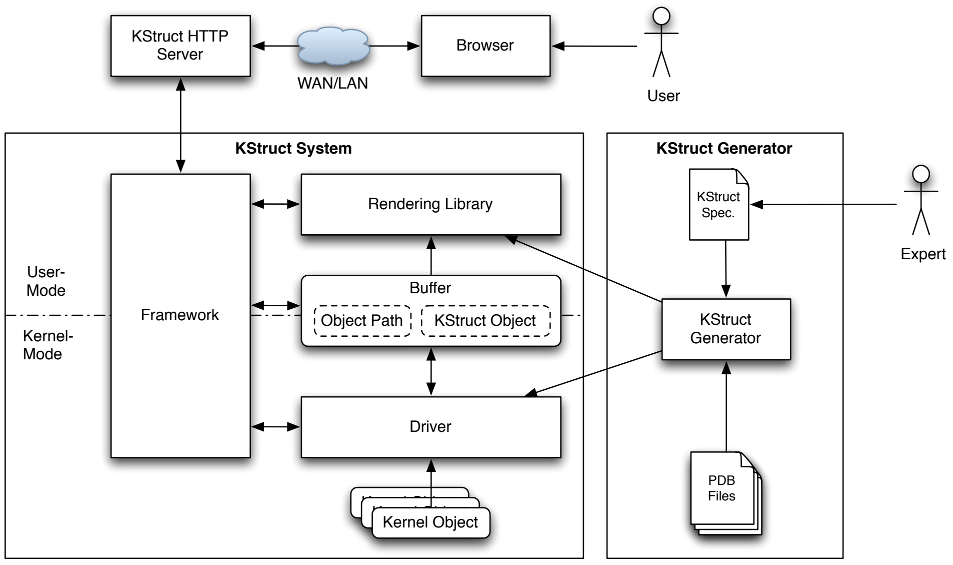


Figure 1: Architecture of the KStruct runtime system.

Windows Research Kernel.

While the KStruct Generator is used for generating the device driver and and the rendering engine, the KStruct Framework is necessary for running those components on a target system. The Framework loads both the rendering library and the device driver and provides an interface for the HTTP server in order to process queries.

A major goal of the KStruct system is to guarantee type safety of displayed objects, *i.e.*, it is not possible to accidentally cast a data structure to another one. Debuggers lack this facility: a user may cast a kernel address in what ever type he/she likes. Also, we wanted to use an addressing scheme that is equally intuitive and comprehensive, *i.e.*, the addressing scheme should reflect the dependencies and the complexity of the kernel's objects.

To solve the problem, we introduce the concept of an *object path*. An object path uniquely determines an object in the heap of the kernel. However, there may be several paths leading to the same object. The following is an example of an object path:

```
/<root>/<member1>/<member2>/.../<membern>
```

The first object in this path, `root`, determines a global or static data structure of the kernel. For convenience, we introduced aliases to these root objects. For example, `Processes` is an alias for `PsActiveProcessHead`. Due to space limitations we will not further discuss aliases here. Each root object may be followed by zero or more member names, delimited by slashes ("/"), where each member name identifies a field within the preceding data structure. If the preceding data structure is actually a list or an array, the member name forms an identifier that uniquely identifies a data structure within the list or array, respectively. The following example identifies the `UniqueProcessId` field of an instance of the `EPROCESS` data structure: `/Processes/81234567/UniqueProcessId`. When a query is issued, the framework will issue the object path to the driver, which

```
struct EPROCESS{
  KPROCESS Pcb;
  [lock] EX_PUSH_LOCK ProcessLock;
  LARGE_INTEGER CreateTime;
  HANDLE UniqueProcessId;
  [lhead(ETHREAD::ThreadListEntry)]
    LIST_ENTRY ThreadListHead;
  [mlock(VadRoot)] KGUARDED_MUTEX
    AddressCreationLock;
  MM_AVL_TABLE VadRoot;
};
```

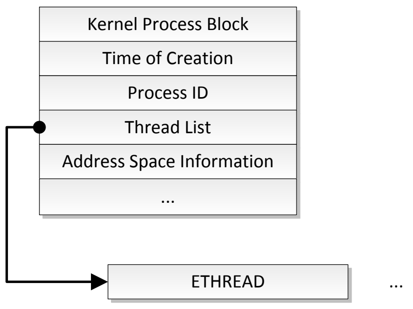


Figure 2: The KStruct definition for the `EPROCESS` data structure [10, 15]. On the right is a block diagram of the specified data structure. Due to space limitations, some fields are omitted here.

will check whether the given object path is valid within the current system. If the object path is valid, a copy of that object is sent to the rendering library, which is responsible for rendering the data structure in a human readable format. Within our current KStruct configuration, the output format is HTML as we provide access over the Internet. However, other formats are reasonable, for instance, a Windows Management Instrumentation (WMI) [22] provider.

1.2 KStruct Access

The architecture of the KStruct framework reflects the different kinds of users and phases necessary to run the KStruct system: the specification phase and the experiment phase. The experiment phase is designed for novice users, which experimentally want to extend their knowledge of an operating system. The specification phase on the other side is designed for domain experts that have thorough knowledge on the matter. Domain experts may include kernel architects and developers or operating system teachers that want to use KStruct for their classes.

In order to specify data structures of interest, we designed KStruct Access, our domain-specific language (DSL). The goal of this language is to allow specification of kernel data structures in order to automatically generate both a kernel-mode device driver and of a user-mode helper library out of this specification. As most data structures in the WRK are defined using the C language, KStruct Access is an extension to the subset of C which allows the definition of structs and enumerations (`enum`). Fig. 2 illustrates the usage of our DSL. The whole language is centered around programming idioms we identified by reviewing the WRK. These idioms usually help programmers to keep the source code maintainable. To capture those idioms in the Windows kernel, a number of extensions, compared to C, are necessary. Collecting these idioms is still work-in-progress. The annotations added to type and field declarations have often been inspired by DCE IDL [1].

Due to the lack of space, we focus here on expressing locking semantics. However, the KStruct Access DSL is capable of expressing the following concepts. A more detailed overview of the KStruct principals and the language itself can be found here [16].

- Lists
- Variable-sized arrays
- Arrays with varying element sizes
- Polymorphic pointers
- Variant data types
- Meta-data for rendering purposes

1.3 Synchronization

Operating system kernels are massively parallel, highly efficient systems. Expressing locking semantics on data structures was therefore one of the major concerns when designing the DSL. In Fig. 2 these semantics are declared by either the *lock* or *mlock* keyword, respectively. While the lock keyword causes the KStruct driver to acquire the specified lock before accessing the data structure that contains the lock, the mlock keyword causes KStruct to acquire a lock only if the specified field is accessed. That is, the KStruct driver will acquire the `AddressCreationLock` lock if and only if `VadRoot` is to be accessed.

Each field of a data structure may itself be a nested data structure or may point to another data structure and so on. A lock is released only, if all objects on an object path have been accessed. This may raise the issue of deadlocks as we may introduce lock dependencies that were not considered when designing the Windows kernel. In order to not halt or crash the system, the KStruct framework will always try to acquire a lock at first place. If the lock can be acquired, KStruct will proceed in parsing the object path. Otherwise, all previously acquired locks will be released. The KStruct driver will eventually attempt to acquire all the locks again.

2 KStruct – The Bigger Picture

KStruct is a system for consistently inspecting kernel data structures, while the kernel is running. It was originally designed for classroom use only. However the key functionality of KStruct, picking arbitrary data structures within the kernel heap, may be beneficial in data center environments too. Which data structures are to be inspected is defined by the users facilitating the KStruct Access DSL. Some data structures may be even indicators for resource shortages or even failures. The following list is by no means exhaustive. However, it shall provider the reader an impression of what these indicators may look like.

Available page frames If the number of available page frames on a node drops below a certain threshold it may be an indicator that the system in its current setting is running out of memory. In a virtualized environment, as described in the introduction of this report, this might be an indicator that too many guest operating systems are *actively* using their resources and thus exhaust the host's resources.

User/kernel time stalls If the amount of time a services spent running in user or kernel mode does not change for a certain period of time, it may be an indicator that the service hangs. However, just considering that value might not be sufficient. A service may also wait for another request! To make sure the service really crashed, KStruct may inspect the objects any thread of that service is waiting on. If there are no such object, we may have an even stronger evidence for a service crash.

Application specific indicators KStruct may even access application or service specific indicators even without any modification. Services may export these indicators by means of the OS, *e.g.*, Performance Counter or the `/proc` file system [6, 8], or their service containers [20]. However, these indicators may not be useful under all circumstances and some indicators may be even missing. In such situations, KStruct offers a simple and flexible solution to support even those indicators.

For those reasons, we focus on integrating the KStruct framework into the reliable service computing platform presented in the last Research School report and continue to investigate the applicability of KStruct in data center scenarios. For the sake of completeness, we would like to give a short review on my architecture proposal for fault-tolerant service computing [18]. While different levels in the software stack may be considerable for achieving fault-tolerance, we focus on the operating system level, as we think this allows for the most transparent and service independent approach.

Although services, like Web services, inherently are stateless, most of those Web service implementations rely on a *session state* that is maintained for each instance of a consumer-provider interaction [21]. For instance, such a session state may be the content of the shopping cart for an on-line shop. Also, Web services rely on persistent storage that manifests the state of the whole business model behind the service. Other approaches for fault-tolerant SOAs do not address this important issue of where to store this data. Typically, this data is stored in databases which in turn need infrastructure and maintenance. From the service provider's point of view, a more holistic approach may be beneficial.

Several middleware approaches exist to replicate services and their state from one node to another. In contrast to existing solutions, our approach uses static, per-service recovery schemes describing a set of nodes where to restore the service in case of a failure. Also, we exploit capabilities of the operating system to provide the static replication scheme of a service and its state. Static recovery schemes are particularly beneficial as they allow determining a replication node in constant time.

2.1 System Model

Our system consists of a set of machines (physical or virtual), called *nodes*, which are interconnected via a reliable network. In the beginning, each node hosts exactly one service instance. Within the fault model introduced by Christian [5], we only tolerate crash faults of either nodes or service instances, *i.e.*, a node or service instance that crashed will remain silent and unfixed until the whole system may eventually be rejuvenated. Due to limitations in the underlying protocol stack, the omission fault class and the timing fault class are mapped to the crash fault class. Tolerating faults in the Byzantine fault class is beyond the scope of this paper.

A service typically processes a request regardless of previous messages (*stateless* service behavior). However, in real-world service implementations there is an urgent need for representing a common state between a service consumer and a service provider [21]. Thus, in case of a node/service crash, it is not enough to simply re-deploy the service on another node. Furthermore, the state has to be propagated in advance to a recovery node. In our model, the state of the service is the aggregation of all operating system provided resources, like open network connections, data base connections or objects allocated on the heap. Further, we make the following assumptions: (1) this propagation is possible in a reliable way, and (2) the transmission delay of a state transfer is much less than the average meantime to failure of a node. Therefore, the communication network must allow end-to-end communication between the cluster nodes and must provide a certain bandwidth with enough transmission capacity to replicate the service state to the recovery node.

We maintain for each service a recovery list. A recovery list describes *where* a service has to be re-deployed after its execution node failed. The set of recovery lists of all services in the cluster compose the *recovery scheme*. In recent research, static recovery schemes were investigated and algorithms were developed which ensure optimal load distribution for a huge amount of cluster nodes [7]. The derived schemes ensure the best achievable load distribution across the remaining cluster nodes in the case of (multiple) node failure. We exploit the advantage of static recovery lists in our services infrastructure: according to our restricted system model, it is sufficient to propagate state changes only to the appropriate recovery node.

2.2 Implementation

Every node uses a modified operating system kernel which provides state replication capability. This kernel must be able of (1) capturing the service state, (2) replicating the service state to other nodes, and (3) restarting processes in the case of node failures.

On the operating system level, the state of a process can be described by its used memory pages, and its kernel resources. For capturing the service state all memory areas which were modified as a result of a service request have to be captured. Resources have to be captured in a different way: File and network operations can be recorded on the API level so that each operation can be “replayed” on the recovery node.

The state transfer is based on static recovery lists. Each process has an associated

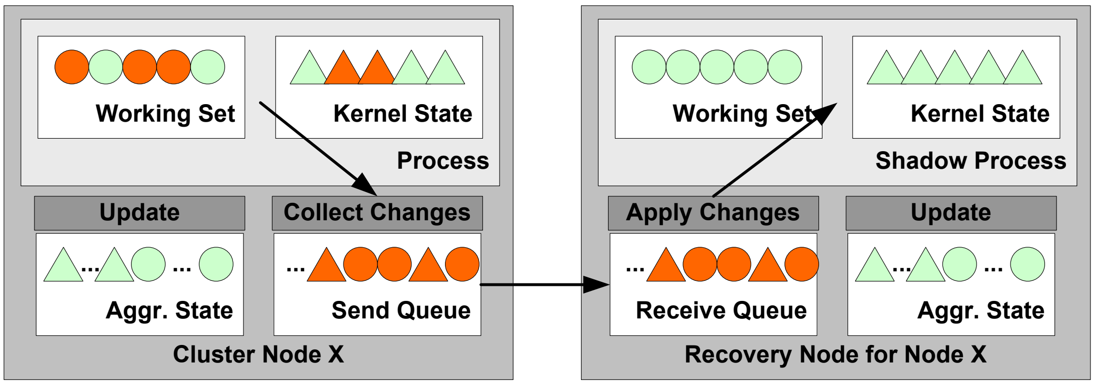


Figure 3: State Replication

recovery list, which determines where the process has to be restarted if the current node is failing. The process state is incrementally replicated to this node. In this way, we avoid some problems of other replication approaches: (1) No global checkpoints are necessary. (2) There is one single replica per process. (3) The network traffic caused by state replication is constant.

For state transfer, we propose an event driven (warm) replication scheme: updates to the state are propagated in the event of a completed request. Figure 3 illustrates the procedure of state transfer. (1) The system waits until the processing of the current request is finished. (2) The state modification is determined: the current working set of page frames is checked for updates. Modified pages are inserted into the transfer queue. Also, modified kernel resources are inserted into the transfer queue. (3) The changes are applied to a (locally) consistent state representation. (4) The content of the queue is transfered to the recovery node. (5) After the state transfer was acknowledged by the recovery node, the next request can be processed. (6) The recovery node applies the state modifications onto the shadow service. This procedure is repeated after every processed request.

In this way the complete service execution state is available in four representations: (1) in the memory of the real service instance, (2) in the local state representation on the same node, (3) in the memory of the shadow service instance on the recovery node, and (4) in aggregated form on the remote node. The second representations is necessary in case the recovery node crashes. In this case the replicated service state is lost. The recovery list of the local process determines the next recovery node. The local state representation is transfered to this next recovery node and the shadow process is reinitialized. Afterwards, the state can be incrementally transfered to this new recovery node as described above.

In the case of node failure, services are migrated from the failing node to the respective recovery nodes. State replication and process migration depend on the recovery lists of the processes. A (static) recovery list determines the sequence of cluster nodes where a specific process has to be restarted. In the following example we assume a four-nodes-cluster and the following recovery list for process 0: $R_0 = \{1, 3, 2\}$, *i.e.*, if node 0 breaks down, the process is restarted on node 1, if node 1 breaks down subsequently, the process is restarted on node 3 and so on. The recovery lists for

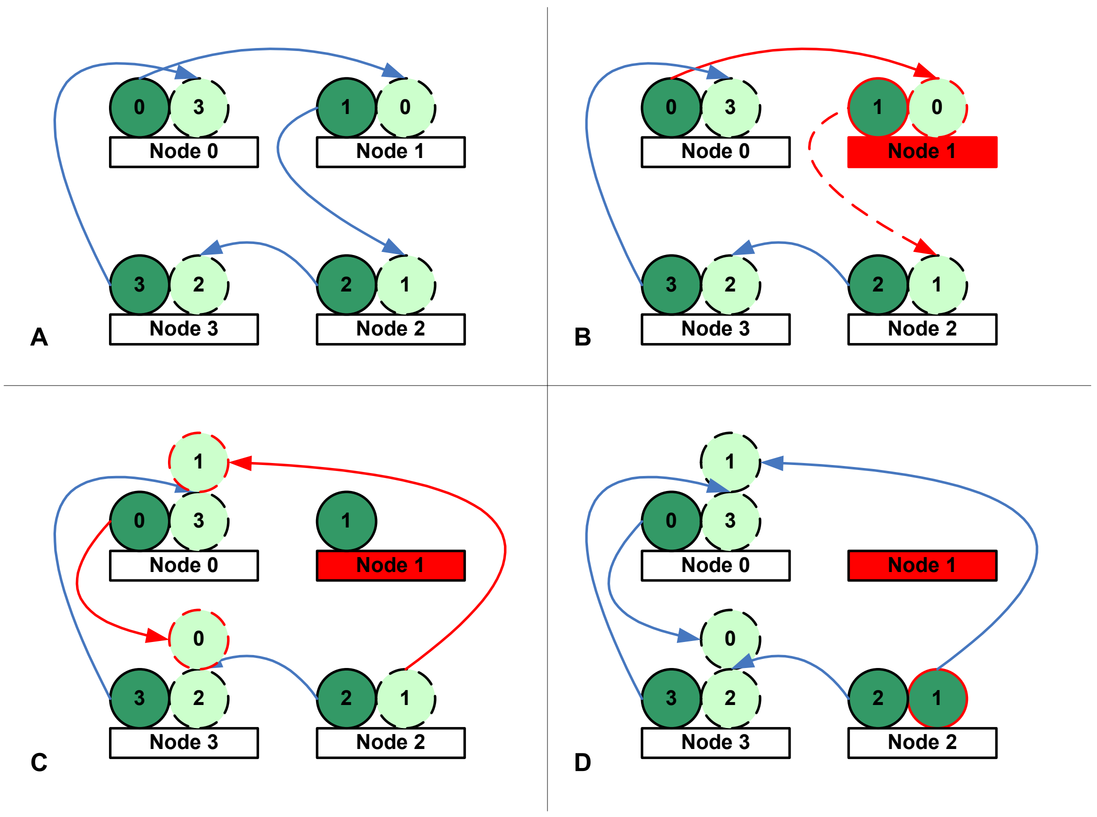


Figure 4: Recovery

all other processes can be derived by adding one (modulo four) to each position; *i.e.*, $R_1 = \{2, 0, 3\}$, $R_2 = \{3, 1, 0\}$, and $R_3 = \{0, 2, 1\}$.

Figure 4 shows the procedure if a node breaks down. (A) A service is executed on four cluster nodes. Each process is replicated to one other node. (B) Node 1 breaks down. Process 1 must be recovered. The replicated shadow process of process 0 is lost. (C) The aggregated process states are transfered to the next recovery node of each process. The state of process 0 is transfered to node 3 and the state of process 1 is transfered to node 0. (D) The shadow process of process 1 is initialized for service execution.

Initial case studies have shown the feasibility of our approach. We have modified the Windows Research Kernel [14] to allow for fault-tolerant service execution on a multicomputer system: while one processor executes an active process, the process' state is replicated on another processor. The system then allows to switch seamlessly between both process instances.

2.3 Summary

We have presented an architecture for supporting a highly reliable, predictable, and fault-tolerant service execution environment, which is most sought-after in SOA. We achieve fault-tolerance by replicating the state of a service. The architecture therefore exploits the benefits of static recovery lists: selecting a recovery node is made in $O(1)$. Also, the network traffic for replicating the service state is minimized, as the state has to be transfered to only one recovery node. The state transfer was further refined by modifying the OS directly which allows for fine-grained state-tracking. The initial architecture was presented in [18].

3 Microsoft Internship

In this final part of my report, I would like to present the work I did during my internship with Microsoft. Although only related through the Windows Research Kernel (WRK) to my research, it was a personal gain and once in a lifetime chance I had to seize.

Among others, my internship had the following goals, which I will cover briefly in the remainder of this report.

1. Leveraging access to the WRK memory management subsystem.

2. Design of a WRK authorization infrastructure.

Beside those goals I had the chance to meet with architects of the Windows operating system and to get a slight insight on how Microsoft sees the future in service computing.

3.1 The Easy Pager Project

The memory management subsystem is the part of the WRK that manages virtually all aspects related to virtual memory. According to the Windows NT Design Documents (part of the WRK [11]), the memory management subsystem is responsible for:

- Supporting virtual memory.
- Mapping files into a process virtual address space.
- Protecting shared memory and memory mapped files.
- Implementing an API for application control on virtual address space allocation and mapping of shared memory.
- Supporting copy-on-write pages.
- Reserving and committing private memory without creating any memory objects.

A core feature of virtual memory is paging, which allows multiple processes to have (1) their own address space and (2) to use more memory than is actually available on the system. This is implemented by dividing the physical memory into page frames and to assign at each point in time at most one virtual page of some process to that frame. Pages that do not reside in memory are stored in a backing file.

Whenever a thread allocates memory, the virtual address space of its process is increased by the amount needed. The WRK distinguishes *reserving* from *committing* virtual address space. Only virtual addresses that are committed to a process are valid for use by a thread of that process. Each process has its own private address space, although some parts of it may be shared with other processes. On x86 architectures that address space can range from 2 GB to 3 GB depending on the option Windows is booted with. Physical and virtual memory are separated into portions called pages. To distinguish virtual pages from physical pages, we henceforth refer to physical pages as page frames. Not all pages of a process can be resident at the same time, *i.e.*, a page is mapped to a page frame. Thus the WMM must maintain information about which pages are currently resident and which are not. This information is called the working set and each process has its own working set.

The working set becomes particularly interesting when the page frames run short or even are exhausted. In that case, the virtual page cannot be mapped directly to a page frame. Instead the memory management subsystem has to decide, which page to remove in order to free that page frame. The decision, which page to sacrifice is crucial with respect to the overall performance of the OS. This is first, because pages are usually written to a persistent storage, which has significantly higher access times, and second, paging-out pages that are referenced again shortly afterwards keeps the OS unproductively busy. Thus, paging or page replacement strategies or policies are a fundamental part of operating system courses.

The WRK memory management subsystem is a very highly optimized system. Also, dealing with working sets and paging is not as simple as in the model described above.

For example, working sets are not static but can grow or shrink depending on the memory pressure. Accessing pages may result in additional page faults due to the memory management architecture provided by the hardware, *e.g.*, when a page table containing the page is also paged out. Although dealing with those facets is interesting and obligatory for operating system architects, it might be too sophisticated for undergraduate students.

The *Easy Pager* project thus tries to abstract from all the nifty implementation details given by the WRK memory management subsystem. It focuses particularly on paging and page replacement strategies. Some goals include but are not limited to:

- Students shall implement their own working set structure.
- The Easy Pager and the WRK memory management subsystem co-exist, *i.e.*, the WRK will manage working sets as usual and is ignorant of the Easy Pager.
- The Easy Pager shall allow for managing only a subset of the address space, in order to simplify the evaluation process of different implementations.
- The Easy Pager is event driven, *i.e.*, in order to implement a different page replacement strategy, only a few event handlers have to be implemented.

The Easy Pager design has been implemented and will eventually be released in the companion of the next WRK release. The release date has not been announced yet.

3.2 Sharing WRK Related Content

Since the epoch of the WRK, the HPI and other universities around the world started using it for their research and teaching. Both projects presented in this paper and the Windows Monitoring Kernel [17] are for example based on the WRK. Although the WRK license agreement explicitly encourages publication in conferences and workshops, it is however difficult to share and distribute the content over the Internet. This is basically because of the restriction that WRK related content may be used by universities only. Especially in the unconstrained Internet it is a challenging task to (1) authenticate users and (2) verify that they are eligible for the respective content.

A practical scenario should illustrate the dilemma. At the Operating Systems and Middleware group, we host an HTML version of the WRK source code. This format has several advantages over the static source code as it makes the WRK explorable and provides a more systematic approach to the huge repository of information. Students at the HPI profit from this repository in their operating systems courses. They may also benefit from accessing our KStruct system. However, making the HTML documentation available over the Internet is a challenging task as we must ensure that prospective clients are eligible for the content.

During my internship I was able to design an infrastructure that enables participating universities to verify the identity of prospective clients and to verify their eligibility for the content. The proposed solution may not be well aligned with current ongoing research in the field of identity management or trust relationships. However it furthermore facilitates already existing building blocks within the Microsoft and Windows Research Kernel ecosystem.

The proposed infrastructure can easily be adopted by participating universities. However, the infrastructure has not yet been tested. With the HPI being the first institution to test the infrastructure outside of Microsoft, further details and evaluations will soon be available.

3.3 Summary

Within this section, I presented those parts of my internship which were not covered by a non-disclosure agreement. However, my internship lead to a fruitful collaboration which is very likely to continue in one form or another.

4 Conclusion

I spent the past 3 months with an internship with Microsoft. Although a huge personal gain, I was unable to proceed in my research efforts in a way I would have liked to proceed during the past semester. Thus, the progress towards my Ph.D. thesis is rather limited compared to the past report. However, I will continue my research to a more holistic approach towards unified data center management, the basic infrastructure for the *cloud* and service computing, with respect to fault-tolerance and load balancing.

In this report, I presented the KStruct framework as a promising candidate to selectively monitor aspects of an operating system kernel. Applicability to virtualized environments like data centers will be a next milestone. My proposed architecture on how to further integrate and unify fault-tolerance capabilities for service computing into the operating system is still a wide field that has to deal with reliable communication, group membership protocols, achieving consensus in the presence of failures. I will continue my investigation on how service-oriented architectures may benefit from existing solutions and where possible flaws may occur. This is however future work and subject to ongoing research.

References

[1] *DCE: Remote Procedure Call.* Number C309. The Open Group, August 1994.

[2] Hp systems insight manager. http://h18013.www1.hp.com/cpq-products/servers/management/hpsim/index.html, 2008.

[3] N. Aghdaie and Y. Tamir. Client-transparent fault-tolerant web service. *Performance, Computing, and Communications, 2001. IEEE International Conference on.*, pages 209–216, Apr 2001.

[4] Bryan Cantrill, Michael W. Shapiro, and Adam H. Leventhal. Dynamic instrumentation of production systems. In *USENIX Annual Technical Conference, General Track*, pages 15–28, 2004.

[5] F. Christian, H. Aghili, R. Strong, and D. Dolev. Atomic broadcast: From simple message diffusion to byzantine agreement. Technical Report RJ 5244 (54244), IBM, 1985.

[6] T. J. Killian. Processes as files. In *Proceedings of the USENIX Summer Conference*, Salt Lake City, 1984.

[7] Kamilla Klonowska, Håkan Lennerstad, Lars Lundberg, and Charlie Svahnberg. Optimal recovery schemes in fault tolerant distributed computing. *Acta Inf.*, 41(6):341–365, 2005.

[8] Michael W. Knop, Peter A. Dinda, and Jennifer M. Schopf. Windows performance monitoring and data reduction using watchtower. September 2001.

[9] Jim Lau, Lau Cheuk Lung, J. da Fraga, and G.S. Veronese. Designing fault tolerant web services using bpel. *Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference on*, pages 618–623, May 2008.

[10] Microsoft. Windows Operating System Internals Curriculum Resource Kit. http://www.microsoft.com/resources/sharedsource/windowsacademic/curriculumresourcekit.mspx.

[11] Microsoft. The Windows Research Kernel. http://www.microsoft.com/resources/sharedsource/Licensing/researchkernel.mspx, 2006.

[12] L. E. Moser, P. M. Melliar-Smith, and Wenbing Zha. Making web services dependable. *ares*, 0:440–448, 2006.

[13] Sajeeva L. Pallemulle, Haraldur D. Thorvaldsson, and Kenneth J. Goldman. Byzantine fault-tolerant web services for n-tier and service oriented architectures. *icdcs*, 0:260–268, 2008.

[14] Andreas Polze and Dave Probert. Teaching operating systems: the Windows case. In *SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education*, pages 298–302, New York, NY, USA, 2006. ACM Press.

[15] Mark E. Russinovich and David A. Solomon. *Microsoft Windows Internals*. Microsoft Press, One Microsoft Way, Redmond, Washington 98052-6399, 4 edition, 2005.

[16] Alexander Schmidt. Kstruct: A language for kernel runtime inspection. In *Proceedings of the Fall 2007 Workshop of the HPI Research School*, Reinsberg, Germany, October 2007. Hasso-Plattner-Institut.

[17] Alexander Schmidt and Michael Schöbel. Analyzing System Behavior: How the Operating System Can Help. In *LNI GI Proceedings 110*, 2007.

[18] Alexander Schmidt, Michael Schöbel, and Andreas Polze. Operating system support for reliable service computing. In *Proceedings of the 5th Service Availability Symposium*, volume 2, Tokio, Japan, June 2008.

[19] Alexander Schmidt, Martin von Löwis, and Andreas Polze. KStruct: An adaptive kernel inspection framework. In *USENIX Annual Technical Conference, Poster Session*, Boston, MA, June 2008.

[20] Peter Tröger. *Dynamische Ressourcenverwaltung für dienst-basierte Software-Systeme*. PhD thesis, Hasso-Plattner-Institut at Potsdam University, 2007.

[21] Peter Tröger, Harald Meyer, Ingo Melzer, and Markus Flehmig. Dynamic provisioning and monitoring of stateful services. In *Proceedings of the 3rd International Conference on Web Information Systems and Technology (WEBIST 2007)*, pages 434–438, Setúbal, Portugal, 2007. INSTICC.

[22] Craig Tunstall and Gwyn Cole. *Developing WMI Solutions*. Addison-Wesley, 2002.

Active Information Graphs

Hagen Overdick
hagen.overdick@hpi.uni-potsdam.de

1 Introduction

The question ”does anybody have a good idea?” rarely is a good sign in the day to day dealings of a knowledge worker. Ideas do not materialize out of thin air and while systems like *design thinking* can help shape an ”idea production process”, the key enabler to being innovative and productive is a clear goal and full attention to the project at hand. However, work no longer has clear boundaries for a knowledge worker. Private and business matters do compete for attention making it difficult to even tell which project *should be* at hand.

Nevertheless, only actions drive a project towards closure. Deciding the next action involves providing the right information and the right tools to the right actor at the right time. However, what is the right information, when many people already complain about information overload? How is information delivered and processed? Who gets the information for what purposes and which tools are involved? How are progress and results documented for future reference?

The difficult answer: It depends. A flood of information is pouring in on us everyday. We need to collect and process this information, put it into relationship, adhere to and discover dependencies between them, and to make things even more complicated, all of these actions produce more information all interlinked in graph structures. Traditionally, we treat these information graphs as passive data and write software to operate on them.

In the course of this paper, we explore a paradigm shift towards treating information graphs as active components. Instead of software operating on such graphs, the graphs expose functionality themselves. This functionality includes modelling the graph, visualizing it, as well as expressing and executing workflows with them.

The rest of this paper is structured as follows: In section 2 we look at business processes and why a large class of them has to be regarded as an evolving information graph. In section 3 an existing meta-process to structuring and evolving information graphs in a workflow context is outlined. Section 4 summarizes the requirements for an IT implementation of such a meta-processes. In section 5 resource orientation is revisited from an implementers view, rather than the common client’s view description. Section 6 explains how an Active Information Graph may be implemented in a resource-oriented approach. The paper closes with related work in section 7 and an outlook and conclusion in section 8.

2 Processes

In Business Process Management (BPM) [29], the focus is on processes as the key instrument to organizing actions into defined activities and to making their interrelationship explicit to improve understanding and consequently increase efficiency, flexibility, and sustainability of the organization.

Generally speaking, with the exception of leisure time, everything ongoing should relate to a project with a defined result. In BPM there are several terms used for *project*, such as *workflow instance* or *case*, however they tend to be bound to specific mindsets, which should be avoided in a general discussion. In the context of this paper, a *project* is defined as any desired result which requires more than one action step. Moreover, the word *action* refers to loosely defined tasks (e.g. call Gero about the presentation next week) while *activity* referes to tasks that can ultimately be automatized (e.g. reserve beamer for next thursday at 13:00 in room C2.4)

The differentiation between actions and activities already hints at different classes of processes. The easiest processes to handle a routine ones, characterized by a fairly static structure, a-priori planability, and defined need for information and communication. Traditional workflow systems focus on static processes as they are easy to plan. Industrial manufacturing is the arch type of static processes and can be characterized by high a-priori knowledge (i.e. clear differentiation between design time and runtime) and a very high activity to action ratio (i.e. tasks are automatable). Semi-structured process are less static and have a lower activity to action ratio, but they still follow general patterns. A typical example for a semi-structured process as taking a customer from initial contact to closing the deal. There a certain fix points (e.g. writing an offer, signing the contract), but the process is more driven by the response of the customer than a static process model. Finally, there is the class of ad-hoc processes. Ad-hoc literally means *for this purpose* and signifies non-generalizable processes. Examples include both one-time processes without a-priori knowledge (e.g. building a Large Hadron Collider) as well as creative processes of knowledge workers (e.g. the output is unknown at start and every exact output is produced only once).

Knowledge intensive processes are called *wicked problems* [25], as they have unpredictable patterns of behavior. Most of the observable behavior is interdisciplinary communication and information exchange. There is no clear end to such processes, instead there are frequent review phases leading to either an iterative loop or termination (successful or unsuccessful) of the process, some approaches even discuss making these loops explicit [18].

3 Getting Things Done

In Getting Things Done (GTD) [2], David Allen shares his insights from coaching thousands of people on being productive. His conclusions are based on the premise that productivity is indirectly proportional to the things one has to keep in mind. Here, keeping in mind is ment literally: if your brain does not trust you to be able to organize and store all things current, it will unconsciously do so for you, but by doing so keep you in a unrelaxed, unproductive state.

Consequently, one has to avoid remembering things actively and establish an organization system your brain will ”trust”. David Allen suggest a basic workflow for processing incoming information as shown in Figure 1. From the perspective of BPM this process can be regarded as a meta-process creating an environment in which processes instances in the more traditional BPM sense are rooted. In GTD, process instances are called projects. A project is defined as any desired result which requires more than one action step, the definition inherited by this paper. Consequently, actions may happen outside of projects (just one action to reach goal, e.g. a college is asking for a paper, no need to create a project, simple provide the PDF). The GTD meta-process is focused on emptying the inbox, which holds any kind of incoming information, including notes to oneself. Experience recommends to simply process it linearly until it is empty. Processing the inbox in low frequency is sufficient, there is no need to being distracted e.g. by incoming mail and disrupt work for ever single incoming item. However, once any item in the inbox is processed, it should never stay there. Again, the goal is to empty the inbox. To do so, the first question to ask is, whether the item at hand is actionable, i.e. does it require an action. If not, it may be irrelevant and should immediately be trashed. If it is relevant but not actionable, it can either be a reference information to be stored for quick retrieval (remember knowledge work is information intensive) or it might be an idea not be acted upon right now. These should be kept on a review list, a tickler file for future projects.

However, if the item currently processed is actionable, the very next question should be for the *next action*, i.e. what can be done next. If this turns out to be a multi-step action, the item in the inbox should be related to a project, either an existing one or implicitly create a new one. Again, projects represent a desired outcome and as such are the foundation for project plans. Project plans are a set of milestones along the way towards the desired outcome. Another function of project plans is for reviewing progress. According to Allen, it is sufficient to know the very next action within a project only. If we go back to the discussion about processes, this approach appears to be a perfect match for ad-hoc processes. Actions may take place without projects (i.e. process instances), but are created on demand as driven by input. Once an ad-hoc process is created, the focus is on creating a set of milestones leading to the desired result. Within any given ad-hoc process, it is sufficient to determine the very next action to take. Any ad-hoc process without a determined next action is in planning, but the project plan is not focused on activities, but milestones.

Once the next action to take is determined - either ad-hoc while processing the inbox or as the result of a project planning - the very next question to ask is how long it will take. Here, Allen suggests a 2-minutes-rule, i.e. if it takes less than 2 minutes to complete the action than simply do it. If not, there is yet another question to be answered: are you the right actor of this action? If not, delegate the activity and keep a reference on a ”waiting for” list. If yes, the action should be deferred. Either, to a fixed time (e.g. a meeting) in a calendar or to one or more next actions lists. Why several lists? Because context matters to most of these actions (e.g. a reminder to call someone is only useful, when you are at a phone and have some quiet time).

The GTD meta-process is not bound to an IT solution. In fact, it can be implemented

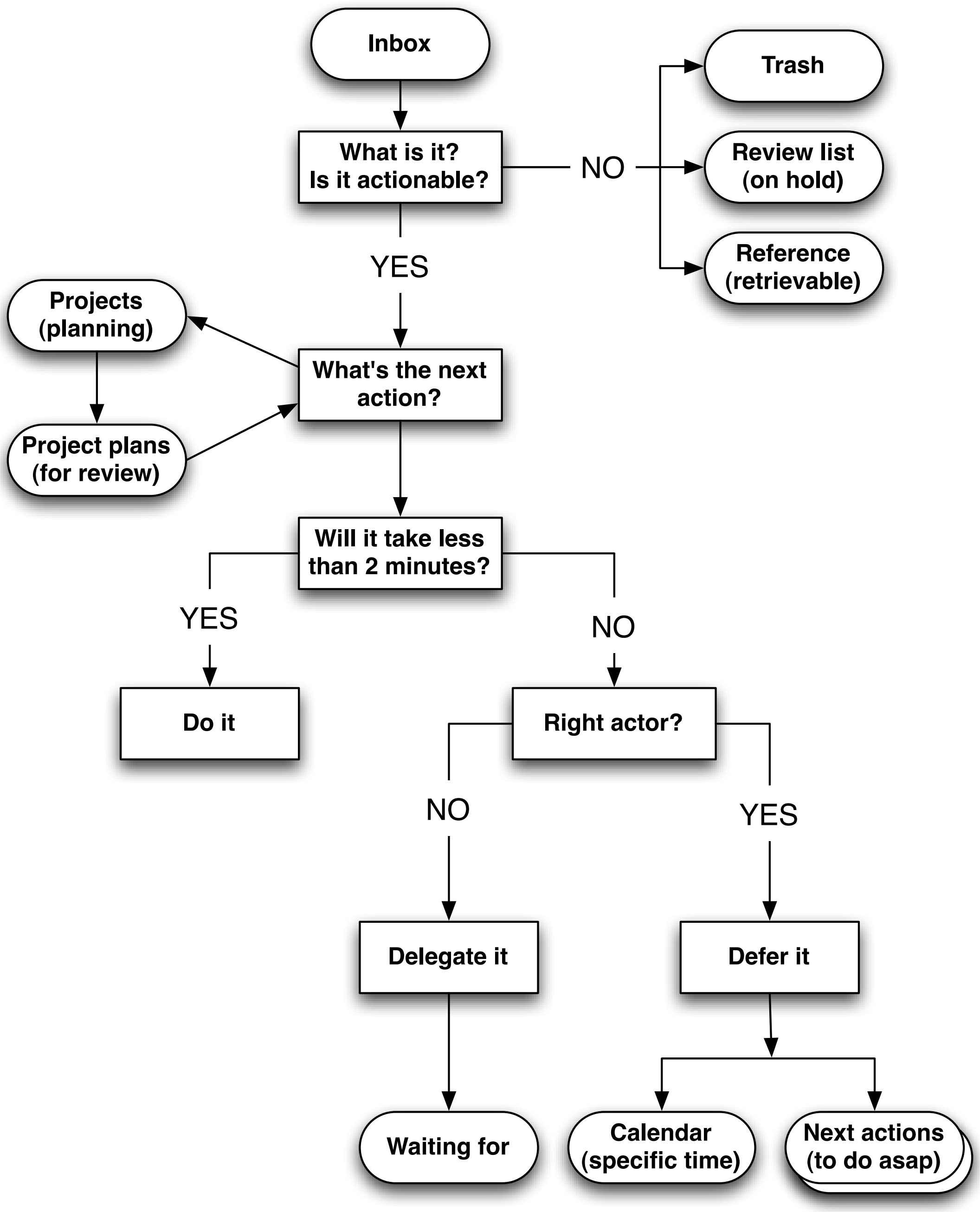


Figure 1: Gettings Things Done - The Meta Process

without any IT solution at all and at the time of the book's creation no IT implementations where available. In the meantime, several such implementations where create, such as OmniFocus[1] and Things[2]. However, all available implementations focus on task management, i.e. they act as reminder systems and do not integrate BPM features such as automated activities and/or control flow between actions.

In the next section, requirements for an IT implementation of the GTD meta-process allowing for more advanced BPM features are discussed.

4 Requirements for an IT-based BPM supporting GTD

From the above discussion we can deduce the following requirements:

- There is a differentiation between *context* and *project*.
- Projects are a grouping mechanism for various elements, including goals, milestones, actions, activities, as well as supporting information relevant to the project.
- Contexts are orthogonal to projects. They group roles, locations, available actions and activities, and project independent information.
- There needs to be a dynamic mapping from project elements into contexts. The closest equivalent in traditional BPM systems would be task lists, however task lists are traditionally bound to roles only.
- The central component to feed new elements into the system is an inbox, taking untyped data and with the help of the GTD meta-process either evolving them into typed elements or leading to the creation of typed elements based upon them.
- Having different contexts implies a distributed system as contexts explicitly incorporate different physical locations. Even today, not all locations can provide an internet connection (e.g. airplanes), hence offline capabilities are desirable. Also, distributing for scalability implies offline capabilities as servers will fail and dealing with failure does not differ from dealing with offline operations.
- Eventually projects should be able to support control flow capabilities as described in workflow patterns [30]. It has been shown [32] that not all workflow patterns have local decidability, but nevertheless should be supported.
- Information, i.e. data objects, may reside outside of a project and may be referenced by several projects at the same time.

[1] `http://www.omnigroup.com/applications/omnifocus/`
[2] `http://www.culturedcode.com/things/`

5 Resource Orientation

In [8] the underlying principals of the world wide web where described. Out of this work, the concept of resource orientation emerged. Resource orientation is an architectural style with the following key priniciples:

- All members of a communication network are independent of it, instead they expose functionality as resources. Interaction with resources should never be transparent, as latency and network failures can not be made transparent.
- Resources have globally unique identifiers (URI) and each URI exposes a uniform interface. All interfaces operate on synchronous and stateless request-response interactions. The uniform interface does not prescribe the exchanged messages, but helps identify content type (e.g. via mime-types [9]) and intention (e.g. via verbs). Making intentions explicit allows for intermediate processing without explicit knowledge of the application. Best example is caching for side-effect free reading.
- The preferred message type is hypermedia, i.e. content capable of encoding URIs. Given hypermedia, the client is enabled to keep the application state as the sum of all received messages. All possible state transitions are encoded into these messages as URIs. A client should never be forced to guess or compute URIs, but receive them explicitly.

While these principals capture a large portion of the success of the world wide web, there description is purely from a client perspective. As this paper focuses on the server side, let’s describe resource orientation from the server side as visualized in Figure 2:

Entity An entity is a set of persistent data and is primarily characterized by a disjoint scope of transactional serializability. The concept was named and described in [12], where the author also argues for an entity to not grow beyond the capabilities of a single machine to allow for almost-infinite scaling of applications. Notice that a traditional relational database must be regarded as a single entity, as arbitrary transactions across all contained data is explicitly allowed by design. However, the recent rise of discussion of *sharding* for scalability [13] gives validity to the requirement of keeping entities smaller than the capabilities of a single machine.

Persistent Data may be stored in arbitrary form. This includes relational data as well as documents. The persistence is guaranteed by the enclosing entity. Consequently, transactional operations on persistent data can only happen when they all belong to the same entity.

Function Functions operate on any subset of entities, including creating and deleting entities themselves. For the sake of simplicity, we focus on functions operating on persistent data only. Function may operate transactional on persistent data within a single entity or exchange message with other resources (see below). However, the

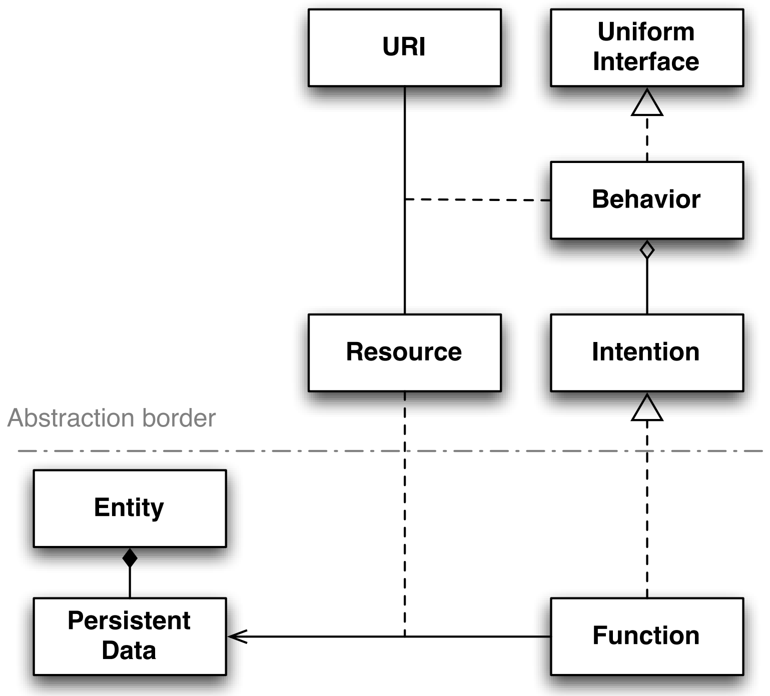


Figure 2: Visualizing an resource oriented entity

length of any transaction is limited to a single function call, i.e. calling a function will start a transaction, returning from it will end it. Any interaction caused by the function with other resources is not part of the transaction.

Intention Every function is an instance of an intention, i.e. all effects of executing a function (both on persistent data as well as messages originated) must comply to the definition of the intention implemented. In the case of the Hyper Text Transfer Protocol (HTTP) [7] the following intentions[3] are defined:

- `GET`: Interactions with the intention `GET` have an empty request body and are guaranteed to have no substancial effect within the receiver of such request, i.e. they are *safe* to call. `GET` responses are expected to be a description of the current state of the targeted resource (see below). These attributes allow `GET` to act as a universal reflection mechanism, it can be issued without any prior knowledge of the resource. Also, as `GET` does not alter the state of the targeted resource, the response can be cached. This has great benefits to a distributed architecture and both aspects can be seized without prior semantic knowledge of the targeted resources.

- `PUT`: Interactions with the intention `PUT` do cause an effect in the targeted resource, but do so in an *idempotent* fashion. An idempotent interaction is defined as replayable, i.e. the effect of N messages is the same as that of 1. In a distributed

[3]RFC 2616 refers to intentions as *verbs*

system, where transactions may not be readily available, this is a great help to error recovery.

- `DELETE` : Interactions with the intention `DELETE` do cause an effect in the targeted resource, where that effect is expected to be some kind of termination. Just as `PUT`, `DELETE` is defined as *idempotent*. However, as with all interactions, the interpretation is solely the responsibility of the receiver.

- `POST`: All other types of interactions default to the intention `POST`, i.e. they cause an effect in the receiver and they are not safe to replay. This is a catch all mechanism for all interactions that can not be described by the prior intentions. Without a uniform interface, all interactions must be treated like this, doing so causes loss of context free reflection, caching and replayability at the protocol level.

Uniform Interface In resource orientation, all functionality is exposed via a uniform interface. A uniform interface defines how interactions are executed, especially how intentions are signaled and content types are negotiated. This leads to simplified communication, loose coupling, and the possibility of intermediaries to intervene in the message exchange (best example caching) without application knowledge.

Behavior From a client's view only the behavior of a resource is relevant. Because of the uniform interface the expressiveness of a behavior is limited. Hence, instead of one complex behavior entities expose many simple ones. A behavior groups a set of intention instances into a meaningful concept.

Resource A resource is the binding of a behavior to a specific subset of an entity according to the uniform interface. This is the implementation view, from the client's perspective a resource is synonymous with an instance of a domain specific behavior.

URI Uniform Resource Identifiers allow for globally unique referencing of resources. Note that the binding from URI to resource is not required, i.e. a URI can exist without any actual binding, which allows references to be made to a concept before any realization of that concept exists.

This description of resource orientation differs from previous papers, e.g. [20], where resource was taking the position of entity. However, with this picture it is much easier to explain why exposing resources does not automatically imply better scaling, as the size of the entity is relevant. Also, showing which resources are actually within a transactional scope is much easier.

To a client the entity is never exposed, nor should the client be required to be aware of it. Only active resources are exposed and the architectural decisions of resource orientation simplify dealing with a myriad amount of available resources.

6 Active Information Graphs

As outlined elsewhere before [4, 22], the author strongly believes resource orientation to be well-suited for workflow applications. The current state is modeled as a resource, all possible transitions are offered to the client as URIs embedded into hypermedia (i.e. links) returned by interaction with these resources. This ensures loose coupling and simplifies the introduction of new capabilities, they simply represent new links.

But in the case of ad-hoc processes the loose coupling goes beyond just client and server. It also extends to decoupling data from projects. In the GTD meta-processes, data exists before a project and might be used in more than one project at the same time (e.g. reference material). Also, incoming information may evolve its type.

To address all these issues, we now introduce the concept of Active Information Graphs. In the concepts of section 5 a graph is an entity consisting of nodes and edges. Each node and edge is an individual persistent data object. In [21, 23] a serialization format was introduced including a resource-oriented protocol for creating, updating, and deleting of nodes and edges of such Active Information Graphs, the Potsdam Encoding for Models (PoEM).

Mapping the GTD meta-process onto Active Information Graphs, an individual Active Information Graph for each project appears to best suited. All project elements are represented by nodes and relationships between them by edges. Mapping a project to an individual Active Information Graph (and consequently to an entity) is a good compromise between the desire to keep entities as small as possible and the need for transactional scopes. Our use cases from the Oryx project [5, 6] indicate a transaction scope at the project level to be useful, both for modelling (e.g. moving several elements graphically at the same time) as well as executing (e.g. transition semantics of various workflow patterns).

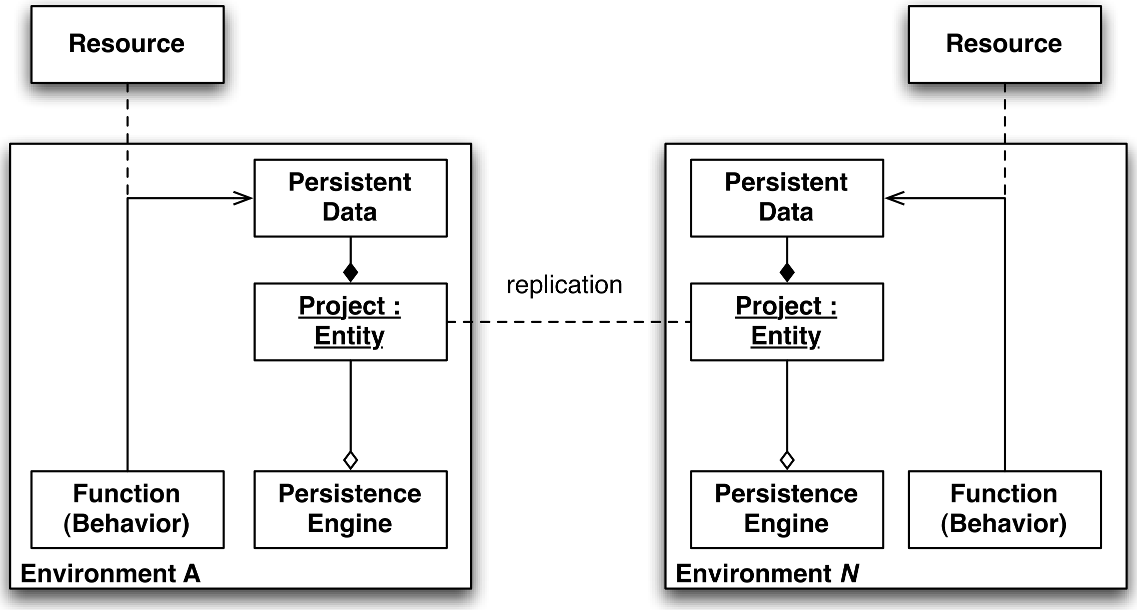


Figure 3: Proposed architecture for Active Information Graph environments

In section 4 we described contexts to be orthogonal to projects and the need for offline capabilities, both from the use cases as well as from the aspect of scalability. Offline capabilities are only possible with distributed hosting environments and in a first step we simply map contexts to different execution environments. An execution environment consists of a persistence engine and a set of available function. Execution environments replicate Active Information Graphs completely, i.e. each environment is capable of establishing a transactional scope over all elements of a graph. Additionally, an environment contains a set of functions. Based on the available functions and graph elements, resources are exposed analog to the outline in section 5. This architecture is visualize in Figure 3.

Notice that the execution environments are completely abstracted away. To a client, only resources are exposed and the sum of one entity's resources jointly form one Active Information Graph. The minimum active behaviors of an Active Information Graph is to be editable are:

Active Information Graph behavior

- `GET` lists all graph elements in PoEM notation.
- `PUT` may be used to create Active Information Graphs. To enable this, the persistence layer must expose a valid URI template [11], where all instances of this template map to this behavior. Semantically, this is an update from void, as the URI is chosen by the client and consequently bound conceptually before an Active Information Graph was actually created.
- `POST` creates new graph elements from the content of the interaction. By using `POST` the non-trivial task of naming a URI for the graph element is delegated to the Active Information Graph itself.
- `DELETE` destroys the Active Information Graph and all its elements.

Each graph element provides at least one resource:

Active Element behavior

- `GET` returns the current representation of the element in PoEM notation.
- `PUT` updates the element. The lost update problem is solved via [19]. Notice, that resource orientation favors a replace semantic, i.e. a complete representation is send.
- `DELETE` removes this element from the Active Information Graph.
- `POST` is undefined for this behavior.

These two behaviors already allows for the implementation of the GTD meta-process as outlined in section 3. However, actually knowing and adhering to the process is left completely to the client, i.e. even though we have an Active Information Graph, we would simply treat it as data storage. The Active Information Graph truly becomes active, when we start internalizing the process into the graph.

In PoEM typing a graph element is expressed as zero or more `<category term="concept" scheme="domain.uri" />` tags. The `scheme` attribute references the defining domain of the type and the `term` attribute a specific type within the referenced domain. Consequently, evolving an element's type simply maps to altering the set of `<category />` tags.

Going back to the GTD meta-process in Figure 1 we could introduce a behavior binding to all untyped notes and implement the first step in the GTD meta-process:

Process Input behavior Upon GET a small hypermedia form with the buttons *Actionable*, *Review list*, *Reference*, and *Trash* is returned. The later button can directly link to a `DELETE` on the *Active Element behavior* URI. The review list we assume to be an independent Active Information Graph, hence its form will map to `POST`, as moving an element from one graph to the other is crossing a transaction boundary. The actual move is implemented by the bound function, i.e. it will `POST` the element to the targeted graph and then execute a `DELETE` on the local node. *Reference* and *Actionable* we consider to be types. Consequently, changing the type of an element can be mapped to `PUT` on the *Process Input behavior* URI. The implementation will augment the matching `<category />` tag to the node. This in turn will allow other behaviors to bind to the node, e.g. a *Evaluate Next Action behavior* to *Actionable* nodes.

Discovering, documenting, and implementing such behaviors will be a vital part of future research, executed by evolving use cases in the domain of ad-hoc processes. Eventually, this should lead into the support of all workflow patterns on top of the GTD meta-process.

6.1 Performance considerations

Even if the name Active Information Graph is introduced in this paper, it is not the first attempt to bind functionality to a graph, e.g. the original Oryx design included a wiki-oriented execution environment directly driven by the modeled process graph. The underlying technology was reused by a Olaf Märker in his master thesis [17], again exposing resources based on modeled graphs[4]. All these former attempts did not prosper, because of performance issues. These issues can be summarized as decisions about indexing and granularity. Dynamically binding functionality to a graph requires fine-grained access to the graph's elements and fast lookups. The finer the granularity of the graph's elements is, the easier it is to realize fine-grained access. However, the indexing overhead increases dramatically. The initial design was to keep a persistent XML Document Object Model [15], simplifying dynamic queries. However, the required parsing and indexing are just overwhelming and lead to a poor performance. In [3] a different approach is shown. Instead of preprocessing data for random queries, specific queries are prepared asynchronously via the map/reduce pattern. Applying

[4]To be precise: PiVM is tree-based

this approach to Active Information Graphs, each node and edge is stored in whole as an individual persistent data element and asynchronously operating map/reduce operation create the desired indicies, providing dramatically improved reading and writing operations and proven scalability (Google uses this approach internally). Going back to the `<category />` example, a map/reduce function could realize the binding of functions to persistent data (i.e. graph elements exposing resources) by generated an index of all `<category/>` tags and matching functions, effectively creating an index of all possible bindings. Note, that such binding does not have to be limited to a single node, in fact complex workflow patterns will bind to several nodes at the same time, made possible by the enclosing transactional scope of the Active Information Graph. Binding to several nodes should be easy to achieve via the map/reduce pattern, a statement to be validated empirically shortly by a reference implementation.

6.2 Replication of projects

Another non-trivial feature of the proposed architecture is replication of Active Information Graphs between environments. From the design of the architecture it is apparent, that a multi-master replication is desirable, to allow operations on Active Information Graphs in multiple environments concurrently. Traditionally, multi-master replication is realized using distributed locks. However, as offline capabilities are also desirable, the use of distributed locks is ill-advised [12]. Without locks, conflicts will arise and need to be handled. In CouchDB [16] a very interesting approach is implemented and as of writing the author is unaware of any scientific discussion of this approach:

Revisiting ACID properties [10] and contrasting them to the discussion in section 5 we can provide atomicity to an entity only individually for each replication, i.e. a single environment. Once replications is used, there may be conflicts on merge, violating the consistency requirement. However, if we introduce the concept of versioning to the persistence layer, consistency and conflict resolution may be separated into distinct features. On conflict, one version is declared the winner and all losers will be added to the history as prior and conflicting versions. If all persistence engines use the same deterministic approach (e.g. the version with the highest update frequency wins), global consistency can still be guaranteed without the need for all persistence engines to communicate with each other. Conflict resolution is pushed into the application realm. As Active Information Graphs are designed to model and execute workflows anyhow, exposing such conflict resolution as a behavior should be acceptable in the general case. In the specific case, e.g an OR-join workflow pattern, this might not be acceptable, as conflicts will be the norm in such cases. These specific cases need to be addressed at the environment level, e.g. binding the execution behavior of an OR-join to a specific environment.

7 Related work

Intentional Programming [27] has the concept of an active source graphs, with pluggable intentions. However, Intentional Programming is focused on software development itself, i.e. the active graph is not the program, but the source code to a program, while Active Information Graphs do not differentiate between development and runtime, they are the applications themselves, evolving at runtime. In WASA2 [31] workflow instances and schemas are represented as object graphs. However, elements of this graphs have a fixed semantics and they do not expose functionality themselves. Here, the object graph is primarily used to simplify runtime modifications. Regarding workflow flexibility and ad-hoc processes, several approaches exist [1, 24, 26, 28] and may be evaluated on-top of Active Information Graphs. The notion of context awareness is a central aspect of Context-oriented Programming [14], but focused on programming languages. If and how the research results of Context-oriented Programing may be applied to Active Information Graphs is future work, most likely they will aid the configuration of environments.

8 Conclusion and Outlook

In the course of this paper, we have introduced Active Information Graphs as a framework for managing, evolving, and working with graphs of information. While the general approach may be applicable to a wide range of applications, it is motivated by and focused on supporting knowledge workers dealing with projects in there daily work. Knowledge-intensive processes are characterized by very little a-priori knowledge and evolution of the processes at runtime. Active Information Graphs provide a framework for creation, distribution, and binding of functionality to information graphs and in turn provide a platform for further research and development. A high-performance implementation of the architecture outlined in this paper is a short-term goal providing distributed, scalable, and offline-capabale Active Information Graphs. Next, an integration with the Oryx editor for visual editing of Active Information Graphs is planed. In combination, a development platform for applications based on Active Information Graphs is within reach, allowing for empirical validation of the suitability of Active Information Graphs for knowledge workers.

References

[1] Michael Adams, Ter, David Edmond, and Wil van der Aalst. Worklets: A service-oriented implementation of dynamic flexibility in workflows. *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE*, pages 291–308, 2006.

[2] David Allen. *Getting Things Done. The Art of Stress-Free Productivity*. Penguin, 2003.

[3] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *Sixth Symposium on Operating System Design and Implementation*, 2004.

[4] Gero Decker, Alexander Lueders, Hagen Overdick, Kai Schlichting, and Mathias Weske. RESTful Petri Net Execution. In *Proceedings of the 5th Workshop on Web Services and Formal Methods (WS-FM)*, LNCS, Milan, Italy, Sep 2008. Springer Verlag.

[5] Gero Decker, Hagen Overdick, and Mathias Weske. Oryx - An Open Modeling Platform for the BPM Community. In *Demo Session of the 6th International Conference on Business Process Management (BPM)*, 2008.

[6] Gero Decker, Hagen Overdick, and Mathias Weske. Oryx - Sharing Conceptual Models on the Web. In *Demo Session of the 27th International Conference on Conceptual Modeling (ER)*, 2008.

[7] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. Technical report, The Internet Engineering Task Force, 1999. `http://www.ietf.org/rfc/rfc2616`.

[8] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000. Chair-Richard N. Taylor, `http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm`.

[9] N. Freed and N. Borenstein. Multipurpose Internet Mail Extensions (MIME) Format of Internet Messaged Bodies. Technical report, The Internet Engineering Task Force, 1996. `http://www.ietf.org/rfc/rfc2045`.

[10] Jim Gray. The transaction concept: Virtues and limitations (invited paper). In *Very Large Data Bases, 7th International Conference, September 9-11, 1981, Cannes, France, Proceedings*, pages 144–154. IEEE Computer Society, 1981.

[11] J.C. Gregorio, M.H. Hadley, M.N. Nottingham, and D.O. Orchard. URI Template. Technical report, IETF, 2008. `http://bitworking.org/projects/URI-Templates/`.

[12] Pat Helland. Life beyond Distributed Transactions: an Apostate's Opinion. In *Third Biennial Conference on Innovative Data Systems Research*, 2007. http://www-db.cs.wisc.edu/cidr/cidr2007/papers/cidr07p15.pdf.

[13] Cal Henderson. *Building Scalable Web Sites*. O'Reilly, 2006.

[14] Robert Hirschfeld, Pascal Costanza, and Oscar Nierstrasz. Context-oriented Programming. *Journal of Object Technology*, 7(3):125–151, 2008.

[15] Arnaud Le Hors, Philippe Le Hégaret, Lauren Wood, Gavin Nicol, Jonathan Robie, Mike Champion, and Steve Byrne. Document object model (dom) level 2 core specification. Technical report, W3C, 2000. http://www.w3.org/TR/DOM-Level-2-Core/.

[16] Damien Katz et al. The CouchDB Project. http://couchdb.org.

[17] Olaf Märker. Eine virtuelle Maschine für den π-Kalkül zur Implementierung von Ressourcen mit dynamischen Verhalten. Master's thesis, Hasso Plattner Institut, 2008.

[18] Raul Medina-Mora, Terry Winograd, Rodrigo Flores, and Fernando Flores. The action workflow approach to workflow management technology. In *CSCW '92: Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, pages 281–288, New York, NY, USA, 1992. ACM.

[19] Henrik Frystyk Nielsen and Daniel LaLiberte. Editing the web. Technical report, W3C, 1999.

[20] Hagen Overdick. Resource-oriented Application Building, 2007.

[21] Hagen Overdick. Supporting Design Thinking with IT, 2008.

[22] Hagen Overdick. Towards Resource-Oriented BPEL. In Thomas Gschwind and Cesare Pautasso, editors, *Emerging Web Services Technology*, volume II, pages 129–140. Birkhäuser, 2008.

[23] Hagen Overdick and Martin A. Czuchra. PoEM - Potsdam Encoding for Models. *Services Computing, IEEE International Conference on*, 2:619–620, 2008.

[24] Artem Polyvyanyy and Mathias Weske. Flexible process graph: A prologue. In *Proceedings of the 16th International Conference on Cooperative Information Systems (CoopIS)*, November 2008.

[25] Horst W. J. Rittel and Melvin M. Webber. Dilemmas in a general theory of planning. *Policy Sciences*, 4(2):155–169, June 1973.

[26] Shazia W. Sadiq, Wasim Sadiq, and Maria E. Orlowska. Pockets of flexibility in workflow specification. In *ER '01: Proceedings of the 20th International Conference on Conceptual Modeling*, pages 513–526, London, UK, 2001. Springer-Verlag.

[27] C. Simonyi. Intentional programming: Innovation in the legacy age, 1996.

[28] W. van der Aalst, M. Weske, and D. Grunbauer. Case Handling: A New Paradigm for Business Process Support. *Data and Knowledge Engineering*, 53:129–162, 2005. `http://bpt.hpi.uni-potsdam.de/twiki/pub/Public/MathiasWeske/sdarticle.pdf`.

[29] W. M. P. van der Aalst, A. Hofstede, and M. Weske. Business process management: A survey, 2003. `citeseer.ist.psu.edu/article/vanderaalst03business.html`.

[30] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. Technical Report BETA Working Paper Series, WP 47, Eindhoven University of Technology, 2000.

[31] Gottfried Vossen and Mathias Weske. The WASA2 object-oriented workflow management system. *SIGMOD Rec.*, 28(2):587–589, 1999.

[32] Moe Thandar Wynn, David Edmond, W.M.P. van der Aalst, and A.H.M. ter Hofstede. Achieving a General, Formal and Decidable Approach to the OR-Join in Workflow Using Reset Nets. In *Lecture Notes in Computer Science*, pages 423–443. Springer Berlin / Heidelberg, 2005.

FMC-QE - Hierarchies, Transformations and Rules

Stephan Kluth
stephan.kluth@hpi.uni-potsdam.de

This report FMC-QE - Hierarchies, Transformations and Rules complements former FMC-QE reports. After a general introduction and an introduction of FMC-QE, the hierarchical modeling in FMC-QE and related work in this area are discussed. This mainly includes decomposability, the adaption of Norton's theorem to the Queueing Theory, research on the formalization of hierarchies and aggregation methods in Petri Nets. In the section model transformations, the while loop and the feedback loop transformation are described, as well as open research questions like the transformation of a parallel execution to a serial execution are raised. The rules, how to model multiplex scenarios, in which multiple logical servers are mapped to a single real server, are described in the section modeling rules. In this section also the modeling of multiclass scenarios is described and again some open questions like the modeling and evaluation of a semaphore synchronization scenario and exception handling are raised. Finally some conclusions and an outlook are given in the last section.

1 Introduction

This report *FMC-QE - Hierarchies, Transformations and Rules* follows a series of reports on FMC-QE, the Fundamental Modeling Concepts for Quantitative Evaluation. After the 1st report in spring 2007: *FMC-QE - Positioning, Basic Definitions and Graphical Representation* [22] defining FMC-QE basics and diagram types, the 2nd report in fall 2007: *FMC-QE - Case Studies* [21] summarizing two case studies in FMC-QE and the 3rd report in spring 2008: *FMC-QE - Calculus* [23] describing fundamental mathematical laws, the load generation model and the FMC-QE Calculus including the FMC-QE Tableau and the formulas in FMC-QE, this report describes the hierarchical modeling, model transformations and modeling rules in FMC-QE.

After a short introduction into FMC-QE in section 2, this report is structured as followed:

Section 3 summarizes related work in the area of hierarchical modeling and aggregation and decomposition in the focus of quantitative modeling and evaluation from the perspective of FMC-QE. The chapter 3.1 summarizes important publications on the topic decomposability done by Simon and Ando [32], Courtois [10, 12] and Vantilborgh [34]. Norton's Theorem of the electrical circuit theory was adapted to queueing theory networks by Chandy, Herzog and Woo [9]. This results were further generalized and specialized for the quantitative evaluation of systems. Chapter 3.2 abstracts publications in this area and shows the relations to FMC-QE. Chapter 3.3 Formal Hierarchies and Combination of Models summarizes approaches of formalizing hierarchies and evaluating different subsystem with different kinds of modeling and evaluation techniques. Chapter 3.4 summarizes some publications on introducing hierarchies in Time Augmented Petri Net models. Literature on the Forced Traffic Flow Law, which is the fundamental law for the hierarchical decomposition in FMC-QE, is described in chapter 3.5. Main ideas of the hierarchical modeling and evaluation in FMC-QE are given in chapter 3.6.

Section 4 proposes model transformations in FMC-QE. These non automatic transformations are made in order to support the evaluation of models which does not follow all FMC-QE assumptions due for example a previous model in a different modeling technique, like the Queueing Theory. These transformations are for example the transformation of the while loop in chapter 4.1 and the transformation of a feed backward loop in chapter 4.2. Also some open questions in the area of FMC-QE model transformation are raised.

Section 5 describes some FMC-QE modeling rules. This section will provide ideas and methods, how to model different scenarios in FMC-QE. In the first chapter 5.1, the modeling of a system in which multiple logical servers are mapped to one real server is described. In FMC-QE, the modeling and evaluation of multiclass scenarios is also supported. The description, how to model such systems is given in chapter 5.2. Again some open questions in FMC-QE modeling rules are discussed.

Finally some conclusions and an outlook are given in section 6.

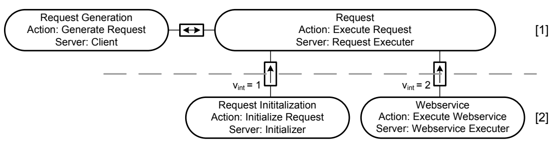
2 FMC-QE

The Fundamental Modeling Concepts for Quantitative Evaluation (FMC-QE) is a modeling and evaluation calculus that originates from and extends the modeling technique Fundamental Modeling Concepts (FMC) [24, 33] for the description and evaluation of the quantitative system parameters. FMC-QE also integrates ideas, results and advantages of the Queueing Theory and the Time Augmented Petri Nets.

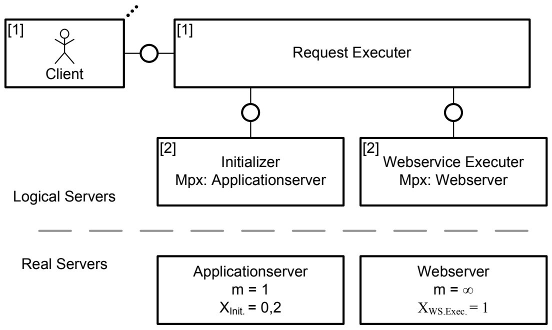
The hierarchical service request, which is the origin of every service provisioning process, is in the main focus of the FMC-QE modeling [38]. In order to realize the hierarchical decomposition, the service requests are modeled, comparable to physics, as a tuple of value and unit ($SRq_i=\{SRq_i\}[SRq_i]$).

In FMC-QE, systems are modeled in three different views from the perspective of the hierarchical service request. These three views, based on FMC, are the hierarchical service request structures, modeled in Entity Relationship Diagrams (example in figure 1(a)), the static (server) structures, represented in Block Diagrams (example in figure 1(b)) and the dynamic (control flow) behavior, described in Petri Nets (example in figure 1(c)).

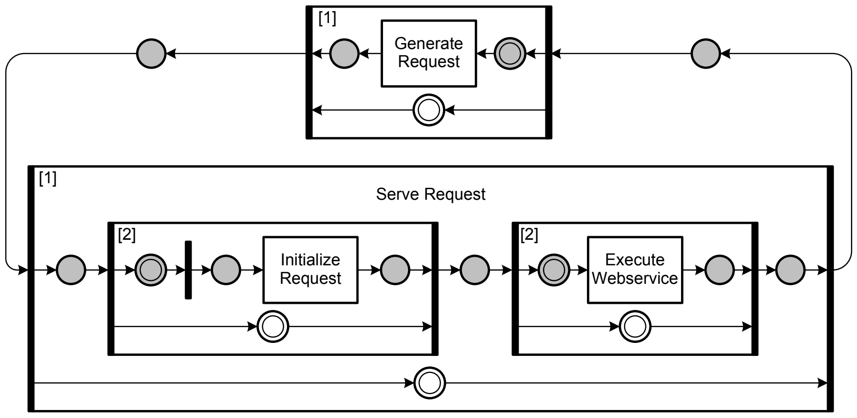
After modeling the performance parameters of the systems in the FMC-QE diagrams, different performance parameters are used as input for the FMC-QE Tableau. In this hierarchical balance sheet, based on the FMC-QE Calculus [23, 37], the performance values of the system are then derived on the assumption of stationary pro-



(a) service request structures (E/R-Diagram)



(b) static structures (Block Diagram)



(c) dynamic behavior (Petri Net)

Figure 1: FMC-QE Example

cesses. In the FMC-QE Tableau (example in table 1), the different performance values, like queue lengths or response times are calculated.

Table 1: FMC-QE Tableau

Experimental Parameters:	
n_{ges}	30
λ_{bott}	5,0000
f	0,8000
λ	4,0000

Service Request Section							Mapping		Multiplicity			Dynamic Evaluation Section						
[bb]	$SRq_i^{[bb]}$	$p_{[bb-1],i}$	$v_{i,ext}^{[bb-1]}$	$v_{i,int}^{[bb]}$	$v_i^{[bb]}$	$\lambda_i^{[bb]}$	$Server_i$	$X_{i,measured}^{[bb]}$	$m_{i,ext}^{[bb-1]}$	$m_{i,int}^{[bb]}$	$m_i^{[bb]}$	$X_{i,mpxed}^{[bb]}$	$\mu_i^{[bb]}$	$\rho_i^{[bb]}$	$n_{i,q}^{[bb]}$	$n_{i,s}^{[bb]}$	$n_i^{[bb]}$	$R_i^{[bb]}$
2	Webservice	1	1	2	2	8,0000	Webserver	1,0000	1	1	1	1,0000	1,0000		0,0000	8,0000	8,0000	1,0000
2	Request Initialization	1	1	1	1	4,0000	App. Server	0,2000	1	1	1	0,2000	5,0000	0,8000	3,2000	0,8000	4,0000	1,0000
1	Request	1	1	1	1	4,0000			1	1	1		5,0000		3,2000	8,8000	12,0000	3,0000
1	Request Generation	1	1	1	1	4,0000			1	1	1	4,5000	0,2222		0,0000	18,0000	18,0000	4,5000

Server Section				
$Server_j$	m_j	$X_j^{[1]}$	$\mu_j^{[1]}$	$\mu_j^{[1]}*m_j$
App. Server	1	0,2000	5,0000	5,0000
Webserver	∞	1,0000		

The fundamental laws applied in the calculation of the performance values in the FMC-QE Tableau are Little's Law [26] and the Forced Traffic Flow Law [19]. Little's Law is used for the calculations of values inside a hierarchical level (horizontal). The Forced Traffic Flow Law is the key to the vertical relationships among the different hierarchical levels.

A main aspect in the development of FMC-QE is the scalable applicability to complex systems. This is achieved through complexity reduction through the modeling in different views and the distinction between operational and control states [38]. But the most important aspect in the complexity reduction is the hierarchical modeling and evaluation in FMC-QE, which is the key to large systems. Related work in hierarchical quantitative modeling and the hierarchical modeling in FMC-QE is described in the following section.

3 Hierarchical Modeling and Aggregation in Quantitative Models

Hierarchical modeling is the key to complexity in the modeling of quantitative systems in FMC-QE. This section gives an overview over related work in the area of quantitative hierarchical modeling and also gives insights in the hierarchical modeling in FMC-QE.

In order to classify the different approaches and to clarify the understand of hierarchies in FMC-QE, some definitions in are given: There are two different kind of hierarchies: organizational hierarchies with super-/subordinate relations and abstraction hierarchies with compose/decompose relations [38]. In organizational hierarchies, one instance orders a service request to a hierarchically lower instance. In the abstraction hierarchy, some parts are composed to a whole. Often there is a misunderstanding between refinement and hierarchies. In the understanding of FMC-QE, there is only a hierarchy if the service requests are transformed. A refinement of a huge flat network is therefore not mandatory a abstraction hierarchy.

The decomposability in chapter 3.1 is often a hierarchical refinement, while Norton's Theorem, described in chapter 3.2 is often a flat refinement and not a hierarchical aggregation of a subnet. The models and techniques shown in chapter 3.3 could be both. The articles and books on the Forced Traffic Flow Law, summarized in chapter 3.5, are very closely related to the FMC-QE hierarchical understanding (described in chapter 3.6) and therefore the models often include a hierarchical decomposition and not only a non hierarchical refinement.

3.1 Decomposability

Simon and Ando [32] introduced decomposability and nearly-decomposable systems in order to support aggregation of variables in the analysis of large and complex systems. Completely decomposable systems [10] are systems in which state variables could be classified in groups where:

- interactions within groups can be studied as if interactions among groups did not exist and
- interactions among groups can be analyzed without referring to the interactions within groups.

This hypothesis is correct [10], but often to rigorously. As mentioned, Simon and Ando [32] introduced nearly completely decomposable systems. In the analysis of such systems, there are good approximations [10] when interactions among groups are weak compared to the interactions within groups. Simon and Ando [32] showed, that in such systems, short-run dynamics and long-run dynamics can be distinguished:

- short-run dynamics represent the interactions of the variables within each subsystem and
- long-run dynamics represent the interactions among the subsystems.

Therefore, the subsystems could be analyzed by local equilibriums and the whole system could be analyzed by aggregate variables and a global equilibrium.

Courtois [10, 11] transferred the results of Simon and Ando to queueing networks and computer performance analysis. He also introduced a hierarchy of aggregate variables in order to cope with the complexity of large systems. Through the dissection of systems into subsystems with different models, the different models could be evaluated separately and through the representation of the subsystems in aggregate variables, these systems could be analyzed at different levels. Courtois also mentioned, that through this technique and the related state-space partitioning, it is possible to analyze the different subsystems with different methods like queuing theory, simulation and deterministic models.

In the hierarchical aggregation technique of Courtois, there remains a known approximation error [12,34]. This error is of the same order of magnitude as the maximum degree of coupling between the different aggregation variables. As a result of this, the aggregation variables or systems should be well conditioned and indecomposable [12]. Courtois also introduced multilevel aggregation [12].

A special case in the aggregation of matrices are nearly-completely decomposable matrices which are also block stochastic [12]. Here, the eigenvalues of the aggregates matrices are summations of the eigenvalues of the original matrices, which yields to exact values for the steady state probabilities. Then the remaining error results only from the aggregation of stochastic matrices to subsystems.

Based on this results, Vantilborgh [34] introduced conditions under which the aggregation yields to exact results. He states and proves that the Perron-Frobenius eigenvector of an aggregated matrix is correct if, and only if the Perron-Frobenius eigenvectors of the non aggregated base matrices are subparallel [1] to the Perron-Frobenius eigenvector of the aggregated matrix. The outcome of this is, that the aggregative analysis of a system through the successive analysises of the subsystems yields exact results for all steady-state distributions if and only if the Perron-Frobenius eigenvectors of the customer behavior matrices of the subsystems are subparallel to the Perron-Frobenius eigenvector of the customer behavior matrix of the aggregated system. Vantilborgh also proves, that the equilbrium distribution of the aggregated system is equal to the conditional distributions on the servers of the subsystems connditioned on the customers served of queued at the servers, if and only if the Perron-Frobenius eigenvectorsof the subsystems are subparallel to the Perron-Frobenius eigenvector of the aggregated system.

[1]subparallel [34]: An m-element vector v is subparallel to a vector $u = [u_1u_2..u_n]$, $n > m$ if there exists a scalar k such that $v = k[u_1u_2..u_n]$

The decomposability is very interesting for the research on FMC-QE, because it shows the strengths and the assumptions of the decomposition. The results of the approximation error and the exact aggregation could be applied to the hierarchical decomposition of FMC-QE in order to show the accuracy of the hierarchical evaluation.

3.2 Norton's theorem

Norton's Theorem of the electrical circuit theory states, that in an electrical circuit of batteries and resistors, it is possible to replace a subsystem by a single current source and a parallel internal resistance with the same 'equivalent' behavior. Referring to [20] this 'current source equivalent' was published in parallel in the year 1926 by Edward Lawry Norton [30] and Hans Ferdinand Mayer [29], who called this 'Ersatzschema' (equivalent circuits). Both results are based on the studies on Hermann von Helmholz and Léon Charles Thévenin, who also were apparently unaware of the work of each other. Therefore the Norton Theorem is sometimes also called Helmholtz-Thévenin, Helmholz-Norton or Mayer-Norton Theorem [20].

Chandy, Herzog and Woo [9] adapted this results for the use in Queuing Networks in order to replace a subsystem by a single composite queue. They proved Norton's Theorem for closed Gordon-Newell Networks and showed that it is also applicable to open networks and networks which satisfy local balance. With this results, it is possible to replace a subsystem by a composite queue in order to simplify the analysis of the rest of the system. They prove, that for closed networks, the queue length distribution for a queue in an equivalent network is the same as in the given network and the queue length distribution at arrival for a queue in the equivalent network is also the same as in the given network. For open networks with exponential service times and Poisson arrivals they state, that the queue length distributions for all queues in a subsystem in the equivalent network are the same as in the given network. They also show that Norton's Theorem holds for the class of networks which satisfy local balance. Here in a closed network, the service rates at the composite queue is set equal to the throughput to the short with the same number of service requests in the short. For open networks, all uninteresting queues in the subsystem are replaced by a single composite Poisson source which generates service requests of all classes, where each class is generated independently. The generation rates are set equal to the throughput through the short of the subsystem under consideration.

Balsamo and Iazeolla [3] extended Norton's theorem for the use of any number of queues and and arbitrary interface between the subsystem and the rest of the system. They proposed a solution for BCMP-Networks [4] with one class of customers (extension to several classes is immediate [3]). They first construct a closed networks, called the *reduced network* where the servers of the subnetwork are shorted (the service times are set to zero) and the service rate of the composite queue is set to the throughput of the original network with the same number of service requests in the network. In the article they prove that the marginal probabilities for the queues and the queue length distributions at arrival on the queues are the same for the original and the equivalent network. They also describe an algorithm for the solution of the reduced network and the equivalent network.

Walrand [35] proved, that Norton's Theorem is also true for multiclass quasi reversible networks.

For FMC-QE the results of the transmission of Norton's Theorem to Queuing Theory is interesting in the development of aggregation methods in the hierarchical modeling and analysis as well as the development of new model transformations like the serial-parallel transformation.

In the related work discussed in this chapter, Norton's Theorem is used for the study of a subsystem without solving for the entire network. The idea is, to replace the uninteresting parts of the entire system by the equivalent network and then study the behavior of the rest of the system - hierarchies are not mentioned there. In contrast to this, the purpose of the Norton's Theorem in FMC-QE would not be to reduce a flat network, but to develop methods for the analysis of hierarchical networks and not for the refinement of flat networks.

3.3 Formal Hierarchies and Combination of Models

Malhotra and Trivedi [27] propose a formal expression of hierarchies in model specification and solution. In this methodology, the models are a composition of different hierarchical models. Every model is treated as a black box and in the hierarchical overall model the input and output variables are combined with the help of an interconnection matrix. Though the treatment as a black box and the output and input connection, many different types of sub models are possible. If parameters are passed from model M_i to model M_j, an order $\succ$ is defined. The relation $\succ\succ$ further defines the transitive closure between the models. If the overall model is acyclic (If $M_i \succ\succ M_j$ then not $M_j \succ\succ M_i$) the solution of the overall model is achieved through the solving of the model from the most-inner sub model to the overall model. If the overall model is non-acyclic, an iterative solution is possible for the solution of the overall model.

Bause and Buchholz [6] use aggregation and disaggregation in their Product Form Queueing Petri Nets [5]. Their results for the exact aggregation are based on Norton's Theorem.

Balbo, Bruell and Ghanta [2] propose a technique in which queuing network models and Generalized Stochastic Petri Nets are combined in a hierarchical modeling approach. In their approach, the hierarchical subsystems which doesn't violate the BCMP-Theorem [4] are modeled as Product Form Queueing Networks (PFQN). Subsystems, which violate the BCMP assumptions, like systems with synchronization of processes or simultaneous possession of resources by a process, are modeled as Generalized Stochastic Petri Nets (GSPN [28]). Later the different results are combined to an overall solution. Through this combination, it is avoided, that the whole system is modeled as a GSPN model, which results in a state space reduction. In this approach every subsystem is evaluated in isolation and the results are then aggregated. In their paper they refer to the decomposability approach of Courtois [10,11] and the approximation error of this approach. They [2] state, that through the solving of the single sub models in isolation with the appropriate exact model, only the aggregation would lead to an approximative error.

The formal hierarchies are closely related to FMC-QE are therefore very interesting. Also the combination of models could be used to solve special problems, like the semaphore synchronization, described in chapter 5.3.

3.4 Aggregation and Hierarchies in Time Augmented Petri Nets

Bucci and Vicario [7] propose Communication Time Petri Nets (CmTPN). CmTPNs augment the basic model of Petri Nets with inhibitor arcs, the timing constraints of Time Petri Nets, and a module construct which permits the decomposition of a complex model into smaller sub models [7]. This module construct consists of *writing* and *reading ports* connected with *communication links* with transitions and places that are referred as *writing transitions* and *reading places*. Writing and reading ports of the different sub models can then be connected through *channels* to support the communication and interaction between the different sub models. The analysis of such models consists of two steps: *unit analysis* and *integration analysis*. In the unit analysis, the different subsystems are analyzed. The different sub models are separated from each other and communicate only through interfaces. Through that interfaces, the environment of each sub model is defined. Then every sub model is analyzed similar to a Time Augmented Petri Net. In the integration analysis, the whole model is computed, by aggregating the results of the different subsystems.

Haddad and Moreaux [17] combine aggregation and decomposition in order to evaluate Petri Nets. In their understanding, aggregation reduces the state space by grouping states and solving the corresponding Markov chain on the set of state classes. In the decomposition method they follow the decomposition of Plateau and Fourneau ([31] referring to [17]), in which the state space is a cartesian product of smaller state spaces. In their article, they also propose the concept of *internal* and *external synchronization*, where internal synchronization means synchronization within one class and external synchronization means synchronization between several classes.

Buchholz [8] proposes the hierarchical structuring of Superposed Generalized Stochastic Petri Nets (SGSPNs). In his work, he presents a technique in which a hierarchical structure is generated in a preprocessing step in order to compute a compact generator matrix. Through this approach, the problem of unreachable states is avoided. In the approach, *macro states* and the generation of a *macro transition system* is proposed. Macro states combine states of a component state space and a set of macro states define a partition of the component state space. Each macro state represent a set of detailed states. In the macro transition system, the state transitions of one macro state to another (not locally) are defined. Through this macro transition system a *global reachability analysis* is performed. The unreachablility of a global macro state implies the unreachability of a detailed state, which allows the exclusion of unreachable states.

Freiheit and Zimmermann [14] propose a methodology for the automatic decomposition of models. Their results are based on the decomposition of Stochastic Petri Nets but could be extended to other modeling techniques. Their "‘divide and conquer approach"’ is divided in three steps: In the first step, the system is decomposed into smaller subsystems. In the second step, the dependencies between the different subsystems are derived and extracted in "‘low-level subsystems"’ and an aggregative "‘basic skeleton"’. In the third step, the performance values of the system are computed by an iterative approximation method.

3.5 Forced Traffic Flow Law

The Forced Traffic Flow Law (also called Forced Flow Law) [13,25] is one of the main laws in FMC-QE. Though the Forced Traffic Flow Law, the hierarchical service request transformation of the server requests into subrequests is possible, which enables the hierarchical decomposition in FMC-QE. In the Forced Traffic Flow Law, an arrival rate $\lambda_{sup(i)}^{[bb-1]}$ on hierarchical level $[bb-1]$ will be transformed into an arrival rate $\lambda_i^{[bb]}$ on the hierarchical level $[bb]$ through the use of a traffic flow coefficient $v_i^{[bb]}$:

$$\lambda_i^{[bb]} = v_i^{[bb]} * \lambda_{sup(i)}^{[bb-1]} \tag{1}$$

The results of Denning and Buzen [13] are closely related to FMC-QE. Like in FMC-QE, the Forced Traffic Flow Law is used for the hierarchical decomposition of service requests in subrequests. They also define units for the service request in the definition of rates like service rates. In their two level models the overall service request is called ‘job’ and the sub requests are called ‘request’. In their definition, the traffic flow coefficient is called visit ratio and expresses the mean number of requests per job for a device.

Lazowska et al. [25] described the specialization of the term ‘request’ on different levels of details. For example a request at a disk is called disc access and a request at the level of the entire system could be defined as a user-level interaction.

In Haas and Zorn [16], the Forced Traffic Flow Law (there: ‘Verkehrsflussgesetz’) is used for the decomposition of a system into subcomponents. Through the decomposition, the overall request is departed into subrequests at every component through the visit ratio V_i and the throughput D_i at every component could then be expressed as $D_i = V_i * D$. They also define a distinction of the different throughputs for every service request class.

3.6 FMC-QE

In FMC-QE, the systems are modeled from the perspective of the hierarchical service request [38]. On the basis of the three FMC [24,33] diagram types - Entity/Relationship-Diagram, Block Diagram and Petri Net, the quantitative system properties are modeled and evaluated.

The hierarchies of the service request structures are the main focus in FMC-QE, because the service request in the origin of every service provisioning process [38] and the hierarchical decomposition is the key to cope with complex systems. The systems are modeled from three different hierarchical views: the service request structrues, the (logical) server structures and the dynamic control flow behavior. Furthermore the service requests are strictly modeled as a tuple of $\{Value\}$ und $[Unit]$ ($SRq_i = \{SRq_i\}\,[SRq_i]$) in order to make the hierarchical decomposition possible through the service request transformation on the hierarchical borders of the model. On the basis of clerical conventions, SRq_i equals N_i and therefore:

$$N_i = \{N_i\} [N_i] \tag{2}$$

The fundamental laws of the quantitative evaluation used in FMC-QE are Little's Law [26]:

$$N_i^{[bb]} = \lambda_i^{[bb]} * R_i^{[bb]} \tag{3}$$

and the Forced Traffic Flow Law) [19]:

$$\lambda_i^{[bb]} = v_i^{[bb]} * \lambda_{sup(i)}^{[bb-1]} \tag{4}$$

In the scope of hierarchies, Little's Law defines relations inside an hierarchical level $[bb]$ (horizontal). According to Little's law, the number of service requests i on the hierarchical level $[bb]$ is a product of the arrival rate $\lambda_i^{[bb]}$ of service requests N_i per time unit and the response time $R_i^{[bb]}$.

The Forced Traffic Flow Law defines inter-hierarchical relations and defines therefore vertical relations between the hierarchical levels. It is the key to the hierarchical modeling in FMC-QE. The Forced Traffic Flow Law defines, that an arrival rate $\lambda_{sup(i)}^{[bb-1]}$ of service requests $N_{sup(i)}$ per time unit on hierarchical level $[bb-1]$ will be transformed into an arrival rate $\lambda_i^{[bb]}$ of service requests N_i on the hierarchical level $[bb]$. This transformation is done with the help of the traffic flow coefficient $v_i^{[bb]}$. As already mentioned, in FMC-QE, the service requests are modeled as a tuple of $\{Value\}$ und $[Unit]$. So the Traffic Flow Coefficient is not only a scalar of $\{Value\}$ but also the basis for the service request transformation from the unit service request $N_{sup(i)}^{e[bb-1]}$ to a number of unit service requests $N_i^{e[bb]}$:

$$v_i^{[bb]} = \left\{v_i^{[bb]}\right\} \left[v_i^{[bb]}\right] = \left\{v_i^{[bb]}\right\} \left[\frac{N_i^{e[bb]}}{N_{sup(i)}^{e[bb-1]}}\right] = \left\{v_i^{[bb]}\right\} \frac{\left[N_i^{e[bb]}\right]}{\left[N_{sup(i)}^{e[bb-1]}\right]} \tag{5}$$

Here the service requests $N_{sup(i)}^{e[bb-1]}$ on the hierarchical level $[bb-1]$ are on superordinate $(sup(i))$ of the service request $N_i^{e[bb]}$ on the hierarchical level $[bb]$.

4 Model Transformations

In FMC-QE, the system is modeled from the perspective of the hierarchical service request structures, processed by the system. In order to evaluate the model in the FMC-QE Tableau, the model must follow a tree shaped structure. If the modeled system does not follow this assumption, due to modeling with a different modeling technique or special system behavior, the model is transformed into a tree shape structure. After the transformation, the behavior of the transformed model is equivalent to the behavior of the original model, but analyzable with the FMC-QE Tableau.

In the following chapters, some model transformations are defined. Chapter 4.1 describes the transformation of a while loop and chapter 4.2 defines the transformation of a feed backward loop. Finally some open questions are raised in chapter 4.3.

4.1 While Loop

In FMC-QE, a while loop is transformed in order to compute the performance values of the model in the FMC-QE Tableau. Figure 2 shows an FMC-QE model of a while loop. The loop body is represented by the execution of the service request SRq_i.

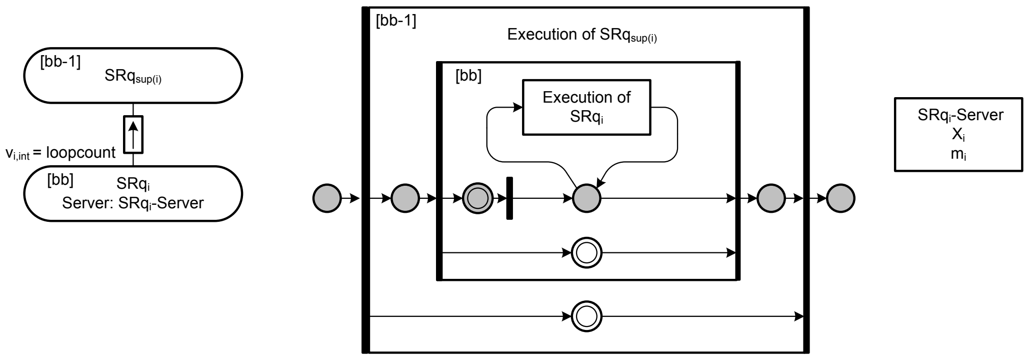


Figure 2: Original While Loop

In the service request structures, the loop is represented by a hierarchy of service requests. The service request $SRq_{sup(i)}$ represents the whole execution of all iterations, while the service request SRq_i represents one single loop body execution. The traffic flow coefficient $v_{i,int}$ represents the mean number of loop iterations. In the Petri Net, the inner service request loop represents the while loop and the corresponding real server has the multiplicity m_i and the Service time X_i.

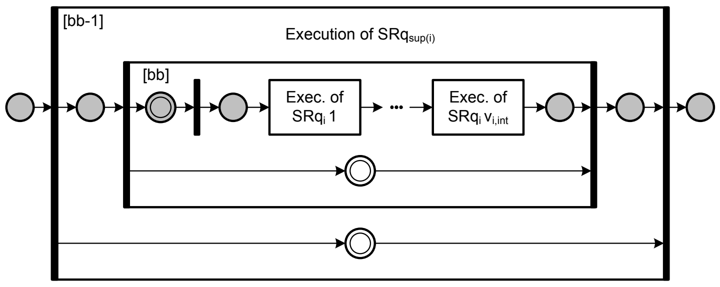


Figure 3: While Loop Transformation (Serialization)

In the first step of the transformation, the loop is transformed to a sequence of executions of the loop body (SRq_i), like shown in figure 3.

Then the single executions of the loop body are combined to one execution in which, the service time is the sum of the single service times. This step is shown in figure 4. In this transformation, the arrival rate of the service requests SRq_i is not changed:

$$\lambda_i'^{[bb]} = \lambda_i^{[bb]} \tag{6}$$

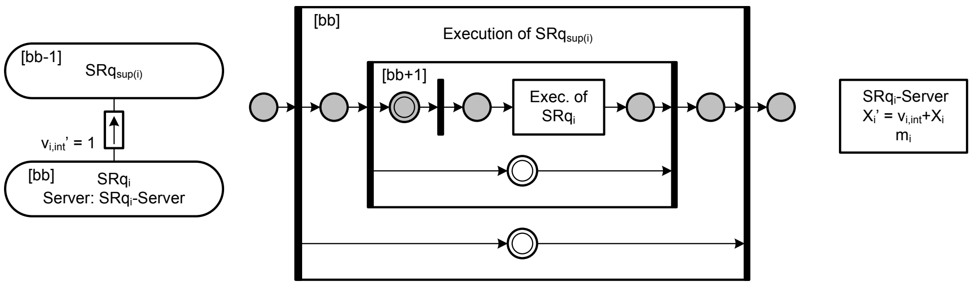


Figure 4: While Loop Transformation (Combination)

But the service time is the sum of the single executions of the loop bodys, respectively the product of the mean loopcount $v_{i,int}$ and the original service time X_i:

$$X_i^{'\,[bb]} = v_{i,int} * X_i^{[bb]} \tag{7}$$

4.2 Feed Backward Loop

In FMC-QE, feed backward loops, like shown in figure 6 are transformed into a feed forward execution.

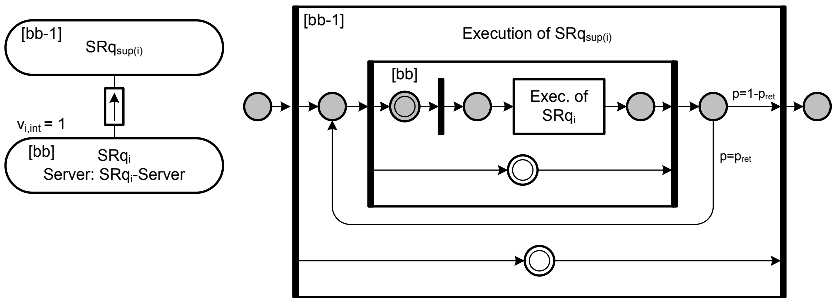


Figure 5: Feed Backward Loop

In the feed backward execution there is a probability p_{ret}, that the execution service request SRq_i has to be repeated (e.g. failure).

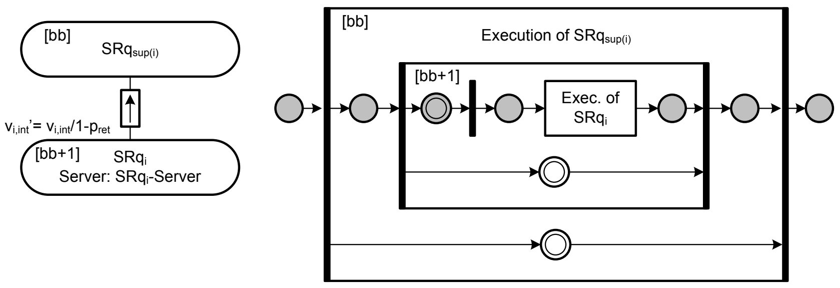


Figure 6: Feed Forward

In the feed backward - feed forward transformation not the service time X_i but the arrival rate λ_i is transformed. The increased arrival rate corresponds better to the real system than the extension of the service rate. The new arrival rate λ_i' is the product of the traffic flow coefficient $v_{i,int}$ and the old arrival rate λ_i:

$$\lambda_i'^{[bb]} = v_{i,int}'^{[bb]} * \lambda_i^{[bb]} \tag{8}$$

The traffic flow coefficient is calculated due to the formula of the geometric sum [15]:

$$s = \sum_{n=1}^{\infty} a_1 q^{n-1} = \frac{a_1}{1-q} \tag{9}$$

as followed:

$$v_{i,int}'^{[bb]} = \frac{v_{i,int}^{[bb]}}{1 - p_{return}} \tag{10}$$

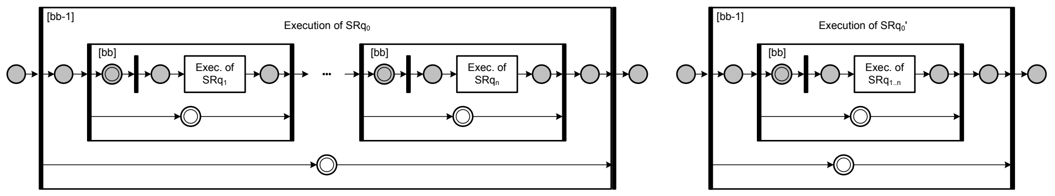
In comparison to the transformation of the while loop, the service time X is not transformed:

$$X_i'^{[bb]} = X_i^{[bb]} \tag{11}$$

4.3 Open Questions

In the following, some other interesting model transformations are shown, which are under research.

Serial to Single Station A transformation of a sequence of n service requests $SRq_1^{[bb]}$ .. $SRq_n^{[bb]}$ to an equivalent single service request $SRq_{1..n}^{[bb]}$ would help in reducing huge flat models into smaller aggregated ones. In comparison to a hierarchical serial execution, the flat structure is combined on the same hierarchical level $[bb]$ and this is not a service request transformation to a hierarchically higher level. The corresponding FMC-QE Petri Nets are shown in figure 7 (in an equivalent transformation $SRq_0^{[bb-1]} = SRq_n'^{[bb-1]}$).



(a) Serial (b) Single Station

Figure 7: Serial to Single Station Transformation

The first results are, that the service time of the service request $SRq_{1..n}^{[bb]}$ is the sum of all service times $SRq_1^{[bb]}$ .. $SRq_n^{[bb]}$:

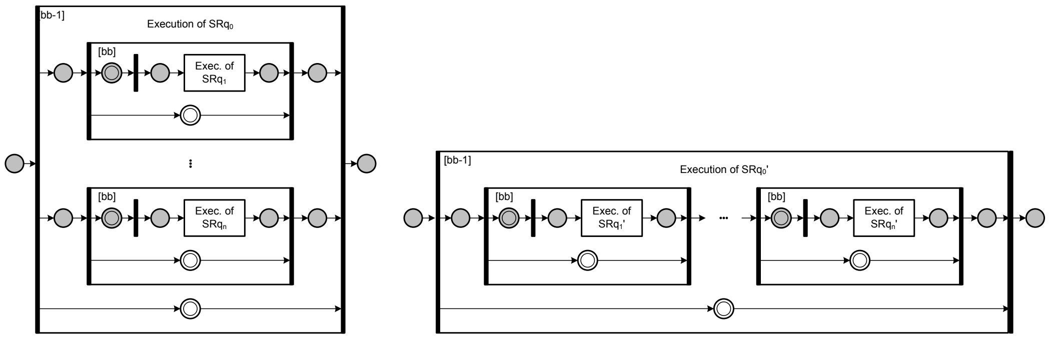
$$X_{1..n}^{[bb]} = \sum_{i=1}^{n} X_i^{[bb]} \tag{12}$$

The service rate is the minimum of all service rates:

$$\mu_{1..n}^{[bb]} = min(\mu_1^{[bb]}..\mu_n^{[bb]}) \tag{13}$$

The definition of the multiplicity of the combined service request server $m_{1..n}^{[bb]}$ is not yet defined. Also the hadling of different traffic flow coefficients $v_i^{[bb]}$ is not yet defined.

Parallel to Serial Another possible transformation, derivated from the electrical circuit theory, is the transformation of a parallel execution into an equivalent serial execution, like shown in figure 8. Here n service requests $SRq_1^{[bb]}$ .. $SRq_n^{[bb]}$ are transformed to the service requests $SRq_1^{'[bb]}$ .. $SRq_n^{'[bb]}$ and $SRq_0^{[bb-1]}$ is the hierarchical superordinated service request (in an equivalent transformation $SRq_0^{'[bb-1]} = SRq_n^{[bb-1]}$).



(a) Parallel (b) Serial

Figure 8: Parallel to Serial Transformation

The service rate of the parallel execution should be the same as the service rate in the serial execution:

$$\mu_0^{'[bb-1]} = min(\mu_1^{'[bb]}..\mu_n^{['bb]}) = min(\mu_1^{[bb]}..\mu_n^{[bb]}) = \mu_0^{[bb-1]} \tag{14}$$

In the parallel execution, the service time of the whole execution is the maximum of all service requests:

$$X_0^{[bb-1]} = max(X_1^{[bb]}, .., X_n^{[bb]}) \tag{15}$$

In the serial execution, the service time of the whole execution is the sum of all service requests:

$$X_0^{'[bb-1]} = \sum_{i=1}^{n} X_i^{['bb]} \tag{16}$$

If the transformation is equivalent then:

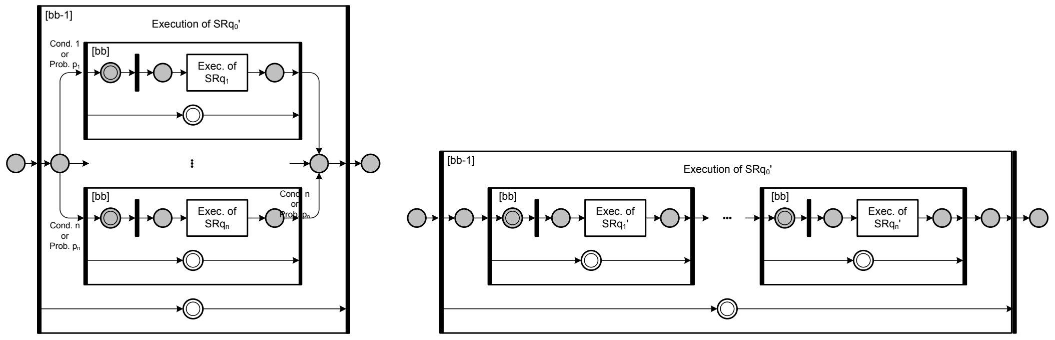
$$X_0^{'[bb-1]} = X_0^{[bb-1]} \tag{17}$$

So:

$$X_0'^{[bb-1]} = \sum_{i=1}^{n} X_i'^{[bb]} = max(X_1^{[bb]}, .., X_n^{[bb]}) = X_0^{[bb-1]} \quad (18)$$

Which would reduce the execution all service requests executed in serial to the longest executed service request in the parallel case.

Branch to Serial Another open question is the transformation of a branch to a serial execution, like shown in figure 9.



(a) Branch (b) Serial

Figure 9: Branch to Serial

In this transformation, the single service times of the transformed execution should be weighted, but it remains the same problem as in the parallel serial transformation, that the execution collapses to the longest service request.

5 Modeling Rules

In the following section, some special modeling rules are described. In chapter 5.1 the modeling of multiplexers (one real server for multiple logical servers) are defined. The modeling of a multiclass senario is descibed in chapter 5.2. In chapter 5.3 some open modeling questions are raised.

5.1 Multiplex

The mapping of several logical servers to one or more real servers very common. An example for this scenario are several threads mapped to a processor. The multiplexer is the key to model and evaluate such scenarios in FMC-QE. A multiplexer in signal transmission is shown in figure 10.

Figure 11 shows exemplary FMC-QE hierarchical structures (service request, static and dynamic). In this example, a service request *A* is decomposed into service request *A.1* and *A.2*. *A.2* is further decomposed into service request *A.2.1* and *A.2.2*. In this example, the logical server *Request A.1 Server* and the logical server *Request A.2.1*

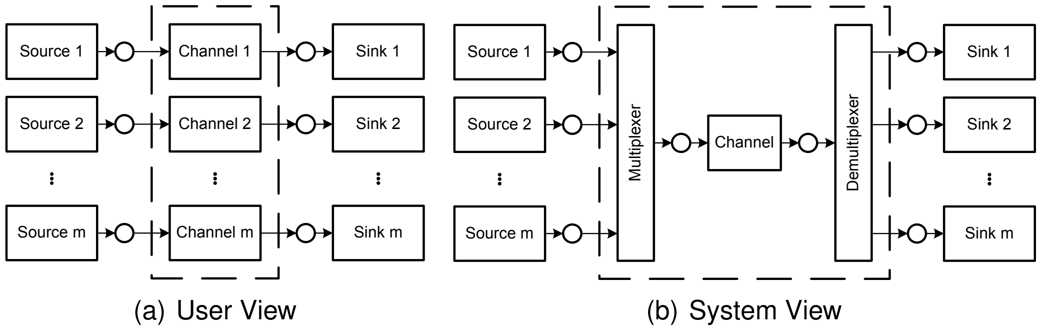


Figure 10: Multiplexer/Demultiplexer [36]

Server share one real server called *Server X*. This could be seen in the block diagram in figure 11(b).

In the diagram of the real servers in figure 12 it could be seen, that *Server X* has the multiplicity $m = 1$.

In order to calculate the performance values of the modeled system in the FMC-QE Tableau, a multiplexed service time $X_{i,mpxed}$ must be calculated for every logical server. Referring to the last report [23] the calculation of the multiplexed service time is as followed:

The service time ($X_{i,mpxed}^{[bb]}$) a service request received from a server depends on the number of all logical servers handled by a real server ($\sum_{\forall\, SRq_i\ of\ Server_j} m_i$) and the multiplicity of the real server (m_j). If $\sum_{\forall\, SRq_i\ of\ Server_j} m_i \leq m_j$ then, every logical server is handled by a dedicated real server and the measured service time and the multiplexed service time are the same ($X_{i,mpxed}^{[bb]} = X_{i,measured}^{[bb]}$). In this case the redundant real server would idle. If $\sum_{\forall\, SRq_i\ of\ Server_j} m_i > m_j$, every service request receives a longer service time, because the server or the servers are handling other service request types in parallel. The received service time is then calculated as: $X_{i,measured} * \frac{X_j}{X_{i,measured}^{[bb]} * v_i^{[bb]}} * \frac{m_i}{m_j}$. In the special case infinite servers (both logical and real are infinite servers) $X_{i,mpxed}$ is defined like $\sum_{\forall\, SRq_i\ of\ Server_j} m_i = m_j$, $X_{i,mpxed}^{[bb]} = X_{i,measured}^{[bb]}$. To summarize the paragraph:

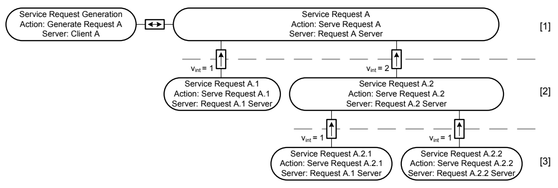
$$X_{i,mpxed}^{[bb]} = \begin{cases} X_{i,measured}^{[bb]}\ if\ \left(\sum_{\forall\, SRq_i\ of\ Server_j} m_i\right) \leq m_j \\ X_{i,measured} * \frac{X_j}{X_{i,measured}^{[bb]} * v_i^{[bb]}} * \frac{m_i}{m_j}\ else \end{cases} \tag{19}$$

While X_j is the service time a server needs to served the its parts of the top level service request. It is defined as the normalized sum of all measured corresponding service times.

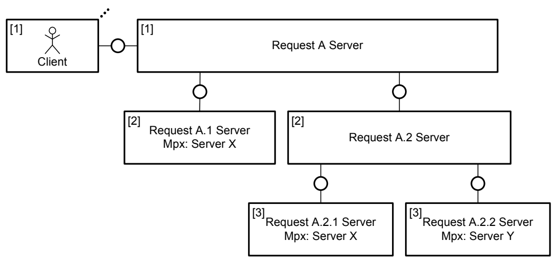
$$X_j = \sum_{\forall mpxed\, SRq_i} \left(X_{i,measured}^{[bb]} * v_i^{[bb]} \right) \tag{20}$$

For the rest of the calculated values, there is no distinction between multiplexed servers and non-multiplexed (dedicated) servers.

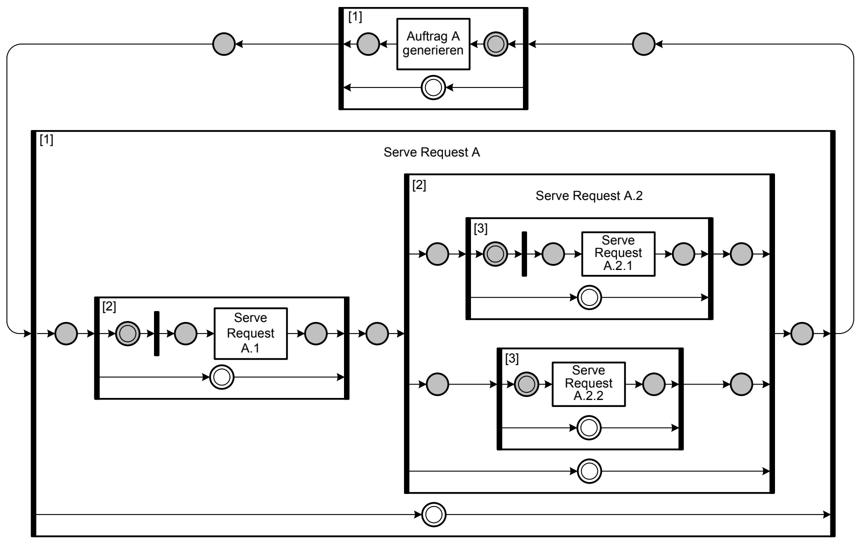
The corresponding FMC-QE Tableau of the example in shown in table 2.


Service Request Generation
Action: Generate Request A
Server: Client A
Service Request A
Action: Serve Request A
Server: Request A Server
[1]
v_{int} = 1
v_{int} = 2
Service Request A.1
Action: Serve Request A.1
Server: Request A.1 Server
Service Request A.2
Action: Serve Request A.2
Server: Request A.2 Server
[2]
v_{int} = 1
v_{int} = 1
Service Request A.2.1
Action: Serve Request A.2.1
Server: Request A.1 Server
Service Request A.2.2
Action: Serve Request A.2.2
Server: Request A.2.2 Server
[3]

(a) service request structures



(b) static structures



(c) dynamic behavior

Figure 11: FMC-QE Multiplex Example - Logical Structures

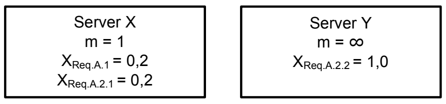


Figure 12: FMC-QE Multiplex Example - Real Servers

Table 2: FMCQE Multiplex Example - Tableau

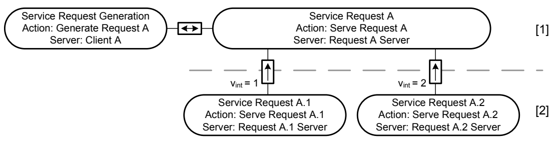
Experimental Parameters:	
n_{ges}	**30**
λ_{bott}	**1,6667**
f	**0,8000**
λ	**1,3333**

Service Request Section							Mapping		Multiplicity			Dynamic Evaluation Section						
[bb]	$SRq_i^{[bb]}$	$p_{[bb-1],i}$	$v_{i,ext}^{[bb-1]}$	$v_{i,int}^{[bb]}$	$v_i^{[bb]}$	$\lambda_i^{[bb]}$	$Server_i$	$X_{i,measured}^{[bb]}$	$m_{i,ext}^{[bb-1]}$	$m_{i,int}^{[bb]}$	$m_i^{[bb]}$	$X_{i,mpxed}^{[bb]}$	$\mu_i^{[bb]}$	$\rho_i^{[bb]}$	$n_{i,q}^{[bb]}$	$n_{i,s}^{[bb]}$	$n_i^{[bb]}$	$R_i^{[bb]}$
3	Service Request A.2.2	1	2	1	2	2,6667	Server Y	1,0000	1	1	1	1,0000	1,0000	2,6667	0,0000	2,6667	2,6667	1,0000
3	Service Request A.2.1	1	2	1	2	2,6667	Server X	0,2000	1	1	1	0,3000	3,3333	0,8000	3,2000	0,8000	4,0000	1,5000
2	Service Request A.2	1	1	2	2	2,6667			1	1	1		1,0000		0,0000	2,6667	2,6667	1,0000
2	Service Request A.1	1	1	1	1	1,3333	Server X	0,2000	1	1	1	0,6000	1,6667	0,8000	3,2000	0,8000	4,0000	3,0000
1	Service Request A	1	1	1	1	1,3333			1	1	1		1,0000		3,2000	3,4667	6,6667	5,0000
1	Generate Request A	1	1	1	1	1,3333			1	1	1	17,5000	0,0571		0,0000	23,3333	23,3333	17,5000

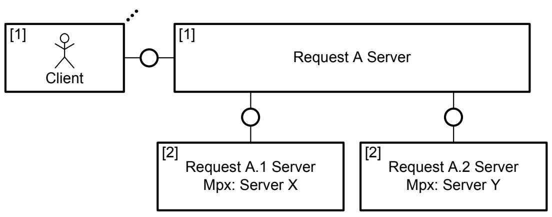
Server Section				
$Server_j$	m_j	$X_j^{[1]}$	$\mu_j^{[1]}$	$\mu_j^{[1]}*m_j$
Server X	1	0,6000	1,6667	1,6667
Server Y	∞			

5.2 Multiclass

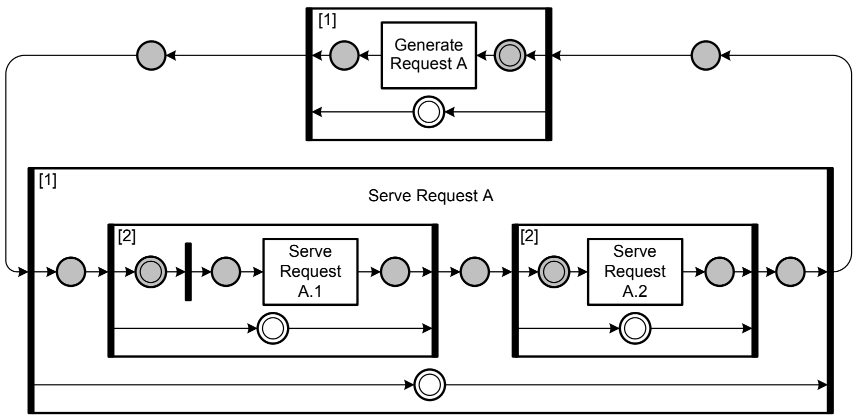
If there are different kinds of service requests and it is not reasonable to combine the different arrivals so a single class of service requests, multiclass models are the way of modeling such systems. In FMC-QE, the multiclass scenario is just an extension of the multiplex case. In multiclass models, the logical structures differ from each other, so there is a different model for every class. In figure 13 the service request structures, the logical server structures and the dynamic behavior of service request class *A* is shown.


Service Request Generation
Action: Generate Request A
Server: Client A
Service Request A
Action: Serve Request A
Server: Request A Server
[1]
v_{int} = 1
v_{int} = 2
Service Request A.1
Action: Serve Request A.1
Server: Request A.1 Server
Service Request A.2
Action: Serve Request A.2
Server: Request A.2 Server
[2]

(a) service request structures



(b) static structures



(c) dynamic behavior

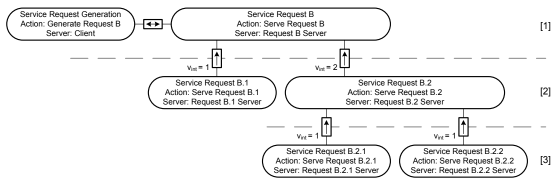
Figure 13: FMC-QE Multiclass Example - Class A

In class *A*, a service request *A* is decomposed into a serial execution for two service requests *A.1* and *A.2*. *A.1* is handled by server *X* and *A.2* is handled by server *Y*.

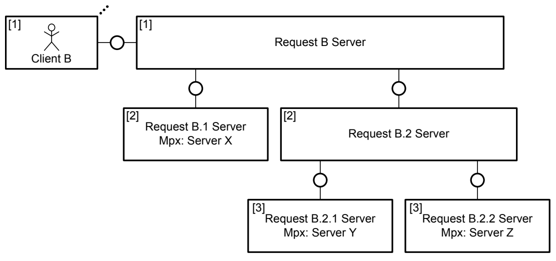
In the other class *B* (described in figure 14), there is also a decomposition into two service requests *B.1* and *B.2*. The service request *B.2* is furthermore decomposed into two parallel executed service requests *B.2.1* and *B.2.2*. The logical server is mapped to the real server *X*, is mapped to *Y* and is mapped to *Z*.

In addition to the logical structures of the two classes *A* and *B* in the figures 13 and 14, the real servers are defined in figure 15. There is a multiplex of server *X* for the logical servers *Request A.1 Server* and *Request B.1 Server* and a multiplex of server *Y* for the logical servers *Request A.1 Server* and *Request B.2.1 Server*. The server *Z* is a dedicated server for the service request *B.2.2* in class *B*.

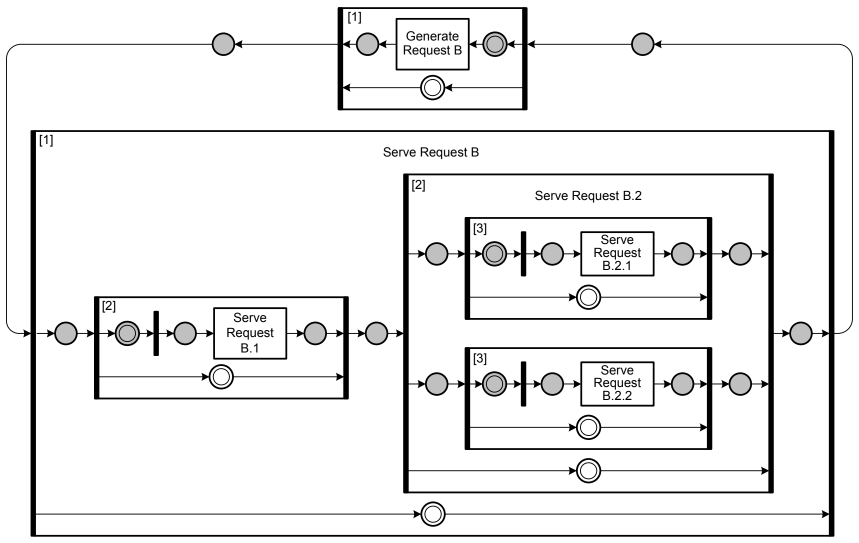
The corresponding tableau of of the multiclass example is shown in table 3. In comparison to a non multiclass model, this table consists of more than one experimental parameters and service request section.


Service Request Generation
Action: Generate Request B
Server: Client
Service Request B
Action: Serve Request B
Server: Request B Server
[1]
$v_{int} = 1$
$v_{int} = 2$
Service Request B.1
Action: Serve Request B.1
Server: Request B.1 Server
Service Request B.2
Action: Serve Request B.2
Server: Request B.2 Server
[2]
$v_{int} = 1$
$v_{int} = 1$
Service Request B.2.1
Action: Serve Request B.2.1
Server: Request B.2.1 Server
Service Request B.2.2
Action: Serve Request B.2.2
Server: Request B.2.2 Server
[3]

(a) service request structures



(b) static structures



(c) dynamic behavior

Figure 14: FMC-QE Multiclass Example - Class B

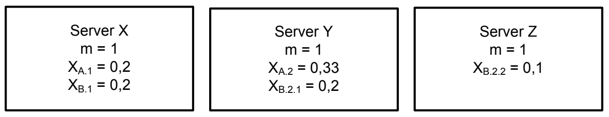


Figure 15: FMC-QE Multiclass Example - Real Servers

Table 3: FMC-QE Multiclass Example - Tableau

Experimental Parameters:	
n_{ges}	**30**
λ_{bott}	**0,9434**
f	**0,8000**
λ	**0,7547**

Service Request Section							Mapping		Multiplicity			Dynamic Evaluation Section						
[bb]	$SRq_i^{[bb]}$	$p_{[bb-1],i}$	$v_{i,ext}^{[bb-1]}$	$v_{i,int}^{[bb]}$	$v_i^{[bb]}$	$\lambda_i^{[bb]}$	$Server_i$	$X_{i,measured}^{[bb]}$	$m_{i,ext}^{[bb-1]}$	$m_{i,int}^{[bb]}$	$m_i^{[bb]}$	$X_{i,mpxed}^{[bb]}$	$\mu_i^{[bb]}$	$\rho_i^{[bb]}$	$n_{i,q}^{[bb]}$	$n_{i,s}^{[bb]}$	$n_i^{[bb]}$	$R_i^{[bb]}$
2	Service Request A.2	1	1	2	2	1,5094	Server Y	0,3300	1	1	1	0,5300	1,8868	0,8000	0,0000	0,8000	0,8000	0,5300
2	Service Request A.1	1	1	1	1	0,7547	Server X	0,2000	1	1	1	0,4000	2,5000	0,3019	0,1305	0,3019	0,4324	0,5730
1	Service Request A	1	1	1	1	0,7547			1	1	1		2,5000		0,1305	1,1019	1,2324	1,6330
1	Generate Request A	1	1	1	1	0,7547			1	1	1	38,1170	0,0262			28,7676	28,7676	38,1170

Experimental Parameters:	
n_{ges}	**30**
λ_{bott}	**0,9434**
f	**0,8000**
λ	**0,7547**

Service Request Section							Mapping		Multiplicity			Dynamic Evaluation Section						
[bb]	$SRq_i^{[bb]}$	$p_{[bb-1],i}$	$v_{i,ext}^{[bb-1]}$	$v_{i,int}^{[bb]}$	$v_i^{[bb]}$	$\lambda_i^{[bb]}$	$Server_i$	$X_{i,measured}^{[bb]}$	$m_{i,ext}^{[bb-1]}$	$m_{i,int}^{[bb]}$	$m_i^{[bb]}$	$X_{i,mpxed}^{[bb]}$	$\mu_i^{[bb]}$	$\rho_i^{[bb]}$	$n_{i,q}^{[bb]}$	$n_{i,s}^{[bb]}$	$n_i^{[bb]}$	$R_i^{[bb]}$
3	Service Request B.2.2	1	2	2	4	3,0189	Server Z	0,1000	1	1	1	0,1000	10,0000	0,3019	0,0000	0,3019	0,3019	0,1000
3	Service Request B.2.1	1	2	1	2	1,5094	Server Y	0,2000	1	1	1	0,5300	1,8868	0,8000	3,2000	0,8000	4,0000	2,6500
2	Service Request B.2	1	1	2	2	1,5094			1	1	1		10,0000		0,0000	0,1509	0,1509	0,1000
2	Service Request B.1	1	1	1	1	0,7547	Server X	0,2000	1	1	1	0,4000	2,5000	0,3019	0,1305	0,3019	0,4324	0,5730
1	Service Request B	1	1	1	1	0,7547			1	1	1		2,5000		0,1305	0,4528	0,5834	0,7730
1	Generate Request B	1	1	1	1	0,7547			1	1	1	38,9770	0,0257			29,4166	29,4166	38,9770

Server Section				
$Server_j$	m_j	$X_j^{[1]}$	$\mu_j^{[1]}$	$\mu_j^{[1]}*m_j$
Server X	1	0,4000	2,5000	2,5000
Server Y	1	1,0600	0,9434	0,9434
Server Z	1	0,4000	2,5000	2,5000

5.3 Open Questions

Other interesting modeling and evaluation questions are concurrent processes and the quantitative modeling of exception handling.

Concurrent Processes The usage of FMC-QE in the performance modeling and evaluation of concurrent processes and synchronization would be an interesting research question.

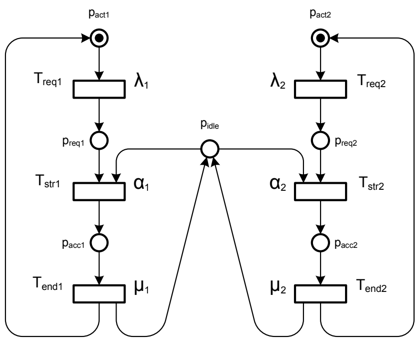


Figure 16: Semaphore Synchronization [28]

There were attempts to model and analyze concurrent processes, more precisely a semaphore synchronization, with FMC-QE. A model and a tableau of the semaphore synchronization shown in figure 16 was already set up. Parts of this model are shown in figure 17 and figure 18.

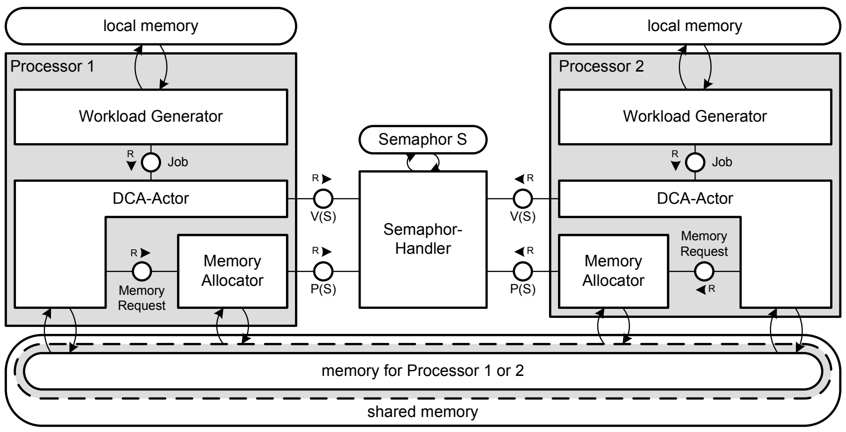


Figure 17: Semaphore Synchronization - Server Structures (FMC)

There is also an FMC-QE Tableau with an iterative approximative solution in use [39]. In further research, other concurrent processes will be modeled and evaluated. Also other iterative solutions like [18] will be investigated.

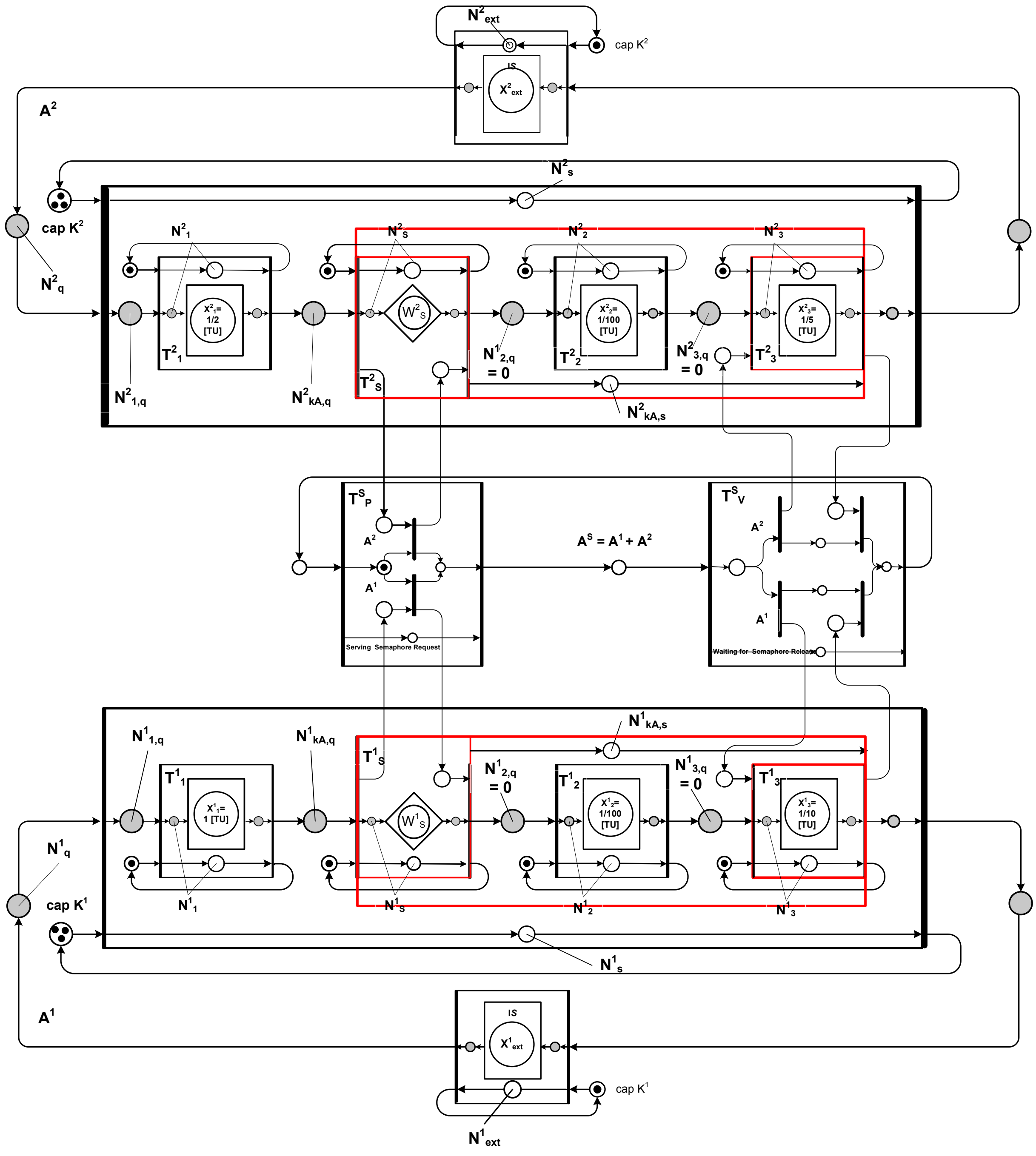


Figure 18: Semaphore Synchronization [39]

Exception Handling Often it is useful to neglect the exception handling in the modeling of systems, because in makes the models much more complex for the (hopefully) rare events of exceptions. But, if the exception handling would be relevant for the quantitative evaluation of the system, a corresponding model should be established. In first ideas, the FMC-QE model would follow the FMC modeling styles, like shown in figure 19.

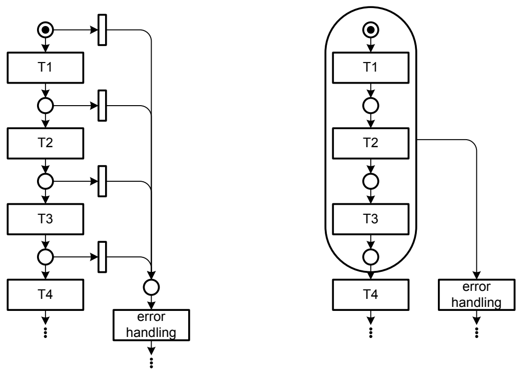


Figure 19: Exeption Modeling in FMC [1]

But here are open questions like: How to cope with exception handling over many hierarchy levels and multiple *throw* statements and so on.

6 Conclusions and Outlook

The hierarchical modeling in FMC-QE, described in section 3 is the key to model complex systems. Though the hierarchical service request structures and the service request modeled as a tuple of value and unit, the quantitative evaluation of the models in the FMC-QE Tableau is scalable to complex systems, which were unaccessible to quantitative evaluation before. The related work on this field gives also basics and ideas for further research and concretions.

The section model transformations 4 extend the applicability of FMC-QE to the modeling and evaluation of scenarios which does not follow the FMC-QE hierarchical tree shaped model.

The multiplex modeling and the possibility of modeling multiclass scenarios, described in section 5: modeling rules, open the availability of FMC-QE for the modeling and evaluation of another wide range of systems. Also the considerations of how to model exception handling, addresses interesting research questions.

After the definition and description of FMC-QE in the last three reports, including basic definitions, diagram types, case studies, the calculus as well as hierarchical modeling, model transformations and modeling rules in this report, the next research will be focused using FMC-QE in order to model Product Form and Non Product From Queueing Networks in order to find a clear and simple criterion in the distinction between these types of networks. Also the open questions raised in this report will be addressed.

References

[1] Rémy Apfelbacher, Andreas Knöpfel, Peter Aschenbrenner, and Sebastian Preetz. FMC Visualization Guidelines. Technical report, Hasso-Plattner-Institute, Potsdam, January 2005.

[2] G. Balbo, S. C. Bruell, and S. Ghanta. Combining Queueing Networks and Generalized Stochastic Petri Nets for the Solution of Complex Models of System Behavior. *IEEE Transactions on Computers*, 37(10):1251–1268, October 1988.

[3] S. Balsamo and G. Iazeolla. An Extension of Norton's Theorem for Queueing Networks. *IEEE Transactions on Software Engineering*, 8(4):298–305, July 1982.

[4] Forest Baskett, K. Mani Chandy, Richard R. Muntz, and Fernando G. Palacios. Open, Closed, and Mixed Networks of Queues with Different Classes of Customers. *Journal of the ACM*, 22(2):248–260, 1975.

[5] Falko Bause and Peter Buchholz. Product Form Queueing Petri Nets: A Combination of Product Form Queueing Networks and Product Form Stochastic Petri Nets. Technical Report 529, Fachbereich Informatik Universitt Dortmund, Dortmund, Germany, 1994.

[6] Falko Bause and Peter Buchholz. Aggregation and Disaggregation in Product Form Queueing Petri Nets. In *Proceedings of the 7th IEEE International Workshop on Petri Nets and Performance Models (PNPM'97)*, pages 16–25, Los Alamitos, CA, USA, 1997. IEEE Computer Society.

[7] Giacomo Bucci and Enrico Vicario. Compositional Validation of Time-Critical Systems Using Communicating Time Petri Nets. *IEEE Transactions on Software Engineering*, 21(12):969–992, December 1995.

[8] Peter Buchholz. Hierarchical Structuring of Superposed GSPNs. *IEEE Transactions on Software Engineering*, 25(2):166–181, March/April 1999.

[9] K. M. Chandy, U. Herzog, and L. Woo. Parametric Analysis of Queuing Networks. *IBM Journal of Research and Development*, 19(1):36–42, January 1975.

[10] P. J. Courtois. Decomposability, instabilities, and saturation in multiprogramming systems. *Communications of the ACM*, 18(7):371–377, 1975.

[11] P. J. Courtois. *Decomposabilty, Queueiing and Computer System Applications*. ACM Monograph Serices. Academic Press, Inc., New York, San Francisco, London, 1977.

[12] P.J. Courtois. Error Analysis in Nearly-Completely Decomposable Stochastic Systems. *Econometrica*, 43(4):691–709, July 1975.

[13] Peter J. Denning and Jeffrey P. Buzen. The Operational Analysis of Queueing Network Models. *ACM Compututer Surveys*, 10(3):225–261, September 1978.

[14] Jörn Freiheit and Armin Zimmermann. A Divide and Conquer Approach for the Performance Evaluation of Large Stochastic Petri Nets. In *Proceedings of the 9th IEEE International Workshop on Petri Nets and Performance Models (PNPM'01)*, Los Alamitos, CA, USA, 2001. IEEE Computer Society.

[15] Baerbel Grimm, Willi Woerstenfeld, Peter Pfeil, and Karlheinz Martin. *Das grosse Tafelwerk*. Volk und Wissen Verlag GmbH, Berlin, first edition, 1994.

[16] Martin Haas and Werner Zorn. *Methodische Leistungsanalyse von Rechensystemen*. R. Oldenbourg Verlag GmbH, München, Wien, 1995.

[17] S. Haddad and P. Moreaux. Evaluation of High Level Petri nets by Means of Aggregation and Decomposition. In *Proceedings of the 6th IEEE International Workshop on Petri Nets and Performance Models (PNPM'95)*, pages 11–20, Los Alamitos, CA, USA, 1995. IEEE Computer Society.

[18] Philip Heidelberger and Kishor S. Trivedi. Analytic Queueing Models for Programs with Internal Concurrency. *IEEE Transactions on Computers*, 32(1):73–82, January 1983.

[19] Raj Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, New York, USA, 1991.

[20] Don H. Johnson. Origins of the equivalent circuit concept: the voltage-source equivalent. *Proceedings of the IEEE*, 91(4):636– 640, April 2003.

[21] Stephan Kluth. FMC-QE - Case Studies. Presented at the Fall 2007 Workshop of the HPI Research School on Service-Oriented Systems Engineering, Hasso Plattner Institute for Software Systems Engineering, Potsdam, Germany, October 2007.

[22] Stephan Kluth. FMC-QE - Positioning, Basic Definitions and Graphical Representation. Presented at the Spring 2007 Workshop of the HPI Research School on Service-Oriented Systems Engineering, Hasso Plattner Institute for Software Systems Engineering, Potsdam, Germany, April 2007.

[23] Stephan Kluth. FMC-QE - Calculus. Presented at the Spring 2008 Workshop of the HPI Research School on Service-Oriented Systems Engineering, Hasso Plattner Institute for Software Systems Engineering, Potsdam, Germany, April 2008.

[24] Andreas Knöpfel, Bernhard Gröne, and Peter Tabeling. *Fundamental Modeling Concepts: Effective Communication of IT Systems*. John Wiley & Sons, March 2006.

[25] Edward D. Lazowska, John Zahorjan, G. Scott Graham, and Kenneth C. Sevcik. *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*. Prentice-Hall, Inc., Februar 1984.

[26] John D. C. Little. A Proof of the Queueing Formula $L = \lambda * W$. *Operations Research*, 9:383–387, 1961.

[27] M. Malhotra and K.S. Trivedi. A methodology for formal expression of hierarchy in model solution. In *Proceedings on the 5th International Workshop on Petri Nets and Performance Models*, pages 258–267, Toulouse, France, October 1993. IEEE Society Press.

[28] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. Wiley Series in Parallel Computing. John Wiley & Sons, Inc., New York, NY, 1995.

[29] Hans Ferdinand Mayer. Über das Ersatzschema der Verstärkerröhre [On equivalent circuits for electronic amplifiers]. *Telegraphen- und Fernsprech-Technik*, 15:335–337, 1926.

[30] Edward Lawry Norton. Design of finite networks for uniform frequency characteristic. Technical Report TM26-0-1860, Bell Laboratories, 1926.

[31] Brigitte Plateau and Jean-Michel Fourneau. A methodology for solving Markov models of parallel systems. *Journal of Parallel and Distributed Computing*, 12(4):370–387, 1991.

[32] Herbert A. Simon and Albert Ando. Aggregation of Variables in Dynamic Systems. *Econometrica*, 29(2):111–138, April 1961.

[33] Peter Tabeling. *Softwaresysteme und ihre Modellierung*. Springer, Berlin, Heidelberg, 2006.

[34] Hendrik Vantilborgh. Exact Aggregation in Exponential Queueing Networks. *Journal of the ACM (JACM)*, 25(4):620–629, 1978.

[35] J. Walrand. A Note on Norton's Theorem for Queuing Networks. *Journal of Applied Probability*, 20(2):442–444, June 1983.

[36] Siegfried Wendt. *Nichtphysikalische Grundlagen der Informationstechnik. Interpretierte Formalismen*. Springer, Berlin, 2 edition, 1991.

[37] Werner Zorn. Calculus Paper. Hasso Plattner Institute for Software Systems Engineering, Internal, Status 20070313, March 2007.

[38] Werner Zorn. FMC-QE - A New Approach in Quantitative Modeling. In Hamid R. Arabnia, editor, *International Conference on Modeling, Simulation and Visualization Methods (MSV 2007) within WorldComp '07*, pages 280 – 287, Las Vegas, USA, June 2007. CSREA Press.

[39] Werner Zorn. Hierarchische Modellierung auf Basis von Bedienanforderungen. Presented at the Institut für Operations Research, Humboldt-Universität zu Berlin, Berlin, April 2007.

Identity Management for Cross-Organizational SOA

Ivonne Thomas
ivonne.thomas@hpi.uni-potsdam.de

Identity Management describes the process of establishing, representing and recognizing a person's identity as digital identities in computer networks. Managing identities is a critical prerequisite to determine the legitimate actions this identity may perform. For this reason, it is the foundation for identity-based access control mechanisms. Traditional approaches for identity management work well as long as systems are used by a small set of users within the trusted circle of their organization. However, SOA facilitates not only the seamless connection of systems located in one trust domain, but also of systems provided by partner organizations or third-parties which are under foreign administration and therefore represent a completely different trust domain. The vision of the future SOA is to have an *Internet of Services* – a global market place on which services are offered and consumed by independent organisational units. In such a setting traditional identity management solutions lack of scalability, user convenience and security. New approaches as federated identity management and decentralized identity management face the new situation, but still bear many open problems.

This report identifies the new challenges for identity management in cross-organizational SOA environments and describes identity federation as a concept for the controlled sharing of identity information between trust domains. To evaluate the trustworthiness of shared identity information, it is important to analyse the underlying trust relationships in order to assess the trustworthiness of the information's source. In this report the concept of organizational trust and the concept of identity trust are introduced which can be used to classify these trust relationships. Based on this, the report describes possible approaches to establish and manage trust for cross-organizational identity management and outlines the future research direction.

1 Motivation

In the past, there was no need to share identities from one organization or company to another, since each organization acted as its own authority for managing their users – thus forming a closed trust domain. However, this has changed quite a bit with SOA, services and the vision of an Internet of Services. Collaborations across the borders of the own trust domain are not only common, but required to keep pace with fast changing business needs. The design of Service-oriented Architectures allows a seamless communication between applications independent from the platform on which they run and across domain boundaries; therefore, making them perfectly suitable for the integration of services provided by independent business partners. The vision is to choose services from a broad portfolio and to compose and re-compose them according to concrete business requirements. The management of user identities across organizational borders is a key element to follow through such a vision since each service regardless whether located in the same or another trust domain needs identity information in order to perform access control and to prevent unauthorized access. The following section describes the new challenges for identity and access management in open environments and highlights why traditional identity management mechanisms lack in the context of cross-organizational SOA.

1.1 New Challenges for Identity and Access Management in SOA

Consistency In todays online world, digital identities are exploding in numbers, are dispersed over the network and are difficult to keep consistent. As a consequence, the registration and maintenance of digital identities (user accounts) becomes not only annoying, but also bears significant security risks. User are loaded with dozens of accounts - each one requiring separate user identifier and credentials such as passwords. This leads to the situation that the same passwords is used for multiple accounts or that passwords are chosen in a way in which they are easy to remember resulting in an increased security risk. Furthermore, along with an increased number of digital identities, difficulties arise to maintain identity data and to keep it up-to-date, since each account needs separate consideration. Data which is not up-to-date bears the risk that a user gains access although he is not authorized anymore, for example, when he left the company or was granted a different job position. To address such issues, solutions are required, which either provide consistency between accounts and/or which facilitate a reduced number of accounts.

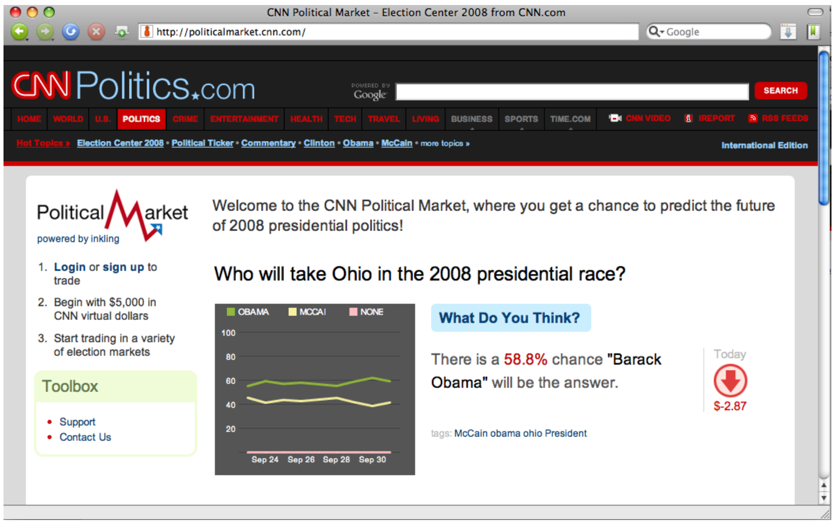
Scalability Each partner in the network needs to be identied and authorized to access another partners condential resources. Traditional approaches for identity management like the isolated model (cf. Jøsang [4]) require users to register with every single service and to reauthenticate each time they use a service in another trust domain. As businesses have become more distributed, registering and authenticating for each service leads to an explosion of accounts and expensive maintenance cost to manage users from many different domains. Accounts need to be created, but also need to be changed or deleted completely once legitimations change. If users are not located within the same administrating domain, tracking changes in a person's position or role is difficult. In particular with regard to business partners which are having their own security requirements and procedures, traditional approaches do not scale with an increased number of service users, service providers and security requirements. The challenge is to apply mechanisms which work well regardless of the number of users and trust domains involved.

Availability vs. Protection of Services On a service market place, on which services can be used by everyone, organizations are faced with the task to simultaneously protect their resources from unauthorized access while at the same time making them available to a wide range of users. However, each service call should have the appropriate level of access control, not to low to prevent security breaches and not high to preserve user convenience and high costs. This means, each transaction is associated a different security risk in case someone gets unauthorized access to the assets it provides. Such trust requirements can range from low-level access control for non-confidential information to high-level financial transactions which would result in high financial loss in case of security breaches. Therefore depending on the value of the transaction or information assets, the level of access control needs to be higher or lower. The optimal balance between availability and protection of services is needed to allow for secure, but not inmoderate access to services.

Costs Costs with regard to identity management are a crucial factor in most companies. Identification processes are expensive and time-consuming and the management of users as for example in a company's active directory requires permanent updates to reflect changes in the personnel structures as new or leaving employees. Therefore, with regard to collaborations, building on existing identity and access management infrastructures does not only save time and money, but also has an advantage with regard to security. That is, the responsibility for the user account to be up-to-date is left at the domain which is administrating this user and thus prevents misuse of accounts of users which have left the organization. Therefore, identity and access management mechanisms should support the sharing of identity and access management infrastructures between the organizations.

Federation Partnerships between organizations with the aims to use services and other resources together are increasing and come along with the challenge to bring together existing identity management systems. In order to enable resource access across completely unrelated security domains, users of one domain need to be known in the foreign domain. Since identification processes are expensive and organizations will stick to their own identity management infrastructure, mechanisms are required which introduce the own users to the foreign trust domain. This requires mechanisms to express accounts in an understandable format and to transfer the whole account or parts of the account to the foreign domain.

Trust In order to use services and to communicate identity information between different security domains a pre-existing trust relationship is required. If a relying party should accept identity information received from somewhere outside its own control, it needs to trust the other organizations identity infrastructure.This means, it need to trust the partner organization to manage their identity information in a way that is consistent with the security the company requires for those resources.


CNN Political Market – Election Center 2008 from CNN.com
http://politicalmarket.cnn.com/
Google
CNN Politics.com
POWERED BY Google
SEARCH
HOME WORLD U.S. POLITICS CRIME ENTERTAINMENT HEALTH TECH TRAVEL LIVING BUSINESS SPORTS TIME.COM CNN VIDEO IREPORT RSS FEEDS
Hot Topics » Election Center 2008 • Political Ticker • Commentary • Clinton • Obama • McCain • more topics »
International Edition
Political Market
powered by inkling
Toolbox
Support
Contact Us
OBAMA MCCAI NONE
100 80 60 40 20
Sep 24 Sep 26 Sep 28 Sep 30
What Do You Think?
Today
$-2.87
tags: McCain obama ohio President

Welcome to the CNN Political Market, where you get a chance to predict the future of 2008 presidential politics!

1. Login or sign up to trade
2. Begin with $5,000 in CNN virtual dollars
3. Start trading in a variety of election markets

Who will take Ohio in the 2008 presidential race?

There is a 58.8% chance "Barack Obama" will be the answer.

Figure 1: CNN Political Markets Start Page

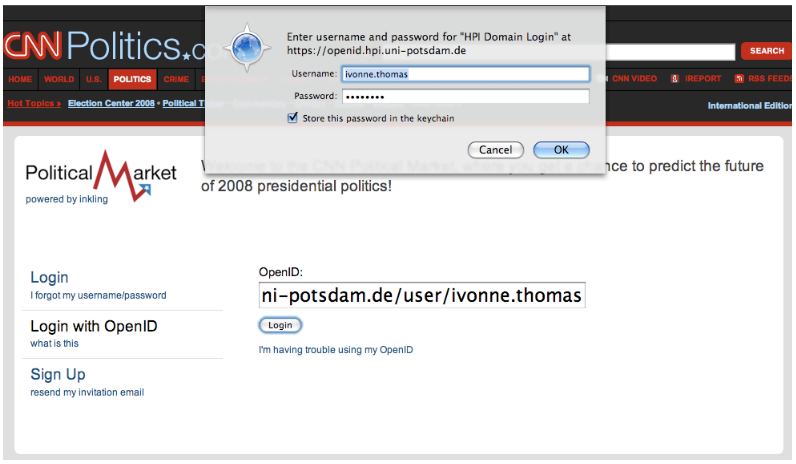
2 Sharing Identity Information for Access Control: The Concept of Identity Federation

Sharing identity information across the internet is a concept which is already used by a number of technologies such as CardSpace or OpenID. This section will start with a real-life example for a Login to a foreign web site using OpenID. Given this small example, the basic procedures for identity federation are demonstrated. Afterwards, the concept of identity federation is explained in detail. Finally, Section 2.3 will give an overview about exisitng solutions which use the concept of identity federation.

2.1 Example for Identity Federation: OpenID

OpenID is a very lightweight technology to enable the sharing of identity information across the web. Due to its simplicity it is well suited to demonstrate the basic steps necessary to perform authentication and authorization across different trust domains. As an example, a web page of CNN is used, which requires an authenticated user. This page, called CNN Political Market [1] (cf. figure 1), allows to place virtual puts and gets on politicians. In order to login, a user can either provide a password or a so-called OpenID Url. An OpenID Url identifies a user and contains the information about the entity, which administrates the user's identity information, the so called identity provider. On request CNN uses this information and redirects the user to the login page of his identity provider, which will check for the right credentials. This procedure is shown in figure 2: As the OpenID provider we use the HPI OpenId Provider, which has been implemented as a student project during the last year and which is identified by the URL https://openid.hpi.uni-potsdam.de. Once a user enters its HPI OpenID Url at the CNN

[1] http://politicalmarket.cnn.com/



Enter username and password for "HPI Domain Login" at https://openid.hpi.uni-potsdam.de

Username: ivonne.thomas

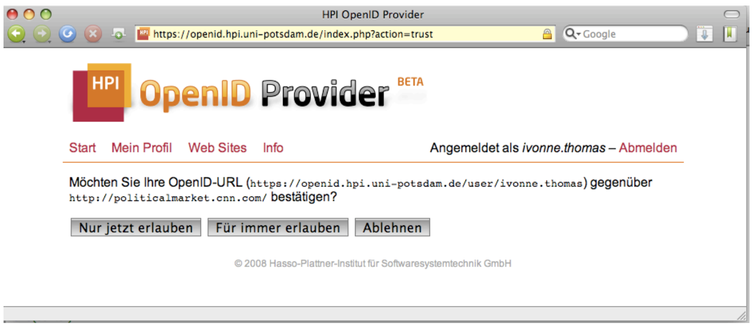
Password: ••••••••

☑ Store this password in the keychain

W…nce to predict the future of 2008 presidential politics!

OpenID: ni–potsdam.de/user/ivonne.thomas

Figure 2: CNN Political Markets Login using OpenID



Angemeldet als *ivonne.thomas* – Abmelden

Möchten Sie Ihre OpenID-URL (`https://openid.hpi.uni-potsdam.de/user/ivonne.thomas`) gegenüber `http://politicalmarket.cnn.com/` bestätigen?

© 2008 Hasso-Plattner-Institut für Softwaresystemtechnik GmbH

Figure 3: The HPI OpenID Provider requests authorization from the user to share identity information.

web site, the request is redirected to the HPI, which will prompts for a username and password of the HPI account of the user. When the user enters the right credentials, s/he is logged in and asked for permission to share its identity information with the foreign domain (cf. figure 3). Once the sharing is permitted, the request is redirected to the CNN web site and contains the information, that the user was authenticated successfully. Based on this information, the user is logged in to the foreign web site without the need to re-authenticate. Furthermore, a session is created, which allows the user to log in to any other web site which supports OpenID without the need to retype a password. The next section will give a detailed description about the sharing of identity information across trust domains, called identity federation.

2.2 The Concept of Identity Federation

Managing numerous digital identities and associated authentication credentials is cumbersome for most computer users and often bears significant security risks. Nonetheless, service providers often need a portion of our identity to perform a service (identity-based service), or to hold us liable in case anything bad happens. As a consequence, a concept for the controlled sharing of identity information was developed, called Identity Federation.

The idea of identity federation is to leave the responsibility for the users account in the managing domain and, instead, propagate required identity information to the relying partner in the foreign domain. The advantage of this approach is that identity management can build on existing Identity and Access Management infrastructures, which saves time and money and also leaves the responsibility for the user account to be up-to-date at the domain which is administrating this user.

The basic building block of Identity Federation is the trusted federation relationship established between identity providers and service providers. An identity provider (IdP) holds digital identities of registered users for the purpose of provisioning these identities, or portions of them, to a party willing to rely on this information (the relying party). A service provider (SP) usually takes the role of the relying party. It allows users to authenticate themselves at a federated identity provider and then relies on the assertion issued by the IdP upon successful authentication. As this authentication can be made persistent over some time period (a session is created), users can be freed from reauthenticating, if they access another service provider that is federated to the same identity provider. This effectively enables single-sign-on (SSO) across security domain boundaries.

In order to establish the unique user identity at the service provider, several methods can be employed. One way is to purely rely on the identity attributes retrieved from the identity provider and have no local user management (no user account at the SP). If the user previously registered an account at the service provider, s/he is offered the option to authenticate at her/his identity provider and link the SP account to the IdP account. A combination of both methods is to initially obtain relevant user data from the identity provider and then automatically create an account for the user.

An important issue in Identity Federation is privacy. Private identity data is shared between various parties over a mainly insecure network, raising the risk that it falls in the wrong hands. This threat is partly addressed by the trust relationship that underlies a federation and the regulations dened in the federation agreement, which oblige federation partners to adhere to certain privacy policies. Further mechanisms to prohibit unauthorized parties from easily deducing an identity is to maskerade unique user IDs by mapping them to pseudonyms.

One of the key mechanisms to implement identity federation is the issuance and exchange of exchange of security tokens. A *security token* in the web service context is an XML construct, which contains information about its holder's attributes signed by an authority. The signature allows the receiver of the token to verify the authenticity of the source of the claims it contains. With such a token issued in one domain, the owner can ask for access to ressources in a completely different security domain. In a typical federation scenario as shown in figure 4, three kinds of entities are involved: the *Identity Provider* (IdP), the *Service Provider* (SP) and the *User*. The *User* as the requesting party registers with his *identity provider*, which will verify the user's attributes and manage this information under a unique identifier *(user account)*. Usually the identity provider provides some kind of service, a *secure token service* (STS), which

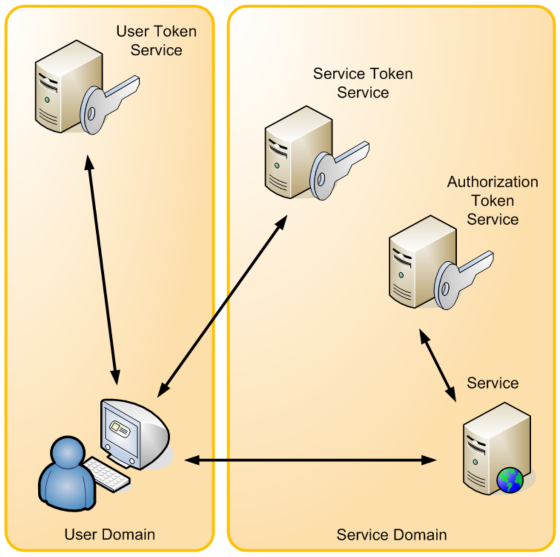


Figure 4: Typical federation set-up

can be accessed by all client applications to obtain security tokens. Once the user authenticates to the client application a request is sent to the STS to verify the user's claims and to issue a security token containing this information. This token can be sent to a relying party, usually the service provider, who is willing to rely on the information in the token. The service provider allows users to authenticate themselves at a federated identity provider and then relies on the assertion issued by the IdP upon successful authentication.

2.3 Existing solutions

Many standardization organisations and initiatives have proposed specifications and solutions for an identity management based on decentralized information. The following sections give a short overview about the main solutions and the fields in which they are used.

2.3.1 OpenID

OpenID defines a very lightweight web protocol to authenticate a user in one domain and to share his identity information with another domain. The latest version of OpenId is version 2.0 [1], whose major improvements compared to the previous version 1.1, concern the security of the protocol. One improvement, for example, is the establishment of a nonce to prevent reply attacks. Another improvement is the support of the new association type DH-SHA256 and session type HMAC-SH256 for stronger encryption.

The central idea of OpenID is to use a URL as the identifier. As this can be any URL, a party has to rely on the Identity Provider to determine correctness of a claimed identity. The OpenID authentication protocol species that the relying party should use the information provided in a XRDS document to discover the authentication endpoint of the Identity Provider. In order to work with the Identity Provider, the relying party has to establish an association with the provider, which constitutes the federation. Essentially, both setup a shared secret using Diffie-Hellman Key Exchange [7]. Necessary trust between them to perform this procedure has to be built beforehand and is out of scope of the OpenID protocol.

If an association was established at one point in time, the relying party can redirect the user's user agent to the Identity Provider with an authentication request. The Identity Provider, in turn, authenticates the user and depending on the outcome of this process either issues an authentication approval assertion, or a failure message. Therewith the user's user agent is redirected back to the relying party. In case of an approval assertion, the relying party verifies the signature (technically an authentication code) in the assertion using the shared secret established earlier.

2.3.2 CardSpace

CardSpace is the new identity management technology that Microsoft introduced after the lessons learnt from the past experiences with .Net Passport. Windows CardSpace is an identity metasystem, which allows transaction-based identity. It is a component of Microsoft's initiative to create an identity metasystem. CardSpace rests on a foundation of WS-Security, WS-Trust, WS-SecurityPolicy and WS-MetadataExchange, and will run on any platform that supports those standards. CardSpace is part of Windows Vista and is also included in the .NET Framework 3.0. It supports both self-issued and managed identities, similar to OpenID. Identities are stored as InfoCards in CardSpace and serve as the digital equivalent to the physical identification cards stored in many people's wallets. In order to authenticate using a CardSpace managed identity, a Security Token Service must be used that generates signed, encrypted tokens conforming to the WS-Trust standard.

2.3.3 Liberty Alliance

The Liberty Alliance [13] is the largest initiative for developing a federated identity architecture with about 200 members from the industry, the government and academic organizations. It is a commercial initiative which was founded in 2001 by 30 major companies like Sun, HP and Oracle in order to foster the development of open standards for federated identity management. Up to today, the Liberty Alliance has published a large set of technical as well as business specications to implement federated identity architectures. While the technical specications propose solutions from a technological perspective, the project also attaches great importance to the business relationships underlying the technical infrastructure. It considers "trust the necessary foundation for secure interoperability, and central to the successful realization of whats possible on the Web" [13]. Therefore, business guidelines have been developed to help establishing strong trust relationships between business partners as well as between enterprises and their customers. The specied architecture of the Liberty Project is comprised of three parts: the Identity Federation Framework (ID-FF), the Web Services Framework (ID-WSF) and the Identity Services Interface Specications (ID-SIS). The ID-FF is a framework to enable identity federation and management by features like identity/account linkage, simplied sign on, and simple session management. ID-WSF is a framework to support web services for requesting, retrieving and updating identity information in a federated environment. It supports, for example, sharing of attributes, service discovery as well as security mechanisms to protect the exchanged messages. Finally, the ID-SIS is based on the ID-WSF framework and aims on building high-level applications. The framework species, for example, geo-location services, directory services, calendar services and many more. Every member of the Liberty Alliance commits to implement the developed guidelines in order to build a large network of Liberty-enabled applications. A first implementation of a subset of the Liberty Alliance Project specications is available from Sun [9].

2.3.4 WS-Federation

WS-Federation [5], the Web Service Federation Language, is a public draft specication which has been developed as a joined initiative of Microsoft and IBM and is part of the WS-Security specications. WS-Federation contains a proposal for a reference model to provide identity security for web services from a technological and business point of view. As the Liberty Project, WS-Federation is also motivated by business use cases. Technically, WS-Federation is built on top of WS-Policy, WS-Trust and WS-Privacy. It uses these specications as building blocks for dening additional federation concepts, such as mechanisms for federating identity, attribute, authentication and authorization assertions between different security domains.

2.3.5 SAML

The Security Assertion Markup Language (SAML) [3] has been approved as an OASIS standard to describe security assertions about the authentication, authorisation or attributes of a subject with the aim to share this information between different trust domains. The latest version, SAML 2.0, was ratified in March 2005. SAML 2.0 has been a major step towards federated identity management, since it provides formats and mechanisms to describe the attributes of a person as well as authentication and authorisation decisions in a standard XML format. Even though SAML was not developed to complement web service standards, it fits perfectly well with them. It addresses precisely well the web services' need for portable identity. Therefore, it has been adopted by the WS-Security standard and extensions for SOAP exist which describe the transport of a SAML assertion within a SOAP message. However, SAML is not bound to a specific transport protocol. For example, the SAML URI binding defines the way in which SAML assertions are communicated using URI resolution.

2.4 Limitations

All technologies and specifications have in common that they assume that trust is established out-of-band and a trust relationship either exists or not. In fact, none of the specications talks about the qualities of the trust relationship. For example to rate a company according to its trustworthiness based on parameters as authentication or privacy agreements is not possible. The specifications only mention that some sort of trust relationship exists. In addition, it is neither clearly specified how complex trust structures can be mirrored to metadata, nor whether this is necessary at all.

3 Layered Trust Model

This section identifies two types of trust relationships which are required to accept identity information from a foreign partner and to perform access control decisions based on the received information. First, a trust relationship is required between the service provider and the identity provider in order to trust the correctness of the assertions and second, for a concrete transaction, the service provider has to decide whether the identity-based information in the assertions are sufficient to reach a certain trust level which is required to perform the request. While in the first case, the trust relationship is of a long-running kind, the trust establishment in the second case is part of identity-based access control mechanisms. We call the first kind of trust, *organizational trust* and the second kind *identity trust*. The following section gives a detailed characterization and comparison.

3.1 The Concept of Organizational Trust

Organizational trust refers to the quality of the trust relationship between the participants of a SOA. When service consumers and service providers are located within the same trust domain, the registration, authentication and management of participants happens under the same administrative control and are, therefore, usually fully trusted. However, with regard to cross-organizational SOA involving services from different organizations, trust between the participants of a SOA is not given per default, but required to allow access to the services of a partner organization. Models for identity management as Federated Identity Management establish cross-organizational trust by setting up federation agreements and contracts to extend the trust domain of an organization to the federation. Having a federation or not, whenever organizational borders are crossed by a SOA, the question of whether the partner is trusted arises. Factors as past experience, the minimum trust settings for, for example, registration and authentication of users or the reputation of a company are important attributes to assess the trustworthiness of the potential business partner. Also, the kind of business relationship is an important factor. A B2B relationship is usually much more trustworthy than a B2C relationship due to contracts which manifest certain obligations and procedures of the business partners. All these factors make up the quality of the trust relationship. This quality can be quantified in a value, which we call *Organizational*

Trust Level in the following. The assessment of the organizational trust level is in particular important with regard to authorisation and access control which is most often based on identity information. An organization will only rely on identity information received from outside its own control, when it recognizes the source of information as trustworthy. Therefore, the trustworthiness of this source has to be assessed and put in relation with the damage that might be caused by trusting on malicious information. The assessment resulting in the organizational trust level directly influences the credibility of identity information, which is the foundation for identity-based access control.

3.2 The Concept of Identity Trust

The identity of a subject is important for most systems in order to provide personalized service or to hold us liable in case anything bad happens. Therefore, reliable authentication mechanisms are required. They ensure the credibility of identity information which provide the foundation to perform access control. A broad range of access control models have been developed in the last decades, defining access control constraints based on particular security information such as the user's role (RBAC [8]) or the user's team affiliation (TBAC [12]). Since all these pieces of information can be considered as attributes of involved objects, the attribute-based access control model (ABAC) can be seen as the most comprehensive access control model, as described in [2]. Within the borders of one organization, the organizational role of a subject is often the attribute of choice to perform access control, since it can be understood as "A set of expectations and behaviours associated with a given position in a social system." [**?**] and therefore implicitly reflects the trust one can put into the correct behaviour of a subject. While this "role behaviour" is predictable within one organization, it is hard to predict for a subject from a foreign organization since role definitions are not or insufficiently known. Here, attributes as the affiliation to a company or a person's credit line are more meaningful. In either case, the provision of such trust-related attributes is required to build up trust in the identity of the user and its behaviour. This trust, which we call *identity trust*, is the concept behind all access control models.

With regard to access control in SOA, each transaction requires a different threshold for access control, not to little to prevent security breaches and not to much in view of user acceptance and convenience. Requirements can range from low-threshold access for services providing non-confidential information to high-risk financial transactions which require strong access control mechanisms. Therefore, depending on the value of a transaction and the risk associated with it, a transaction has specific trust requirements, which are described as part of access control policies. Such requirements are for example reflected as a specific role or stricter requirements for authentication.

To establish identity trust, trust-related information as specified in the access control policy is exchanged, often conveyed in security tokens, so called credentials. Each credential effects the identity trust and can lead to an update or downgrade of the identity trust level. With regard to authentication, approaches exist [10] that describe how the quality of the authentication effects the trust level. Besides attributes as the authentication of a subject, the identity trust is furthermore effected by the organizational trust level of the organization which issued the credential. Since, as mentioned earlier, the organizational trust level of an organization indicates the credibility of security information received by this organization, a low organizational trust level should also lead to a downgrade of the identity trust level.

3.3 Comparison

Table 1 summarizes the concepts of organizational trust and identity trust and compares them. As Organizational Trust refers to the quality of the trust relationship between organization, it implicitly answers the question: ”Can we trust the issuer of a token?”. The decision to trust another entity as an Identity Provider in a SOA infrastructure, is a decision which is drawn before any messages start flying around. Usually, federation agreements or similar contracts are negotiated and signed when setting up the federation. These decisions are then configures in the SOA infrastructure. As compared to this, identity trust is the trust between the subject of the transaction and the service provider. It is service-call specific and therefore is negotiated each time, a call for a new transaction receives.

Organizational Trust	**Identity Trust**
refers to the quality of the trust relationship between organizations	refers to the identity associated with a transaction
Can we trust the issuer of a security token?	Can we trust the subject in the token?
determined out-of-band	determined during service call
configurable	negotiable

Table 1: Comparison of Identity Trust and Organizational Trust

4 Trust in Identity Federations

The previous sections introduced the problems with traditional identity and access management when applied to a SOA environment and described the concept of identity federation as it is already used by existing technologies and standards. As trust is the foundation to share identity information between security domains, a classification into organizational and identity trust has been introduced. This sections points out approaches to establish both identity trust as well as organizational trust, since both are required to share identity information securely between domains.

4.1 Organizational Trust

4.1.1 Using Federations to establish Trust

Federation agreements are written contracts negotiated by the leading members of two or more cooperating companies, which contain common agreements, rules and

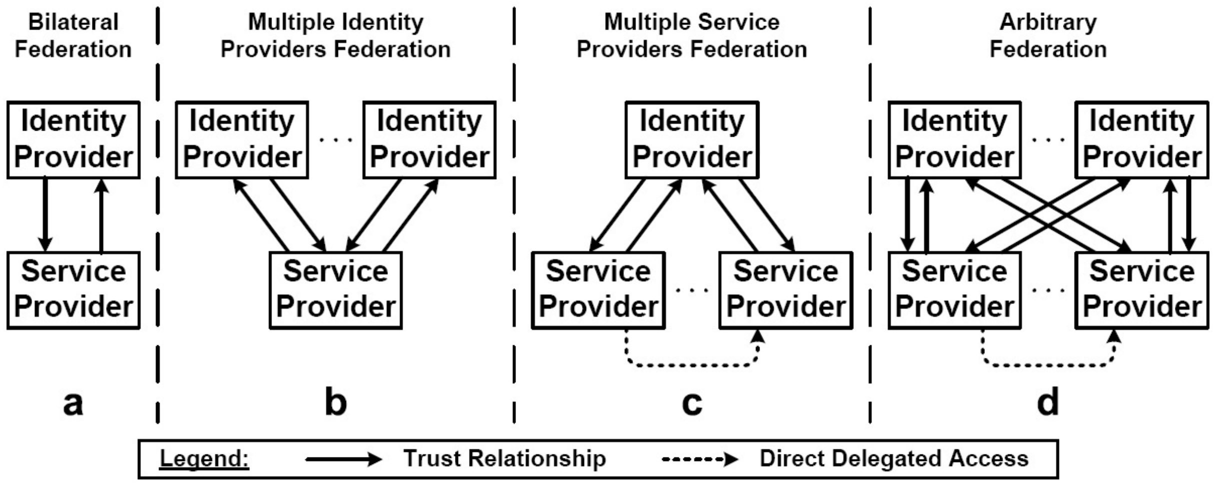


Figure 5: Patterns based on direct trust

conditions both partners need to adhere to. Federation contracts, therefore, ensure that a relying party can trust the processes and methods used by the identity provider. Such common agreements, rules and conditions comprise, for example, regulations on the registration of users, on the authentication of users or privacy agreements. Since the contracts also define penalties in case one partner does not adhere to the agreed upon obligations, federation contracts ensure that all parties are exposed to an equal risk in the case of failure.

4.1.2 Describing Trust Relations and Trust Requirements as Trust Patterns

When looking at the concept of Identity Federation, one can identify certain recurring scenarios how identity providers and service providers affiliate into federations. In these scenarios, different types and qualities of trust are distributed among the federation participants. In particular, in order to achieve a trustworthy collaboration between business partners, it is important to know: (a) which trust requirements need to be met to establish a trust relationship within a federation; and (b) which additional trust requirements arise when crossing federation borders. We have identified those scenarios that are covered by current Identity Federation theory and examined them for their inherent trust requirements. In the following, we will refer to them as trust patterns.

Our trust analysis employs a method similar to the method described by Povey in [6]. In his work, Povey presents an approach to develop trust policies by starting with a risk management analysis. Risk management explicitly deals with risk as the combination of event uncertainty and event impact. Minimizing one or both of them is the goal of risk management. According to [6] this is generally achieved in a fourstep process. First, valuable assets, threats to them and the impact of their compromise need to be identied. Second, threats emerge because of vulnerabilities, which therefore have to be found. Third, the risk of an attack exploiting vulnerabilities is determined. Finally, a decision must be made whether a risk is accepted or mitigated.

We adapted the risk management procedure to our needs and simplied it to some

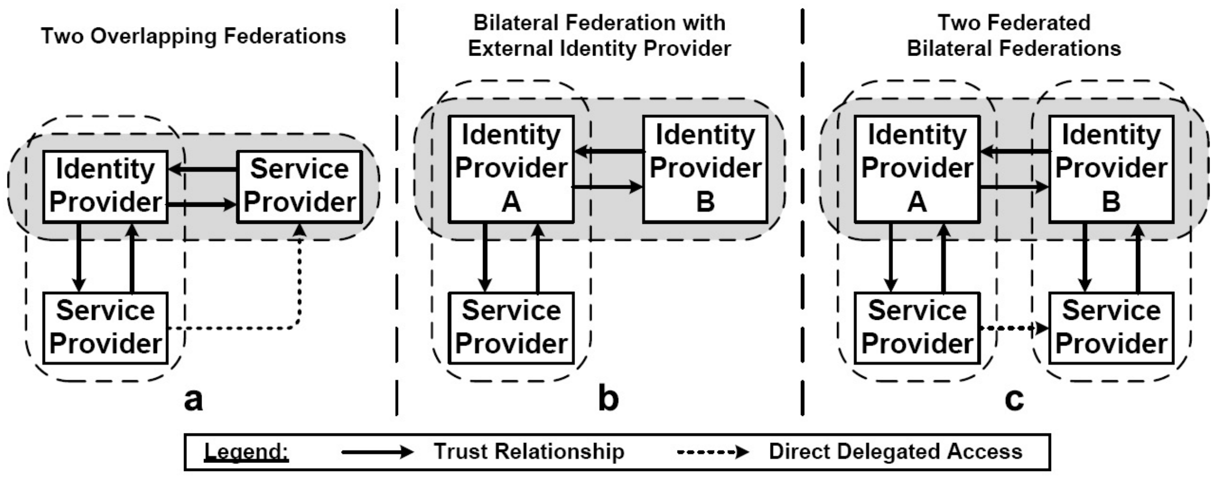


Figure 6: Patterns based on direct and indirect trust

extent. Initially, we identify assets and potential general threats. Because we almost exclusively deal with sensitive identity information and authentication artifacts, the impact of asset compromisation is likely to be severe for all assets. Hence, mitigating risks is absolutely necessary. In a second step we describe threats and vulnerabilities in detail and give a rough estimate of the probability of their exploitation. We have subsumed this examination under the term risks and conduct an examination from the viewpoint of every participant or role. This will be helpful for the derivation of trust requirements in the nal step, since trust was dened to be directional.

We identified recurring patterns in identity federation topologies and classified them into those based solely on direct trust relationships (cf. figure 5) and those which also include indirect trust relationships (cf. figure 6). Our results are summarized in the paper ”Trust Requirements in Identity Federation Topologies” and submitted for publication.

4.1.3 Using Reputation to assess Organizational Trust

Reputation is another approach to assess the trust someone can put into an organization. Reputation facilitates indirect trust relationships. There is no direct trust relationship between an entity A and an entity C. Instead both entities have a direct trust relationship with a third entity B. This relationship is leveraged to assess the trustworthiness of the unknown communication partner. The advantage is that such a trust relationship is established quickly and cost-saving, but has to that effect a lower confidence level than trust relations based on the direct trust model.

4.2 Identity Trust

4.2.1 Further Classification

In an open environment, identity is most often relevant when determining whether to trust a subject. While in the early days of computers, a digital identity was often nothing more than an identifier, a user's digital identity today a complex structure. Further parameters as the authentication of a user, its payment information such as a credit card are important to decide whether a transaction can be performed or not. Therefore, we can further classify trust into an identity into certain subcategories as trust into the registration of a user, its authentication or past experiences.

Authentication Trust Different partners in the federation might have different regulations on the authentication process that even exceed the requirements of a service provider. In these cases, the authentication method should not be stipulated in advance. For a flexible service access, different authentication methods should be allowed that comply with the services requirements. However, more flexibility in the authentication step results into a complicated access control step. For this reason, approaches exist to subsume different properties of the authentication process into a level of trust and grant access to a resource if the requirements of the expected trust level are met. Typical approaches define levels of trust by grouping requirements into categories of similar impact, by considering the economic loss or by using a combination of impact and likelihood. Moreover, ideas exist to describe the strength of the authentication by a numerical value and to subsume them into a quantified trust level. Our approach for an authentication trust level is based on the probability that an attacker can crack the authentication method and is subsumed in the paper ”Using Quantified Trust Levels to Describe Authentication Requirements in Federated Identity Management ” [11].

Registration Trust Identification processes are decisive to assess the trust into a user since they reflect the processes to create a digital identity. A registration which takes place online without any verification whether the user's name or his/her address are correct is much less trustworthy than a registration which requires the user to show an identity document such as a passport to a real person which will check whether the identity of the prospektive user matches the information in the presented passport. Therefore, differences as these should be considered when assessing the trust someone can put into the identity of user and should be reflected by a different trust level.

Reputation Trust Past experiences as for example from previous transactions tells a lot about a user's trustworthiness. Has a previous transaction been successful, it might be more likely that the user performs in the expected way. Therefore, different approaches exists, which take the reputation of a user into account when assessing its trustworthiness and base the decision for granting access on its reputation based on previous transactions or transactions performed with other service providers.

5 Conclusion and Future Work

With federation in place the integration of businesses becomes seamless. Identity Federation enables SSO between independent security domains in Web–based and service–oriented environments. It also forms the foundation for new models of identity management. These models are more collaborative than those employed today and put the user as the owner of identity information in a more controlling position. This report showed the new challenges of identity management systems when applied in a service-oriented environment. As one of the main challenges and a pre-requisite to apply a federated identity management, trust has been identified as the key to share identity information. We have seen, that in order to share identity information, trust has to be considered on two layers: This is on one hand the trust relationship between the organizations and on the other hand the trust relationship between a service and the requester. I presented a classification into *Organizational trust* to describe the first relationship and *Identity trust* to describe the trust into the subject of a request. Various approaches have been introduced to establish either trust relationships. That is for the organizational trust mainly the establishment of a federation by means of federation contracts. To characterize these trust relationships trust patterns have been introduced which describe the trust requirements in dependency of risks and potential threats as well as the assets one has to protect. These ideas have been submitted as as a paper to the IEEE AINA conference. In the area of identity trust, a further classification has been proposed to assess the trust in different aspects concerning an identity, such as trusting its registration, trusting its authentication, or trusting its attributes. Concerning the authentication trust several papers have been successfully published which describe how one can assess the trust into an authentication process.

The research conducted so far lays the foundation for various research directions. Possible extensions include the definition of a formal model to express organizational trust, identity trust and their relationship with the aim to use this model for improved access control decisions. If we introduce the concept of non-absolute trust between organizations, one important question to answer is how this would affect the identity trust, that is the trust into the subject certified by this organization. And last, but not least, it is necessary to define the parameters which influence a trust relationship and which decide about the fact whether a relying party can trust another party or not. Future work also comprises the integration of these trust considerations in existing identity management solutions as the HPI OpenID provider and an implementation of the concepts as a proof of concept.

References

[1] M. Atkins, J. Bufu, J. Ernst, B. Ferg, B. Fitzpatrick, M. Glover, H. Granqvist, D. Hardt, C. Howell, J. Hoyt, D. Recordon, D. Reed, M. Scurtescu, and K. Turner. OpenID Authentication 2.0 - Final, 2007. non-normative specification.

[2] Hai bo Shen and Fan Hong. An attribute-based access control model for web services. In *pdcat*, pages 74–79. IEEE Computer Society, 2006.

[3] J. Hughes, P. Madsen, E. Maler, R. Philpott, N. Ragouzis, and T. Scavo. Security Assertion Markup Language (SAML) V2.0 Technical Overview, 2008. Commitee Draft.

[4] Audun Jøsang, John Fabre, Brian Hay, James Dalziel, and Simon Pope. Trust requirements in identity management. *Proceedings of the 2005 Australasian workshop on Grid computing and e-research*, 44:99–108, Jan 2005.

[5] Hal Lockhart, Steve Andersen, Jeff Bohren, and Yakov Sverdlov. Web Services Federation Language. http://www-128.ibm.com/developerworks/library/specification/ws-fed/, 2007.

[6] Dean Povey. Developing Electronic Trust Policies Using a Risk Management Model. In *CQRE*, pages 1–16, 1999.

[7] E. Rescorla. RFC 2631: Diffie Hellmann Key Agreement Method, 1999. Request for Comments.

[8] Ravi S. Sandhu and Edward J. Coyne. Role-based access control models. *IEEE Computer*, 29:38–47, 1996.

[9] Sun microsystems Inc. Implementation of the Liberty Alliance Project Specifications, 2007.

[10] Ivonne Thomas, Michael Menzel, and Christoph Meinel. Quantified trust levels for authentication. In *Highlights of the Information Security Solutions Europe (ISSE) 2008 Conference*. Vieweg-Verlag, 2008.

[11] Ivonne Thomas, Michael Menzel, and Christoph Meinel. Using quantified trust levels to describe authentication requirements in federated identity management. *Proceedings of the 2008 ACM workshop on Secure web services*, 2008.

[12] Roshan K. Thomas and Ravi S. Sandhu. Task-based authorization controls (tbac): A family of models for active and enterprise-oriented autorization management. In *DBSec*, pages 166–181, 1997.

[13] J. Tourzan and Koga, Y. et al. Liberty ID-WSF Web Services Framework Overview, Version: 2.0, 2006. non-normative specification.

Taking Trust Management to the next level: Analysis and Formalization

Rehab AlNemr

rehab.alnemr@hpi.uni-potsdam.de

ABSTRACT

Business often develop proprietary reputation systems for their community, with the side effect of locking users into that service if they wish to maintain their reputation (Bonawitz, Chandrasekhar, & Viana, 2004). Reputation is used in multi-agent models like e-commerce, and distributed computation and reasoning. Currently, virtual communities are using their own reputation values only without exchanging. Reputation transfer is a controversial subject that is considered either not applicable or of high potentials.

Trust is used to carry out decisions in case of uncertainty. In that sense it is used in peer-to-peer (P2P) networks to facilitate its interactions. In P2P networks, peers' willingness to share the content they have and forward the queries plays an important role during the content search process. Using reputation in P2P systems can be an incentive for peers to cooperate. The goal is to have dynamic social networks that work on *acquiring*, *processing*, *establishing*, *analyzing*, *exchanging* and *evolving* of knowledge. In this report, the connections of trust management to the classic IT security disciplines authorization, trust, and identity management will be laid out. With this background, a generic architecture for context-aware reputation systems, which can interact with identity-related services like identity providers and policy decision or enforcement points, is presented. More specialized architectures for different environments—business- or consumer-oriented—will be derived from the generic architecture.

Continuing from the point of putting the model that facilitates reputation transfer to the path of analyzing, I have been defining the work domain and the candidate systems that could work with the proposed model, analyzing the existing reputation-based communities, categorizing them, and identifying reputation tools used. Following that, I am working on defining the most suitable ontologies to be used in order to build the knowledge-base used by the model. The goal is to formalize the proposed model to integrate into real life applications and problems, and to finally develop the standardized reputation reference models.

1 Introduction

Since the rise of the so-called Web 2.0, many social Web sites focusing on people and their relationships have attracted large number of users, and became a marketplace for various business interactions. In these Web communities, reputation related to different contexts needs to be exchanged. The perception, calculation and interpretation of this reputation differ from one community to the other creating the belief that reputation transfer is a matter of fiction. By taking a closer look at the actual difficulties of reputation transfer, we can identify the crucial points of a working reputation transfer system and finally present a means of implementing such framework.

The simplicity of current reputation systems resembles the simplicity of early identity management solutions, which basically consisted of simple databases containing usernames and passwords. Existing work on reputation systems focuses on improving the calculation of reputation values, preventing malicious actions, and deployment into the business world. The achievements in other domains, among them decentralization, standardization, and opening datasets for future enhancements, have not been considered for reputation systems. Reputation models should be capable of including and processing information that cannot be foreseen when developing or implementing the model. This will also allow combining reputation information from independent sources into a more comprehensive view on the reputation of users, services, or agents. Here, a framework is proposed that facilitates the transfer of an agent’s reputation from one community to the other by introducing:

- A new representation for the reputation value or profile; Reputation object
- The development of reference models to diminish the distance between multi-perceptions of different communities and platforms.
- The use of reputation centers to facilitate reputation transfer and highlight the importance of their role in analyzing attacks.
- Defining the knowledge domain and the candidate systems that could work with the proposed model.

I am analyzing the existing reputation-based communities, categorizing them, and identifying reputation tools that are used. Following that, I am providing guidelines to define the most suitable ontologies to be used in order to build the knowledge-base used by the model. The goal is to formalize the proposed model to integrate into real life applications and problems, and to finally develop the standardized reputation reference models. The connections of trust management to the classic IT security disciplines authorization, trust, and identity management will be laid out. Later I am presenting the work with my colleague Matthias Quasthoff, where identity management techniques are used to help deploying the model.

The report is organized as follows; first, a summary of the previously presented concepts and models, followed by concepts in identity management that add up to form the future vision for reputation based systems and the use of new identity management approaches to ensure the development of these systems are introduced.

2 Context-aware Reputation-based Framework

In the last report, I have presented a model that can be used to facilitate reputation transfer in service oriented architecture. I will summarize in the next subsections the main constructs and concepts of this model. Later I will introduce some concepts and definitions that are used in the work of combining identity management with reputation management.

Using Reputation Objects instead of Reputation Values

Most of the existing reputation-based systems lack the connection between general reputation and the context of the given reputation. I have suggested the use of an object that has the *context* attached to its value. It is agreed that trust is context-specific even if it holds the same information. For example, a professor in a medical school may simplify treatment information regarding certain disease to his students but will need to use other form of the same information in actual treatment. Therefore the need to link between the *value* and *its meaning* is crucial. Reputation object contains a multidimensional array, a matrix, which represents the reputation linked with its context and the RRTM used to calculate this value.

```
Object Reputation {
 TrustMatrix [context][reputation value][RRTM];
 Time ValidTime;
 Credentials PresentedCredentials; //optional
}
```

Associating context with trust sure increase the complexity of the overall system, but it is critical for deriving meaningful trust.

Developing Reputation Reference Trust Models

The calculation of reputation values and the perception of the meaning of each value differ from one system to another. I suggest the development of Reputation Reference Trust Models (RRTM) that can be used as reference when publishing reputation values, thus narrows the difference in perceptions. Reference trust models can be used in this case, to refer to a set of measures that each person based his opinion on. If trust judgment measures taken by one person are stricter than the others, this person may refuse to take recommendation from others who use softer measures. There is a big difference between *belief in trustworthiness* and *actions due to trustworthiness*. The distinction is important because if a trustier has past interactions with the trustee that does not mean that all future interactions are guaranteed. Though the trustee is trusted but the action due to this trustworthiness differs every time.

Reference trust models can be used in this case, to refer to a set of measures that each person based his opinion on. If trust judgment measures taken by one person are stricter than the others, this person may refuse to take recommendation from others who use softer measures. In this way a participant may make a local evaluation of global information and according to the RRTM which he is using.

Forming the models should be followed by forming mapping functions to facilitate reputation transfer from one model to the other. If it is not sufficient to have one trust model that can be used by different platforms, then the solution is to have standard models and mapping functions between them. This decreases the semantic distance of trust definitions, where each entity, platform, service, or agent has different interpretations and measures for trust judgment.

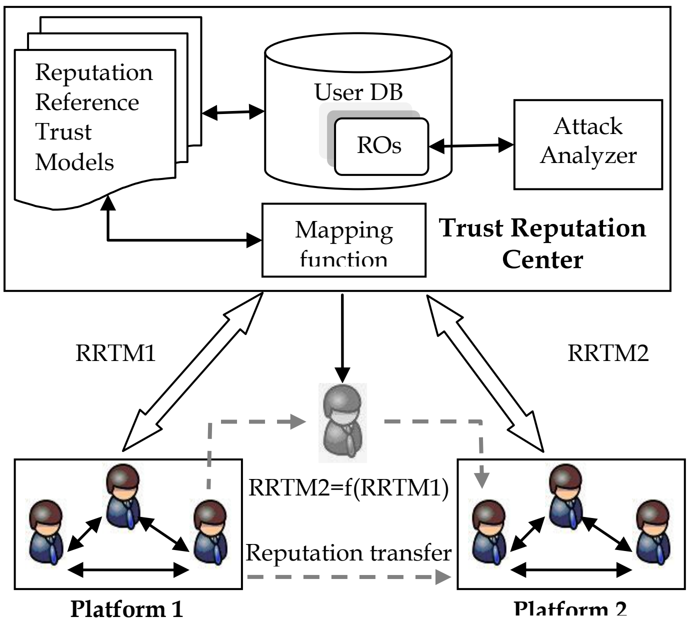


Figure 1: The Framework/Model

Developing and supporting the use of Trust Reputation Centers

The final goal of my research is to be able to implement a framework that facilitates *Reputation Transfer* between different platforms. This can happen only if we use Trust centers, much like using Certificate Authorities. TRC will be a pool of user reputation gathered from different platforms. Each user, agent, or service can have two values that define his reputation: an overall reputation (trusted or non-trusted for malicious users), and a context-based reputation object (RO). When two users from two different platforms (or organizations) establish an interaction, the TRC can be used as a transparent trusted third party. Transparency here means that it will not be enough to get a binary decision (trust or distrust) from the center, but also the RO that details reputation values regarding each context, or a specific context, and the RRTMs used. Interacting users send the center a request to get others reputation objects or values. The request should contain the RRTM used by the platform. The center transfers the stored value to the correspondent value of the requester RRTM using mapping functions.

Another advantage to this approach is that new users in any platform or community won't have a *zero* reputation but rather they will start with a value obtained by the center- *transferred reputation*. The center also can act as a negotiator or investigator of agents' reputations in a network. Moreover, using a centered trust party is the possibility of observing ballot stuffing attacks. Ballot stuffing attack is when a group of user may perform unfairly high or low ratings that may affect positively or negatively the user reputation. An observer will identify large variance of a single user's reputation/rating per context. The detection and attenuation of biased ratings will be one of the center tasks. Existing models for analysis can be used (Sherchan, Loke, & Krishnaswamy, 2006). The action that follows the detection may vary from one center to the other. Center components are:

- User Database with the associated reputation objects.
- Reputation Reference Trust Models
- Mapping functions between these models
- Attack Analyzer

Since it is unfeasible for service-oriented entities to keep information on large scale, some flexible degrees of uncertainties, and hence trust, must be established. Until now there is no legitimate authority per jurisdiction to compute trust values. The future vision of these systems is to be able to transfer reputation values from one service or community to the other (from *eBay* to *Amazon*, from *Facebook* to *Flickr*, etc.). These communities should have some common jurisdictions, so the transferred reputation value is meaningful. As human way of thinking, the perception of the degrees of trust differs from one service to another. Transferring reputation values eliminates the need for having to manage different "accounts" in different communities, which can be essential in the e-business world. It can be beneficial also in other situations: when an agent in one platform requests an interaction with another agent in different platform, or when an agent registers in a new platform and does not want to start with zero reputation value.

Mapping identities and transporting identity and reputation information between services requires new, standardized formats and protocols. The information can be expressed with the help of Semantic Web technologies such as RDF or OWL, and can then be transported using HTTP. However, the architecture model underlying HTTP (Fielding, 2000) leaves it open how to request and transmit these data, given that this involves authorization and access rights delegation. Existing solutions to access rights delegation such as OAuth (OAuth Workgroup, 2007) are not suitable for large-scale non-interactive service interaction. (Quasthoff, Enlightenment 2.0: Facilitating User Control in Distributed Collaborative Applications, 2008) (Alnemr & Meinel, 2008)

3 Identity Management

3.1 Managing digital identity

With the concepts introduced in the previous section, identity management can described as the combination of the fundamental tasks presented in this section, making up the so-called identity management life cycle. Representations of digital identities can be user profile pages on the WWW, SAML tokens (OASIS, 2008), or even just one line of text in a password file.

Establishing and describing identities. Digital identities are used to recognize participants that have been dealt with before. It is up to the designer of an information system which of a participant's properties to choose to accomplish this task. These properties can comprise the name, contact data, or passport number, but also less identifying information such as date and time of first interaction, or some other context information like IP addresses used, or just even hash values of these data. Only after a digital identity has been *established*, i.e. has been linked to the participant, we can refer to the identity and further *describe* the participant, i.e. add user profile information and link other attributes to the digital identity. It is important to see that finally destroying a digital identity can be hard. Destroying an identity is virtually impossible if descriptions of the identity in question are controlled by third parties, something which is quite realistic in an open information system like the WWW or an open service infrastructure.

Authentication. Digital identities serve several purposes. On the one hand, they are used to refer to the participant linked to an identity when issuing statements like trust assessments. On the other hand, participants choose a digital identity linked to them or establish a new one when they are about to interact with other participants in an identifiable manner (Cranor, et al., 2006). Using a digital identity in an interaction requires *authentication*, i.e. the confirmation that a digital identity is indeed linked to the participant in the interaction. Depending on the requirements to data protection and other considerations, authentication mechanisms can be designed for varying complexity and security, e.g. by requiring or doing without multi-factor authentication (OASIS, 2008).

Authorization. Many possible interactions in information systems are actually not desired. Accessing resources or information and triggering specific actions, be it placing orders or placing a call on somebody else's phone, should often be restricted to a subset of potential participants or to specific circumstances. Even in "open" systems where authorization is not required or desired, when excluding spammers and other potential attackers, actually authorization mechanisms are put in place. Authorization decisions can be based on virtually any kind of information, ranging from the digital identity of a participant over other attributes describing the identity, e.g. role or task assignments, to highly dynamic evaluation of context information right before an interaction. Especially in open systems basing authorization on enumeration of authorized digital identities is not feasible and also the definition of pre-defined roles or tasks will be hard (Seigneur & Jensen, 2007). Hence, modern decentralized and open information systems employ more generic attribute-based or context-aware authorization mechanisms, which will also be fed by trust and reputation systems.

3.2 Implementing digital identity management

Any identity management architecture can be broken down to an agglomeration of participants and components. While in final implementations some of the components presented in this section could be combined into a single service, especially in open systems this combination can lead to severe interoperability issues, as will be presented later in this report. On the contrary, by boiling down each type of service to its core functionality, better flexibility and conceptual soundness can be achieved.

Identity Provider (IdP). Digital identities are managed by special services, so-called *identity providers*. Identity providers are responsible for linking a digital identity to participant with the help of an identifier and authentication mechanisms. In systems where relying parties (i.e. services interacting with identity providers) accept digital identities issued by different identity providers, a participant must either specify its identity provider along with the identifier, or the identity provider must be discoverable from the identifier, e.g. as OpenID does with the help of URI (Recordon & Reed, 2006). As has been mentioned in the previous section, different scenarios can require different security levels. Hence, an identity provider truly designed for serving multiple use cases needs to support flexible combinations of authentication factors. The SAML protocol allows relying parties to request certain authentication mechanisms and identity providers to return the information which authentication mechanisms have actually been used (OASIS, 2008).

Security Token Service (STS). These are specified in the WS-Security specification (OASIS, 2004) and help decoupling participants, services, and the identity provider. Tokens themselves can contain various claims about the identity described, authentication and authorization. They can also contain further information restricting possible uses of the token, e.g. by restricting the validity period or relying parties. Separating STS from the identity provider, results in better cross-domain interoperability and protocol independence.

Policy Decision Point (PDP), Policy Enforcement Point (PEP). As a single identity provider can be used with several services, letting the identity provider decide on authorization for all these potentially unknown services is not feasible. Similarly, due to the potential desire for reuse of authorization and trust policies, the reasoning on authorization and trust decisions takes place at the *policy decision point*, which can be independent of the service or resource being accessed. The service itself only features a *policy enforcement point*, which delegates the authorization decision to the PDP.

3.3 Identity management architectures

Identity management silo. Classic closed-world applications and services—desktop- or Web-based—usually feature a built-in user and password database. Identities are established through invitation, e.g. by administrators or existing users, or self-registration, e.g. in large Web communities. Depending of the potential user base of such a system, legal contracts between the service provider and service users, and the mode of registration (invitation or self-registration) a more or less traceable link will exist between the digital identity and the participant. Disadvantages of such setting are obviously the need of carrying the burden of identity management instead of being able to just use an existing identity management system, and lacking interoperability with other identity-enabled computer systems. However this system architecture is being chosen quite often due to relatively low initial system and policy framework complexity. However, as systems designed around a specific user database usually have severe conceptual limitations with regards to identity management, this approach should is unsuitable for modern open systems except the consequences are well-thought and do not interfere with future development of the system.

Identity management in the Social Web. The Social Web, also known as Web 2.0, features a specific kind of Web sites allowing users to publish data about themselves, photographs or other text and multimedia content in dedicated Web sites. Especially Social Networking sites like Myspace, Facebook, or LinkedIn can be seen as Identity Providers, because they are centered around their users identity. However, the reuse of the identity described outside the Social Networking site is often limited. Some few Web sites allow further processing of digital identities by publishing some parts of the identities with the help of RDF and FOAF (Brickley & Miller, 2007) or just by some proprietary interfaces (Winer, 2003). On the other hand, there are emerging Web-scale identity management standards like OpenID (Recordon & Reed, 2006) and CardSpace (Microsoft Developer Network, 2005), which do allow for separation of identity providers and relying parties, and for extensive reuse of digital identities and claims about an identity. Social Web sites offer quite efficient, yet proprietary policy engines based on describing other users from the participant’s perspective (“friend” or “family member”). However, all the Social Web sites are still missing to take the next logical step of clearly structuring the their services into identity provider functionality (“Who am I?”), policy decision point (“Who can see what parts of my data?”), and the specific business segment—describing events, places and creative work (“Where do I love to go? What did I see there?”) etc. This clear structuring along with offering substitution of parts of these services with services from other providers with the help of standardized protocols and data formats would lead to new opportunities to Web users (Quasthoff, 2008).

Identity management in service-oriented architectures. Identity management in SOA approaches the identity and data portability issues from a different perspective compared to identity management on the Social Web. More precisely, many of the concepts introduced in the previous sections have been fixed in SOA-centric documents and specifications (OASIS, 2004). Where identity management and system architecture on the Social Web can be described as being developed in a bottom-up approach, WS-Security standards have been developed in a rather formal top-down approach. Also, the requirements to business-scale identity management are somewhat different. The contractual requirements of a company to their employees’ identity provider will be more detailed compared to what a Web user expects from some Social Web site. Also, the potentially unlimited number of identity providers a specific service has to cope with will be rather manageable in a enterprise SOA, as likely all employees of a single company, e.g. contractors or customers, will share a common identity management system. Besides the formal approach, which can be a solid basis for concrete implementations, there are three obstacles preventing WS-* standards from becoming the straightforward solution to digital identity management in general: The high level of abstraction and universality of the proposed architecture; the focus on SOAP Web services in implementations along with strong security mechanisms, and the constraint of being able to "federate" legacy enterprise identity management systems, disregarding completely decentralized solutions that will be possible with the help of trust and reputation systems.

4 Model's Use case and discussion

4.1 Transferring Reputation Object: A use case

Stock Market and Real estate Market: A use case

The usability of transferring reputation objects between two communities is best illustrated by an abstract use case. The participants of stock market and real estate market communities are shown in figure2. We are using the use case of one agent, a technical analyst, in the stock market who wants to register in a real-estate community. The reputation object $\mathtt{RO}_{\mathtt{StockMarket}}$ in a stock market is viewed as the collective value of his reputation in: Real-estate, Forex, Financial, and Commodity.

$$\mathtt{RO}_{\mathtt{StockMarket}}\mathtt{[ConextS][value]}$$

where *ContextS* $\in$ {*Credentials, Real-estate, Forex, Financial, Commodity*} and is the domain of all related properties belonging to a stock market agent. In real estate market the properties that describe an agent reputation $\mathtt{RO}_{\mathtt{RealestateMarket}}$ are: Location, Value Estimation, Quality, Future Value estimation, Marketing skills, and being Friendly.

$$\mathtt{RO}_{\mathtt{RealestateMarket}}\mathtt{[Conext][value]}$$

where *ContextR* $\in$ {*Location, Value Estimation, Quality, Future Value estimation, Marketing skills, Friendliness*} and is the domain of all related properties belonging to a real estate market agent. It is expected from a technical analyst in a stock market, who has high reputation in real-estate stocks, to be experienced in some of the real-estate properties; namely: *value*, *location*, and part of *future value estimation*, hence the semantic intersection between the two communities' properties. Hence, we can conclude from the real-estate reputation in the stock market reputation object to these reputation values in the real-estate reputation object, but cannot predict any other reputation value, thus he will have zero reputation for *Marketing skills*.

$$\mathtt{NewAgent}\ \mathtt{RO}_{\mathtt{RealestateMarket}} = \mathtt{RO}_{\mathtt{StockMarket}}\mathtt{[ContextS][value]} \cap \mathtt{RO}_{\mathtt{RealestateMarket}}\mathtt{[ContextR][value]}$$

The same can be done for communication between existing systems, like:

- eBay and Amazon,
- Credit Card rating databases,
- Credit Card databases and Western Union,
- eBay and PayPal

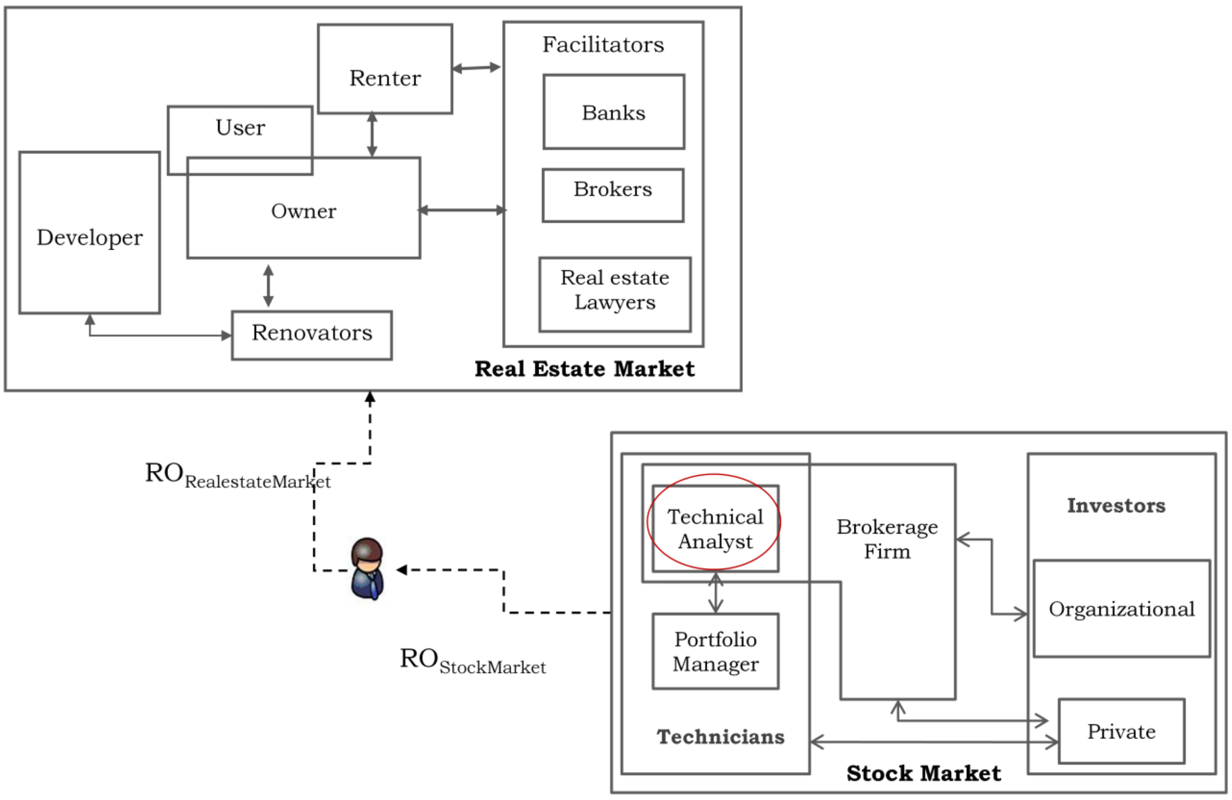


Figure 2: The Stock-Real estate Markets use case

4.2 Framework Strength points

4.2.1 Benefits of using Reputation Object

For communities like *eBay*, *Delicious*, *Amazon*, *Allexperts*, etc., the question is since the ratings are written in natural languages, then what are the benefits of the RO attributes? The answer is there are three proved benefits:

1. Easier to maintain the answers to the eBay and Amazon rating questions. For example: the after buy Questionnaire: *Was the product in the described condition? Was it delivered on time?*
2. Easier to select a seller according to what is most suitable to my requirements. For example: *I need faster delivery service and I don't care about the price*
3. Avoiding Lawsuits by specifying exactly the topic and properties of rating.

That is said, the main reason Reputation object is used is to enforce the connection between general reputation and the context of the given reputation. The context is linked to each reputation value for a single agent.

Reputation Systems and Law

An interesting study about legal challenges that face online reputation systems is being conducted in (Chandler, El-Khatib, Benyoucef, Bochmann, & Adams, 2007).

The authors explore legal cases against systems like eBay (California, Grace vs.eBay) and Amazon (cases in UK and USA). The main reason of most of the cases is rating ambiguity. Users misrepresent their rating in a way that both influence negatively the entity being rated and does not correspond to the rating attributes. For example: A used-books seller who was rated badly because the book was not good or too long, although the book was in a very good condition (which is what matters for rating a used-books seller). From the legal point of view, systems like eBay hold no responsibility for users who are expressing their *taste*. What is important from the legal perspective is the distinction between “expressions of fact” and “opinion”. Though eBay instituted *limited* assurance coverage; Standard Purchase Protection Program, the problem still exists and growing. What these systems need is specific rating attributes categorized semantically according to the sub contexts of the rated subject. The less vague the rating, the less legal issues arise.

This is achieved if Reputation objects are being used. It can break down the rating into attributes like *delivery* and *quality*. Even quality can be further *sub-categorized*. By saving this in the object, the user is able to express his opinion and at the same time his rating corresponds directly to these attributes which leads to less lawsuits.

4.2.2 Benefits of using RRTMs

Going through standardizing reputation trust models is long and needs lots of research. The question is: why standardizing reputation model factors and properties into RRTMs, instead of letting *agents* interact directly and asking directly for the reputation value from the other platform?

The answer is simple, in order to do so, the destination platform has to request for the entire history of the agent from the source platform. This is required to let the destination calculate its own view of trust and reputation, therefore a new value. At the end it is a holistic approach that consumes time and loads the system with too much computation, especially if this is done for every agent who is requesting reputation transfer.

5 Analysis and Formalization of the model

In the previous sections, we have laid out the future vision for reputation based systems and the model that fulfill this vision. To deploy such model, a thorough analysis must take place. This analysis consists of several processes like:

- Defining the domain of reputation systems without being limiting
- Checking the existing systems to avoid current problems and inflexibility
- Defining the categories of the reputation systems
- Defining concepts, terms and relationships that build up a common Ontology to be used by most of reputation systems
- From this ontology, construct a knowledge base that will be used in the model

- Defining the main constructs of the trust relationship
- Determine how identities are handled in the framework
- Give instructions for implementation

5.1 Reputation Systems Categorized

Reputation systems are fitter for e-services because the notions modeled seem more intuitive to the domain. They can be categorized by characteristics (Chandler, El-Khatib, Benyoucef, Bochmann, & Adams, 2007):

1. The subject of rating
2. The providers of the ratings (open to public or restricted)
3. The business model (revenue derived from an associated online auction, retail channel, advertising, or a public service)
4. Relative Reviews (whether users ratings are relative to the attributes of rating)

The subject of the rating varies from individuals like *Allexperts.com*, *eBay* or *Amazon*, Business like *BizRate.com*, to Articles or posting like in *Kuro5hin.org*, *Slashdot.com* and Products and services like *Epinions.com*.

Reputation systems can also be categorized based on the common features and properties of the online communities:

1. E-market places like *eBay*
2. Opinions and activity sharing sites like *Epinions, Del.icio.us, LastFm*
3. Business/Jobs network sites like *Linkedin.com & Ryze.com*
4. Social/entertainment sites like *Friendster.com & Facebook*
5. News site like *Kuroshin.org, Slashdot, & Zdnet*
6. The Web/Semantic Web as for anyone who publish anything – decentralized way
7. P2P networks where peer clients share opinions about other peers.

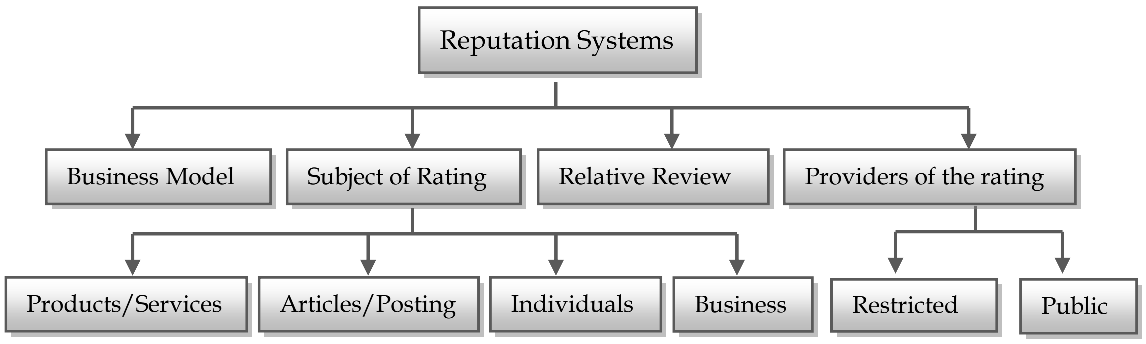


Figure 3: Online reputation systems categorized based on characteristics

In peer-to-peer (P2P) networks, the problem is that the system is open and autonomous by definition. Malicious peers can distribute corrupted files easily or even some peers can download files without letting their files available for download, a technique known as *free riding*. On the other hand, peers can benefit from one active good-peer if they know that he is *good* and *active*. A way to ensure marking both good and bad peers is to let every peer express their opinion to the others.

Using reputation concept can be an incentive for peers to cooperate. Interacting peers can provide recommendations, positive or negative, about other autonomous entities. This encourages more interaction and keeps track of entities behavior.

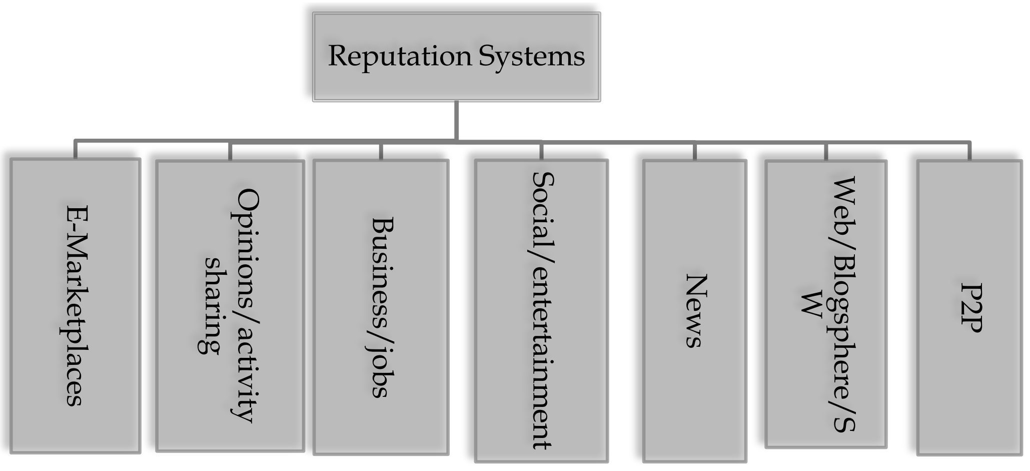


Figure 4: Online reputation systems categorizes based on common features and properties

Reputation systems also can generally be described into two abstract *Reputation Topologies*:

1. Directly: by users' statements about their experiences (all social networks).
2. Indirectly: from behavior (*Google*: view the behavior of creating a hyperlink to a Web Page as evidence of the quality, reliability of that web page). Or like the number of cross citations that a given author or journal has accumulated over a period of time. It is a science known as Scientometrics. It is the study of measuring research outputs such as journal impact factors. (Baumgartner & Pieters, 2000)

5.2 Previous and Existing Reputation Tools and Systems

Most of the existing tools focus on using contextual cues which vary from system to another to evaluate the level of trust that should be placed on a trustee requesting the establishment of a trust relationship. The conceptual function is almost the same in all tools; the change lies in the reputation calculation and dissemination.

For *Poplano 2000*, it uses simple math formulas, specifically designed for solving the problem of searching distributed databases but at the same time constraints cannot be specified. *SECURE* tool reasons about trust, where risk is a significant consideration here. *The Nameless (work of Doyle & Shrobe) 2000* is a probabilistic model that constantly collects security related data about the users from a broad variety of resources including intrusion detection systems, system logs, network traffic analyzers and so forth. In our model, this tool mostly fits into the Attack Analyzer process. *SULTAN 2003* contains lots of building blocks like specification editor, the analysis engine, the risk service, the trust monitor, and the trust consultant. Having all these processes in one system, it is expected that it is not lightweight to be installed. (Grandison, Trust Management Tools, 2007)

The problem with these tools and systems that most of them have more than one of the following shortcomings:

- do not consider interaction between different systems/platforms/services,
- cannot be fit to avoid legal issues,
- produce general reputation values regardless of the context,
- cannot be adjusted to benefit from the ongoing evolution in technology and information science; generally, not adaptive to evolved knowledge,
- have simple calculations, sometimes even unrealistic,
- do not consider *distrust,* or
- do not consider trust monitoring, update and propagation.

5.3 Knowledge base and the base constructs of relationships

To share common understanding of the information structure among agents or services, a common Ontology must be defined. This Ontology should not be used only in one domain but also in multiple domains. This includes defining the system participants, terms used, new and old concepts, and the base constructs of system relationships (Context, measurability and level of trust, cardinality, mathematical properties, and influencing factors like Mistrust, distrust, risk, agreements, experience, diffidence, incentive, and legislation)

This all lead to build up a knowledge base that is needed to help in the decision support process and to enable the reuse of the domain knowledge. Most of the terms and concepts used in the knowledge base are identified in figure 5.

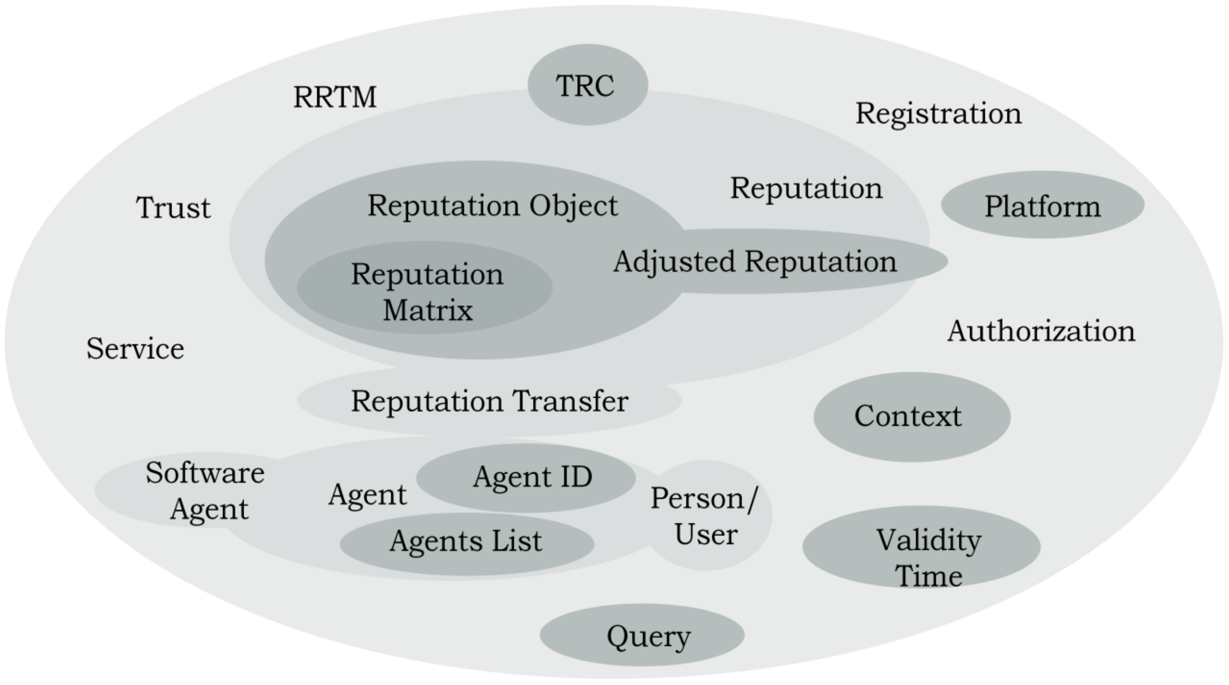


Figure 5: Terms and Concepts used in the model

6 Trust Management and Identity Management

As described earlier in this report, moving from identity management silos to Web-scale identity management has required establishing a relatively complex vocabulary and construction kit of components and interactions. The same holds for the integration of Web-scale trust management. As has been shown, high versatility of identity providers and low dependence of services upon specific identity providers is crucial for achieving an open ecosystem of services and service providers. Similarly the future trust management infrastructure must allow for high flexibility and low interdependence between services, identity providers, and trust reputation centers. Reduced interoperability and intentional hard-wiring should be supported in some scenarios, but must not be enforced in the general architecture.

In terms of the basic service vocabulary presented earlier in this report, the storage of reputation objects, the mapping function, and the attack analyzer are composable services as shown in Fig. 6. Due to the open nature of reputation—we don't know beforehand what interactions, events, and opinions will finally constitute the reputation of a participant, nor can we know all the contexts in which a participant may gain reputation in future—reputation information should be exchanged in a flexible, but well-defined format. RDF with underlying RDF reputation schemas will be the technology of choice for linking descriptions of interactions ("A purchase has been made"), users ("The user's name is …"), and ratings ("Delivery was quick, technical product quality was good, but the product itself is lacking certain features"). It will be the task of the trust reputation center to gather reports by participants, i.e. other users, services, and special services such as the attack analyzer or the mapping function, and to compute meaningful per-context ratings as specified by the requested RRTM.

Due to user privacy concerns, reputation information should not be available for public data retrieval. Rather, the user should have control over which trust reputation centers to use, and which services to allow access to his reputation data. From the perspective of services, the user should not be able to hide reputation information. Hence, a reputation search service is required returning a list of TRCs that store reputation information about the user, without automatically revealing this information. When a user first registers with a new Web site, the following actions should be triggered:

1. The Web site queries the reputation search service for the list of TRCs storing reputation information of the user.
2. The Web site asks the user for permission to query the reputation information from the centers returned by the search service.
3. For each of the centers, the user grants or denies access to his reputation data.
4. If access to all reputation centers is granted, the Web site queries and processes the reputation information according to its own RRTM. If access to some TRCs is denied, the Web site can choose one of the following actions.
    a. *Tolerate:* Accept the partial access and proceed, even at risk of deliberately being excluded from processing low reputation information

b. *Mark:* Accept the partial access as in a), but add information to the reputation object giving the user low reputation for transparency
c. *Punish:* Accept the users decision, but refuse to import any data from TRCs, hence considering the user as a new user, starting his reputation information from scratch
d. *Reject:* Abort user registration

Reject is relevant if the Web site accepts only users with already established reputation, and does not want to *Tolerate* or *Mark* incomplete reputation information.

By the protocol presented, reputation transfer between different Web sites and platforms is possible. At the same time, the user's privacy is respected and the user retains control over his reputation data without giving him the opportunity to hide valuable information.

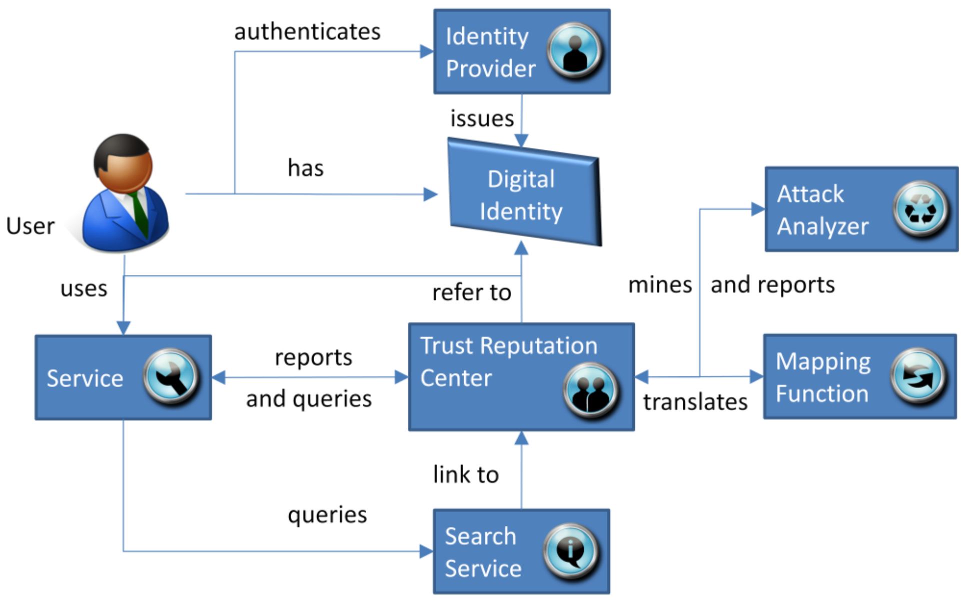


Figure 6: Combining trust management and identity management

7 Conclusion

The above work was conducted with my colleague Matthias Quasthoff and was accepted as a book chapter for *Handbook of Research on P2P and Grid Systems for Service-Oriented Computing: Models, Methodologies and Applications*. In this book chapter, we have given a definition of trust and trust management. We related trust in information systems to the perception of trust in social science. It can be observed from different point of views: trust meaning, components, types, and sources. Trust management has been defined as the process of making assessments and decisions regarding trust relationships. It has been observed that trust management tools are still maturing and not ready for wide deployment in P2P and other open, distributed systems.

As reputation and trust management requires the notion of digital identity and identity management, the relevant concepts including participants, interactions, roles, infrastructure, and information have been defined. Using these concepts, the identity management lifecycle and prototypical identity management architectures have been laid out.

I introduced my context-aware reputation framework and pointed out the need for complex reputation objects instead of plain reputation values per user. We introduced the notion of standardized reputation reference trust models (RRTM), which explain the point of view on reputation for each Web community. The mediation between these points of view is facilitated by trust reputation centers (TRC). Afterwards we analyzed existing trust management tools. All of them are incapable of adapting to the dynamic requirements in open systems. Last, we presented how the identity management architecture should be combined with the trust management framework presented. In this part, we defined a behavior model for reputation transfer and formalized this into a protocol.

From analyzing reputation systems, tools, and theories, the most important aspect is to provide adjusted reputation that reflects evolving knowledge. Trust management is not only about establishing trust relationships, but also about trust monitoring, update, and propagation. The proposed use of reputation objects along with trust reputation centers and reference models, and the combination with recent identity management technologies forms our contribution.

References

Alnemr, R., & Meinel, C. (2008). Getting more from Reputation Systems:A Context–aware Reputation Framework based on Trust Centers and Agent Lists. *The Third International Multi-Conference on Computing in the Global Information Technology* (pp. 137-142). Greece: IEEE Computer Society Press.

Baumgartner, H., & Pieters, R. (2000). *The Influence of Marketing Journals: a Citation Analysis of the Discipline and its Sub-Areas.* Tilburg: Tilburg University, Center of Economic Research.

Bonawitz, K., Chandrasekhar, C., & Viana, R. (2004). *Portable reputations with EgoSphere.* Massuchusetts: MIT Internal Report.

Brickley, D., & Miller, L. (2007, November 1). *FOAF Vocabulary Specification 0.91*. Retrieved from http://xmlns.com/foaf/spec/

Chandler, J., El-Khatib, K., Benyoucef, M., Bochmann, G., & Adams, C. (2007). Legal Challenges of Online Reputation Systems. In L. K. R. Song, *Chapter in Trust in E-Services: Technologies, Practices and Challenges* (pp. 84-111). Hershy: Idea Group Publishing.

Cranor, L., Dobbs, B., Egelman, S., Hogben, G., Humphrey, J., Langheinrich, M., et al. (2006, November 1). *The Platform for Privacy Preferences 1.1 (P3P1.1) Specification*. Retrieved from http://www.w3.org/TR/P3P11/

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures.* Irvine: University of California.

Grandison, T. (2007). Trust Management Tools. In R. Song, L. Korba, & G. Yee, *Trust in E-Services: Technologies, Practices and Challenges* (pp. 198-216). Hershy: Idea Group Publishing.

Microsoft Developer Network. (2005, May 1). *Microsoft's Vision for an Identity Metasystem*. Retrieved from http://msdn.microsoft.com/en-us/library/ms996422.aspx

OASIS. (2008, August 01). *SAML Specifications*. Retrieved from http://saml.xml.org/saml-specifications

OASIS. (2004, March). *Web Services Security*. Retrieved from http://www-128.ibm.com/developerworks/library/specification/ws-secure/

OAuth Workgroup, O. C. (2007, December 4th). *OAuth Core 1.0*. Retrieved December 4th, 2007, from http://oauth.net/core/1.0/

Quasthoff, M. (2008). Enlightenment 2.0: Facilitating User Control in Distributed Collaborative Applications. *In Proc. of the 2008 WI-IAT Doctoral Workshop.* Sydney: IEEE.

Recordon, D., & Reed, D. (2006). OpenID 2.0: a platform for user-centric identity management. *Proc. of the 2nd ACM workshop on Digital identity management* (pp. 11-16). Alexandria: ACM.

Seigneur, J.-M., & Jensen, C. D. (2007). User-Centric Identity, Trust and Privacy. In R. Song, L. Korba, & G. Yee, *Trust in E-Services: Technologies, Practices and Challenges* (pp. 293-322). Hershey: Idea Group Publishing.

Sherchan, W., Loke, S. W., & Krishnaswamy, S. (2006). A fuzzy model for reasoning about reputation in web services. *Proceedings of the 2006 ACM symposium on Applied computing* (pp. 1886 - 1892). Dijon, France: ACM.

Winer, D. (2003, June 1). *XML-RPC Specification*. Retrieved from http://www.xmlrpc.com/spec

Automated Service Composition for Minimal Goals

Harald Meyer
harald.meyer@hpi.uni-potsdam.de

This report gives a brief overview of some of my research in the past six months. The main focus is on automated service composition of minimal goals. Additionally, a brief summary of a joint project with Software AG on service tagging is given.

Optimality of service compositions is seen as the number of activities or additional quality of service metrics. We argue that this is not enough. Charging a credit card if it is not necessary, is wrong. We extend an automated service composition approach by the notion of minimal goals.

Achieving real world effects in excess to the specified ones can be problematic. But only allowing specified real world effects is too restrictive for practical usage. We therefore propose an approach that limits the amount of achieve unspecified effects.

1 Introduction

The goal of service orientation is the alignment of the IT infrastructure to the business goals of a company [2, 5, 14]. Service composition is a fundamental concept of service orientation. Using service composition, existing functionality can be aggregated to form new, more complex functionality. Today, service composition is mostly done manually at design time. This makes adaptation to changes in the service landscape or business requirement changes difficult. Also, the manual creation of service compositions is an error-prone and complex task. Finally, compositions are not tailored to the individual service request. This means that for individual requests unnecessary tasks are performed. Different approaches for the automation of service composition exist [3, 6, 15, 16, 18].

Existing approaches have in common that they try to create a service composition that achieves certain goals. What they do not tackle is to achieve *only* these goals. Optimality of service compositions is often seen as the number of activities, the length of the critical path, or additional quality of service metrics (costs, execution time). We argue that this is not enough. Achieving only the required goals can be crucial when the goals are real world effects. For example, charging a credit card if it is not necessary, is wrong. In this paper, we extend an automated service composition approach by the notion of minimal goals.

A first step towards a better composition approach is to separate effects with regard to their possibly negative impact. Achieving certain effects in excess of the specified goals is more problematic than achieving others. The critical effects are actually the *real* *world effects* in contrast to unproblematic *information space effects*. This separation is not new. In WSMO [17] real world effects are the only effects. Information space effects are called postconditions.

Given the separation of information space effects and real world effects, a first approach might be to change the composition approach as follows. Reaching the goal in information space means to achieve at least all the information space effects from the goal (plus possibly additional ones). Reaching the goal in the real world means to achieve exactly the real world effects from the goal.

While this approach is nice in theory, it is too restrictive for practical usage: One is often not interested in all real world effects. Certain real world effects are mere byproducts. If one is required to model them, specifying a service request becomes much more complicated. It means that one needs to know all the effects of all services and specify the effects one wants to achieve and the effects one does not want to achieve. This essential encodes assumptions about which services will be used into the service composition.

Our approach is different. Instead of requiring a perfect match for real world effects, we instead just try to achieve as few additional real world effects as possible. We call this approach composition for minimal goals. Hence, we allow for real world effects we did not intend. In some cases, these effects might actually be harmful (e.g. charging a credit card). But given the algorithm presented in this paper, we can argue that these additional effects will only the limited.

The rest of the paper is structured as follows. We will introduce in Section 2 a motivating scenario and show why composing for minimal goals makes sense. In Section 3 automated service composition using heuristic search is introduced. We present search strategies and a heuristic to guide them. Section 4 then defines what minimal goals are and how to incorporate a heuristic for overachievement estimation into existing search strategies. After an overview of related work in Section 5, the paper closes with a summary and a look at future work.

2 Motivating Scenario: Online Shopping

E-commerce companies must be flexible. They want to integrate new partners easily, expand their product portfolio, and expand into new regions and countries. Because of its declarative nature, automated service composition can be advantageous over explicitly modeled processes for these companies. We will use one process from E-commerce, namely order processing, to demonstrate in this paper why composing minimal goals is important.

A variant of the order processing service composition is depicted in Figure 1. The input to the process is an order and customer and payment data. The items of the order are packaged and then shipped. In parallel, payment is performed. Payment consists of two services. The first authorizes the payment and blocks the amount on the customers credit card. Only the second then performs the actual payment. Other payment options are available as well. Another service performs the payment without first authorizing it. Finally, it is also possible to pay with a gift voucher. Another option

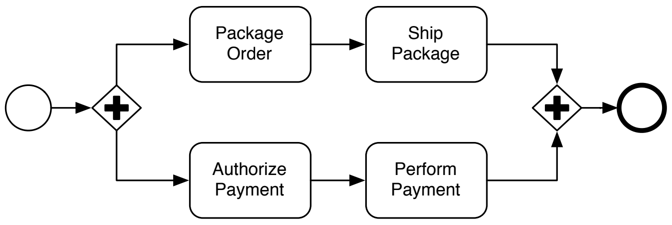


Figure 1: Service Composition of the Order Processing Process.

for customers is to gift wrap individual items. Altogether, the available services are:

- *Package Order*: takes and order (precondition) and packages it (effect)
- *Ship Package*: takes a packaged order (p) and ships it (e)
- *Gift Wrap Item*: takes an item from an order (p) and wraps it as a gift (e)
- *Payment*:
  - *Direct Payment*: takes customer information and the amount of money (p), transfers the amount from the customer to the company, and marks the payment as performed (e)
  - *Authorize Payment*: takes customer information and the amount of money (p) and authorizes the payment by emitting a payment token (e)
  - *Perform Payment*: takes a payment token (p), transfers the amount from the customer of the company, and marks the payment as performed (e)
  - *Pay With Gift Voucher*: takes a gift voucher (p), invalidates the gift voucher, and marks the payment as performed (e)

Given this service landscape, it is easy to show why composing for minimal goal is advantageous. Most services have real world effects. Doing more than intended to do can be expensive (gift wrapping even though the customer did not want it) or annoying to customers (performing unnecessary payments even though the customer used a gift voucher), which, in the end, will be expensive, too. It has to be noted that this can even happen if the composition is supposedly optimal, if optimality means number of services or length of the critical path: A composition that did not use the gift voucher but instead used the *Direct Payment* service has the same number of services. But it is clearly not what the customer wants to happen.

One could argue that composing for minimal goal is not really necessary and could be replaced by more elaborated modeling what to achieve and what not to achieve. For example, one could state for each item of the order that it *should not* be gift wrapped. But of course this is impracticable. Another example is the payment token exchanged between the *Authorize Payment* and *Perform Payment* services. If it is not deleted by the latter, we are required to state it as part of the goal. But as we do not know which payment service is used, this becomes difficult. If we leave it out, we cannot use the two-step payment processing. If we include it, we have to use the two-step payment processing. Does automated service composition make sense then? We encode which service to use in the request. We also need to know which services are available to explicitly exclude their potentially dangerous effects. How to overcome these problems by composing minimal goals will be shown in the remainder of this paper.

3 Automated Service Composition using Heuristic Search

Heuristic search is a frequently used approach in automated planning and automated service composition [10, 12]. Given a service request consisting of an initial state, a goal, and a set of service operations, find a path in state space that leads from the initial state to a state satisfying the goal. The service invocations on the path and their ordering on the path is the service composition. The state space is a graph with states as vertices and service invocations (the instantiations of service operations) as edges. Formally, these concepts are:

Definition 1 *A **logical expression** $e \in E$ defined over a alphabet (R, F, C, V) with the set of relations R, the set of functions F, the set of constants C, and the set of variables V is:*

- *A term t is a logical expression. T is the set of terms. T_{ground} is a subset of T containing only ground terms. T and T_{ground} are defined as:*
  - *A variable $v \in V$ is term ($v \in T$, $v \notin T_{ground}$).*
  - *A constant $c \in C$ is a term ($c \in T_{ground}$).*
  - *If $f \in F$ is a function and terms $t_1, ..., t_n \in T$ then $f(t_1, ...t_n) \in T$. Iff $t_1, ..., t_n \in T_{ground}$ then $f(t_1, ...t_n) \in T_{ground}$*
- *If $r \in R$ is a relation and $t_1, ..., t_n \in T$ are terms then $r(t_1, ..., t_n)$ is a logical expression (i.e. $r(t_1, ..., t_n) \in E$).*
- *If e is a logical expression, so is $\neg e$ ($e \in E \Rightarrow \neg e \in E$).*
- *If e_1 and e_2 are logical expressions, so is their disjunction $e_1 \vee e_2$ and conjunction $e_1 \wedge e_2$ ($e_1, e_2 \in E \Rightarrow e_1 \vee e_2 \in E$ and $e_1 \wedge e_2 \in E$).*

Literals are all $r(t_1, ..., t_n)$ and $\neg r(t_1, ..., t_n)$.

*A logical expression a **satisfies** another logical expression a' (written as: $a \models a'$) if every positive literal of a' is in a ($\forall l \in a'^{+}, l \in a$) and no negative literal of a' is in a ($\forall \neg l \in a'^{+}, l \notin a$).*

Definition 2 *A **service request** $R = (a_0, g, \mathcal{O}p)$ is a triple consisting of the initial state $a_0 \in E$, the goal $g \in E$ and a set of service operations $\mathcal{O}p$. A state is a logical expression. services.*

Definition 3 *A **service operation** is a tuple $s = (I, O, pre, \textit{eff})$ consisting of:*

- *I: List of input parameters*
- *O: List of output parameters*
- *pre: The precondition is a logical expression and must be satisfied in order to invoke the service.*
- *eff: The effect is a logical expression. It describes the changes to the current state resulting from the invocation of the service.*

*A **service invocation** $i = (s, Z)$ is a pair consisting of a service operation $s = (I, O, pre,$eff$)$ and a variable assignment $Z : \mathcal{V} \rightarrow \mathcal{T}_{ground}$ that assigns every variable a ground term. Formally speaking, this is a Herbrand interpretation and T_{ground} is the Herbrand universe [9]. Variables $v \in V$ are all the elements from $\mathcal{I}$ and $\mathcal{O}$ plus the additional variables from pre and eff. pre^Z and eff^Z are the precondition and effect with all variables bound according to the variable assignment Z.*

3.1 Search Strategies with Heuristics

Our work is based upon previous research by Hoffmann and Nebel, who developed the planners FF [8] and Metric-FF [7]. They introduced enforced hill-climbing as a planning algorithm and relaxed Graphplan as its heuristic. Hill-Climbing is a search algorithm guided by a heuristic function $h_{dist} : E_{state} \times E \rightarrow \mathbb{R}_0^+$. This heuristics $h_{dist}(a, g)$ delivers an approximation of the distance (measured in numbers of services to invoke) of the state a to the goal g. Starting with the initial state, a new state is selected from the direct successors.

The first successor that is, according to the heuristic, better (e.g. nearer to the goal) than the current state is selected and assigned as the new current state. This process is continued until the current state satisfies the goal or search fails. It fails if a state a, which is unequal to g, is reached so that no direct successor a' with $h(a', g) < h(a, g)$ exists. Given an admissible heuristics and a mechanism to prevent visiting states multiple times, the algorithm always terminates.

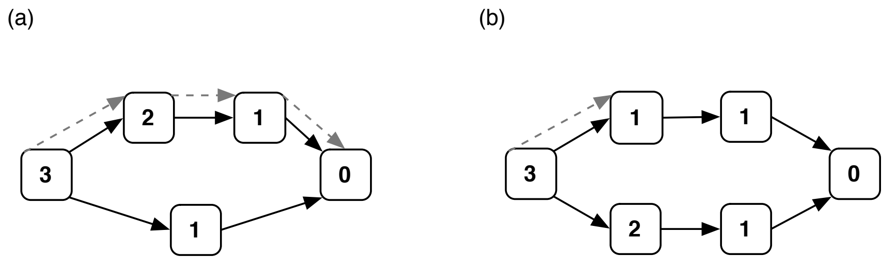


Figure 2: Hill Climbing is not optimal (a) and incomplete (b).

Hill-Climbing does not create optimal compositions and it is incomplete. Figure 2(a) illustrates the reason for its inoptimality. Displayed are states, their heuristic values, and possible state transitions. If the state with heuristic value 2 is evaluated first, it is selected even though a shorter path exists. Another problem is the greediness of

Hill-Climbing. Greediness means that optimization is done locally without taking the path to the current state into account. This is only of importance if a cost function is associated with state transitions. Otherwise the admissible heuristics guarantees that greediness does not affect the composition result. Figure 2(b) demonstrates why Hill-Climbing is incomplete: If the upper path is taken, composition fails after the first state with heuristic value 1 as no direct successor with a better heuristic value can be found. Such a state is called a local maximum. Using an extended version of Hill-Climbing, Enforced Hill-Climbing the problem of local maxima can be solved. If the algorithm gets trapped in a local maximum, it switches the breadth-first search until it reaches the end of the plateau. As classical Hill-Climbing, its enforced variant is incomplete and does not guarantee optimal solutions. Best-first search is a strategy similar to Hill-Climbing. But instead of selecting the first neighbor that is better, it selects the best neighbor. Like Hill-Climbing it is incomplete and inoptimal.

Greediness is the source for incompleteness and inoptimality. It tries to optimize for lower distance estimations not taking into account the costs it took for getting to the current state. A non-greedy algorithm is for example $A*$. To calculate the goodness of a state it does not only take into account the estimated distance to the goal, but also the (known) distance from the initial state. In every step, the unevaluated state with lowest combination of estimated distance and actual distance from the initial state is selected. Situations in which a greedy algorithm is lured into an inoptimal path or into a dead end can be solved by $A*$ because it is able to *turn around* (back track). All unevaluated states together are called the frontier. Figure 3 demonstrates this. Each states contains the estimated distance (top) and the distance from the initial state (bottom). The current state has a bold border while unevaluated states have a dotted one. In the initial state, the upper path it taken because it has a lower distance estimation than the lower path (with equal initial state distances). During phases (b) and (c) it turns out that the distance estimation was too low and in (c) the algorithm stops evaluating the upper path. The sum of actual distance from the initial state (3) and the estimated distance to the goal (1) is higher than for the first state in the lower path $(2 + 1)$. Hence in (d) the lower path is selected, through which in the following the optimal solution for this problem is found. If the heuristics is admissible, this means that the estimations do not overestimate the actual distance, $A*$ is guaranteed to find the optimal solution and is complete.

3.2 Heuristics: Distance Estimation

A common estimation technique is relaxation: some constraints from the original problem are removed, making the problem easier to solve. Then the distance for the relaxed version of the problem is taken as an estimation of the distance in the unrelaxed problem. Hoffmann and Nebel [8] did this with their *Relaxed Graphplan heuristics* h_{dist}. Negative effects are ignored and a solution for the planning problem is found using the Graphplan algorithm [4]. As they showed, the resulting heuristic is admissible and in P. Graphplan is a planning algorithm, which separates planning into two phases: graph building and solution extraction. In the graph building phase a planning graph is built. It is a leveled, directed graph:

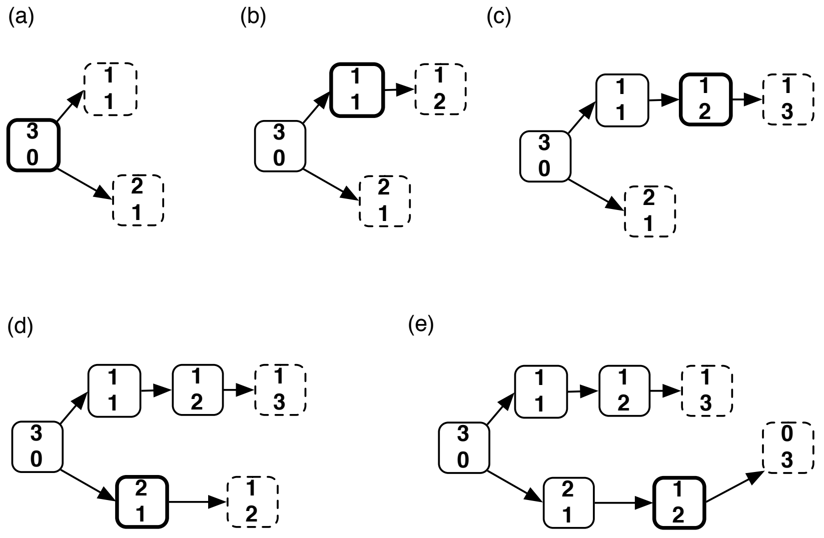


Figure 3: $A*$ finds optimal plans.

Definition 4 *A planning graph is a leveled, directed graph* $G_{planning} = (\bigcup_{i<n} L_i, E)$ *with* L_{i*2} *being the fact layers and* L_{i*2+1} *being the activity layers. A layer consists of vertices* $li \in L_i$ *which are either facts or activities. Edges only connect vertices from adjoining layers:* $E \subseteq L_{i*2} \times L_{i*2+1} \cup L_{i*2+2} \times L_{i*2}$.[1]

Figure 4 illustrates such a graph. A fact layer represents a state. The first fact layer is the initial state. All invocable services are added to the first activity layer. This activity layer *produces* a new fact layer including all the effects of the selected services. This is continued until a fact layer is reached that contains all the goals.

For solution extraction backward search is performed. Starting from the final layer, all service invocations in the previous layer are selected that contribute to goal. In the next step, producing service invocations are selected for the facts required by the previously selected service invocations and the remaining goal facts. This is continued until the first fact layer is reached. The total number of service invocations selected is then taken as the heuristic value for distance from the initial state to the goal.

4 Heuristic Search using Overachievement Estimation

The heuristic used to guide the search in the previous section was an estimation of the distance of given states from the intended goal. The composition algorithms presented so far do not take into account whether the goal achieved by the composition contains unnecessary real world effects. An extension to do this will be presented in this section. As only real world effects are important for this, states need to be separated into real world and information space effects:

[1]The original definition of planning graphs [4] also allows edges inside layers. They are used to denote mutual exclusion relationships between activities or facts.

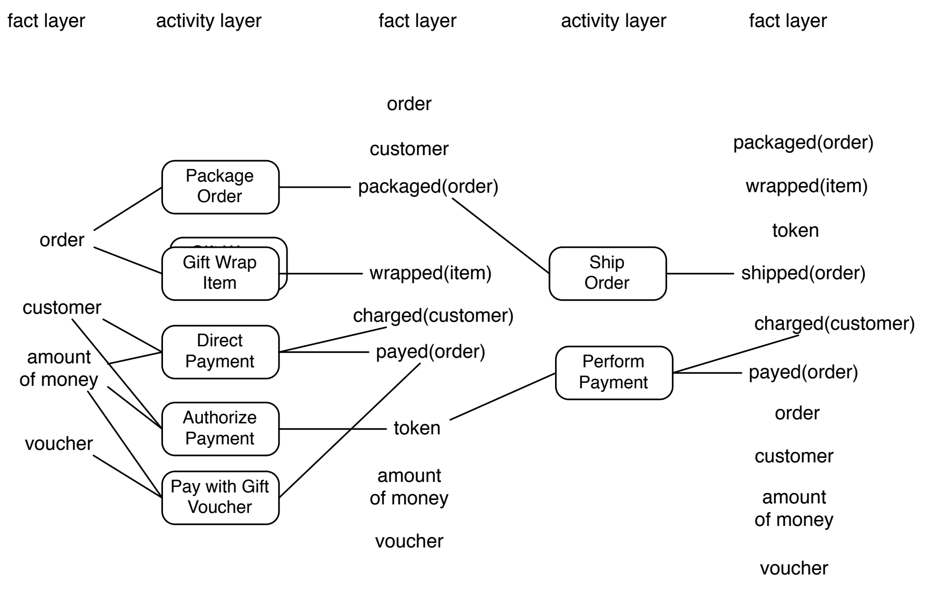


Figure 4: A Planning Graph.

Definition 5 *Given a logical expression e, e_{inf} represents the information space part and e_{rw} represents the real world part of the expression (Note: $e = e_{inf} \cup e_{rw}, e_{inf} \cap e_{rw} = \emptyset$)*

Using real world effects, it is possible to calculate how good or bad a state is with regard to how much more it achieves than intended by the goal. The result of this calculation is the overachievement score:

Definition 6 *Given state s and service request $R = (a_0, g, \mathcal{S}D)$ the overachievement score $score_{over}(s, a_0, g)$ of state s for initial state a_0 and goal g is:*

- $|s_{rw} \setminus a_{0_{rw}} \setminus g_{rw}|$ *if* $s \models g$
- ∞ *otherwise.*

The overachievement score of a state is hence infinite if it does not satisfy the goal and otherwise it is the number of additional facts in the state that are not necessary to satisfy the goal. The lower the overachievement score of the achieved state, the better the composition. That helps to decide which composition is best among a set of compositions. But it does not help in creating a good composition. Another heuristic could be used together with the Relaxed Graphplan heuristics. The overachievement heuristic h_{over} is an estimation of the overachievement score of the achieved goal state of a service composition starting in the current state. To calculate it without too much overhead, we let h_{over} be the overachievement score of the achieved goal state of the Relaxed Graphplan solution.

Let us look at how solution extraction in Relaxed Graphplan works. Figure 5 shows the planning graph from Figure 4. But instead of displaying the whole graph it starts in the final fact layer with only the facts from the goal (the order is shipped and payed).

Now those service invocations from the activity layer are selected that fulfill any of the goals. These are *Ship Order* and *Perform Payment*. Goals might also be fulfilled by service invocations in earlier activity layers. Now as we have fulfilled the goals, we have to fulfill the preconditions of the selected service invocations in the next step. We can do this using *Package Order* and *Authorize Payment*. The other service invocations in this activity layer are not part of the solution. This means the extracted solution is neither optimal with regard to its length (using *Direct Payment* or *Pay with Gift Voucher would have been shorter*) nor with regard to overachievement (using *Payment with Gift Voucher* would have prevented the unnecessary real world effect $charge(customer)$).

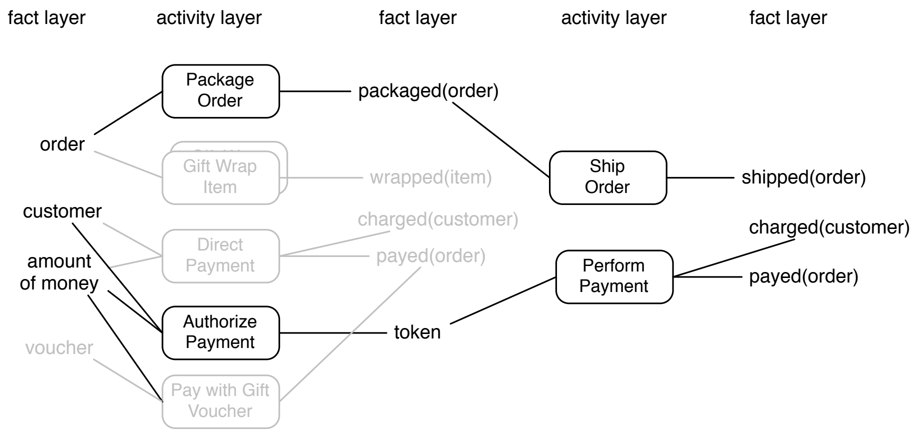


Figure 5: The Planning Graph after Solution Extraction.

The idea now is, to calculate the overachievement score for the state reached by the relaxed solution. In this example, the overachievement score is 2. The facts $token$, $packaged(order)$, and $charged(customer)$ were not part of the goal (which contained $shipped(order)$ and $payed(order)$). The fact $token$ is an information space fact. Hence it is not counted towards the overachievement score.

This example shows why it was not sufficient to allow only specified real world effects: $packaged(order)$ is an unspecified real world effect. It is a required intermediate step to shipping the order. The heuristic calculation is only an estimation and does not deliver the minimal possible overachievement score. If *Pay with Gift Voucher* was used instead of *Authorize Payment* and *Perform Payment*, the overachievement score would have been 1.

How the overachievement heuristic can be used to guide the search depends on what we want do achieve. Minimal overachievement or minimal composition sizes? Depending on the preference different strategies on how to incorporate the overachievement heuristic must be followed. How this can be done will be presented in the next two sections.

The overachievement heuristic can vary dramatically from one state to it predecessors or successors in state space. It can increase, decrease or remain the same. The heuristic is inadmissible. Using it with $A*$ as a search strategy, one can not guarantee optimality with regard to overachievement. Additionally, if one prefers minimal overachievement over minimal composition size, $A*$ is optimal neither with regard to overachievement nor to composition size minimality. But as we will see in the next section, if one prefers composition size minimality one can guarantee it with $A*$. Of course, preferring composition size minimality means that one cannot make such guarantees about minimal overachievement.

4.1 Preference of Minimal Overachievement

We can use the overachievement heuristic with Hill-Climbing by selecting the first neighboring state which has a lower distance estimation and at most the same overachievement estimation as the current state. Only if we do not find such a neighboring state, we take into account states with higher overachievement estimation. With $A*$ we select the unevaluated state from the frontier with the lowest overachievement estimation and if several unevaluated states have the same overachievement estimation, the one with the lowest distance to the goal.

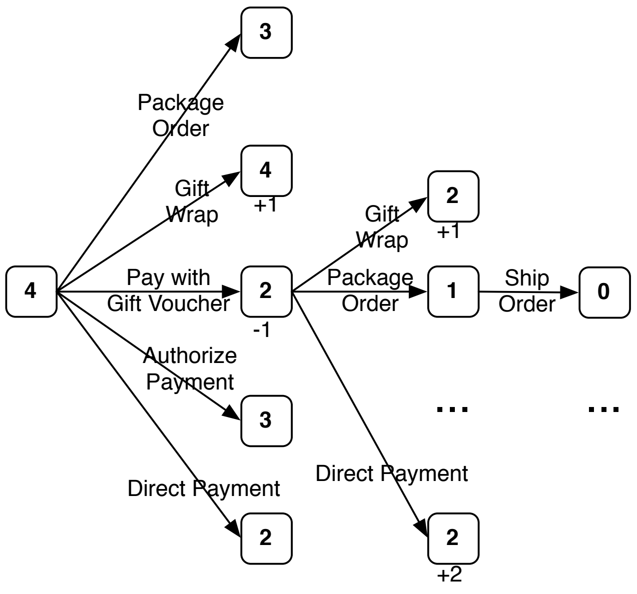


Figure 6: Search with Minimal Overachivement Preference

Figure 6 shows how composition could work when using best-first search. The initial state is an order and the customer wants to pay with a gift voucher. The syntax is similar to previous illustrations of the search space. The rectangles are states connected by service invocations. The number inside the the rectangle is the h_{dist} value for this state. Below some states, their effect on the overachievement heuristic is shown. As we have seen in Figure 5, the overestimation heuristic's value for the initial state is 3. A $+X$ means that the overestimation for this state is estimated to be worse than that of the previous state. Accordingly, $-X$ means that it is better than the previous state.

In the example, we select the state reached by *Pay with Gift Voucher* as the next state after the initial state. It has the best overachievement estimation (h_{over} of value 2). It is also the best state according to the distance estimation. What we can see here, is that without using overachievement estimation, the algorithm could have equally well selected the state following *Direct Payment*.

In the example, the algorithm was able to calculate the exact values for the overachievement score. This is not always the case. Sometimes the estimated overachievement score his higher and sometimes lower than the actual one. Higher than estimated overachievement scores are easy to understand: by ignoring negative effects, the size of the state increases. If one of the services actually deletes something from the state that is not part of the goal or precondition to another service, ignoring this effect makes the estimated overachievement score higher than it actually is. One example is the *Pay with Gift Voucher* service. Its deletion of the voucher after its invocation is ignored in Figure 6. Otherwise it would have a -2 instead of a -1 effect on the estimated overachievement score of the previous state.

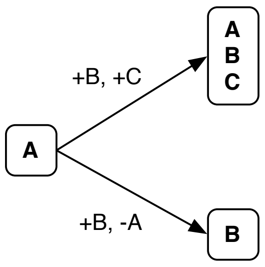


Figure 7: Overachievement estimation can be lower than actual value.

Interestingly, the possibility for lower than actual overachievement score estimations stems from ignoring negative effects, too. By ignoring the negative effects during heuristic calculation, we might come up with a simpler solution than actually possible. During the calculation of the heuristic this is not apparent. Take a look at Figure 7. It shows three states and transitions from one of them to the two others. The letters inside the states are the facts of the state and the transitions are labeled to indicate what is happening: $+X$ means that X is added and $-X$ means that X is removed from the state. If our goal now was to achieve a state in which both A and B are true, only the top-right state is a goal state. It has an overachievement score of 1. But during the calculation of the overachievement heuristics, negative effects are ignored. Hence, the lower-right state seems to be a goal, too. But it has a score of 0. The heuristic would calculate 0 also as the overachievement heuristic for the initial state.

4.2 Preference of Composition Size

Preferring minimal overachievement over composition size means that we can guarantee optimality for neither. Another approach is to prefer composition size: the search strategy is primarily guided by distance estimations and only to choose among several states with the same estimated distance, we choose the one which promises smaller overachievement. Using an optimal search strategy like $A*$, we are guaranteed to find optimal solutions with regard to composition size.

In Figure 6, preference among both heuristics did not matter. The result would gave been the same. In the first step, two states are equally good with regard to composition size: the states following *Pay with Gift Voucher* and *Direct Payment*. Using the overachievement heuristic the algorithm would have chose the state following *Pay with Gift Voucher*. But if the service landscape contains a *Cash on Delivery* service that ships the order to the customer and collects the money on delivery, the result would differ. While, we would still use *Pay with Gift Voucher* with minimal overachievement preference, for composition size preference we would now use the new service. If the customer wanted to pay with the provided gift voucher, this is clearly not what he intended. But it is the smallest possible composition (consisting only of *Package Order* and *Cash on Delivery*).

This means that when preferring composition size, one will normally not only get smaller composition but also compositions with higher consequently overachievement scores. But, on rare occasions, it is actually possible that one will get compositions with lower overachievement scores by using composition size preference. This can be the case if ignored negative effects trick the heuristic into assuming a larger composition has a lower overachievement score than a shorter one.

4.3 Achieving Better Overachievement Score Estimations

Negative effects are ignored during the calculation of both heuristics. One possible extension for the calculation of the overachievement heuristic is to use the real effect of the solution extracted from the planning graph. The planning graph is still build and the solution is still extracted from the relaxed problem (negative effects are ignored). But when calculating the state reached by the composition, negative effects are taken into account. This has three implications. First, overachievement scores will always be lower or equal to the ones where negative effects were ignored entirely. Secondly, the overachievement estimations can still be lower, higher, or equal to the actual overachievement. And finally, the calculated state that is reached might not actually satisfy the goal. This is the case if service invocations delete facts required in the goal. Three solutions are possible to calculate the overestimation score for such states:

- Follow the second condition of Definition 6. States have an infinite score.
- Calculate the score for the state even though it does not satisfy the goal.
- Calculate the score for the state reached without negative effects.

We have not done any experiments on which approach is better. But the first approach might cut away too many viable paths in search space. The second approach might favor paths that will not lead to the goal (dead ends). The third approach seems to be most promising. But it is mixing and comparing scores reached through different calculations. This potentially leads to strange and hard to understand interactions among them.

5 Related Work

To the best of our knowledge, the notion of composing for minimal goals is new. The Web Service Modeling Ontology (WSMO) [17] separates information space and real world effects. In service matchmaking Paolucci et al [13] distinguish degrees of match: exact, plug in, subsumes, and fail. Their approach differs from ours mainly because it is concerned with finding services and ranking them accordingly whereas we create service composition that rank high. Other matchmaking approaches support more elaborate rankings but are still concerned with finding and not creating service (compositions).

Related are also the numerical state variables of for example Metric-FF [7]. Like our approach, Metric-FF extends the Enforced Hill Climbing search algorithm of FF. A numerical state variables are part of logical states. Service invocations can affect the value of such variables. Numerical state variables might be an alternative to the presented overachievement heuristic. We count the number of facts in the current state not required by the goal and modify it accordingly when we add service invocations. Part of the goal then is to optimize for low values for this numerical state variable. There a two problems that prevent an easy implementation. The approach for optimization of numerical state variables in Metric-FF is based on a cost-based notion. Service invocations can only increase the value of the numerical state variable. With the overachievement score this is not the case: service invocations can increase it, decrease it, or leave it unaltered. And the algorithm does not guarantee optimality. Other planning systems supporting PDDL have similar optimization capabilities [1, 6, 16].

6 Summary

The main contribution of this paper is the introduction of the notion of overachievement minimality. It is based on the observations that in automated service composition it can be harmful to achieve more real world effects than required and that specifying all effects of correct solutions is not feasible. We extended an existing heuristic search algorithm by the capability to find solutions with smaller overachievement scores. We present two different ways to *plug* the new overachievement heuristic into the existing search algorithm. In the first approach we prefer overachievement minimality over composition size minimality, but we cannot guarantee optimality for either. With the second approach we prefer composition size minimality and can, with the correct search strategy, also guarantee its optimality. With regard to overachievement scores the first approach will normally deliver better solutions than the second one.

Our future work is aimed at improving the approach to guarantee overachievement minimality. Our first step into this direction will be to look at problems without negative real world effects. We assume that guaranteeing optimality with regard to overachievement is easier in such problems and would require a better solution extraction phase for Relaxed Graphplan that extracts solutions with minimal overachievement from the planning graph. We will also investigate the necessity of real world negative effects and how they could possibly abolished (most likely not possible) or at least be contained.

7 Project Report: SOA Governance using Tagging

The goal in introducing a SOA is to reduce logical dependencies between IT components and to close the gap between business requirements and their representation in the IT landscape [2, 5, 14]. The introduction of a service layer to decrease and control logical dependencies, however, comes at the cost of increased technical complexity. The control and steering of IT assets and their life cycle in such a complex SOA landscape is what is referred to as SOA Governance. Governance is key to the success of SOA initiatives and requires support in terms of methodologies and tools. SOA governance tools aim at regulating the life cycle of a service or other reusable IT asset from its inception to its retirement. During its life cycle a service will pass a number of steps that are all defined and controlled in the SOA governance system. An IT asset's life cycle can be divided into two phases: the providing and the consuming phase. The providing phase covers the part of the life cycle that regulates how a service goes into production. The consuming phase defines how it will be consumed once it is in production.

Discovering services is difficult in large service landscapes. Semantic Web services [11] are a promising approach to find services based on functionality. Service functionality is described through preconditions and effects. But creating them and writing queries to find services according to preconditions and effects is a complex task. Additionally, semantic service specifications only capture functional aspects. Although some standardization efforts like Web Service Modeling Ontology (WSMO) include quality of service specifications, other aspects are important as well. Functional descriptions and the service interface are in many cases not sufficient decision criteria for a service consumer. Additional information that might be important to know for a consumer are:

- In what business contexts has the service been used (successfully)?
- How often has this service been found and bound to consumers already?
- How do other users of the SOA governance system rate this service?
- What other services have been used by consumers of this service?

These questions can hardly be added to the functional description by the service provider. They depend on the usage of the service in the organization. Therefore, it is of high value to let users add their individual functional and technical descriptions to the service meta-data in an easy and informal way like tagging. Furthermore, the SOA governance system can maintain dynamic tags that depend on the usage of a service like its popularity or its performance. The advantage of letting users add their view on the service with individual tags is that the collection of tags provide a common view on the service rather than only the view of the provider. The combination of these tags make it more easy for consumers to discover and bind a service and therefore increase the value and efficiency of an SOA.

7.1 Scenario

The scenario we use is taken from a Software AG Centrasite plugin to manage the life cycle of services and other artifacts in a service landscape. As you can see in Figure 8 the service landscape consists of nine typical ERP services. With this, the landscape is rather small. But to illustrate tagging use cases it is sufficient. Another aspect of this service landscape is that its services are high-level services. A service like *Personnel Development* is rather abstract. In reality it will consist of several, more fine granular services.

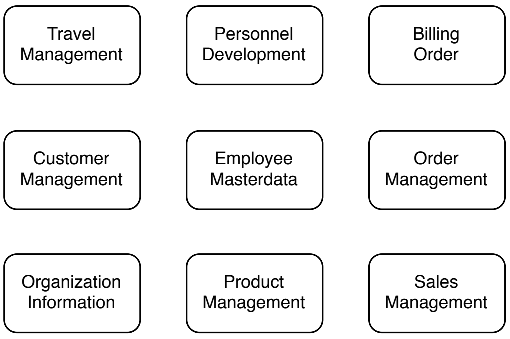


Figure 8: Service Landscape

7.2 Use Cases

The use cases for service tagging are use cases for three distinct roles: service engineer, process engineer, and service landscape manager. The service engineer develops new services. A process engineer develops new processes based on the available services and the service landscape manager is for example responsible for knowing what the service landscape looks like, updating services, or making services easy to find. In short, he manages the service landscape.

Tagging service ownership The owner of a service can be a person, a department, or an organization. The person could be the service engineer who developed the service or somebody who is responsible for the operation of the service. This can be used to know whom to contact in case of problems. Tagging service ownership on the level of departments can give information about which departments participate in a process or whether services with similar functionality are provided by different departments. Tagging service ownership on the level of individual persons is done by the service engineer and the service landscape manager. On the department level it is done by the service landscape manager. One example of a service ownership tag is a tag *HR* for *EmployeeMasterdata*, *OrganizationInformation*, ad emphTravelManagement to denote that they are provided by the human resources department.

Tagging service functionality Using tags to express service functionality is closest to semantic service specifications. But instead of expressing the functionality in formal way using preconditions and effects, tags represent just keywords on what the service does. An initial tagging of service functionality might be done by the service engineer who developed the service. But tagging service functionality gets to its real strength when it is used by service users (e.g. a process engineer) to describe what a service does for them. This can help to capture unintended usages of services as well as bridging terminology differences between service engineers and service users.

The *OrganizationInformation* service might have been developed with users from human resources in mind, but is actually also used in processes of other departments as well (e.g. to find someone responsible for an approval). This use might not be captured in the tags provided by the service engineer. The first process engineer to use the service in this manner might decide to tag this new, unintended usage to help himself or others to find the service in the future. An example for different terminology between service engineers and user could be the *TravelManagment* service: is it used to approve and book business trips or is it used by employees to request leaves of absence or vacations? Users in the human resources department can tag the service to describe what it does in their own words.

Tagging service affiliation Affiliation describes in which the business processes or the applications in which a service is used. Tagging affiliation allows impact analysis of what happens when a certain service is no longer available. Combined with information about in which system a service is implemented, the information can be used by service landscape managers to know whether a system is still necessary or which processes need to be changed if a system is shut down. Tagging service affiliation gives an overview of which services are often used together. This information can be used by process engineers to find services. Based on already selected services, they can browse the service landscape to see which services are used together with these. In an advanced version, this browsing could be done by the process modeling tool. The tool checks which services might be interesting for the current modeling situation and suggests them. Another elaboration of tagging service affiliation is to perform the tagging automatically based on modeled processes.

A slightly different form of tagging for service affiliation, is tagging negative affiliation. Negative affiliation means that services cannot be used together. Most likely, this would have to be done manually to capture experiences about the service landscape.

Tagging service quality and characteristics This use case is done by process engineers and landscape managers to capture experiences of service usage. To a smaller extend, the service engineer can tag non-functional attributes like cost, security, or physical location. This usage is closest to how semantic service specification languages see service quality. These tags are rather static and do not contain information about the actual operation of service. On the other hand, the tagging by process engineers and landscape managers is about how a service operates. This includes the performance and reliability of services.

Tagging Service Life Cycle Tagging the life cycle of service is an easy but limited approach to life cycle management. Much more elaborate approaches exist. A tag can represent a phase in the life cycle of a service. For example, each service from the scenario is available in different versions. As the versions of the services are not aligned, it is often unclear which service versions are the current ones, which are currently in development, or which are already deprecated and should not be used in new processes. The service landscape manager could assign a tag *current* to the current version of each service. As the tags of life cycle management are not predefined, the landscape manager can use his own terminology or follow an existing life cycle.

Tagging for task organization A service engineer might mark services he needs to work on (e.g. to fix bugs). The process engineer might group services he wants to use in a future process. A service landscape manager might do some *landscaping* and mark which services he still has to look at. For all three, the tags will disappear as soon as the tasks are finished and the tags will normally be of no use for others. In these two aspects, tagging for task organization differs from all other use cases.

7.3 Implementation

We integrated the tagging functionality into Centrasite as a plugin. The plugin UI is integrated into Centrasite Control. Oryx is a Web-based modeling tool developed at the Business Process Technology group at the Hasso Plattner Institute. It is currently mainly used to model business processes using the Business Process Modeling Notation (BPMN) but stencil sets exist to, for example, also model EPCs or Petri nets. The non-UI part of the tagging plugin interfaces with Centrasite using the *Java API for XML Registries (JAXR)*. Hence, this part can be used by an registry that supports JAXR. All the tags of user as well as all tags by all users form classifications schemes. The tags for an individual service are classifications.

Figure 9 shows a screenshot of the tagging plugin inside Centrasite Control. The tag cloud lists all tags and the tags are sized according to their frequency. The *deprecated* tag is the biggest as multiple, deprecated versions of each service exist. Using the *current* tag, a user can easily find the newest version of each service. The other tags are the results of the different use cases we presented in the previous section. Besides looking at the tag cloud, the plugin includes screens to edit the tags of services, list services with a specific tag, and search for services based on tags.

Figure 10 is a screenshot of Oryx. On the right hand side, it includes inspectors to access a Centrasite repository. One can browse the repository using tags and if one has found a suitable service, drag it onto the modeling canvas to integrate it into the current process. The aforementioned functionality for suggesting services based on the current modeling context, has not yet been implemented.

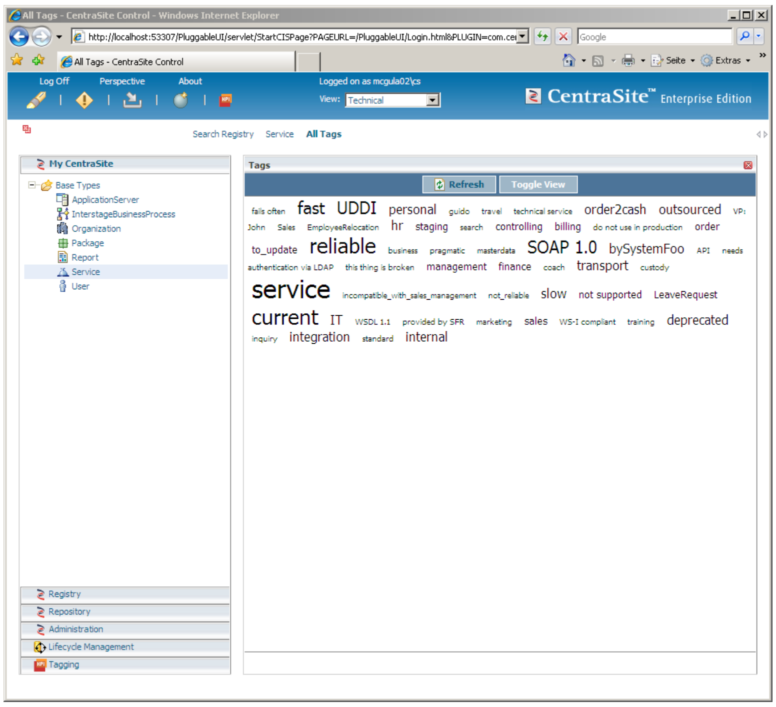

All Tags - CentraSite Control - Windows Internet Explorer
Log Off
Perspective
About
Logged on as mcgula02\cs
View: Technical
CentraSite™ Enterprise Edition
Search Registry
Service
All Tags
My CentraSite
Base Types
ApplicationServer
InterstageBusinessProcess
Organization
Package
Report
Service
User
Registry
Repository
Administration
Lifecycle Management
Tagging
Tags
Refresh
Toggle View
fails often fast UDDI personal guido travel technical service order2cash outsourced
John Sales EmployeeRelocation hr staging search controlling billing do not use in production order
to_update reliable business pragmatic masterdata SOAP 1.0 bySystemFoo API needs
authentication via LDAP this thing is broken management finance coach transport custody
service incompatible_with_sales_management not_reliable slow not supported LeaveRequest
current IT WSDL 1.1 provided by SFR marketing sales WS-I compliant training deprecated
inquiry integration standard internal

Figure 9: Tagging inside Centrasite

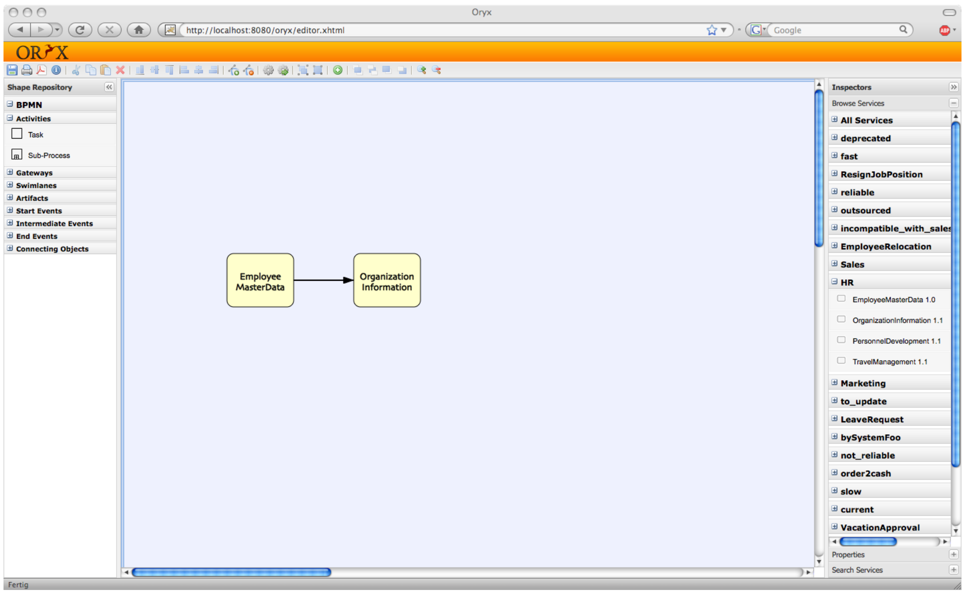


Figure 10: Tags during Process Modeling in Oryx

References

[1] Ivan Serina Alfonso Gerevini, Alessandro Saetti. Planning with numerical expressions in lpg. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI-04)*, 2004.

[2] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Web Services – Concepts, Architectures and Applications*. Data-Centric Systems and Applications. Springer, 2004.

[3] Daniela Berardi, Diego Calvanese, Giuseppe De Giacomo, and Massimo Mecella. Composition of services with nondeterministic observable behaviour. In *Proceedings of the Third International Conference on Service-Oriented Computing*, volume 3826 of *Lecture Notes In Computer Science*, pages 520–526, Heidelberg, 2005.

[4] Avrim Blum and Merrick Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90:281 – 300, 1997.

[5] Steve Burbeck. The tao of e-business services. *IBM developerWorks*, 2000.

[6] Alfonso Gerevini, Alessandro Saetti, and Ivan Serina. Planning through stochastic local search and temporal action graphs. *Journal of Artificial Intelligence Research*, 20:239 – 290, 2003.

[7] Jörg Hoffmann. Metric-FF planning system: Translating ”ignoring delete lists” to numeric state variables. *Journal Of Artificial Intelligence Research*, 20:291 – 341, 2003.

[8] Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253 – 302, 2001.

[9] Stephen Cole Kleene. *Mathematical Logic*. Dover Publications, 2002.

[10] Dominik Kuropka, Peter Tröger, Steffen Staab, and Mathias Weske. *Semantic Service Provisioning*. Springer, 2008. (to appear).

[11] Sheila A. McIlraith, Tran Cao Son, and Honglei Zeng. Semantic web services. *IEEE Intelligent Systems*, 16(2):46–53, 2001.

[12] Harald Meyer and Mathias Weske. Automated service composition using heuristic search. In *Business Process Management (BPM 2006)*, volume 4102 of *Lecture Notes In Computer Science*, pages 81–96, Heidelberg, 2006. Springer.

[13] Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia P. Sycara. Semantic matching of web services capabilities. In *Proceedings of the First International Semantic Web Conference on The Semantic Web*, pages 333–347, London, UK, 2002. Springer-Verlag.

[14] M. P. Papazoglou and D. Georgakopoulos. Service-oriented computing: Introduction. *Communications of the ACM*, 46(10):24–28, 2003.

[15] Marco Pistore, F. Barbon, Piergiorgio Bertoli, D. Shaparau, and Paolo Traverso. Planning and monitoring web service composition. In *Workshop on Planning and Scheduling for Web and Grid Services (held in conjunction with The 14th International Conference on Automated Planning and Scheduling*, pages 70 – 71, 2004,.

[16] Evren Sirin, Bijan Parsia, Dan Wu, James Hendler, and Dana Nau. Htn planning for web service composition using shop2. *Journal of Web Semantics*, 1(4):377 – 396, 2004.

[17] `http://wsmo.org`. *Web Service Modeling Ontology*, 2005.

[18] Liangzhao Zeng, Boualem Benatallah, Hui Lei, Anne Ngu David Flaxer, and Henry Chang. Flexible Composition of Enterprise Web Services. *Electronic Markets – Web Services*, 13:141–152, 2003.

Business Process Model Abstraction and Flexible Process Graph

Fall 2008 Workshop

Artem Polyvyanyy
Artem.Polyvyanyy@hpi.uni-potsdam.de

This report summarizes research activities for the period from April 2008 to October 2008—the second six months at HPI Research School. In this report we present results achieved in two research fields of investigation: business process model abstraction and novel approach for modeling ad-hoc business processes—flexible process graph (FPG).

1 Business Process Model Abstraction

Business process models are the instrument facilitating business process management task in modern companies. Every model is a representation of a business process used by a certain group of stakeholders. The desired level of model granularity depends on a stakeholder and a current task. Top level management prefers coarse grained process descriptions facilitating fast and correct business decisions, while employees directly executing processes appreciate fine granular specifications of working procedures. Thus, it is a common situation when a company maintains several models for one business process. To ease the maintenance, modeling notations like Business Process Modeling Notation (BPMN) [8] or Yet Another Workflow Language (YAWL) [1, 2], allow hierarchical model structuring. A model hierarchy permits organizing process details at different abstraction levels. Unfortunately, these approaches require considerable effort when a process model is changed: keeping separate models consistent as well as preserving inter subprocess dependencies is laborious. Different approaches of presenting significant process model information to a user are discussed in [5–7, 11, 16]. An alternative approach is to derive coarse grained process models from the existing detailed models on demand. This technique can be referred to as a process model abstraction.

Abstraction is generalization that reduces the undesired details in order to retain only information relevant for a particular task. Abstraction mechanisms are used in many domains where users suffer from information overload. One of the most well-known examples is cartography, where geographical maps visualize landscapes on different scales. While a map of a particular town provides detailed information on houses and side streets, the world map captures shapes of continents, main river contours, and marks locations of the largest cities. To stay useful to a reader large scale geographical maps reduce the level of details, but are based on the information derived from the detailed maps. Process model abstraction goal is to produce a model containing significant information based on the detailed model specification.

The ideas presented in this report emerged from a joint research project with AOK Brandenburg—the health insurance company in Teltow, Germany. The operational processes of the company are captured in about $4\,000$ event-driven process chains (EPC) [10]. The goal of the project is to derive methods of automated abstractions from process model details. In [14] we have discussed the developed abstraction mechanisms. This report focuses on the method providing a user control over the abstraction—an abstraction slider.

1.1 Process Model Abstraction

In this section we provide basic definitions, describe several motivating abstraction scenarios, and derive abstraction criteria that can be employed for controlling the process of model abstraction.

1.1.1 Fundamentals

Let us start with the definition of a process model adopted from [18].

Definition 1 $(N, E, type)$ is a *process model* if:

- $N = N_A \cup N_E \cup N_G$ is a set of nodes where $N_A \neq \emptyset$ is a set of activities, N_E is a set of events, and N_G is a set of gateways; the sets are mutually disjoint
- $E \subseteq N \times N$ is a set of directed edges between nodes representing the control flow
- (N, E) is a connected graph
- $type : N_G \rightarrow \{and, xor, or\}$ is a function that assigns to each gateway a control flow construct.

Given Definition 1 we define a business process model abstraction as a function performing a process model transformation.

Definition 2 A *business process model abstraction* is a function $A : P \times S \rightarrow P$, such that:

- P is a set of process models
- S is an abstraction setting, $S \subseteq C \times \mathbb{R}$:
  - $C \subseteq T \times \{asc, desc\}$ is a finite set of abstraction criteria, where T is a set of abstraction criteria types, asc indicates that higher criterion values are of higher significance, $desc$ indicates that lower criterion values are of higher significance

  - $\mathbb{R}$ is the set of real numbers; an element of this set is the criterion value distinguishing significant elements from insignificant

- if $p' = A(p, s)$, where $p, p' \in P$, $s \in S$, $p = (N, E, type)$, $p' = (N', E', type')$, then $|N'| \leq |N|$.

An abstraction process transformation must not increase the number of model nodes. The parameters of an abstraction function are a process model and an abstraction setting. An abstraction setting defines a subspace of abstraction criteria values and, thus, puts restriction on the elements which should appear in the abstracted process model. Only model elements conforming to the abstraction setting should remain in the resulting model. An abstraction criterion is a pair, where the first element is a criterion type and the second element is a hint specifying the relation between the element criterion value and the element significance. An example of the abstraction criterion can be a pair (*activity execution cost*, *asc*), i.e., the higher the execution cost, the higher the activity significance in the model.

Abstraction task implies answering its *what* and *how*:

- What parts of a process model are of low significance?
- How to transform a process model so that insignificant parts are removed?

Answers to both questions should address the current abstraction context, i.e., a business task a user solves at the moment. The choice of an abstraction setting answers the *what* question. A concrete abstraction function implementation answers the *how* question.

1.1.2 Abstraction Scenarios

In this report we aim at learning common principles of process model abstraction. Let us introduce several process model abstraction use cases. The use cases presented are the starting point for analysis and understanding of the abstraction problem.

A business process model analyst might be interested in activities which are executed frequently in a process. Such activities are of high importance, since they noticeably influence execution time and cost of a business process. Consequently, these activities play an important role in such tasks as business process optimization and reengineering.

Alternatively, an analyst can be interested in activities that consume more time in comparison to other process activities. These activities contribute a large share to the overall process execution time and are natural candidates for being studied during the task of process improvement. Once such an activity is optimized, the overall process execution time might drop considerably. Besides, in some situations the execution cost is proportional to the execution time.

Activity execution cost and overall process execution cost are crucial properties of a business process. Since an activity cost has a direct influence on the overall process cost, identification of activities with high costs is another scenario.

Abstractions that reduce insignificant process instances constitute another set of abstraction scenarios. In these scenarios properties of process instances are used as abstraction criteria. For example, one might be interested in “typical” executions of a business process model. A typical execution means that among all possible ways of a business process completion it is the one that is executed most often. Abstractions of this type result in process models describing only process instances which are often observed. Similarly, process instances with the highest duration or cost may be in the focus of process abstraction task. These abstractions result in a process model representing either most time consuming or most “expensive” process instances.

1.1.3 Abstraction Criteria

Abstraction criteria help to tell significant process model elements from insignificant. Abstraction criteria are properties of model elements or model fragments that enable elements comparison and allow identifying information relevant for the task at hand. Analysis of the business scenarios shows that different abstraction criteria can be used for the task of business process model abstraction. A choice of an abstraction criterion or a set of criteria is problem specific. The following abstraction criteria can be derived from the aforementioned scenarios.

Relative probability (p_r) of reaching a process node n from its direct predecessor n_p is the probability of an edge transition from n_p to n, $p_r : \{(n_p, n) \in E\} \rightarrow [0, 1]$.

Mean occurrence number of a node (m_i) is the mean number that the node i occurs in a process instance.

Relative effort of a process activity (e_r) is time required to execute the activity, $e_r : N_A \rightarrow \mathbb{R}^+$.

The relative effort of an activity is measured in time units (e.g., minutes or hours) and quantitatively coincides with the activity duration. However, semantically the effort concept is close to the concept of cost. For instance, if two activities are executed in parallel their total effort is the sum of efforts of both activities.

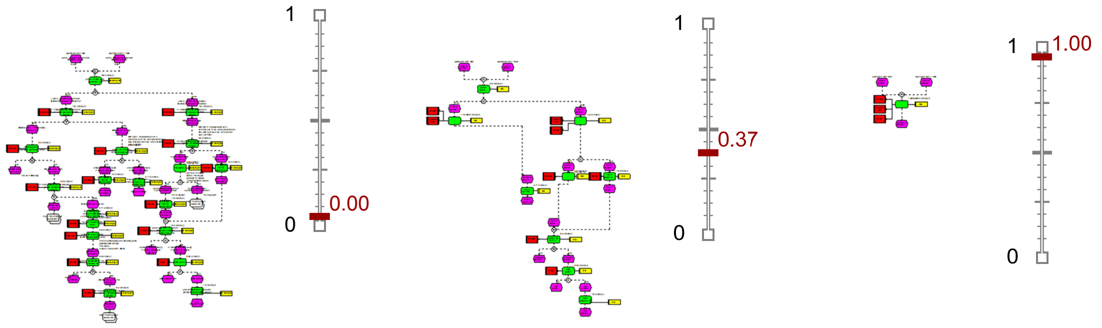
Absolute effort of a process activity (e_a) is the mean effort contributed to the execution of the activity in a process instance, $e_a : N_A \rightarrow \mathbb{R}^+$. Absolute effort can be obtained as the product of relative effort and the mean occurrence number of the activity.

In addition to properties of process model activities, properties of other model elements can be used as abstraction criteria. One can use properties of process instances as abstraction criteria. A model abstraction based on such a criterion preserves significant process instances in a model. Following, we define abstraction criteria relevant to process instances.

Probability of a process instance (P_i) is the probability of a process instance i to happen within a process execution.

Effort of a process instance (E_i) is the effort to be invested in the execution of a process instance i and can be found as the sum of efforts of all the activities executed within this instance.

The proposed list of abstraction criteria does not claim to be a complete one. It can be extended once there is a demand for new abstraction scenarios.



(a) Initial process model (b) Abstracted model with the slider set to 0.37 (c) Abstracted model with the slider set to 1.00

Figure 1: Process model abstraction slider (unreadability intended)

Each abstraction assumes that a process model contains information required for the abstraction procedure or data from which this information can be derived. For instance, an abstraction using relative probability as abstraction criterion requires a process model to possess information about edge transitions. However, most process modeling notations, such as EPC or BPMN, can be extended to allow enriching models with such concepts as probability of edge transition, activity execution time, or activity mean occurrence number.

1.2 Abstraction Slider

In this section we focus on the *what* question of the process abstraction. We use a slider metaphor to propose an approach enabling flexible control over process model abstraction. It is shown how the slider can be employed for distinguishing significant process model elements from insignificant ones. We provide examples demonstrating the approach and illustrating that the slider works effectively with different abstraction criteria.

1.2.1 Slider Concept

Once an abstraction criterion is selected, the required level of abstraction should be specified. Since the desired level of detail cannot be predicted without a priori knowledge about the abstraction context, a decision about a suitable abstraction level is postponed to the moment when there is a demand for a concrete model. Ideally, a user should be able to change an abstraction level continuously within the whole range from an initial detailed process model to a process model containing only one activity. This activity, bounding the abstraction level above, semantically corresponds to the whole process. A model abstraction exhibiting such a behavior can be controlled by an *abstraction slider*.

The slider concept is employed in many engineering systems, where a controlled parameter has to be changed smoothly. Numerous examples of a slider can be found in IT systems. For instance, this control is used in modern geographic information systems (GIS), where a user controls map scale by means of a zoom slider. A slider is a simple entity that can be formalized as follows.

Definition 3 A *slider* is an object that can be described by:

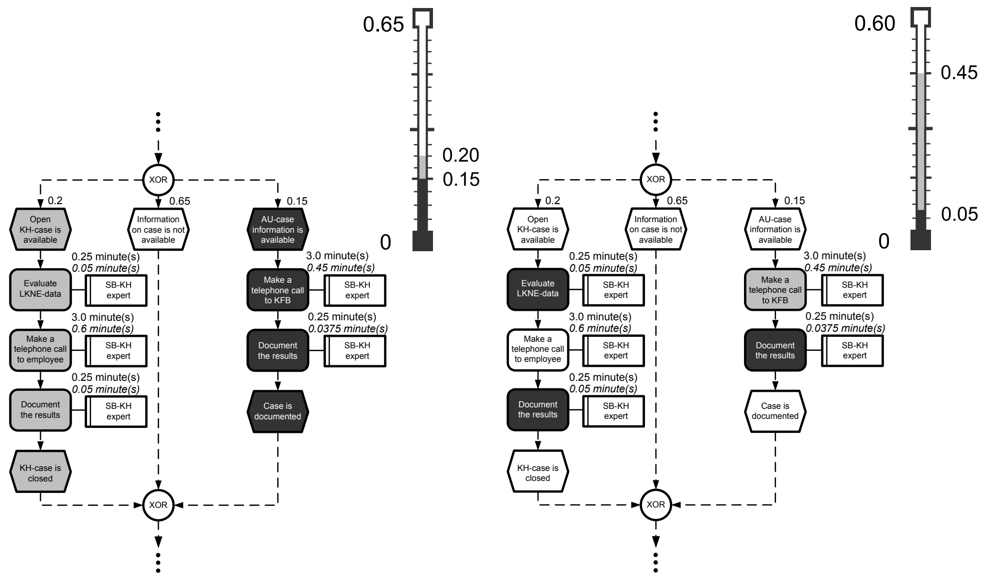
- $[S_{min}, S_{max}]$—a slider interval with a minimum value S_{min} and a maximum value S_{max}
- $s \in [S_{min}, S_{max}]$—a slider state.

Every abstraction criterion discussed in this report (see section 1.1.3) has a quantitative measurement. Therefore, a partial order relation holds for criterion values. Since criteria describe elements of a process model, these elements can be ordered according to the selected criterion. For instance, if activity relative effort is used, an activity taking two minutes precedes an activity taking four minutes. The partial order relation enables element classification. One can choose a value splitting the set into two classes: elements which criterion value is less than the specified value and elements which criterion value exceeds it. Elements of the first class are assumed to be insignificant and should be omitted in the abstracted model, while elements of the other class are significant and should be preserved. A value according to which elements are classified is called an *abstraction threshold*. In the example, an abstraction threshold of three minutes results in the two minutes activity to be assumed insignificant and to be reduced, while the four minutes activity is significant and is preserved in the abstracted process model. Thus, a process model abstraction slider is a function which for a given process model fragment and a specified threshold value tells if this fragment is significant or not. According to the slider definition, an *abstraction slider* is a slider with the slider interval defined on an interval of abstraction criterion values and the slider state associated with the current threshold.

1.3 Abstraction Slider Examples

Figure 1 illustrates application of a slider to control a business process model abstraction. In the example the abstraction criterion is activity absolute effort. Activities with higher absolute efforts are considered to be more significant, i.e., *asc* ordering is used. The business process is captured in EPC notation. Figure 1(a) presents the initial process model. The business process model corresponds to the case when the slider state is 0.00, i.e., no activities are reduced. If the slider state changes to 0.37, the model shown in Figure 1(b) is produced. As a result of abstraction more than 50% of the nodes are reduced. When the slider state is set to 1.00, the process model degenerates into one activity (see Figure 1(c)). Every abstracted business process model contains only elements which properties exceed the specified threshold. Therefore, elements of an abstracted model are more homogeneous in relation to a used abstraction criterion.

From a user perspective a slider control regulates the amount of elements preserved in a business process model. The slider state is directly associated with the



(a) Abstraction criterion is relative probability (b) Abstraction criterion is absolute effort

Figure 2: Process model abstraction sliders with different abstraction criteria

threshold value, classifying model elements into significant and insignificant. In the simplest case a user specifies an arbitrary value used as a threshold (which means that the slider interval is $[-\infty, +\infty]$). An obvious drawback of this approach is that a user has to study a process model thoroughly in order to provide a helpful threshold value. A low threshold value makes all the elements in a process model to be treated as significant, i.e., no nodes or edges are reduced. On the other hand, a threshold which is too high may lead to reduction of the whole process model to one activity. A process model containing one activity provides such a small amount of information about a business process that the abstracted model becomes useless. To avoid confusing situations, the user should be guided by an interval in which all the "useful" values of abstraction criteria lie.

Alternatively, the abstraction slider can control a share of nodes to be preserved in a model. Since abstraction mechanism possesses information about the model element properties, it is always possible to estimate the threshold value which results in the reduction of the specified share of the process model.

As we have mentioned, an abstraction slider can manage abstraction process based on various criteria. Depending on the chosen criteria and the current slider state, abstraction results in different process models. Consider two examples from Figure 2. In both cases the same fragment of an EPC is shown. The fragment presents an exclusive choice taking place during execution of an operational process in health insurance industry. The process model is enriched with the information about the probabilities of connection transitions. Each function has two labels: function relative effort and function absolute effort (in italic). Figure 2 shows what parts of this fragment are considered to be significant depending on the selected abstraction criterion and the slider state.

Two different criteria are used: relative probability (Figure 2(a)) and activity absolute effort (Figure 2(b)). Color coding is used to show correspondence between the range of the slider state change and the elements which are considered to be significant within this range. Let us assume that activity absolute effort is considered as criterion and the slider state changes in the range between 0.05 and 0.45 (colored with light gray). Function "Make a telephone call to KFB" is considered significant in this range, while functions "Evaluate LKNE-data" and "Document the results" are insignificant. At the same time function "Make a telephone call to employee" is significant till slider state exceeds 0.60. Figure 2 vividly visualizes the importance of abstraction criteria choice: the coloring of process fragments substantially differs from one case to another.

1.4 Process Model Transformation

In this section we address the *how* question of the process model abstraction task. We base our solution on process model transformation and reduction rules. Reduction and transformation rules are widely used for analysis of process models and have been extensively studied in literature [9, 15]. In this section two classes of abstraction rules are introduced: elimination and aggregation. Afterwards, requirements for abstraction and their influence on the transformation rules are discussed. We argue when each of the techniques is appropriate. Finally, an example of an abstraction approach is presented.

1.4.1 Elimination and Aggregation

Once it is known which elements of a process model are insignificant, they have to be abstracted from. Different techniques can be used to reduce insignificant elements. We distinguish two approaches: elimination and aggregation.

Elimination means that an insignificant process model element is omitted in the abstracted process model. As a result of elimination a model contains no information about the omitted model element. Elimination can be seen as the simplest abstraction method. Although, it still requires rules assuring that the process model is well-formed and preserving the ordering constraints of the initial model.

Aggregation implies that insignificant elements of a process model are aggregated with other elements. In contrast to elimination, aggregation allows preserving information about the abstracted element in the model. If two sequential activities are aggregated into one activity, the properties of the new activity comprise properties of the aggregated activities. For instance, the execution cost of an aggregating activity can be defined as the sum of execution costs of aggregated activities.

An abstraction approach can be based on the exclusive usage of elimination or aggregation; combination of both techniques is also possible. Elimination can be seen as the simplest technique, since it requires only the rules of correct elements dropping. However, elimination is insufficient in many cases. Aggregation requires more sophisticated specification of how the properties of the aggregated elements influence properties of aggregating elements. The choice of an abstraction methodology depends on the requirements imposed on the abstraction.

1.4.2 Transformation Requirements

An essential requirement for a process model abstraction is preserving the process execution logic: neither new ordering constraints should be introduced, nor the existing ones should be changed. Process transformation rules that satisfy this requirement are discussed in [11].

Further, one may formulate additional requirements on abstraction rules. If a company uses process models for estimation of the workforce required to execute business processes, information about the absolute effort of process execution should be preserved in a process model. Abstractions which preserve process properties are called *property preserving abstractions*. In this particular case *effort preserving abstraction* is discussed. If an abstraction must be property preserving, elimination is not sufficient: once a model element is omitted all the information about its properties is lost. Within a property preserving abstraction elimination can be applied only to those elements which do not influence the property being preserved.

It is an additional requirement for any abstraction to produce well-formed abstracted process models. Thus, features of modeling notations should be taken into account by transformation rules. As a consequence, we can expect different rules to be used, e.g., for EPC and BPMN.

Every requirement which is imposed on an abstraction restricts transformation rules. It could be the case that an insignificant model element cannot be reduced, because of the too restrictive set of rules. Assume an effort preserving abstraction should be performed. If there is an activity to be reduced and the abstraction does not specify a rule how to handle the given activity (so that the process absolute effort is preserved), this activity should be preserved in the model. In this sense an important finding is to show which class of process models can be abstracted to one activity by a given set of rules. As we have argued, not every set of rules allows this. An abstraction which is not capable of reducing a process model to one function is called *best effort abstraction*, since it only tries to assure that a given process model is abstracted to the requested level using the given set of rules.

2 Flexible Process Graph

Businesses document their operational processes as process models. The common practice is to represent process models as directed graphs. The nodes of a process graph represent activities and directed edges constitute activity ordering constraints. A flexible process graph modeling approach proposes to generalize process graph structure to a hypergraph. Obtained process structure aims at formalization of ad-hoc process control flow. In this report we discuss aspects relevant to concurrent execution of process activities in a collaborative manner organized as a flexible process graph. We provide a real world flexible process scenario to illustrate the approach.

2.1 Foundations

In this section we briefly present the main concepts of FPG. FPG was first introduced in [13] and is a formal way for representing ad-hoc process control flow. In the core of FPG lies generalization of a directed process graph edge which defines a sequential execution of adjacent activities. In mathematics, generalization of a graph is a hypergraph [3, 4]. Hypergraph edges (hyperedges) are arbitrary sets of nodes. Thus, a hyperedge is an edge that can connect multiple activities. As opposite to a graph-based sequence control flow pattern, it is allowed that within a hyperedge a process participant can choose which activity to execute next. A process model becomes hypergraph, rather than graph-structured:

Definition 4 *A flexible process graph (FPG) is a triple (A, E, T) where:*

- *A is a finite set of activity nodes*
- *E is a finite set of edges $e = \langle I(e), O(e)\rangle \in E$, $A \cap E = \emptyset$*
    - *$I : E \to \mathcal{P}(A)$ is a function defining edge input activities*
    - *$O : E \to \mathcal{P}(A)\backslash\emptyset$ is a function defining edge output activities*
    - *$\forall e \in E : I(e) \cap O(e) = \emptyset$*
- *T is an edge type function, $T : E \to \{and, xor, or\}$.*

Each edge $e \in E$ in FPG is split into two subsets of input $I(e)$ and output $O(e)$ activities to obtain a *directed* hypergraph. Unlike regular graph-structured process models that contain special routing nodes—gateways, FPG introduces edge types that implement routing decisions. The structure of FPG is fixed and does not change during execution of a process instance. Dynamics of a process represented as FPG is specified by process state transitions:

Definition 5 *A state of a flexible process graph (A, E, T) is defined by a state function $S : A \to \mathbb{N}_0 \times \mathbb{N}_0$ mapping a set of activity nodes onto the pairs of natural numbers including zero $(\mathbb{N}_0 = \mathbb{N} \cup \{0\})$.*

When in a certain state, each activity node $a \in A$ of FPG is assigned two numbers $S(a) = (\omega, \beta) \in \mathbb{N}_0 \times \mathbb{N}_0$. $S_\omega(a) = \omega$ (*white* tokens) specifies the number of instances of activity a that need to be accomplished from now on in the process instance. Respectively, $S_\beta(a) = \beta$ (*black* tokens) specifies the number of activity instances so far accomplished in the process instance.

2.1.1 Process Instantiation

FPG process initialization is performed in two steps: 1. $S(a)$ is set to $(0,0)$ for all $a \in A$, 2. For each activity $a \in A$ the initial enabling is performed. An activity a is enabled at process start if $\epsilon^*(a)$ holds:

$$\epsilon^*(a) = \exists e \in E : a \in O(e) \wedge I(e) = \emptyset \wedge cond(e,a)$$

The $cond$ predicate implements edge type $t \in T$ routing decisions (e.g., $\forall a \in O(e) : cond(e,a) = true$, if $T(e) = and$). If $\epsilon^*(a)$ holds, the process state S is modified to give S', such that $S'(a) = S(a) + (1,0)$.

2.1.2 Activity Firing

An activity $a \in A$ can fire in an FPG process instance if it is enabled ($S_\omega(a) > 0$). Activity firing results in the process state S change to S', such that $S'(a) = S(a) + (-1,1)$, i.e., one white token gets painted black. Activity firing is instantaneous, consumes no time, and indicates a completion of the corresponding activity. After activity a has fired, the activity enabling has to be performed on a set composed of output activities of a: $\bigcup_{\{e \in E | a \in I(e)\}} O(e)$.

2.1.3 Activity Enabling

An activity $a \in A$ can be enabled after execution of an activity a_β if $\epsilon(a_\beta, a)$ holds:

$$\epsilon(a_\beta, a) = \exists e \in E \forall a_i \in I(e) : a_\beta \in I(e) \wedge a \in O(e) \wedge S_\beta(a_i) \geq S_\beta(a_\beta) \wedge cond(e,a)$$

An activity a enabling depends on execution of the preceding activity, e.g., a_β. An activity a can be enabled if there exists an edge $e \in E$, such that a is the output activity of e and a_β is the input activity of e. Further, for each input activity a_i of the edge e it holds that the number of accomplished instances of a_i is at least the number of accomplished instances of a_β. Also, the edge e type $t \in T$ condition must hold. If $\epsilon(a_\beta, a)$ holds, the process state S is modified to result in state S', such that $S'(a) = S(a) + (1,0)$.

2.1.4 Process Termination

A process instance terminates when there is no activity to execute, i.e., no activity is enabled ($\forall a \in A : S_\omega(a) = 0$).

2.2 From Formalism to Real World Business Processes

In this section we formally define the mechanism of activity assignment to different process participants and address the FPG activity concept as a time lasting phenomenon.

2.2.1 Process Roles

Activities in business processes are either automated by software systems or executed manually by people. Following, we discuss aspects concerning the assignment of process activities to agents that actually execute them. For the sake of simplicity we abstract from differentiating human and software agents and refer to them as roles. Each role is a sequential system, i.e., can be in the process of execution of only one activity at each moment in time. Therefore, concurrent activity execution can only be achieved by several roles executing different activities. Following, we formally define process role assignment:

Definition 6 *A flexible process graph $FPG = (A, E, T)$ role assignment is a pair (R, W) where:*

- *R is a finite set of roles*
- *$W : A \to \mathcal{P}(R)\backslash\emptyset$ is a roles assignment function.*

Each activity in FPG must have at least one role assigned. Each activity in FPG can be associated with several roles. Once enabled, an FPG activity $a \in A$ can only be executed by a role $r \in W(a)$.

During FPG process instance execution, each participating role can observe a subset of activities currently available for execution by the role—a role task list. By selecting and executing an activity from the proposed list the role contributes to the achievement of a process goal. The assignment of roles to FPG activities allows us to formally define a concept of a role task list.

Definition 7 *A role task list for the role $r \in R$ from the role assignment (R, W) for the flexible process graph $FPG = (A, E, T)$ is a function L, where:*

- *$L : R \to \mathcal{P}(A)$ is defined on a subset of FPG activities*
- *$L(r) = \{a \in A | r \in W(a) \wedge S_\omega(a) > 0\}$, where $r \in R$.*

Thus, a role task list is a subset of enabled activities of the FPG that are assigned to a certain role. Note, that a process participating role can consult on the number of enabled activity instances pending for execution by referring to the FPG state function S (cf. Definition 5).

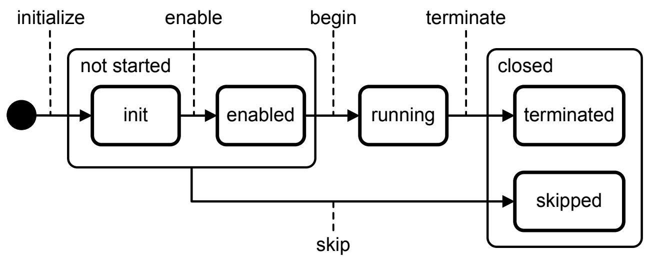


Figure 3: Simple activity instance state transition diagram (adopted from [19])

2.2.2 Modeling Parallelism

So far we have presented FPG as the mechanism to define flexible activity enabling scenarios. Similar to Petri net [12] transitions, activity firing in FPG consumes no time. However, in real world scenarios instantiated business processes consist of activity instances that actually take time. Each activity instance is represented by its state transition system. Following, we discuss issues relevant to the interpretation of FPG when providing a structure to a process on a set of activities that can not be assumed instant by nature.

Figure 3 shows a simple activity instance state transition diagram. When an activity instance is created it enters the *init* state. The enable state transition transfers the activity to the *enabled* state. Before an activity instance enters the *running* state it can still be *skipped* by the skip transition. An enabled activity instance can begin and enter the *running* state. Once accomplished, an activity instance enters the *terminated* state.

[19] also proposes a complex model of the activity instance state transition system. It allows an enabled activity instance to get *disabled* for some period of time. Also, a running activity instance can get *suspended* and afterwards return to the running state. Finally, the closed state in addition to terminated or skipped can also be *failed*, *undone*, or *cancelled*.

It is always possible and is allowed to come up with other state transition systems to represent activity instances. Therefore, instead of focusing on a one particular solution we rather state a list of generic requirements we expect any activity instance state transition system to fulfill if designed to be used as an FPG building block. An activity instance internal state transition system must contain the following generic states:

- *enabled* state—a state which means that the activity instance has to be accomplished in the process instance in order to realize the process goal
- *running* state—a state signals that work is currently conducted for the purpose of accomplishing the activity instance
- *terminated* state—a state which means that the activity instance was accomplished for the purpose of reaching the process goal.

Additionally, an activity instance state transition system must allow only a strict order on proposed activity states: first enabled, then running, and finally the terminated state.

Once an activity is in one of the proposed states it can not return to the previous one given by the order. However, other states might be injected in between, e.g., an enabled activity can be disabled for some period of time or a running activity can be suspended and afterwards returned back to the running state.

The state transition diagram from Figure 3 satisfies the proposed requirements. It contains enabled, running, and terminated states and does not allow any scenarios that are forbidden by the proposed ordering constraints. Note, that one can decide on desired behavior by selecting appropriate mapping of generic states, e.g., one might decide to map the closed or the terminated state from Figure 3 onto the generic terminated state.

Once a direct correspondence between the generic activity instance states and the concrete activity implementation states is done, one can automatically map FPG transition states onto activity instance states. The generic enabled activity instance state corresponds to the FPG activity enabling and the generic terminated state corresponds to the FPG activity firing (cf. section 2.1). Thus, once an FPG activity is enabled following the FPG execution semantics a new activity instance should be transferred to the generic enabled state. Once an activity instance is terminated, has reached its generic terminated state, the corresponding FPG activity should fire to mark the FPG process state transition. The impact of the decision of a mapping between concrete and generic activity instance states should become clear now. In case we decide to map the generic terminated state onto the terminated state from Figure 3 the decision to skip the activity will not trigger the FPG state transition. Alternatively, if decided to accept the closed state as the generic terminated state, the FPG state transition will be triggered regardless of actually performing some work on accomplishing an activity instance or skipping it.

There is no direct mapping of the generic activity instance running state onto the FPG formalism. However, there is an additional constraint that no two activities executed by one role can be in the running state. We have already presented a concept of roles (cf. section 2.2.1). A role is a sequential system capable of executing assigned process activities. A role can consult its task list (cf. Definition 7) prior of selecting an activity for execution. Once started with the selected activity (entered the generic running state) the role should not be able to work on other activities. Only when the running state of the activity instance is left, the role can proceed with other activities.

In the simplest case it should be restricted that only one assigned role can execute an activity and that once started with the activity execution the role should accomplish it and bring it to the generic terminated state. However, more sophisticated scenarios can be envisioned. A role can suspend current activity execution in order to switch to another enabled activity and then return to the execution of the prior activity. Also, one can think of scenarios where several roles collaboratively accomplish an activity, i.e., several roles select the same activity for execution from their role task lists.

2.3 Flexible Business Process Scenario

In this section we present a real world flexible business process scenario. We show how this scenario can be formalized as a FPG.

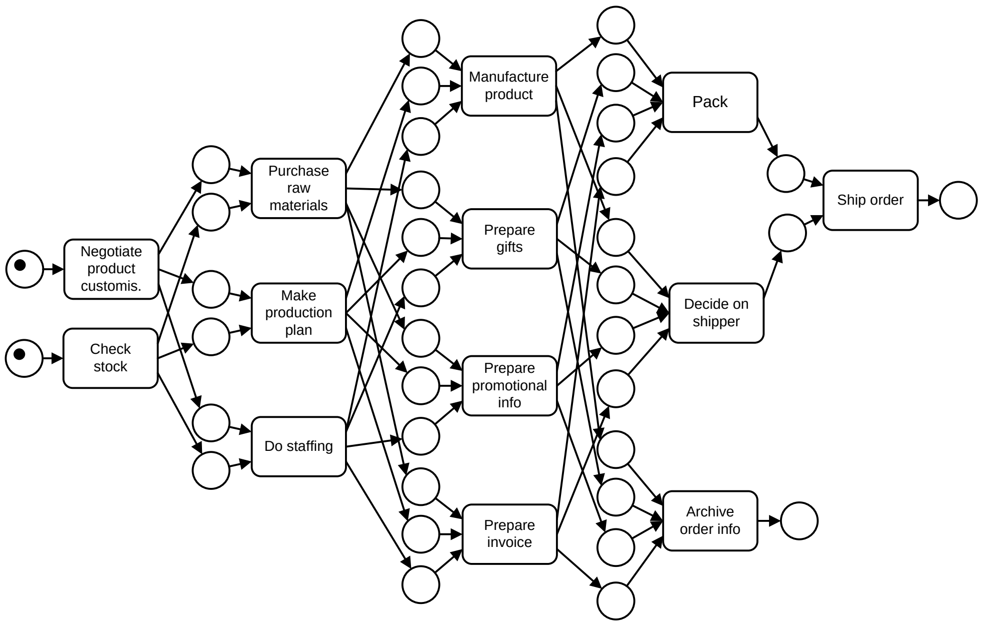


Figure 4: Petri net model that captures flexible business process scenario

The scenario describes a process of customizing, preparing, and shipping an order to a customer. The business process is decomposed to activities as follows. Once an order is confirmed by the customer, "*negotiate product customization*" (NPC) activity takes place. In the scenario we do not concentrate on a specific product but assume a generic one, e.g., this can be a backpack. Following, "*check stock*" (CS) activity takes care of determining whether all basic materials are available to realize the order. One can "*purchase raw materials*" (PRM) required to customize a product, "*make production plan*" (MPP), and "*do staffing*" (DS) by assigning responsible for the task "*manufacture product*" (MP). Some additional work packets need to be performed prior of shipping the order to the customer; these are "*prepare gifts*" (PG) and "*prepare promotional info*" (PPI) to include into the order shipment. Also, somebody needs to take care and "*prepare invoice*" (PI). Once the order is ready someone has to "*decide on shipper*" (DOS), "*pack*" (P), and "*ship order*" (SO). Finally, it is required to "*archive order info*" (AOI).

It is clear that one might come up with several reasonable process models on the proposed set of activities. Following, we specify designed flexible activity execution constraints we assume for our scenario. A process instance can start with execution of either NPC or CS. Once both are accomplished, it is allowed to proceed in any order with execution of PRM, MPP, and DS activities. Once all the materials are available, production plan is ready, and workers are identified, it is possible to start with MP activity. At the same time somebody can take care of order supplements and perform PG, PPI, and PI activities. Once the ordered product is manufactured and all the supplements are prepared, activities concerned with order finalization can take place. It is required to accomplish AOI, DOS, and P activities. If order is packed and the delivery method is determined it is possible to proceed and do SO activity.

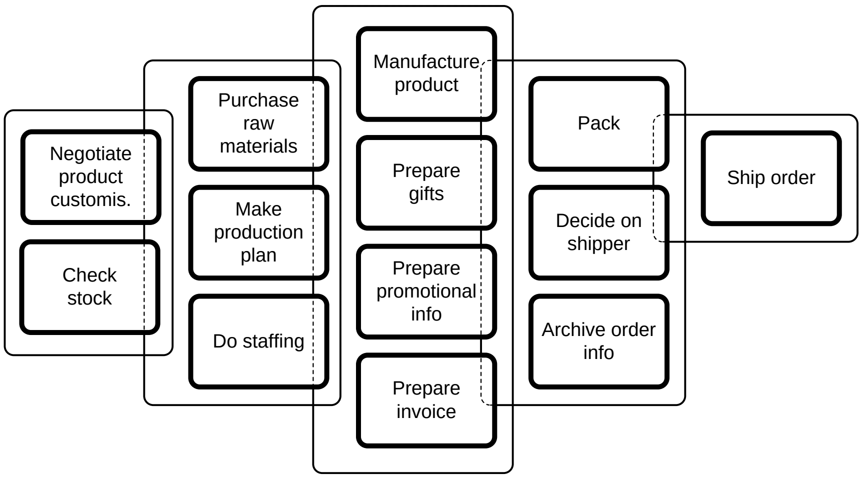


Figure 5: FPG model that captures flexible business process scenario

The Petri net model from Figure 4 captures the flexible process scenario. The model suffers from explosion of modeling constructs, in particular Petri net places, that attempt when combined to represent all possible states of the flexible process scenario.

Figure 5 shows a graphical representation (for details refer to [13]) of the FPG model (A, E, T) that also captures our flexible process scenario, $E = \{e_1, e_2, e_3, e_4, e_5\}$ such that: $e_1 = \langle \emptyset, \{NPC, CS\} \rangle$, $e_2 = \langle \{NPC, CS\}, \{PRM, \ \ MPP, DS\} \rangle$, $e_3 = \langle \{PRM, MPP, DS\}, \{MP, PG, PPI, PI\} \rangle$, $e_4 = \langle \{MP, \ \ PG, PPI, PI\}, \{P, DOS, AOI\} \rangle$, $e_5 = \langle \{P, DOS\}, \{SO\} \rangle$, and function T is such that $T(e_1) = T(e_2) = T(e_3) = T(e_4) = T(e_5) = and$.

After the process initialization phase, activities NPC and CS get enabled, $S(NPC) = S(CS) = (1, 0)$. Eventually both activities are accomplished in the ad-hoc manner and result in the FPG state S such that $S(NPC) = S(CS) = (0, 1)$. Once in such a state further activities enabling takes place and $S(PRM) = S(MPP) = S(DS) = (1, 0)$. The process continues by obeying the FPG execution semantics until $S = (0, 1)$ for all the process activities. Then, the termination condition holds and the process terminates.

The amount of FPG model elements from Figure 5, contrary to the case of the model from Figure 4, clearly exhibits a linear behavior in respect to the amount of modeled execution constraints. One might introduce concurrency into the FPG model by distributing activities among different roles, e.g., supplementary activities PG, PPI, and PI can be assigned to a different role as MP activity.

3 Conclusions

Business process model abstraction is a way to derive high level process models from the detailed ones. This report proposed a slider as the mean for controlling model abstraction level. We argued that the abstraction task can be decomposed into two independent subtasks: learning process model elements which are insignificant (abstraction *what*) and abstracting from those elements (abstraction *how*). The work reported primarily focuses on the former problem. Several abstraction scenarios were provided to motivate the task of business process model abstraction. These scenarios were further reused to extract abstraction criteria. We proposed to adopt a slider concept in order to manage abstraction criterion interval and specify desired abstraction level. The principles of abstraction slider were explained, as well as examples of its work were provided. As the direct continuation of this work we foresee development of transformation rules which can be used together with the abstraction slider concept.

Also, this report briefly presented the FPG formalism and contributed to the interpretation of FPG when modeled for concurrent execution of process activities by different process participants. This report together with [13] finishes overall introduction of a novel approach for representing ad-hoc process control flow. The future work will be concerned with validation of the approach. The applicability of FPG for formalization of Service Science [17] environment is currently under investigation.

Acknowledgments

The author acknowledge the support of the project partner AOK Brandenburg in Teltow, Germany, in particular Dr. Anke-Britt Möhr, Norbert Sandau, and Anja Niedersätz. The work in the domain of business process model abstraction is conducted jointly with Sergey Smirnov and Prof. Mathias Weske. The topic of Flexible Process Graph is developed together with Prof. Mathias Weske.

References

[1] W. Aalst, L. Aldred, M. Dumas, and A. Hofstede. Design and Implementation of the YAWL System, 2004.

[2] W. M. P. van der Aalst and A. H. M. ter Hofstede. YAWL: Yet Another Workflow Language (Revised version). Technical Report FIT-TR-2003-04, Queensland University of Technology, Brisbane, 2003.

[3] C. Berge. *Graphs and Hypergraphs*. Elsevier Science Ltd., 1985.

[4] C. Berge. *Hypergraphs: The Theory of Finite Sets*. Amsterdam, Netherlands: North-Holland, 1989.

[5] R. Bobrik, M. Reichert, and T. Bauer. Parameterizable Views for Process Visualization. Technical Report TR-CTIT-07-37, Enschede, April 2007.

[6] R. Bobrik, M. Reichert, and T. Bauer. View-Based Process Visualization. In *BPM 2007, volume 4714 of LNCS*, pages 88–95, Berlin, 2007. Springer Verlag.

[7] Ralph Bobrik, Thomas Bauer, and Manfred Reichert. Proviado—Personalized and Configurable Visualizations of Business Processes. In *EC-Web*, pages 61–71, 2006.

[8] BPMI.org. *Business Process Modeling Notation*, 1.0 edition, May 2004.

[9] B. Dongen, M. Jansen-Vullers, H. Verbeek, and W. Aalst. Verification of the SAP Reference Models Using EPC Reduction, State-space Analysis, and Invariants. *Comput. Ind.*, 58(6):578–601, 2007.

[10] G. Keller, M. Nüttgens, and A.W. Scheer. Semantische Prozessmodellierung auf der Grundlage "Ereignisgesteuerter Prozessketten (EPK)". Technical Report Heft 89 (in German), Veröffentlichungen des Instituts für Wirtschaftsinformatik University of Saarland, Saarbrücken, 1992.

[11] D. Liu and M. Shen. Workflow Modeling for Virtual Processes: an Order-preserving Process-view Approach. *Information Systems*, 28(6):505–532, 2003.

[12] C.A. Petri. *Kommunikation mit Automaten*. PhD thesis, University of Bonn, Bonn, Germany, 1962. (In German).

[13] A. Polyvyanyy and M. Weske. Hypergraph-based Modeling of Ad-Hoc Business Processes. In *Proceedings of the 1st International Workshop on Process Management for Highly Dynamic and Pervasive Scenarios*, Milan, Italy, 9 2008.

[14] Artem Polyvyanyy, Sergey Smirnov, and Mathias Weske. Reducing the Complexity of Large EPCs. Technical Report 22, Hasso Plattner Institute at University of Potsdam, 2008.

[15] W. Sadiq and M. E. Orlowska. Analyzing Process Models Using Graph Reduction Techniques. *Information Systems*, 25(2):117–134, 2000.

[16] M. Shen and D. Liu. Discovering Role-Relevant Process-Views for Recommending Workflow Information. In *DEXA*, pages 836–845, 2003.

[17] Jim Spohrer, Paul P. Maglio, John Bailey, and Daniel Gruhl. Steps Toward a Science of Service Systems. *Computer*, 40(1):71–77, 2007.

[18] M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer Verlag, 2007.

[19] M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer Verlag, 2007.

Publications

Table 1 presents the current list of published scientific papers, as well as submitted papers and ideas under investigation.

	Paper title	Paper type	Conference	Status
1	A Quantitative Evaluation of the Enhanced Topic-based Vector Space Model	Tech.rep.	HPI'08	published
2	An Ontology-based Service Discovery Approach for the Provisioning of Product-service Bundles	Conf. paper	ECIS'08	published
3	Hypergraph-based Modeling of Ad-Hoc Business Processes	W/s paper	PM4HDPS'08	published
4	Reducing Complexity of Large EPCs	W/s paper	EPK'08	accepted
5	Semantic Querying of Business Process Models	Conf. paper	EDOC'08	published
6	Process Model Abstraction—A Slider Approach	Conf. paper	EDOC'08	published
7	Flexible Process Graph—A Prologue	Conf. paper	COOPIS'08	published
8	Business Process Model Abstraction	Book chap.		submitted
9	On Formalizing Service Science Environment	W/s paper	ISSS'09	in preparation
10	Business Process Model Abstraction by SESE Decomposition (working title)	Conf. paper	to be decided	in preparation
11	Triconnected Business Process Model Abstraction (working title)	Conf. paper	to be decided	in preparation

Table 1: Publications overview

ContextJ
Context-oriented Programming for Java

Malte Appeltauer
Software Architecture Group
Hasso-Plattner-Institut
Universität Potsdam

malte.appeltauer@hpi.uni-potsdam.de

Technology for mobile devices is continuously evolving. This causes a demand for new applications that help mobile users to cope with challenges in their every-day life. Such applications often contain user-specific computations, which are based on dynamic context information. However, programming languages used for their implementation do not explicitly support context-dependent behavior. Instead, context-aware functionality must be modeled at application level, hindering separation of concerns and further software evolution. Context-oriented Programming (COP) is an approach to support modularization and context-dependent dynamic composition of cross-cutting concerns. There are several COP implementations for dynamic languages, however, no statically typed language, such as Java, in which many of those applications are implemented, has been extended yet.

In this paper, we present our two approaches to COP for Java: *ContextLogicAJ*, an aspect-based COP pre-compiler, and *ContextJ*, a full Java language extension. We give an overview of the implementation of our ContextJ compiler and introduce its language constructs. Furthermore, we exemplify software development with ContextJ on a service-oriented mobile application.

1 Introduction

Research and technology in the area of mobile and distributed environments have recently achieved significant improvements. Many software systems based on this infrastructure provide personalized, context-dependent functionality. For such systems, context reasoning and representation is an important technical issue. They are typically implemented in a service-oriented approach and composed by services, which provide context-dependent functionality. Due to the cross-cutting nature of context-dependent functionality, developers have to consider an additional complexity in the software model. To cope with the complex task of developing context-aware applications, we need appropriate means for the representation of context-dependent behavior.

COP [10] supports modularization and dynamic system composition. While various implementations of COP exist for dynamic languages [4, 10], an implementation of a complete COP language extension for a complex, statically typed language - such as

Java - is not described in literature. However, to benefit from COP in large software systems and to validate its modularization and composition features for complex applications, it is necessary to provide support for languages, such systems are typically written in. COP extensions to dynamic languages can be implemented via their meta-object protocols. Java does not provide meta-level access with equivalent expressibility as dynamic languages, thence language extensions need at least pre-compiler or compiler adaptations, if not virtual machine support. For experimental purposes, we implemented both a pre-compiler based on an aspect-oriented library and a compiler for Java-based COP.

In this paper, we report on our implementation of COP functionality as extension to Java. We describe, how COP features can be mapped to Java byte code using rewrite techniques provided by a compiler framework. Section 2 introduces to Context-oriented Programming. The features of ContextJ are described in Section 3. An overview of the ContextJ compiler implementation is given in Section 4. Section 5 presents an application implemented with ContextJ, while Section 6 discusses future work and summarizes the paper.

2 Context-oriented Programming for Java

2.1 Overview

The COP paradigm features a new approach of software modularization by supporting an explicit representation of context-dependent functionality. Such functionality can be dynamically activated for a certain control flow and trigger dynamic behavioral variations. Due to dynamic composition, software evolution becomes more flexible and accessible. We do not restrict the definition of context to a certain domain. Instead, context is everything that is computable, such as a variable's value, the control flow, or the system state. Context can also consist of more complex information, as, for example, personalization, security settings, location-awareness, and more. A broad introduction to COP is provided by [10]. COP language extensions have been implemented for some dynamic programming languages, such as Lisp, Squeak/Smalltalk, Python, Ruby, and Groovy. These language extensions implement at least the following core features of COP.

Modularization. Layers are an orthogonal modularization concept to classes, in which cross-cutting, context-specific functionality can be encapsulated. Layers can range over multiple modules and provide alternative or additional definition of their entities, e.g., classes, methods, or functions. Feature redefinitions are declared within layers. They represent behavior, which should be executed while the enclosing layer is active.

Dynamic composition. Behavioral variations can be executed before, after, or instead of the original functionality. During its execution, a behavioral variation can proceed to the next variation that is provided by another active layer. If no other active layer provides a variation, the original functionality is executed. Layers can be activated and deactivated at run-time.

```java
public class Person {
  private String name
  private String address;

  public Person(String name, String address) {
    this.name = name;
    this.address = address;
  }
  public String getAddress() {
    return this.address;
  }
  public String getName() {
    return this.name;
  }
  public String toString() {
    return this.getName();
  }
}
```

Figure 1: A simple Java class that models a person.

Dynamic scoping. To achieve fine-grained composition, layer activation is scoped for a certain block of execution statements. For all control flows within this block the behavioral variations of the current active layers will be executed instead of the default behavior.

2.2 ContextJ*

The former COP implementations take advantage of the language's meta-level protocol to enable dynamic composition. This method is not feasible for the extension of a statically typed language, such as Java, because of its restricted meta-level access. In [10], the authors present *ContextJ**, a plain Java 1.5 library-based implementation that features the core concepts of COP. This implementation demonstrates that COP can be simulated by means of the Java programming language without any extension to the syntax or semantics of the language.

Figure 1 contains a simple Java class to which we will refer in the following examples. The class `Person` provides the fields `name` and `address` and corresponding accessors. The `toString` method returns the person's name. The listing in Figure 2 shows layer declarations and activation using ContextJ*. The method `toString` (Lines 8-10) calls a `LayerDefinition` object `layers` to select the appropriate layered method at run-time. Within the class initializer (Lines 11-25), layered method variations of `toString` are assigned to `layer`. A layer activation of `Address` for the execution of `toString` is shown in Lines 26-34. For more details of the ContextJ* library, see [10].

As this simple example shows, the proper use of ContextJ* requires developers to follow several idioms. This code tangles the implementation of the core concerns and hinders seamless software development. In the following, we describe a pre-compiler that we developed based on an aspect-oriented language that overcomes some of

```java
import static be.ac.vub.prog.contextj.ContextJ.*;

public class Person implements IPerson {
  //... see Figure 1
  private LayerDefinitions<IEmployer> layers =
    new LayerDefinitions<IPerson>();

  public String toString() {
    return layers.select().toString();
  }
  { layers.define(RootLayer,
      new IPerson() {
        public String toString() {
          return name;
        }
      }
    );
    layers.define(Layers.Address,
      new IPerson() {
        public String toString() {
          return layers.next(this) + address;
        }
      }
    );
  }
  public void someMethod(){
    with(Layers.Address).eval(
      new Block() {
        public void eval() {
          print(toString());
        }
      }
    }
  }
}
```

Figure 2: Layer definition and composition using ContextJ* [10].

these drawbacks.

2.3 ContextLogicAJ

In collaboration with the University of Bonn we implemented a pre-compiler, *ContextLogicAJ* [2] that is based on a generic aspect library for the aspect-oriented language *LogicAJ* [13]. Layer activation management is encapsulated in advice declarations in a special COP aspect. Section 4.1 describes the dispatching technique that is implemented in ContextLogicAJ. To use the aspect as COP pre-compiler, the following idioms must be considered:

- Layers are represented by (empty) subclasses of `contextj.Layer`.

```
 1 import contextj.Layer;
 2 import contextj.ContextJ;
 3
 4 public class Person {
 5   //... see Figure 1
 6   public String toString(Address _layer) {
 7     return ContextJ.proceed() + getAddress();
 8   }
 9   public void someMethod() {
10     Layer.activateLayer(Address.class)
11     print(toString());
12     Layer.deactivateLayer(Address.class)
13   }
14 }
```

Figure 3: Layer declaration and composition using ContextLogicAJ.

- The layer, to which a certain method belongs to, is represented by the type of its first parameter.
- The scope of layer composition must be manually controlled by the methods `activateLayer` and `deactivateLayer`.
- The aspect library provides a static `proceed` method which forwards the execution to the next layer or to the original method.

Figure 3 shows the implementation of our running example using ContextLogicAJ. The first definition (Lines 6-8) in the figure represents a partial definition of `toString` for the layer `Address`. By convention, the type of the first parameter of a method denotes it's layer, thus, `toString` belongs to layer `Address`. The method uses the `proceed` function which calls the original method or the partial method definition of the next layer of the current composition. It is implemented as a static method with an empty body. The aspect uses this method call as join point to inject the behavior of the `proceed` function. The last method in the listing presents an activation of `Address` for the control flow of `print`.

Compared to the ContextJ* implementation, the code for layer definition is clearly reduced. However, developers still have to mind some coding conventions. With this two prototypes we gathered experiences that we use for the implementation of a first ContextJ compiler.

3 ContextJ

3.1 Language Features

One drawback of the preceding COP approaches to Java is that the application code is tangled with infrastructural code that is necessary for proper use of the COP library.

```
declarelayer Address;

public class Person {
  //... see Figure 1
  layer Address {
     public String toString() {
       return proceed() + ", " + getAddress();
     }
   }
  public void someMethod() {
    with(Address) {
      print(person.toString());
    }
  }
}
```

Figure 4: The ContextJ variant of `Person`. Layers encapsulate behavioral variations.

This is error-prone and does not help to gain better program modularity. It is more desirable that developers can focus on the implementation of layer declarations and layer compositions, supported by dedicated language constructs for this work. This section presents the ContextJ language, which extends the Java programming language with COP features. In the following, we refer to standard Java elements using '...' and present only the ContextJ constructs and their entry points in the Java syntax.

3.2 Modularization

In ContextJ, layers are provided as class members. The syntactic structure of the construct is shown below.

$$\begin{aligned}&\mathit{ClassMemberDeclaration} := \ldots \mid \mathit{MethodDeclaration} \mid \mathit{Layer}\\&\mathit{Layer} := \texttt{layer}\ \mathit{Identifier}\ \{\ \mathit{MethodDeclaration}^*\ \}\end{aligned}$$

A layer declaration begins with the keyword `layer`, followed by the layer's identifier and a list of method definitions, which either specify new methods or extend existing methods of the class. New methods are only visible while the layer is active. In Java, the visibility of methods can be declared *public*, *default*, *protected*, or *private*. Due to dynamic composition, ContextJ extends this feature with dynamically scoped method access. For different contexts, objects provide different interfaces. A layered method definition that has the same signature as another method of the class, is a *partial method definition*. It overrides the original method with its context specific behavior.

To trigger the execution of a behavioral variation provided by a following layer or the original method, the build-in expression `proceed` can be used.

$$\begin{aligned}&\mathit{Expression} := \ldots \mid \mathit{Proceed}\\&\mathit{Proceed} := \texttt{proceed(}\ \mathit{Argument}\ \texttt{)}\\&\mathit{Argument} := \mathit{Expression} \mid \mathit{Expression}\texttt{,}\ \ \mathit{Argument}\end{aligned}$$

```
declarelayer HTML;
//...
layer HTML {
  public String getName() {
    return "<b>" + proceed() + "</b>";
  }
  public String getAddress() {
    return "<i>" + proceed() + "</i>";
  }
  public String toString() {
    return "<html><body>" + proceed() + "</body></html>";
  }
}
```

Figure 5: Layer declaration with multiple methods in ContextJ.

In the example of Figure 4, the layer `Address` contains a partial method `toString` (Lines 6-8) that is called instead of the default definition while the layer is active. The partial method definition calls the original method (using `proceed`) and extends its result with layer specific behavior (adding the address). Similar to Java's `super` method it is possible to pass arguments to the next method using `proceed`.

In our experiments with the language constructs, we observed that typos in the identifier often occur in layer definitions, such as declaring 'adress' instead of 'Address'. Methods of the mistyped layer 'adress' are then not executed at activtion of 'Address'. To avoid this problem, we introduced a declaration statement for layers, whose syntax is shown below.

$$\begin{array}{l} \mathit{TopLevelDeclaration} := \ldots \mid \mathit{LayerDeclaraion} \\ \mathit{LayerDeclaration} := \texttt{declarelayer}\ \mathit{Identifier}\ \texttt{;} \end{array}$$

To define a layer in a class, it must be declared in its enclosing compilation unit. Typos within layer declarations can now be detected at compilation time. Figure 4 shows such a declaration for the layer `Address` (Line 1).

3.3 Dynamic Composition

To control layer activation and composition, ContextJ supports a special block construct that can be used in method bodies.

$$\begin{array}{l} \mathit{Statement} := \ldots \mid \mathit{LayerActivation} \\ \mathit{LayerActivation} := \texttt{with}\ \texttt{(}\ \mathit{Argument}\ \texttt{)}\ \{\ \mathit{Statement}^*\ \}\ \texttt{;} \end{array}$$

The keyword `with` is followed by a list of arguments denoting the layers to be activated. The actual block contains the statements that are executed while the layers are active. After the execution of the last statement all layers of this composition are deactivated, even if the block is left by an exception or `return` statement. More than one layer can provide a partial method definition at run-time, making the order of layer activation

```java
public void someMethod() {
  with(layersForCurrentMode()) {
    print(person.toString());
  }
}
public List<String> layersForCurrentMode(){
  ArrayList layers = new ArrayList();
  if(inAddressBookMode())
    layers.add("Address");
  if(inWebExportMode())
    layers.add("HTML");
  return layers;
}
```

Figure 6: Layer composition in ContextJ.

relevant for method execution. The list of active layers works according to the *last-in-first-out* principle: If more than one layer provides a redefinition of a certain method, the layer that was activated at last is the first layer to which the method call is delegated. If a behavioral variation contains `proceed`, but no following layer provides a corresponding behavioral variation, the original method definition is called. Figure 5 introduces a second layer, `HTML`, to our example. For web export, it renders the class properties with HTML tags. The layer contains a partial definition of `toString`. When both layers, `HTML` and `Address`, are composed together, their activation order controls, which partial definition of `toString` is called first.

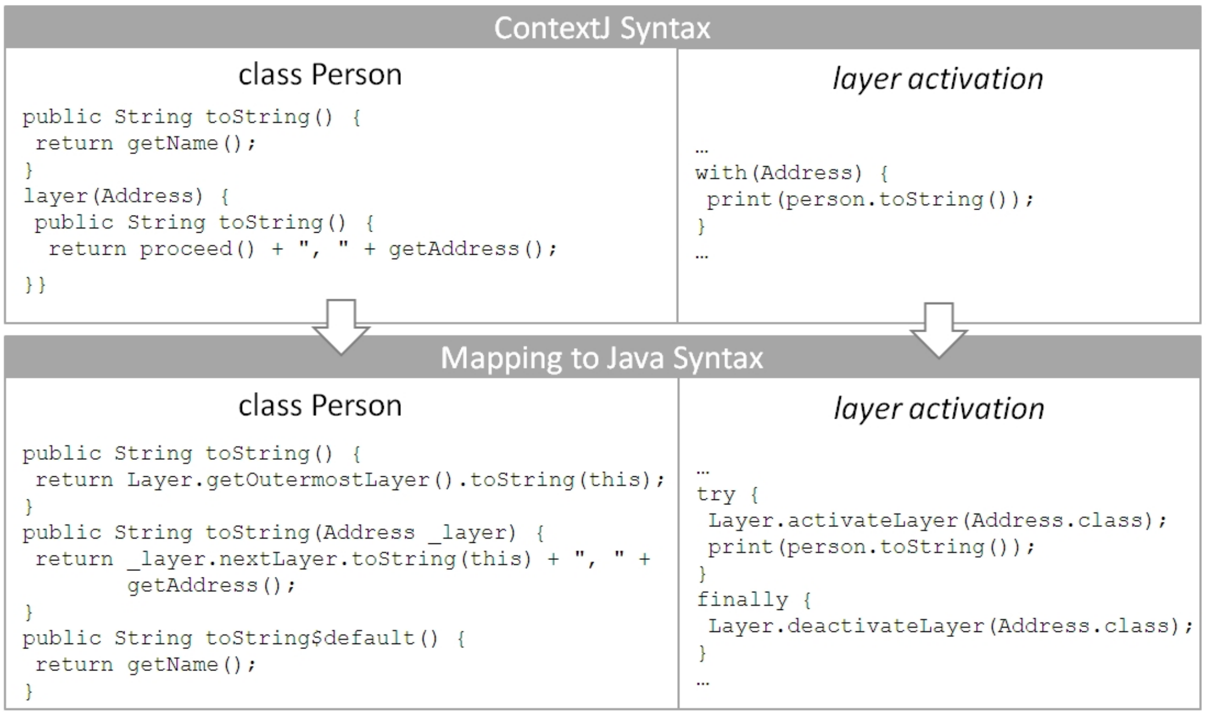
ContextJ supports *direct* and *indirect* layer activation. For direct activation, the layer identifiers are passed to the argument list. Figure 4 shows an explicit activation of the layer `Address` (Lines 11-13).

Often, the computation of the layers to be used is complex, or the layer list cannot be specified at compile-time. For more flexibility, we allow expressions of type `Iterable` as arguments. The elements of `Iterable` must be strings, otherwise an exception is thrown at run-time[1]. If the argument expression returns an empty list, the default behavior is executed. An example for a layer computation is given by Figure 6 the layer activation construct calls a method that computes the layer composition. Consider, `layersForCurrentMode` returns the list `["Address", "HTML"]`. Then, the call `toString` is passed to the layer `HTML` first[2]. This layered method creates some HTML wrapper and accesses the next partial definition via `proceed`. Layer `Address` invokes the original `toString` method and attaches the address to its return value. For all calls within this control flow, it is checked if an active layer provides a behavioral variation. Thus, calls within the control flow of `Address` are wrapped by partial definition of the `HTML` layer.

The ContextJ language allows layer declaration and composition without bothering developers with COP specific glue code. This advantage over the preceding solutions

[1]To be conform to the Java type system, we do not introduce a new primitive for layers. Instead, a list of strings can be used as argument for `with`.

[2]The list is accessed according to the last-in-first-out principle.



```
public String toString() {
 return getName();
}
layer(Address) {
 public String toString() {
  return proceed() + ", " + getAddress();
}}
```

```
…
with(Address) {
 print(person.toString());
}
…
```

```
public String toString() {
 return Layer.getOutermostLayer().toString(this);
}
public String toString(Address _layer) {
 return _layer.nextLayer.toString(this) + ", " +
        getAddress();
}
public String toString$default() {
 return getName();
}
```

```
…
try {
 Layer.activateLayer(Address.class);
 print(person.toString());
}
finally {
 Layer.deactivateLayer(Address.class);
}
…
```

Figure 7: Mapping of ContextJ syntax to Java.

ContextJ* and ContextLogicAJ lead to a seamless integration of our new language constructs into Java. In the following section, we describe the implementation of our ContextJ compiler.

4 The ContextJ Compiler

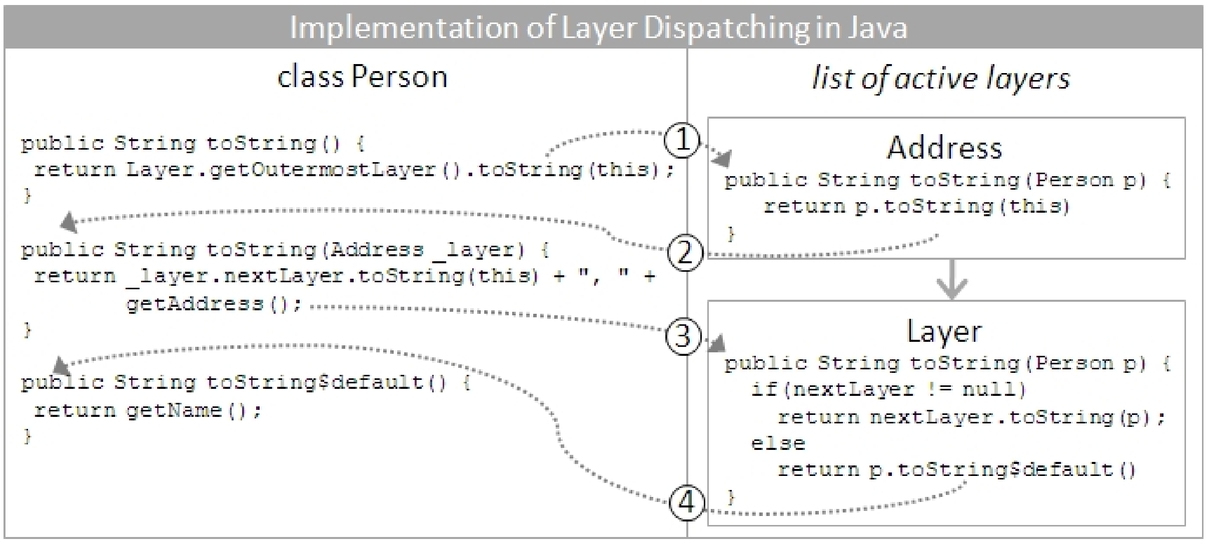
4.1 Layer-aware Method Invocation in Java

Since we want to use ContextJ with existing Java tools and environments, our compiler should be byte code compatible to Java. To generate plain Java byte code from ContextJ source code, we require a generic mapping from ContextJ to Java syntax. This mapping and its implementation in our compiler is described in Section 4.2. In the following, we present a pattern to execute COP semantics without any language extension in Java. This approach has been developed for the ContextLogicAJ precompiler. We adapt the pattern to transform ContextJ into Java elements within the re-write phase of the ContextJ compiler.

For layer-aware method call adaptation in Java, we need to follow some idioms in our class design. For a call to a method M and a list of active layers L^{active}:

1. Compute the next layer $L_i \in L^{active}$ that contains a partial method definition (M_{L_i}) for method M.

2. Execute M_{L_0}



```
public String toString() {
 return Layer.getOutermostLayer().toString(this);
}

public String toString(Address _layer) {
 return _layer.nextLayer.toString(this) + ", " +
        getAddress();
}

public String toString$default() {
 return getName();
}
```

```
public String toString(Person p) {
   return p.toString(this)
}
```

```
public String toString(Person p) {
  if(nextLayer != null)
    return nextLayer.toString(p);
  else
    return p.toString$default()
}
```

Figure 8: Implementation of method call dispatching with layers.

(a) If M_{L_i} contains a `proceed` command, lookup the next layer $L_x \in L^{active}, x > i$ that contains M_{L_x} and execute this method

(b) If no other layer is found, execute the original method definition

For a plain Java implementation of the above algorithm, we note some observations. The selection of the appropriate method definition depends on the dynamic structure of L^{active} that can be implemented as a list consisting of layer objects. The list members L_i decide, if they provide a definition for the currently called method M. Thus, layers provide a method corresponding to each of their partial method definition. Because partial definitions can access the state of M's class, L_i cannot host the code of M_{L_i}. Instead, M_{L_i} must be declared within M and L_i's method must call it. To distinguish multiple partial definitions of M, L_i passes itself as an argument to M_{L_i}. This behavior is well known as *double dispatch*, see, for instance, the *Visitor Pattern* [8]. Layer activation can be implemented straight forward: Two methods control the add and replace functionality of L^{active}.

Figure 7 presents a mapping of a layer to Java constructs using double dispatch. Figure 8 shows a control flow of a method call with active layer. A partial method definition M_{L_i} is a method with the same signature as it's base version M, except that it's first parameter denotes the type of the enclosing layer L_i. Layers are classes that inherit from `contextj.Layer`. L^{active} is represented by a thread-local list of layer objects. For each method M_{L_i}, a layer L_i contains an *accept* method, which calls M_{L_i} and passes its own instance as first parameter. If L_i does not contain M_{L_i}, the next layer in the list is called. Therefore, the superclass `contextj.Layer` contains a default implementation for the *accept* method that calls the next layer in the list.

4.2 Compiler Implementation

For the implementation of our compiler we use the *JastAdd* [9] compiler framework. Typically, compiler extensions require adaptations on several parts of the compiler, such as the lexer/scanner, parser, abstract syntax tree (AST), and semantic analysis. JastAdd is a modular compiler framework that uses aspect-oriented techniques to encapsulate these specifications into dedicated modules. In a build process the single specifications are woven into an executable compiler.

For byte code compatibility we extend the *JastAdd Java Extensible Compiler* [7]. In the following, we describe the implementation of our compiler.

Lexical Tokens

ContextJ extends the set of Java keywords with `layer`, `proceed`, `declarelayer`, and `with`. The extension of the scanner is shown in Figure 9. For lexical analysis JastAdd uses *JFlex* [12], a scanner generator for Java. Each keyword specification provides a corresponding terminal symbol that can be used in the parser. The keyword specification is woven into the scanner at build-time.

```
<YYINITIAL> {
  "with"            { return sym(Terminals.LAYER_ACTIVATION); }
  "layer"           { return sym(Terminals.LAYER); }
  "proceed"         { return sym(Terminals.PROCEED); }
  "declarelayer"    { return sym(Terminals.DECLARE_LAYER); }
 }
```

Figure 9: Specification of ContextJ terminal symbols in Beaver syntax.

Abstract Syntax

JastAdd provides an object-oriented abstract grammar from which the Java AST representation is generated. The abstract grammar does not contain any behavior specification; this is done by separate attribute and equation specifications. For a modularized specification, inter-type declarations are used to extend existing trees. For ContextJ, the Java AST is extended with four node types. `Layer` is defined as subtype of a class member declaration. As child nodes it contains a `TypeAccess` node and a list of method declarations. The statement `LayerActivation` contains an expression, denoting the layers to be activated, and a block of statements, which scopes the layer activation. The delegation function `proceed` contains a list of arguments. Finally, the node `LayerDecl` represents the import declaration for layers. The AST specification is shown in Figure 10.

```
Layer:MemberDecl ::= TypeAccess:Access MethodDecl*;
LayerActivation:Stmt ::= Arg:Expr Block ;
Proceed:Access ::= Arg:Expr*;
LayerDecl:ImportDecl ::= <ID:String>;
```

Figure 10: Declaration of the AST nodes for ContextJ

Parsing

By default, JastAdd uses the Java-based parser generator *Beaver* [5], a LALR(1) parser generator. The system is able to consume the tokens that are generated by JFlex. Beaver accepts a context free grammar, expressed in the Extended Backus-Naur Form (EBNF), and converts it into a Java class that implements a parser for the language described by the grammar. The parser extension shown in Figure 11 specifies the syntactic structure of the `with` construct. The declarations of the other ContextJ constructs are implemented in the same fashion. The symbol `LAYER_ACTIVATION` is followed by an expression which denotes the layer to be activated and which is embraced by parenthesis and a block. This declaration is assigned to the variable `block_stmt`, which is initialized within the parser specification of Java and represents all variations of block statements for this language. This clause will be added as a new block statement. The variables `expression` and `block` used within the declaration refer to the declaration of expressions and block statements. The specification also declares, which code should be executed for the initialization of the AST object.

```
Stmt block_statement =
  LAYER_ACTIVATION LPAREN expression.p RPAREN block.b
    {: return new LayerActivation(p,b); :};
```

Figure 11: Parser extension for the `with` statement.

AST Transformations

For the implementation of the behavior shown in Section 4.1, we make use of JastAdd's re-writing facilities. Typically, re-write rules change or replace a certain AST node or subtree with an other. For instance, the JastAdd Java Compiler uses re-write rules to transform Java 1.5 constructs into 1.4 syntax. We use this technique to replace `LayerActivation` nodes with Java methods that initialize the list of active layers, and to transform `Proceed` nodes into delegations to the next active layer, see Figure 7. The re-write rule of `LayerDecl` contains a side-effect. The node is re-written into a Java import declaration and creates a corresponding class declaration node for the specified layer, if the class has not already been generated by an other layer import declaration. The class is a subtype of `contextj.Layer`, which is provided by the compiler. This super class contains methods to manage the thread-local list of active layers, handle layer activation, and traverse the layer list.

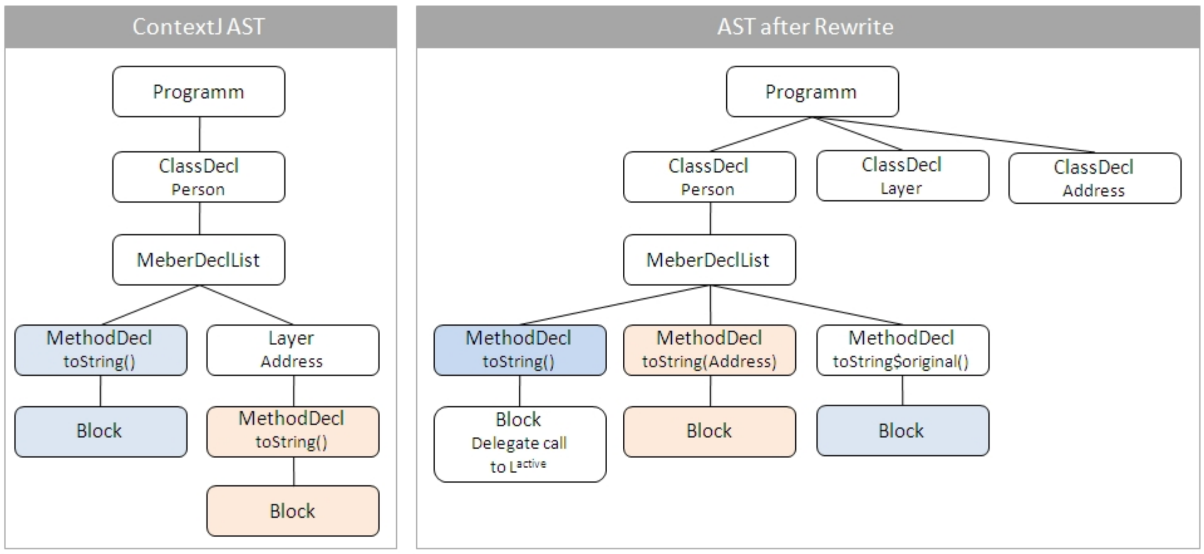


Figure 12: AST transformation of layer nodes to Java elements.

While these rules are implemented straight forward, it takes more effort to specify transformations for the declaration of a layer L. To use the double dispatch mechanism described in 4.1, we need *accept* and *visit* methods for each layer:

- A new parameter of type L is inserted into the parameter list of each partial method definition M_{L_i}. With this parameter, M_{L_i} can be used as an *accept* method. When all partial methods have been transformed, the encapsulating layer L is removed from the class member list.

- For each M_{L_i} a forwarding method $M^{forward}$ is created in L's class. It acts as *visit* method and calls M_{L_i} with its own instance as first parameter.

- A default version of $M^{forward}$, $M^{defaultforward}$, is created in the super class `contextj.Layer`. This method traverses L^{active} for the first layer that provides an implementation of $M^{forward}$.

- The body of M will be replaced with a call to $M^{defaultforward}$.

- A new method wrapper for the original behavior of M is added to M's enclosing class. This method is called by $M^{defaultforward}$ if no layer in L^{active} provides a $M^{forward}$.

Finally, the compiler generates byte code for the transformed layers. The application can then be executed as plain Java program. Figure 12 describes the transformation step at AST level. After re-write, the layer `Address` is replaced with a class. The class `Person` is extended with additional methods. The method body of `toString` has been moved to a new wrapper method.

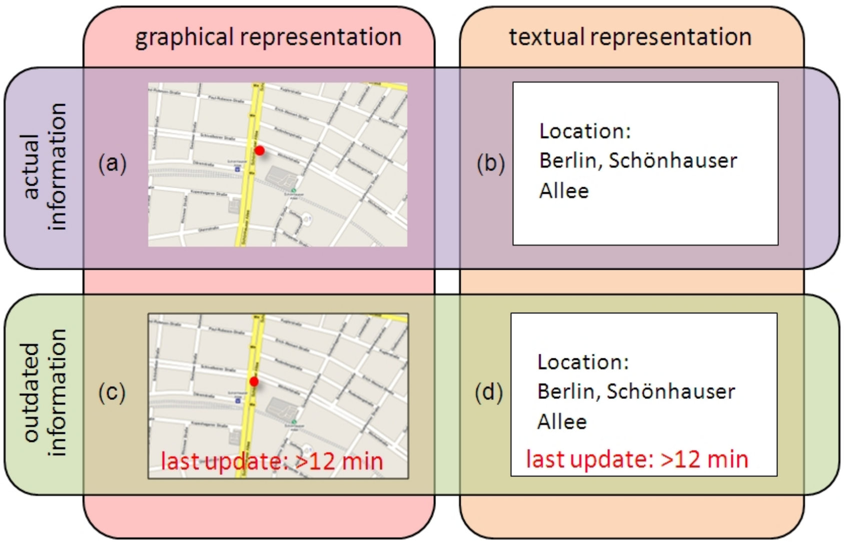


Figure 13: Behavioral variations based on layer compositions [2].

5 An Application of ContextJ

Most context-dependent systems are developed based on context-management frameworks, such as ContextWatcher [3, 14]. Such an infrastructure supports context reasoning, for instance, based on ontologies, and passes context information and changes to applications. We do not focus on context management frameworks but on enhancing the support of context-aware system adaptation at programming language level [1]. We believe, that COP provides good abstractions to specify the behavior or behavior adaptations which have to be executed dependent on the current context.

We will give an example for a service-based mobile application in the following subsection. Since we want to point out the interaction of services and context information, we chose a domain with frequent context changes.

5.1 Sharing Context Data

Mobile community applications (e.g., [11, 14]) allow users to share information about their mood, activities, location, and more. In this scenario, we focus on the location representation of buddies on mobile devices. Consider, Lucy and Tim are using our community software on their mobile clients. When Lucy checks Tim's current location with her cell phone, the graphical representation of this information depends on Lucy's context: If Lucy's device is currently connected to the Internet at high-bandwidth, a web map service is requested to render Tim's location on a map. Figure 13a gives an example for this map representation. Contrary, if the bandwidth is low, Tim's representation depends on Lucy's need for active information. This information is stored in Lucy's user profile and is accessible for applications. In the case that Lucy prefers a nice graphical representation over the refresh period, a map with Tim's outdated location is shown (Figure 13c). The map is labeled with the time stamp of the last update. Alternatively,

```
layer ActualInformation {
  public UIComponent renderUser() {
   String buddyLocation = requestService("getBuddyLocation", "Tim");
   return renderMap(requestService("getMap", buddyLocation));
  }
}
layer OutdatedInformation {
  public UIComponent renderUser() {
    UIComponent component = proceed();
    component.add(renderTimeStampOfLastUpdate());
    return component;
  }
}
layer TextualRepresentation {
  public UIComponent renderUser() {
    String buddyLocation = requestService("getBuddyLocation", "Tim");
    return renderText(buddyLocation);
  }
}
```

Figure 14: Alternative layer definitions for a method.

if Lucy insists on up-to-date information, Tim's position data is simply shown as a text (Figure 13b) as long as the bandwidth is too low for updating the map image.

When Tim arrives at the underground station, his GPS device is unable to receive data any more. Hence, Lucy receives the information that Tims location information is not actual (Figure 13d).

5.2 Implementation using ContextJ

In the following, we present a COP-based implementation of parts of our scenario. We focus on the *modularization* of context-dependent behavioral variation with layers and the *dynamic composition* of layers.

Modularization. Consider, the functionalities for the different location representations shown in Figure 13 are modularized into different layers.

Dynamic Composition. Each combination of these layers yield to a different composition. Dynamic compositions are controlled by dynamic layer activations. The next listing shows such an activation. The service call `getBuddyActivity` is invoked within the *low bandwidth* context, which is inferred by the method call `Bandwidth.isLow`. Thus, Lucy receives the compressed list of activities computed by the layered method definition of the previous listing. Note that layer activation is thread-local; a parallel service request of Tim, for instance, would not be dispatched to the layered method.

Different Combinations of layer activations lead to different compositions. The listing shown in Figure 15 contains an activation of multiple layers to compose the behavior shown in Figure 13(d). The invocation of `renderUser` is passed to the `OutdatedInformation` layer. First, the `proceed` function delegates the call to the next layer. The `TextualRepresentation` layer renders the location simply as a text

```
Activity a;
if(getBandwidth().isLow()) {
  with(OutdatedInformation,TextualRepresentation) {
    a = client.requestService("getBuddyActivity", "Tim");
  }
}
```

Figure 15: Conditional layer activation.

and returns the component back to `OutdatedInformation` that adds a time stamp to it.

6 Summary and Future Work

In this paper, we presented two systems that provide COP for Java. We introduced *ContextLogicAJ*, an aspect library that can act as a pre-compiler to inject COP specific behavior into Java applications. COP functionality is implemented by advice declarations, thus it is easy to change layer-specific behavior at source code level by changing its advice code. However, application developers have to follow some coding conventions for proper use of the COP functionality of the aspect. To overcome this drawback, we developed *ContextJ*, a language extension to Java. We constructed a compiler for ContextJ, whose design and implementation is described in this paper. We presented an application written with ContextJ to show the benefits of COP for service-oriented mobile applications.

With the first version of the ContextJ compiler we can support a broad range of context-dependent Java-based applications. In future, we plan to expand the domain of COP to distributed systems. For real applications based on distributed or service-oriented infrastructures, layer composition and activation must be propagated in the whole system. Therefore, we will implement a service-oriented middleware that allows exchanging layer information between distributed objects. For an assessment of the support of modularity, development and evolution of software, we plan to apply this middleware to a mobile client of the *IYOUIT* [6,14] platform that we currently implement for *Android* [15]. With this system we can support the development and evolution of context-aware programming in distributed applications.

References

[1] Malte Appeltauer and Robert Hirschfeld. Explicit Language and Infrastructure Support for Context-aware Services. In Heinz-Gerd Hegering, Axel Lehmann, Hans Jürgen Ohlbach, and Christian Scheideler, editors, *Beiträge der 38. Jahrestagung der Gesellschaft für Informatik*, volume INFORMATIK 2008 - Beherrschbare Systeme dank Informatik of *Lecture Notes in Informatics*, pages 164–170, München, Germany, September 2008. Gesellschaft für Informatik.

[2] Malte Appeltauer, Robert Hirschfeld, and Tobias Rho. Dedicated Programming Support for Context-aware Ubiquitous Application. In *UBICOMM 2008: Proceedings of the 2nd International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, Valencia, Spain, September 29 - October 4 2008. IEEE Computer Society Press.

[3] Jakob Bardram. The Java Context Awareness Framework (JCAF) – A Service Infrastructure and Programming Framework for Context-Aware Applications. In *Proceedings of the 3rd International Conference on Pervasive Computing*, Albrecht Schmidt, University of Munich, May 2005.

[4] Pascal Costanza and Robert Hirschfeld. Language Constructs for Context-oriented Programming: An Overview of ContextL. In *DLS '05: Proceedings of the 2005 symposium on Dynamic languages*, pages 1–10, New York, NY, USA, 2005. ACM.

[5] Alexander Demenchuk. Beaver - a LALR Parser Generator, October 2008. http://beaver.sourceforge.net.

[6] DoCoMo Communications Laboratories Europe GmbH and Telematica Instituut. IYOUIT. http://www.iyouit.eu/portal/.

[7] Torbjörn Ekman and Görel Hedin. The JastAdd Extensible Java Compiler. In *OOPSLA '07: Proceedings of the 22nd annual ACM SIGPLAN conference on Object oriented programming systems and applications*, pages 1–18, New York, NY, USA, 2007. ACM.

[8] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley, Boston, MA, USA, 1995.

[9] Görel Hedin and Eva Magnusson. JastAdd: an aspect-oriented compiler construction system. *Sci. Comput. Program.*, 47(1):37–58, 2003.

[10] Robert Hirschfeld, Pascal Costanza, and Oscar Nierstrasz. Context-oriented Programming. *Journal of Object Technology*, 7(3):125–151, March-April 2008.

[11] Ralf Kernchen, David Bonnefoy, Agathe Battestini, Bernd Mrohs, Matthias Wagner, and Mika Klemettinen. Context-awareness in MobiLife. In *15th IST Mobile & Wireless Communication Summit*, Mykonos, Greece, 2006.

[12] Gerwin Klein, Steve Rowe, and Régis Décamps. JFlex - The Fast Scanner Generator for Java, October 2008. http://www.jflex.de.

[13] Günter Kniesel and Tobias Rho. Generic Aspect Languages - Needs, Options and Challenges. JFDLPA, Sep 2005.

[14] Johan Koolwaaij, Anthony Tarlano, Marko Luther, Petteri Nurmi, Bernd Mrohs, Agathe Battestini, and Raju Vaidya. Context Watcher – Sharing context information in everyday life. In J.T. Yao, editor, *Web Technologies, Applications, and Services*, Calgary, Canada, July 17-18 2006. IASTED, ACTA Press.

[15] Open Handset Alliance. Android. http://www.code. google.com/android.

Modeling and Verification of Self-adaptive Service-oriented Systems

Basil Becker
basil.becker@hpi.uni-potsdam.de

1 Introduction

It is a fact of life that software is always embedded in a social and technical context which changes over time. Therefore to preserve the value of the software changes are unavoidable which adjust the software to its current context [29]. The rate at which such adjustments are required can vary from the short term reaction to frequent context changes (e.g., mobility) or very slow and only gradual changes (e.g., the user behavior, new requirements). While the former need rapid system adaptation using pre-defined heuristics or adaptation rules, effective support to change, extend, or migrate complex software systems is today responsible for a large fraction of the maintenance efforts to operate a complex software.

Adaptive and self-managed systems [27] which realizes properties such as self-healing, self-configuring or self-optimization employing self-adaptation [32,34,42] have been proposed as a means to address these different challenges.

Advanced software-intensive systems in the future are expected to consist of autonomous agents which coordinate with each other and exploit their context knowledge to enhance their behavior and exhibit self-adaptive behavior ([35, 40]) exploiting the flexible nature of software. Such self-adaptive behavior can be organized either in a top-down manner when considering an individual system, or bottom-up when considering cooperative systems. Top-down self-adaptive systems usually assess their own behavior and change it when the assessment indicates a need to adapt using usually an explicit internal representations and goals. Bottom-up self-adaptive systems in contrast are usually composed of a number of elements which interaction is governed by often rather simple rules.[1]

Traditional modeling approaches such as component-based modeling do not cover the dynamic character of the envisioned self-adaptive systems well. Therefore we suggest using instead a service-oriented approach, which employs collaborations of multiple roles in form of service contracts (cf. [6, 11]) to address systems with bottom-up self-adaptive behavior in form of dynamical structural adaptation as it occurs for example when autonomous vehicles build convoys. We present how such service-oriented view can be modeled by a well-defined subset of UML class diagrams with UML collaborations as well as well-defined behavioral rules for the structural changes and service

[1]When the overall behavior *emerges* from the strict local interaction of the elements, this class of system is also often named self-organizing systems [40].

contract instantiation or termination (cf. [1, 4]). The details of the role and link behavior of the collaborations not considered in this paper are covered by state machines extended with real-time constraints [17, 19].

As outline in [46] it is necessary to ensure that the complex self-adaptive behavior does not result in any harm. In [1], we therefore addressed the formal verification inventing a technique to automatically check inductive invariants for potentially infinite state models for such systems without time. In [4], we extended this approach to also cover timed behavior. In both cases, the rules result in an underspecification for the involved roles which allow them to exploit the resulting degrees of freedom within the rule-based coordination to incorporate also local optimization and planning behavior. Due to the often rather limited mental and sensorial capabilities of the involved agents and their distribution, we made however no assumptions that the self-adaptation of a system will fix any problems (cf. [21]).

In [19], we presented our approach which allows checking the local timed coordination for collaborations with a finite number of roles. We employ model checking to verify the real-time behavior of the interaction of collaboration and its roles separately. We then combine this in a compositional manner with the verification of the component synchronization by ensuring that the components refine the collaboration roles. The proper combination of the former approach to verify the real-time coordination for one collaboration and the pure structural rules has been presented in [17].

However, besides the emergent effects which result in bottom-up self-adaptive behavior and the incorporated local optimization and planning as addressed by these techniques, also the rules themselves might be subject to evolution when also compositional adaptation [30] is considered. To approach this problem we present in this paper a first idea how to provide suitable run-time checks for changes of these rule sets such that the required safety properties which have to be guaranteed by the distributed rule-based self-adaptive behavior can still be guaranteed. We therefore demonstrate how our existing verification technique for invariant checking for complex rules [1,4] can be adjusted such that it becomes an incremental technique which is more suitable for online checks.

The rest of this paper is two fold. The first part (Sections 2 - 6) covers the incremental verification of inductive invariants and has been published in [2]. The second part (Sections 7-10) then focuses on the modelling aspects of service-oriented self-adaptive systems and is based on work, presented in [3]. The report closes with a conclusion and an outlook to future work.

2 State of the Art

In contrast to other approaches which address the verification of self-adaptive systems [21, 46], we cover timed infinite state systems with structural adaptation and also evolution in form of changing sets of rules.

A number of related approaches [8, 16, 37, 44] for the verification of systems with structural changes exist. However, they do not support time dependent behavior, require an initial configuration, only support finite state systems (or systems for which an

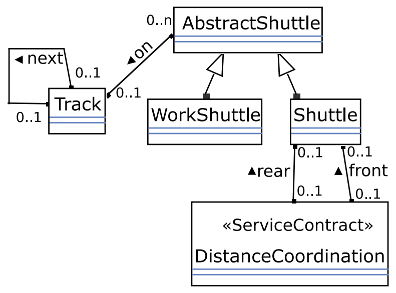


Figure 1: UML class diagram describing the system's elements and their relation to each other

abstraction to a finite state model of moderate size exist), and do not support evolution.

For the verification of infinite state systems with changing structure only a few attempts exist which however do not support time or evolution as approached in this paper: Graph transformation systems are transformed in [7] into a finite structure, called Petri graph which consists of a graph and a Petri net. Both can be analyzed with existing tools and for infinite systems the authors present an approximation. The approach is, however, not appropriate for our needs as it requires an initial configuration and the formalism is rather restricted, e.g., rules must not delete anything. In [9], partner graph grammars are employed to check topological properties of the platoon building. The partner abstraction and abstract interpretation are used to compute over approximations of the set of reachable configurations. However, the supported partner graph grammars restrict not only the model but also the properties which can be addressed a priori.

Real-Time Maude [33] is the only approach we are aware of that addresses structural changes as well as time. It is based on rewriting logics and provides tool support for simulation and bounded model checking. However, it like other approaches requires an initial configuration and is limited to finite state models.

3 Modeling

Throughout the paper we will use a common application example to explain and visualize the theoretical findings. The application example is based on a research project named Railcap[2], which aims at developing a new intelligent transportation system. The Railcap system is build around small, autonomous shuttles. This enables providing a public transportation system, which will serve the customers' individual needs. A single shuttle has an enormous energy consumption. Hence, techniques for achieving an even energy balance are required. Railcap proposes the building of convoys to achieve this goal. The convoys are only efficient w.r.t. the energy balance if the shuttles are in close proximity and therefore arises a need for the real-time coordination among them.

The set of possible states of our system is modeled through UML class diagrams. In Figure 10 the class diagram we use as ontology is shown. Within our system we

[2]http://www.railcab.de

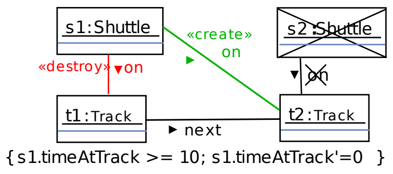


Figure 2: moveSimple rule Story Pattern

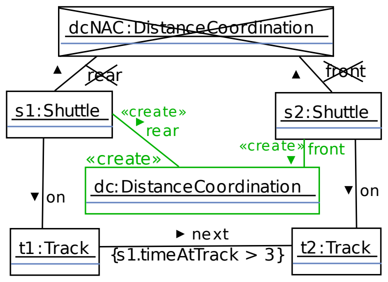


Figure 3: createDC rule Story Pattern

distinguish between Shuttles and WorkShuttles, whose difference is described through the fact that Shuttles can instantiate the DistanceCoordination service contract and WorkShuttles can't. Both types can be located at Tracks and each Track can have a successor that can be reached via the next association. Shuttles are said to be in a convoy if they have the DistanceCoordination service contract instantiated. WorkShuttles are used to maintain the Tracks and therefore could not build any convoys.

For modeling behavior we facilitate Story Pattern. In Figure 2 such a Story Pattern is shown. Story Pattern are a special extension of UML object diagrams [25] that are able to describe changes to the object structure in a compact fashion. The meaning of the mentioned Story Pattern can be translated as follows: The Shuttle s1 only advances from Track t1 to Track t2 if there is no other Shuttle located at Track t2. In addition to the structural requirements, this Story Pattern has a guard ($s1.timeAtTrack \geq 10$) and an update statement ($s1.timeAtTrack' = 0$) attached to it.[3] Detailed semantics follow in Section 4.1.

Due to space limitation we cannot show all rules in this paper. Beside the rule shown in Figure 2 our system further consists of rules that allow movement with an instantiated DistanceCoordination service contract and a move rule dedicated to WorkShuttles. Creation and deletion of the DistanceCoordination service contract is established by two rules, which are depicted in Figures 3 and 4.

[3]The employed rules can only express local atomic changes (effects). We assume that self-adaptive systems typically are distributed systems and following investigating local atomic changes is not a restriction but a necessity.

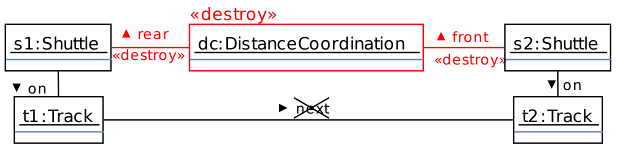


Figure 4: deleteDC rule Story Pattern

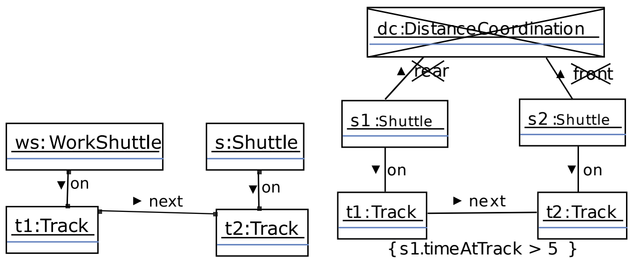


Figure 5: Forbidden Story Pattern

To guarantee the safety of the autonomous vehicles, we have to exclude states which represent an accident or a hazard. Those states are specified through instance situations, i.e. Story Pattern without creation or deletion elements.

For our application example those states have to be excluded where two Shuttles are in close proximity without having a DistanceCoordination service contract instantiated (cf. right graph in Figure 5) or a WorkShuttle is close to any other vehicle. The left graph in Figure 5 depicts this for a WorkShuttle being close to a Shuttle.

4 Invariant Checking

In this Section we roughly describe how invariant checking for timed and untimed graph transformation systems works. More detailed description of this verification technique could be found in [1] and [4]. We will first describe the employed formal model, review the basic idea of our approach [1] for untimed models and then finally describe the extension to also cover time dependent behavior [4].

4.1 Formal Model

For a given set $\mathcal{A}$ of attributes a *attributed graph* is a pair (G, α) consisting of a graph $G = (N, E)$ with node set N and a partial function $\alpha : N \times \mathcal{A} \to \mathbb{R}$ providing evaluations for the attributes defined for each node. A system state, given as an object diagram, can be encoded as an attributed graph by modeling objects as nodes with attributes and links as edges.

A condition ϕ over node attributes can then be evaluated for any evaluation function α. To describe more complex properties we use *graph pattern* P which consist of two sets of positive and negative nodes, and two sets of positive and negative edges. An *attributed graph pattern* is a pair (P, ϕ_P) consisting of a graph pattern P and a condition ϕ_P over the attributes of the positive nodes. An attributed graph pattern matches an attributed graph (G, α) if the positive elements of P matches G, the negative ones cannot be matched, and the condition ϕ_p is also true for α when the nodes are mapped according to the match of P and G.

A graph transformation rule $(L, R)_r$ consists of two graph patterns, a left hand side L (LHS) and a right hand side R (RHS). L consists of those elements of the Story Pattern that are not annotated with ≪create≫, including negative elements, whereas R

consists of all elements not annotated with ≪destroy≫. The elements annotated with ≪create≫ will be created by the rule, while those annotated with ≪destroy≫ will be deleted. Elements without annotations are preserved by the application. We write $G \rightarrow_r G'$ if rule r can be applied to graph G and the application results in graph G'. We write $G \rightarrow^* G'$ if G is transformed into G' by a (possibly empty) sequence of rule applications. For the Story Pattern moveSimple in Figure 2 we thus have that L contains the positive nodes for s1, t1, and t2 while s2 is the only contained negative node.

We have attributed graph rules $((L, \phi), R, \mu)_r$ with ϕ a condition over L and μ an evaluation for all positive nodes new in R and an arbitrary subset of the nodes which are in R and L where (L, ϕ) instead of L must be matched and the resulting attribute evaluation is determined by the update μ. We write $(G, \alpha) \rightarrow_r (G', \alpha')$ if rule r can be applied to graph (G, α) and the application results in graph (G', α'). We write $(G, \alpha) \rightarrow^* (G', \alpha')$ if (G, α) is transformed into (G', α') by a (possibly empty) sequence of rule applications. In the moveSimple rule of Figure 2 the condition ϕ equals the guard $s1.timeAtTrack > 10$ whereas μ equals the update statement $s1.timeAtTrack' = 0$.

A *graph transformation system* (GTS) $S = (\mathcal{R}, p)$ consists of a set of graph transformation rules $\mathcal{R}$ (defined by a set of Story Patterns), defining all possible transformations in the transformation system, and a priority function $p : \mathcal{R} \rightarrow \mathbb{N}$, which assigns a priority to each rule (the higher the number assigned to a rule the higher the rule's precedence). An additional set of initial graphs may describe the initial states of the system.

To also cover timed behavior we decompose the set of attributes into a set $\mathcal{C} \subseteq \mathcal{A}$ of clocks and a set of normal attributes $\mathcal{A} \setminus \mathcal{C}$. Given an evaluation function α, we refer to that evaluation function where the values of all clock variables is incremented by x as $\alpha \oplus x$ (resp. $\alpha \ominus x$ for decrement).

A *timed graph transformation system* (TGTS) $S = (\mathcal{R}, \mathcal{R}_u, p)$ is then defined for $(\mathcal{R}, p)$ a valid attributed graph transformation system and $\mathcal{R}_u \subseteq \mathcal{R}$ the subset of all urgent rules.

In the case of TGTS, we have additional *time steps* where the clock values increase over time. For $\delta \geq 0$ we have $(G, \alpha) \rightarrow_\delta (G', \alpha \oplus \delta)$ if for all $x \leq \delta$ holds $(G, \alpha \oplus x)$ does not match any urgent rule $r' \in \mathcal{R}_u$.

4.2 Untimed Case

In our approach a set of forbidden graph patterns $\mathcal{F} = \{F_1, \ldots, F_n\}$ are employed to represent possible safety-violations (hazards, accidents) of our system. The related property $\Phi_\mathcal{F}$, denoted by $G \models \Phi_\mathcal{F}$, holds iff G matches none of the graph patterns in $\mathcal{F}$. We call G a *witness* for the property $\neg\Phi_\mathcal{F}$ if G in contrast matches a forbidden graph pattern $F \in \mathcal{F}$.

The property $\Phi_\mathcal{F}$ is an *operational invariant* of the GTS S iff for a given initial graph G^0 for all reachable G $(G \in \{G' \mid G \rightarrow^* G'\}$ holds $G \models \Phi_\mathcal{F}$ (cf. [15]). However, due to the Turing-completeness of graph transformation systems with types checking them is restricted to finite models and thus does not fit to the considered class of problems. We therefore instead tackle the problem whether the property $\Phi_\mathcal{F}$ is an *inductive invariant*. This is the case if for all graphs G and for all rules $r \in \mathcal{R}$ holds that $G \models \Phi_{\mathcal{F}} \ \wedge \ G \rightarrow_r G'$ implies $G' \models \Phi_{\mathcal{F}}$. If we have an inductive invariant and the initial graph G^0 fulfills the property, then $\Phi_{\mathcal{F}}$ is also an *operational invariant* as inductive invariants are stronger than their operational counterparts.

The definition of an *inductive invariant* can be reformulated as follows to have a falsifiable form: a property $\Phi_{\mathcal{F}}$ is an inductive invariant of a GTS $S = (\mathcal{R}, p)$ if and only if there exists no pair (G, r) of a graph G and a rule $r \in \mathcal{R}$ such that $G \models \Phi_{\mathcal{F}}$, $G \rightarrow_r G'$ and $G' \not\models \Phi_{\mathcal{F}}$. Such a pair (G, r) which witnesses the violation of property $\Phi_{\mathcal{F}}$ by rule r is then a *counterexample* for the initial hypothesis.

As explained in detail in [1], we can exploit the fact that the application of a rule can only have a local effect to verify whether a counterexample exists. A counterexample (G, r) can only exist when the local modification of G by rule r is necessarily responsible for transforming the correct graph G into a graph that violates the property. In addition, to have a possible counterexample we require that the rule r is not preempted by a rule r' with higher priority.

As we can represent the infinite many possible counterexamples by an only finite set of representative patterns $\Theta(R_l, F_i)$ of graph patterns P' that are combinations of a RHS R_l of a rule r_l and a forbidden graph pattern $F_i \in \mathcal{F}$ (cf. [1]), we can check that no counterexample exists (and $\Phi_{\mathcal{F}}$ is thus an inductive invariant) only considering this finite set.

4.3 Timed Case

Verification of inductive invariants for TGTS differs from the untimed case. According to the time model we have introduced in Section 4.1 the system's behavior is given by the alternation between discrete and time steps. Therefore, reaching a forbidden graph pattern in principle could involve a rule application as well as a time step. Basically the approach for the timed case maps the time related aspects on a system of linear inequalities which can then be checked by a constraint solver which require that all conditions of the timed graph transformation system are linear.[4] The discrete aspects are handled similar to the untimed case.

For a property $\Phi_{\mathcal{F}}$ being an inductive invariant for an TGTS $S = (\mathcal{R}, \mathcal{R}_u, p)$, with $\mathcal{F} = (F_1, \psi_1) \ldots (F_1, \psi_1)$ it is required that there exists no pair $((G, \alpha), r)$ of an attributed graph (G, α) and an attributed rule $r \in \mathcal{R}$ such that $(G, \alpha) \models \Phi_{\mathcal{F}}$, $(G, \alpha) \rightarrow_r \rightarrow_\delta (G', \beta)$, and $(G', \beta) \not\models \Phi_{\mathcal{F}}$. If such a pair $((G, \alpha), r)$ exists it is, analogously to the untimed case, called a *counterexample* for the timed case.[5]

Again, only a finite set of representative patterns $\Theta((F_i, \psi_i), r_l)$ of graph patterns P' that are combinations of a RHS R_l of a rule $r_l = ((L_l, \phi_l), R_l, \mu_l)_{r_l}$ and a forbidden graph pattern $(F_i, \psi_i) \in \mathcal{F}$ have to be considered.

In Figure 6 a scheme for the verification of a TGTS is depicted. According to this we have to check for any graph pattern $(P', \phi_{P'}) \in \Theta((F_i, \psi_i), r_l)$ for some $(F_i, \psi_i) \in \mathcal{F}$

[4]It is to be noted that also early versions of UPPAAL (cf. [45]) relied on constraint solving for the verification of real-time systems.

[5]This condition in fact requires that for an initial state (G, α) we check not only $(G, \alpha) \models \Phi_{\mathcal{F}}$ but also $(G, \alpha \oplus x) \models \Phi_{\mathcal{F}}$ for all x with $(G, \alpha) \rightarrow_x (G, \alpha \oplus x)$.

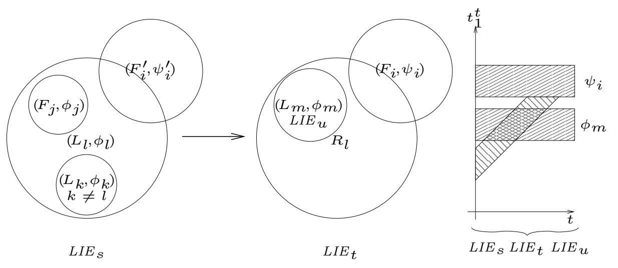


Figure 6: Schema to check a potential counterexample $((P, \phi_P), r_l)$ with resulting graph pattern $(P', \phi_{P'})$ that is a combination of a RHS R_l of a rule r_l and a forbidden graph pattern $(F_i, \psi_i) \in \mathcal{F}$ in the timed case

and $r_l \in \mathcal{R}$ whether the pair $((P, \phi_P), r_l)$ with (P, ϕ_P) defined by $(P, \phi_P) \rightarrow_r \rightarrow_\delta (P', \phi_{P'})$ is a counterexample for $\Phi_{\mathcal{F}}$ or not as follows:

1. Check that the rule r_l can be applied to attributed graph pattern (P, ϕ_P) and that the $(P', \phi_{P'})$ results from this application plus a time step of length $\delta \geq 0$ (this implies that no $r_k \in \mathcal{R}_u \setminus \{r_l\}$ exists with $p(r_k) > p(r_l)$ that matches (P, ϕ_P) and that for all $x \leq \delta$ holds that $(P', \phi_{P'} \ominus x)$ is matched by no $r_m \in \mathcal{R}_u$, due to the definition of rule application).

2. Check that there exists no $(F_j, \phi_j) \in \mathcal{F}$ with $(F_j, \phi_j) \sqsubseteq (P, \phi_P)$ (otherwise (P, ϕ_P) is already invalid).

For rules with higher priority ((L_k, ϕ_k)) as well as forbidden attributed graph patterns ((F_j, ψ_j)) holds in the case they also contain clock constraints that a match found in the source graph pattern does not directly invalidate the counterexample but rather restrict the possible clock values. We derive a system of linear inequalities LIE_s to encode this. Further the application of the rule r may either update or not affect clock variables (cf. 4.1) this has to be encoded in the system of linear inequalities, too.

We have to show the target graph pattern's reachability. In the untimed case this has been done, if the graph rule could not be preempted. This check is required but not sufficient for the timed case. Urgent rules may in fact prevent that we reach a clock evaluation which fulfills the clock constraints of the structural embedded forbidden graph patterns. This effect has to be encoded in a system of linear inequalities including boolean conditions.

Thus, we can finally use a solver for linear inequalities to check the forbidden graph pattern's reachability. In case the system of linear inequalities is not satisfiable the current pair is not a witness against system's correctness. Otherwise, the found clock and attribute valuations serve together with the current pair as witness against the system's correctness.

Listing 1: Complete Algorithm

```
Boolean check(($\mathcal{R}, \mathcal{R}_u, p$), $\mathcal{F}$)
begin
  forall ($F_i, \psi_i$) $\in \mathcal{F}, r_l \in \mathcal{R}$ do
    C := checkPair(($\mathcal{R}, \mathcal{R}_u, p$), ($F_i, \psi_i$), $r_l$);
    if (C == $\emptyset$) then
```

```
        return false;
      fi
    done
    return true;
end
```

The complete algorithm performs this check for any given rule $((L_l, \phi_l), R_l, \mu_l)_{r_l} \in \mathcal{R}$ and forbidden graph pattern $(F_i, \psi_i) \in \mathcal{F}$ by computing in a subroutine `checkPair` as depicted in Listing 1.

`checkPair` computes all possible target graph patterns ($\Theta((F_i, \psi_i), R_l, \mu_l)$ and then derives the related source graph patterns. The above outlined cases are then employed to decide whether the source graph pattern (P, ϕ_P) represents potentially safe graphs that can be transformed into unsafe graphs by applying r plus a time step δ.

5 Incremental Invariant Checking

For the incremental checking we have to store some information gathered during the checking. The `checkPair` function therefore returns the information which elements permit to exclude the existence of a counterexample in form of a check.

Definition 1 *If for a TGTS $S = (\mathcal{R}, \mathcal{R}_u, p)$ no counterexample for any $(P', \phi_{P'}) \in \Theta((F_i, \psi_i), r_l)$ for some $(F_i, \psi_i) \in \mathcal{F}$ and $r_l \in \mathcal{R}$ could have been found using higher priority rules $\mathcal{R}' \subseteq \mathcal{R}$ with $\forall r' \in \mathcal{R}' : p(r') > p(r_l)$ that preempt r_l, urgent transitions $\mathcal{R}'_u \subseteq \mathcal{R}_u$ which preempt that (F_i, ψ_i) can be reached and forbidden graph pattern $\mathcal{F}' \subseteq \mathcal{F}$ that exclude that the initial graph was correct, we call the tuple $(\mathcal{R}', \mathcal{R}'_u, \mathcal{F}', (F_i, \psi_i), r_l)$ a valid* check *for $\Theta((F_i, \psi_i), r_l)$.*

The checking of $S \models \Phi_{\mathcal{F}}$ as outlined in Listing 1 therefore equals to derive a complete set of checks defined as follows:

Definition 2 *A set of checks C is* complete *iff for each forbidden pattern $(F_i, \psi_i) \in \mathcal{F}$ and all rules $r_l \in \mathcal{R}$ of a TGTS $S = (\mathcal{R}, \mathcal{R}_u, p)$ a valid check $(\mathcal{R}', \mathcal{R}'_u, \mathcal{F}', (F_i, \psi_i), r_l)$ exists.*

To be able to cope with incremental changes to the TGTS S or forbidden graph pattern $\mathcal{F}$, we can exploit the following result:

Lemma 1 *A valid check $(\mathcal{R}', \mathcal{R}'_u, \mathcal{F}', (F_i, \psi_i), r_l)$ for a TGTS $S = (\mathcal{R}, \mathcal{R}_u, p)$ and forbidden graph pattern set $\mathcal{F}$ is also a valid check for a TGTS $S'' = (\mathcal{R}'', \mathcal{R}''_u, p'')$ and forbidden graph pattern $\mathcal{F}''$ if*

$$\mathcal{R}' \cup \{r_l\} \subseteq \mathcal{R}'' \wedge \mathcal{R}'_u \subseteq \mathcal{R}''_u \wedge \mathcal{F}' \subseteq \mathcal{F}'' \wedge$$
$$\forall r \in \mathcal{R}' : prio(r) > p(r_l) \Rightarrow prio''(r) > p''(r_l).$$

Proof: *The result follows from the fact that (1) all higher priority rules in $\mathcal{R}'$ which preempt r_l are still present in $\mathcal{R}''$ and still have a higher priority than R_l, (2) all urgent transitions $\mathcal{R}'_u$ which preempt that (F_i, ψ_i) can be reached are still present in $\mathcal{R}''_u$ and (3) that all forbidden graph pattern in $\mathcal{F}'$ that exclude that the initial graph was correct are also still present in $\mathcal{F}''$.* □

An algorithm that for a given change of the TGTS S reconstruct a complete set of checks `W` rather than derived it fully anew by exploiting the result of Lemma 1 can thus work as outlined in Listing 2.

Listing 2: Incremental algorithm

```
Set check($(\mathcal{R}'', \mathcal{R}''_u, p''), \mathcal{F}'', (\mathcal{R}, \mathcal{R}_u, p), \mathcal{F}$,W)
begin
   Set RP := $\emptyset$; Set W' := $\emptyset$;
   if ($\mathcal{R}'' \subset \mathcal{R} \vee \mathcal{R}''_u \subset \mathcal{R}_u$) then
      forall $(F_i, \psi_i) \in \mathcal{F}'' \cap \mathcal{F}, r_l \in \mathcal{R}'' \cap \mathcal{R}$ do
         // check condition of Lemma 1
         if ($(\mathcal{R}', \mathcal{R}'_u, \mathcal{F}', (F_i, \psi_i), r_l) \in$W $\wedge$
              $\mathcal{R}' \cup \{r_l\} \subseteq \mathcal{R}'' \wedge \mathcal{R}'_u \subseteq \mathcal{R}''_u \wedge \mathcal{F}' \subseteq \mathcal{F}''$
              $\wedge \forall r \in \mathcal{R}' : p(r) > p(r_l) \Rightarrow p''(r) > p''(r_l)$
            ) then
            //store valid check in W'
            W' := W' $\cup$ $\{(\mathcal{R}', \mathcal{R}'_u, \mathcal{F}', (F_i, \psi_i), r_l)\}$;
         else
            //not by W covered pairs
            RP := RP $\cup$ $\{((F_i, \psi_i), r_l)\}$;
         fi
      done
   else
      //all old checks remain valid
      W' := W;
   fi
   //pairs for all new forbidden pattern
   forall $(F_i, \psi_i) \in \mathcal{F}'' \setminus \mathcal{F}, r_l \in \mathcal{R}''$ do
      RP := RP $\cup$ $\{((F_i, \psi_i), r_l)\}$;
   done
   //pairs for all new rules
   forall $(F_i, \psi_i) \in \mathcal{F}'', r_l \in \mathcal{R}'' \setminus \mathcal{R}$ do
      RP := RP $\cup$ $\{((F_i, \psi_i), r_l)\}$;
   done

   //check for all accumulated pairs
   forall $((F_i, \psi_i), r_l) \in$RP do
      C := checkPair($(\mathcal{R}'', \mathcal{R}''_u, p'')), (F_i, \psi_i), r_l$);
      if (C == $\emptyset$) then
         return $\emptyset$;
      else
         W' := W' $\cup$ C;
      fi
   done
   return (W');
end
```

The following theorem proves that the incremental algorithm is also correct even though in several cases no new checks are done.

Theorem 1 *The algorithm presented in Listings 2 returns a complete set of valid checks iff the non-incremental algorithm presented in Listing 1 returns true.*

Proof: *Assuming that this is not the case, we have to consider the following cases: (1) If one algorithm returns an empty set resp. false while the other algorithm did not, we can conclude that the `checkPair` returns different results when called in both algorithms for at least one pair which is not possible. (2) If the incremental algorithms returns a non empty `W'` and the old one true, the incremental algorithm might still contain invalid checks. We can exclude that the call of `checkPair` has resulted in such an invalid check (as this would result in the same error for the non-incremental algorithm) and restrict us to the case where the check has been reused. However, as the check has been only reused when the criteria of Lemma 1 are fulfilled (either by the check in the inner if-statement in line 7-9 or when the rule and forbidden graph pattern set have been only extended as checked by the outer if-statement in line 4), such an invalid check in the set `W'` of the incremental algorithm can be excluded.* $\square$

The non-incremental algorithm requires $|\mathcal{F}''| * |\mathcal{R}''|$ calls to `checkPair`. For the incremental algorithm we can observe the following worst-case complexity results: (1) If the we add a new rule, this can result in $|\mathcal{F}''|$ required checks. (2) If the we add a new forbidden graph pattern, this can result in $|\mathcal{R}''|$ required checks. (3) If we remove a rule or a forbidden graph pattern and no old check can be reused, we require $|\mathcal{F}''| * |\mathcal{R}''|$ checks. Therefore, in case (1) and (2) the worst-case complexity can be improved from quadratic to linear only while in case (3) no speedup can be guaranteed. However, as outlined in the following section, in the average case we pretty often reuse the stored checks to also speedup the treatment of case (3).

6 Implementation & Evaluation

In this Section we want to give some insights into the implementation details of our algorithm. The Section concludes with a small evaluation of our findings.

The basic idea underlying our implementation is to store the set `W'` not as set but as a bipartite graph. This graph is called dependency graph G_D. G_D is defined as follows: $G_D = (V_D, E_D, L_D)$ where $V_D = V_D^S \uplus V_D^T$ is a set of nodes, $E_D \subseteq V_D^S \times V_D^T$ is a set of edges and two labeling function $L_D^S : V_D^S \mapsto (\mathcal{R} \times \mathcal{F})$ and $L_D^T : V_D^T \mapsto \mathcal{R} \cup \mathcal{F}$. A node $v \in V_D^S$ represents a pair of a graph rule and a forbidden graph pattern, whereas a node $v' \in V_D^T$ stands for a graph rule or a forbidden graph pattern respectively. When a call to `checkPair` returned a new check $(\mathcal{R}', \mathcal{R}'_u, \mathcal{F}', (F_i, \psi_i), r_l)$, we memorize this by adding for s the node with $L_D^S(s) = (r_l, (F_i, \psi_i))$ the following edges to the dependency graph:

$$E'_D = E_D \cup \{(s,t) | L_D^T(t) \in \mathcal{R}' \cup \mathcal{R}'_u \cup \mathcal{F}'\}.$$

In the case of removal of an graph rule or forbidden graph pattern j the incremental algorithm traverses the removed node's incoming edges and receives a list of pairs, whose checks relies on the removed node.

$$E'_D = E_D \setminus \{(s,t) \in E_D | L_D^T(t) = j\}$$

For those pairs s with $(s,t) \in E_D$ and $L^T_D(t) = j$ we have to check whether we can find another check.

We have evaluated our approach by applying several successive modifications to the system. The applied modifications subsume the update of a rule, the addition of new forbidden graph pattern and the addition of a new rule. We started the evaluation with a configuration of our system that only contained Shuttles. This system consists of four rules and two forbidden subgraphs and its verification took 1553ms. For this initial system configuration (C_0) obviously no incremental check was applicable. The first applied modification (yielding configuration C_1) replaced the rule moveSimple (cf. Figure 2) by a version which also considered WorkShuttles. In configuration C_2 we added the forbidden graph pattern required for WorkShuttles and finally in configuration C_3 we added a rule, which allows for moving of WorkShuttles. All of the checked configurations are safe.

	t_{all}	t_{inc}	Δ_t	#R	#P	Δ_R	Δ_P
C_1	1690	411	137	4	2	2	0
C_2	3347	742	1657	4	8	0	6
C_3	5075	894	1728	5	8	1	0

Table 1: Evaluation results

The evaluation results are depicted in Table 1. The rows contain the data for the configurations an the columns denote the time for old algorithm t_{all}, the time for the incremental one t_{inc}, the time difference between C_i and C_{i-1} for the old algorithm Δ_t. #R, #P, Δ_R and Δ_P refer to the numbers of rules and forbidden subgraphs and the changes applied to these sets.

The results show that the incremental algorithm can also in case of rule removal (C_1) perform much better than the non-incremental algorithm. In two out of three configurations the incremental algorithm is even faster than the differences between the durations of the non-incremental algorithm for both configurations.[6] While the presented results are promising, the available data set is not sufficient to derive any more general observations.

7 Modeling of correct self-adaptive systems

It is a fact of life that software is always embedded in a social and technical context which changes over time. Therefore to preserve the value of the software changes are unavoidable which adjust the software to its current context [29]. The rate at which such adjustments are required can vary from the short term reaction to frequent context changes (e.g., mobility) or very slow and only gradual changes (e.g., the user behavior, new requirements). While the former need rapid system adaptation using pre-defined heuristics or adaptation rules, effective support to change, extend, or migrate complex

[6]This unexpected observation can result as the old algorithm checks all pairs with the changed sets of rules and forbidden graph patterns, while the incremental algorithm only checks the newly created pairs with the extended system.

software systems is today responsible for a large fraction of the maintenance efforts to operate a complex software.

Adaptive and self-managed systems [27] which realizes properties such as self-healing, self-configuring or self-optimization employing self-adaptation [32,34,42] have been proposed as a means to address these different challenges.

The outlined requirements also hold for the next generation of advanced mechatronic systems which will adjust their behavior to the changing system goals leading to self-adaptation respectively self-optimization (cf. [38]). In this context the MECHATRONIC UML approach has been developed [12, 13] which due to the restrictions of the domain can provide a comprehensive model-driven approach for the development. However, no such approach for the more general case exists where the limitations exploited in the MECHATRONIC UML approach do not apply.

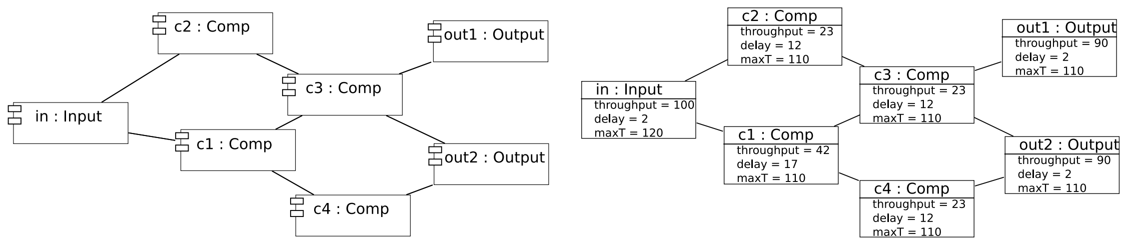
It is a fact that for a systematical engineering of self-adaptive systems modeling and early analyzation is required. But as far as we know there already exist several approaches that tackle the modeling of self-adaptive or dynamic systems. According to [10] none of the existing approaches is able to model structural as well as behavioral changes. The dynamic change of structural aspects has been investigated in [22, 28, 31,36,43], whereas the authors of [5,14,26,46] have put an emphasize on the modeling of behavioral aspects.

In this paper a proposal for a comprehensive modeling approach for self-adaptive systems with a rigorously defined formal semantics that enables to also address the question of correctness will be presented. The approach requires that all required modeling artifacts can be covered at the theoretical level by means of attributed graph and attributed graph transformations. This formal model provides powerful concepts to describe the structural changes which are initiated by the several layers of self-adaptive software and thus is well suited to serve as the underlying formal model.

Instead of the underlying formal model of attributed graphs we suggest to employ well proven software engineering techniques such as object diagrams consisting of instances of a class diagrams to capture the state in a more convenient way. For the different forms of behavior also specific graphical notations that cover the required behavior in form of graph transformations are suggested. An extended combination of activity diagrams and collaboration diagrams named Story Diagrams (cf. [24]) capture the required operational parts. Goals are captured by a specification technique for combine structural and temporal properties [18]. These four diagrams are employed for a case study to demonstrate how the software structure and behavior including also self-adaptive software with parameter as well as structural changes can be modeled.

For the modelling of behavior with graph transformations several well founded approaches exist (e.g. AGG[7]). Both, the work of Taentzer et al. [43] and Le Mètayer [31] also used graph transformations to describe system changes. In contrast to the presented approach both approaches restrict themselves to pure graphs and changes described by a single graph transformations while we will suggest UML structural modeling and combine graph transformations with control flow concepts. In addition, both approaches use graphs but not attributed graphs and are therefore more constrained concerning their expressiveness than the presented approach.

[7]http://tfs.cs.tu-berlin.de/agg/



(a) Instance Situation of a component network (b) Component network shown in figure 7(a) as typed and attributed graph

Figure 7: Different representations for a component model

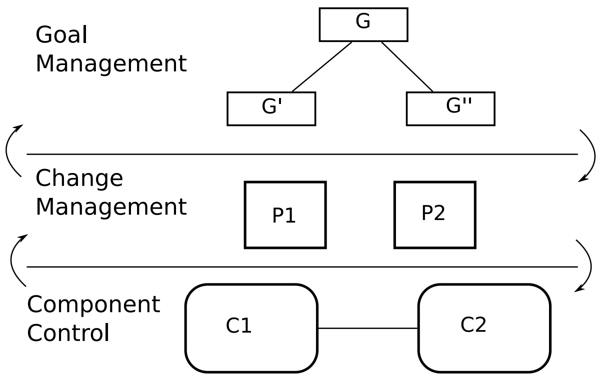


Figure 8: Reference architecture for self-adaptive systems (from [27])

The application example we have chosen for this paper is a self-adaptive Pipe and Filter architecture. In Figure 7(a) each filter is represented by a component and each pipe by a connection between two components. Thus the components build a network. The network is able to perform an operation on the input that it receives from the input component. The results are available at the output components. The self-adaptive system we have in mind can be optimized for two contradictory objectives. The two objectives are the data rate that could be measured at the output components or the delay that means the amount of time if takes until results are available for a given input.

We assume that each component provides three attributes throughput, maxT and delay that hold the values for the data rate the component currently is able to provide, the component's maximum data rate and the time the component delays computation, respectively. The components communicate with each other by message passing.

8 Requirements

We will use this section to point out what requirements an approach has to meet in order to be considered useful for modeling correct self-adaptive systems.

We will use the reference architecture for adaptive and self-managed systems proposed in [27] depicted in Figure 8 to outline our approach and characterize the possible different forms of self-adaptation. The component control layer (CCL) contains the system's components and their computational logic. The CCL is monitored by the change management layer (CML) which is able to change the CCL's structure in order to restart components etc. The topmost layer is the goal management layer (GML) that monitors both the CCL and indirectly the CCL. In the GML the system's "intelligence" is located. Whenever the GML recognizes unwanted behavior, a plan is computed and passed to the CML in order to regain an acceptable state (for details see [27]).

In system design one has to decide what a system's characteristics are. For most more complex systems this includes decisions among centralized and decentralized systems, deterministic and nondeterministic systems etc. Of course all these choices are valid for a self-adaptive system and following each formal model should support them. Nevertheless, we have identified the following other requirements: (1) internal and external change initiation (2) addition / removal of architectural elements (3) different levels of abstraction (4) possibility to change system's environment (5) possibility to express time constraints (6) modeling of control flow (7) intuitive modeling and (8) correctness. This list only contains the most relevant properties. Special domains or intentions might require additional features. Special for the modeling of self-adaptive systems are properties (1) to (4). The requirements (5) to (8) charaterize a good modeling approach in general. In the following we will present the arguments that support our decision.

In some systems the system decides on its own how to change its structure. In a different setup this change could be intiated by an component, which is located outside the system's bounds. Following an approach for the modeling of self-adaptive systems should support internal and external change initiaion (1).

The addition and removal of architectural elements (2) is a basic property a formal approach has to provide. In case the system has to react on some events that prevents some of the system's goals to be feasible, the system has to change its goals. Changing goals usually means that new goals are added (or activated) and old ones are removed (or deactivated) from the system. Therefore it must be possible to add and remove elements. Of course the same holds for other layers as well, if goals change this could require an addition or a removal of architectural elements at the component level.

The different levels of abstraction (3) are heavily associated to the three layers of the reference architecture for self-adaptive systems. The topmost layer (goal management) might be interested in a much more abstract view of the system's state than the lowest level. Further verification techniques are only capable to verify relatively small models, so a high degree of abstraction is useful as well.

Self-adaptive systems are much more context aware than other systems are. This requires to explicitly model the system's context. Of course this also affects the system, so it could be necessary to monitor the environment to be able to react on changes. Following the changes in environment (6) have to be expressible by the approach.

The complexity of self-adaptive systems requires a modeling approach which is able to provide some techniques for the modeling of control-flow (6). The system's goals typically include if-then constructs that, especially if interaction with the environment is needed, can be augmented by timing constraints (5).

Especially for complex systems it is of high importance that the cognitive gap between modeling notation, formal model and developed system is not too big. The bigger the gap, the harder mistakes are to identify. Correctness (8) should be assured for all systems of course. For self-adaptive systems this requirement is of increased importance as those systems change their behavior on their own without or with few human interaction. Followingly, it should be assured that the system is safe.

9 Modeling Self-Adaptation

We propose to use a graph transformation based approach to model self-adaptive systems at a high level of abstraction. To explain the benefits of this approach we will first present the different modeling notations used at each layer and then outline how the presented notations relate to graph transformation systems. Finally we will discuss how the explicit modeling of representative models employed in the change management and goal management can be avoided by idealization.

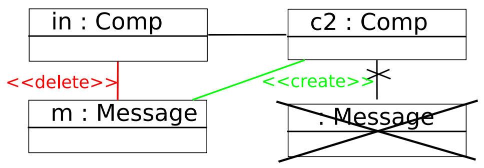


Figure 9: Story Pattern that describes the message passing between two components

Even if our approach has some similarities with MDA - i.e. it also uses model transformations between different layers - it can not be compared to MDA. The main difference is that MDA performs activities at design time, whereas we propose a modeling approach self-adaptive systems and following the acitivies are scheduled at run-time. However, we use techniques such as automatic code generation for all employed modeling techniques and following our approach can be seen as MDE.

9.1 Layers

Throughout this section we will present which modeling notations relate to which system layer and explain how these modeling notations could be mapped to the formal model introduced earlier. We will introduce a possible mapping of the requirements and the three-layer-architecture for self-adaptive systems to the beforehand introduced formal model.

The state and the state space of the CCL, CML and GML is described through one UML class diagram, that is partitioned in disjoint views assigned to each layer, and UML object diagrams. For our application example the class diagram is depicted in Figure 10. The different layers can be distinguished by the different background colors. The CCL is the lowest layer, the CML is aligned in the middle and the topmost layer is the GML. The abstract class Marker is not assigned to any layer. It is only used for making modeling easier. Each class diagram view could easily be translated into a set of types and a mapping of attributes to types. At instance level the object diagram

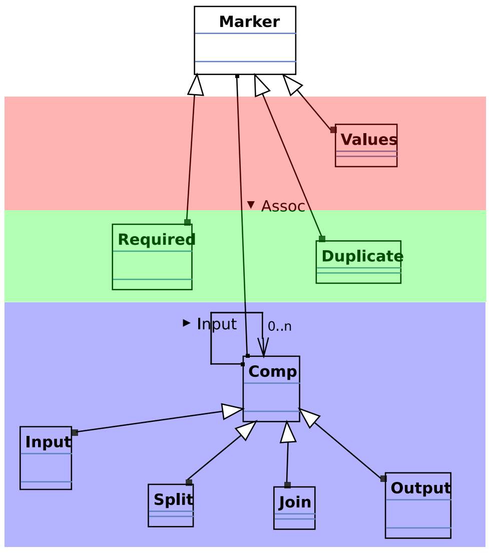


Figure 10: Class diagram of the example system showing the separation into three layers

provides information on the object's type, which then will be used to create an typed and attributed graph. The type sets of the different layers are named $\mathcal{T}_{CC}$, $\mathcal{T}_{CM}$ and $\mathcal{T}_{GM}$ respectively.

Component Control Layer Beside structure the CCL also consists of some behavior we have to model, too. We employ Story Patterns to describe the behavior within this layer. In Figure 9 an example for such a Story Pattern is given. This Story Pattern is sufficient as long as all components only have one successor. In case of multiple successors a little more complicated Story Pattern has to be used that implements a loop. For our model we abstract from the reception of the three attributes throughput, delay and maxT. For simulation purposes etc. the values can be set non-deterministically. Story Pattern in the CCL formally correspond to graph transformations over the state which only refer to node and edges of $\mathcal{T}_{CC}$. Together with the information given in the respective UML class diagram it is possible to completely describe the CCL as a typed and attributed GTS.

- Attributed graph transformations which are restricted to $\mathcal{T}_{CC}$ are internal to the CCL. Other attributed graph transformation are not allowed at the CCL.

Change Management Layer For the CML we use the same modeling techniques as for the CCL, but we also use an extended variant of Story Pattern called Story Diagrams. Story Diagrams augment Story Pattern with control flow elements as they are known from UML activity diagrams. In our application example the CML is responsible for the duplication of marked components. The reunion of duplicated components also lies in the change management's scope, but is not to be discussed in this paper. Figure 11 shows one of the Story Diagrams used at this layer. The Story Diagram's first activity ensures that only components that have an marker attached are considered. The activity's shape indicates that it is a for-each activity. For-each activities could be used to model loops as they could be executed as long as the Story Pattern contained in them matches in the host graph. So, for each component that has to be duplicated the following is done. First Split and Join objects are created. Then each of the components successors is connected to the newly created Join instance. Afterwards the predecessor is connected to the Split instance. Finally the component gets duplicated, the connections with Split and Join are set, the Duplicate mark gets deleted and the throughput each component has to provide gets halved.

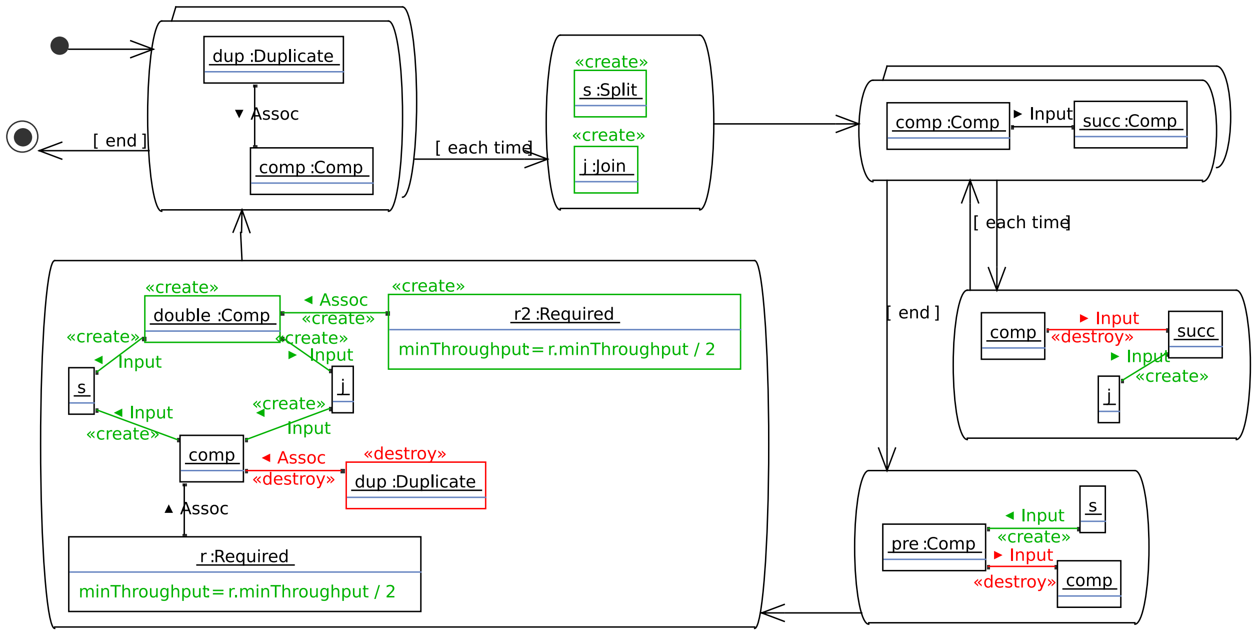


Figure 11: Story Diagram that duplicates a component if demanded by the GML

As the atomic actions of a Story Diagram are Story Pattern, a Story Diagram could be translated into a sequence of graph transformations. The CML's behavior is defined above the classes of the CML and the CCL. The behavior described by the attributed graph transformation rules over the state of the CCL and CML can be classified as follows:

- Attributed graph transformations which are restricted to $\mathcal{T}_{CM}$ are internal to the CML.
- Attributed graph transformations which pre-condition requires elements from $\mathcal{T}_{CM}$ and $\mathcal{T}_{CC}$ relates to *status* checks of the CML with respect to the CCL.
- Attributed graph transformations which post-condition requires elements from $\mathcal{T}_{CM}$ and $\mathcal{T}_{CC}$ relates to *change action* of the CML with respect to the CCL.

Therefore, the CML can be characterized by its capability to change its own state as well as the state of the CCL. Please note that changing a system's structure also yields a change in the systems behavior (cf. [27])

Goal Management Layer The GML is used to plan the system's overall behavior in order to meet its goals. For our rather simple application example this means finding components that have to be duplicated. A component has to be duplicated if it has a Required mark attached that demands a higher throughput than the component currently provides. The Story Diagram in Figure 12 implements a simplified heuristic to identify such components. In the first row the Story Diagram computes the input the component currently has to process. The second and third row of contain if-else activities (those contain a read-only Story Pattern and have exactly two outgoing transitions that labeled with success and failure respectively) that evaluate the prior results. In case the amount of data is less than the required throughput the Required mark is set to any of the preceding components. In case the amount of data is too high for the component to handle, the component is marked with a Duplicate mark.

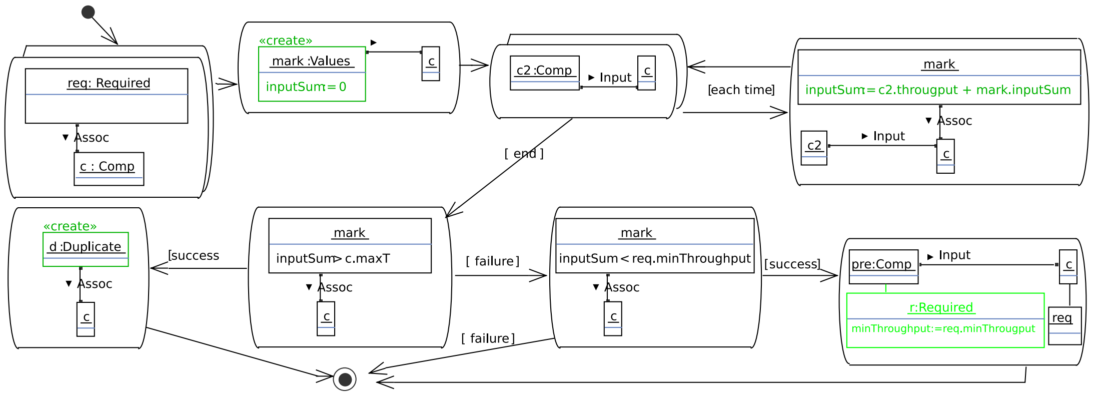


Figure 12: Story Diagram that describes the Goal Management's actions

According to [27] the GML is not only responsible for the computation of change plans, but also for the management of goals, which the system is supposed to meet. In our approach goals are captured by a specification technique for combine structural and temporal properties [18] called TSSDs. TSSDs are very similar to Story Diagrams but do not change the system. Further they allow to quantify single objects instead of a whole graph pattern as Story Diagrams does. In Figure 13 an example TSSD is depicted. The TSSD declares that whenever an Output component has a Required marker attached then the component's throughput must be higher than the required throughput within 100 time units.

Currently it is impossible, given goal specifications, to generate behavior which fulfills the goals.[8] But it is possible to generate code, which monitors behavior and identifies violations of the specified goals (cf. [41]).

- Attributed graph transformations which are restricted to $\mathcal{T}_{GM}$ are internal to the GML.

- Attributed graph transformations which pre-condition requires elements from $\mathcal{T}_{GM}$ and $\mathcal{T}_{CM}$ relates to *change request* of the CML with respect to the GML.

[8]Note that this is a general problem typically referred to as protocol synthesis, which is not specific to our approach

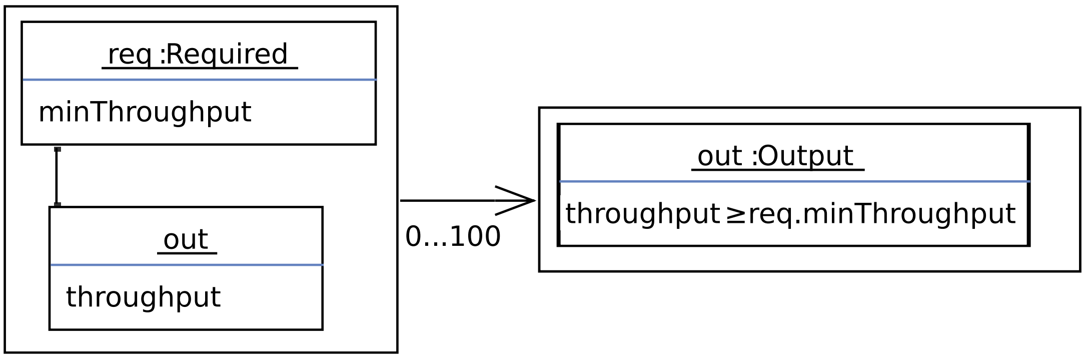


Figure 13: TSSD demanding high throughput after at least 100 time units

- Attributed graph transformations which post-condition requires elements from $\mathcal{T}_{CM}$ and $\mathcal{T}_{CC}$ relates to *change plans* of the GML provided to the CML.

Consequently, the GML can be characterized by its capability to change its own state as well as the state of the CML. It is not expected that the GML has direct access to the component control layer.

In addition, goals can be captured by properties which combine structural properties with temporal constraints on the occurrence for the sequences of attributed graphs denoted by $TRACE((G, \alpha))$ which result from the execution of the graph transformation systems with initial state (G, α).

At this point it becomes obvious that our approach is able to handle distributed changes. There is no need that all layers of one system are deployed to the same computing node. They could be distributed and even access the same layer concurrently.

9.2 Representative Models

In the previous paragraphs we have seen that each higher layer works on or with the state of the layers below. So for example the Story Diagram shown in Figure 11 (which is assigned to the CML) iterates over the CCL's components and duplicate them if necessary. Following the CML requires either direct access to the CCL or indirect via a copy of a CCL, which is periodically synchronized with the CCL. Assuming a decentralized system direct access is not an option, as the CCL must not be stopped.

In practice the CML and GML have to interpret the data they receive concerning the CCL and CML respectively to locally maintain a related model of their current state. This model is usually incomplete and may also not be fully up-to-date all the time.

A first solution might be techniques such as triple graph grammars (TGGs) [39] which permit to update these models of the lower layer system in form of a graph when required which could simplify the modeling and developing efforts to a great extent. To make even frequent updates feasible even incremental versions for the synchronization of models via TGGs [20] may be applicable. TGGs will be translated into a set of operational rules, which are Story Pattern again. Following it is possible to describe the synchronization in terms of a GTS.

If we want to abstract from this additional effort and from the delayed updates of the representative models, we can, however, go one step further and permit the CML and GML to ”reuse” the real CCL as substitute for their model. Required constraints on the visibility of information about the CCL have than to be respected by the rules of the CML and GML. This can be accomplished at the type and attribute level by restricting the known types accordingly such that the rules cannot access any additional information. While the employed idealization has its limitations, the following step-wise process concerning the modeling of representative models is suggested.

1. Start modeling assuming full access to all layers below.
2. Refine your model taking known or expected access restrictions into account.
3. If necessary also consider the extraction of the representative models and their frequent updates using techniques such as TGGs.

Using TGGs it is easily possible to either clone one layer to make it visible or available in another layer or to abstract from the concrete representation. The representative models ensure that our approach is able to work with different levels of abstraction (cf. requirement 3).

9.3 Discussion

So far we have presented a formally underpinned modelling approach which fullfills all the requirements 2,4 and 5 as well as the requirements 1,3,6 and 7. The Marker elements set in our application example initiate the required changes. As the system does not distinguish who has set the mark internal and external change initation are supported. Reference Models allow us to introduce different level of abstractions, and Story Diagrams provide the required control flow for more complex operations. Requirement (7) is obviously met, too.

10 Correctness

Throughout this Section we will present how the correctness of our modelled self-adaptive systems could be ensured. Therefore we present two different approaches: simulation and verification.

10.1 Simulation

In the early stages of the development process developers generally are more interested in a system solving the given problem rather than having a system which already ensures all safety properties. Thus, simulation is of great interest within the early stages. For our approach, presented in this paper, this is achieved by code generation techniques, which are available for all of our presented modeling notations, and a tool called Dobs (cf. [25]). Dobs allows the visualization of objects currently in memory and the execution of methods provided by these objects. Dobs could be seen as a debugger who is able to graphically display the current heap.

In Figure 14 a screenshot of Dobs is shown. The screenshot depicts a situation equivalent to the situation shown in Figure 7(b). The visualization used within the simulation of our models is very similar to our proposed modeling notations and following understanding and judging program behavior is eased.

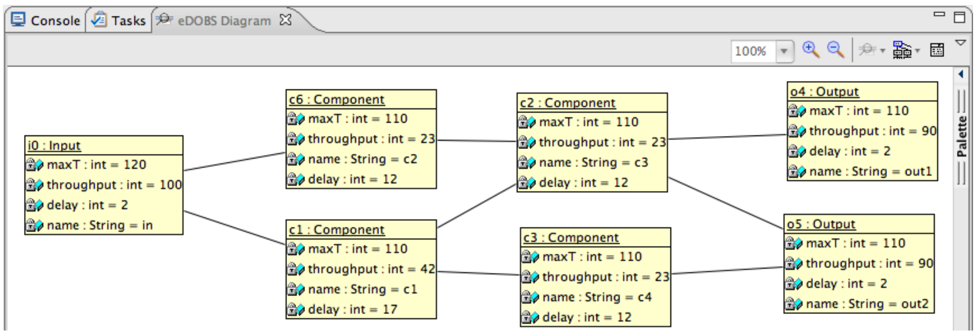


Figure 14: Screenshot of Dobs showing an instance situation

10.2 Formal Verification

Beside the strict semantics another advantage of having a formally underpinned modeling approach is the capability to verify the modelled system. Our approach is mainly based on graph transformations and thus the verification technique we use is tailored to this expressive formal model. We have developed a verification technique called invariant checking (cf. [1,4]) which is dedicated to the verficiation of graph transformation systems.

Invariant Checking could be used to verify that a given set of graph transformations will never reach a forbidden state, assuming the system starts in a valid state. Graphs are used to model forbidden situations and a state is called invalid or forbidden if it contains one of those graphs. The verification result only depends on the verified set of graph transformations and forbidden situations. Followingly the verification result is independent of the system's start situation and hence a set of graph transformations, which have been verified to satisfy a given specification, can be used in diffent systems without verifying them again.

In Section 8 we required the addition and removal of components being one of a modeling approach's properties. Following this should also be supported by a verification technique. Due to the fact that Invariant Checking only considers the set of graph transformations and forbidden situations this obviously holds. Using Invariant Checking we do not have to build the complete state space of the system - in fact we are unable to do it, because the verification is independent of the start state. Thus, typical problems such as state space explosion does not apply to Invariant Checking. On the other hand the checked invariants are structural invariants and thus properties known from temporal logics could not be verified. The same holds for the proof of deadlock freedom.

With Invariant Checking we are also able to verify graph transformation systems, which are augmented with timing constraints (cf. [4]). Therefore, clocks are modeled as arguments of the nodes and the graph transformations are augmented with constraints, which restrict the attributes valuations. Forbidden situations also have an additional constraint.

Invariant Checking has been evaluated in several papers (cf. [1, 4]) and has proven to be an efficient verification technique. The verification's time complexity is linear in the number of rules and properties to be checked.

For smaller systems it is also possible to translate our models into the input language used by GROOVE (cf. [23]). GROOVE then tries to build a labeleed transition system containing all reachable states. Given this transition system it is possible to apply standard model-checking algorithms to our modelled self-adaptive systems. Of course this requires a finite and considerably small system. For a more detailed discussion and related work to this aspect please see [1].

11 Conclusion and Future Work

We presented in this paper the formal model of graph transformation systems and discuss how it serves most of the needs for modeling self-adaptive systems. Our proposed approach for modeling self-adaptive systems based on this formal model employs UML class and object diagrams and related extension of UML behavioral diagrams for the modeling of structural changes. We use a simplified application example to present how the approach can be employed to model self-adaptive systems at a high level of abstraction.

The presented approach still has some limitations we have not solved, yet. The constraints that restrict the applicability of Story Pattern have to be linear inequalities. Further, the attributes are limited to either clocks or constants. For the modeling of complex physical environments this is insufficient.

In the future we plan to further improve our approach. To be able to create even more flexible rules we plan to add some reflection capabilities to GTS. A reflective GTS would be much more expressive but the formal analysis might be impossible. In order to be able to build a reflective GTS we first need an interpreter for Story Pattern and Story Diagrams. Beside this an interpreter can help to ensure the separation into layers (as stated in 9.1)

Concerning the verification of self-adaptive system we developed an approach to incrementally check that changed rule sets which describe bottom-up self-adaptive behavior adhere to the still present or newly added safety properties. The incremental nature of the procedure permits to considerably reduce the checking effort in many cases. While in some cases of changes also an speedup in the worst case could be guaranteed, in case of erasing rules or parts of the invariant no speedup for the worst case can be guaranteed.

The presented technique is a first step towards checking the run-time evolution of systems with bottom-up self-adaptive behavior. It can be applied when an atomic switching between different rules sets can be guaranteed. As future work we plan to provide more evaluation results and look into the problem of decentralized evolution and how this can be supported by run-time checks.

Own Work

[1] Basil Becker, Dirk Beyer, Holger Giese, Florian Klein, and Daniela Schilling. Symbolic Invariant Verification for Systems with Dynamic Structural Adaptation. In *Proc. of the* 28^th *International Conference on Software Engineering (ICSE), Shanghai, China*. ACM, 2006.

[2] Basil Becker and Holger Giese. Incremental Verification of Inductive Invariants for the Run-Time Evolution of Self-Adaptive Software-Intensive Systems. In *Proc. 1st Intl. Workshop on Automatic Engineering of Autonomous and Run-Time Evolving Systems*. IEEE Computer Society Press, 2008. to appear.

[3] Basil Becker and Holger Giese. Modeling of Correct Self-Adaptive Systems: A Graph Transformation System Based Approach. In *Proceedings 1st IEEE Intl. Workshop on Autonomous and Autonomic Software-Based Systems*. IEEE Computer Society Press, 2008. to appear.

[4] Basil Becker and Holger Giese. On Safe Service-Oriented Real-Time Coordination for Autonomous Vehicles. In *In Proc. of 11th International Symposium on Object/component/service-oriented Real-time distributed Computing (ISORC)*, pages 203–210. IEEE Computer Society Press, 5-7 May 2008.

References

[5] Robert Allen, Rémi Douence, and David Garlan. Specifying and analyzing dynamic software architectures. *LNCS*, 1382:21, 1998.

[6] Jim Amsden, Pete Rivett, Kolk Henk, Fred Cummins, Jishnu Mukerji, Antoine Lonjon, Cory Casanave, and Irv Badr. *UML Profile and Metamodel for Services*, June 2007. http://www.omg.org/docs/ad/07-06-03.pdf.

[7] Paolo Baldan, Andrea Corradini, and Barbara König. A Static Analysis Technique for Graph Transformation Systems. In *Proc.*
CONCUR, volume 2154 of *LNCS*, pages 381–395. Springer, 2001.

[8] Luciano Baresi, Reiko Heckel, Sebastian Thone, and Daniel Varró. Modeling and Validation of Service-Oriented Architectures: Application vs. Style. In *ESEC/FSE-11: Proceedings of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 68–77, New York, NY, USA, 2003. ACM.

[9] Jörg Bauer and Reinhard Wilhelm. Static Analysis of Dynamic Communication Systems by Partner Abstraction. In *Proceedings of the 14th International Symposium, SAS 2007, Kongens Lyngby, Denmark, August 22-24, 2007*, volume 4634 of *Lecture Notes in Computer Science*, pages 249–264. Springer Berlin / Heidelberg, 2007.

[10] Jeremy S. Bradbury, James R. Cordy, Juergen Dingel, and Michel Wermelinger. A survey of self-management in dynamic software architecture specifications. In *WOSS 2004*, pages 28–33. ACM, 2004.

[11] Manfred Broy, Ingolf Krüger, and Michael Meisinger. A formal model of services. *ACM Trans. Softw. Eng. Methodol.*, 16(1):5, 2007.

[12] Sven Burmester, Holger Giese, and Matthias Tichy. Model-Driven Development of Reconfigurable Mechatronic Systems with Mechatronic UML. In Uwe Assmann, Arend Rensink, and Mehmet Aksit, editors, *Model Driven Architecture: Foundations and Applications*, volume 3599 of *Lecture Notes in Computer Science (LNCS)*, pages 47–61. Springer Verlag, August 2005.

[13] Sven Burmester, Matthias Tichy, and Holger Giese. Modeling Reconfigurable Mechatronic Systems with Mechatronic UML. In *Proc. of Model Driven Architecture: Foundations and Applications (MDAFA 2004), Linkoping, Sweden*, June 2004.

[14] Carlos Canal, Ernesto Pimentel, and José M. Troya. Specification and refinement of dynamic software architectures. In *Proc. IFIP*, pages 107–126. Kluwer, 1999.

[15] Michel Charpentier. Composing Invariants. In *Proc. of International Symposium of Formal Methods Europe*, volume 2805 of *Lecture Notes in Computer Science*, pages 401–421. Springer, 2003.

[16] Marcelo F. Frias, Juan P. Galeotti, Carlos Lopez Pombo, and Nazareno Aguirre. DynAlloy: Upgrading Alloy with actions. In *Proc. ICSE*, pages 442–451. ACM, 2005.

[17] Holger Giese. Modeling and Verification of Cooperative Self-adaptive Mechatronic Systems. In Fabrice Kordon and Janos Sztipanovits, editors, *Reliable Systems on Unreliable Networked Platforms - 12th Monterey Workshop 2005 . Laguna Beach, CA, USA, September 22-24,2005 . Revised Selected Papers*, volume 4322 of *Lecture Notes in Computer Science*, pages 258–280. Springer Verlag, 2007.

[18] Holger Giese and Florian Klein. Systematic Verification of Multi-Agent Systems based on Rigorous Executable Specifications. *International Journal on Agent-Oriented Software Engineering (IJAOSE)*, 1(1):28–62, April 2007.

[19] Holger Giese, Matthias Tichy, Sven Burmester, Wilhelm Schäfer, and Stephan Flake. Towards the Compositional Verification of Real-Time UML Designs. In *Proc. of the European Software Engineering Conference (ESEC), Helsinki, Finland, Proc. of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering (ESEC/FSE-11)*, pages 38–47. ACM Press, 2003.

[20] Holger Giese and Robert Wagner. Incremental Model Synchronization with Triple Graph Grammars. In Oscar Nierstrasz, John Whittle, David Harel, and Gianna Reggio, editors, *Proc. of the 9th International Conference on Model Driven Engineering Languages and Systems (MoDELS), Genova, Italy*, volume 4199 of *Lecture Notes in Computer Science (LNCS)*, pages 543–557. Springer Verlag, October 2006.

[21] Matthias Güdemann, Frank Ortmeier, and Wolfgang Reif. Formal modeling and verification of systems with self-x properties. In *Proceedings of the Third International Conference on Autonomic and Trusted Computing (ATC-06)*, 2006.

[22] Dan Hirsch, Paolo Inverardi, and Ugo Montanari. Graph grammars and constraint solving for software architecture styles. In *Proc. of ISAW'98*, pages 69–72. ACM, 1998.

[23] Harmen Kastenberg and Arend Rensink. Model Checking Dynamic States in GROOVE. In *Model Checking Software, 13th International SPIN Workshop, Vienna, Austria, March 30 - April 1, 2006, Proceedings*, volume 3925 of *Lecture Notes in Computer Science*, pages 299–305. Springer Berlin / Heidelberg, 2006.

[24] Florian Klein, Ulrich Nickel, Jorg Niere, and Albert Zündorf. From UML to Java And Back Again. Technical Report tr-ri-00-216, University of Paderborn, Paderborn, Germany, September 1999.

[25] Hans J. Köhler, Ulrich A. Nickel, Jörg Niere, and Albert Zündorf. Integrating UML Diagrams for Production Control Systems. In *Proc. of the 22^nd International Conference on Software Engineering (ICSE), Limerick, Ireland*, pages 241–251. ACM Press, 2000.

[26] J. Kramer and J. Magee. Analysing dynamic change in software architectures: A case study. In *Proc. of the International Conference on Configurable Distributed Systems*, page 91. IEEE, 1998.

[27] Jeff Kramer and Jeff Magee. Self-Managed Systems: an Architectural Challenge. In *FOSE 0́7: 2007 Future of Software Engineering*, pages 259–268, Washington, DC, USA, 2007. IEEE Computer Society.

[28] Jeff Kramer, Jeff Magee, and Morris Sloman. Configuring distributed systems. In *Proc. SIGOPS*, pages 1–5. ACM, 1992.

[29] Meir M. Lehman. Softwareś Future: Managing Evolution. *IEEE Software*, 15(01):40–44, 1998.

[30] Philip K. McKinley, Seyed Masoud Sadjadi, Eric P. Kasten, and Betty H.C. Cheng. Composing Adaptive Software. *IEEE Computer*, 37(7), July 2004.

[31] Daniel Le Métayer. Software architecture styles as graph grammars. In *Proc. SIGSOFT*, pages 15–23. ACM, 1996.

[32] David J. Musliner, Robert P. Goldman, Michael J. Pelican, and Kurt D. Krebsbach. Self-Adaptive Software for Hard Real-Time Environments. *IEEE Inteligent Systems*, 14(4), July 1999.

[33] Peter C. Olveczky and José Meseguer. Specification and Analysis of Real-Time Systems Using Real-time Maude. In Tiziana Margaria and Michel Wermelinger, editors, *Proceedings on Fundamental Approaches to Software Engineering (FASE2004)*, volume 2984 of *Lecture Notes in Computer Science*. Spinger-Verlag Heidelberg, 2004.

[34] Peyman Oreizy. A Flexible Approach to Decentralized Software Evolution. In *Proc. of the 21th International Conference on Software Engineering (ICSE)*, pages 730–731, 1999.

[35] Peyman Oreizy, Michael M. Gorlick, Richard Taylor, Dennis Heimbigner, Gregory Johnson, Nenad Medvidovic, Alex Quilici, David S. Rosenblum, and Alexander L. Wolf. An Architecture-Based Approach to Self-Adaptive Software. *IEEE Intelligent Systems*, 14(3):54–62, June 1999.

[36] Peyman Oreizy, Nenad Medvidovic, and Richard N. Taylor. Architecture-based runtime software evolution. In *Proc. ICSE*, pages 177–186. IEEE, 1998.

[37] Arend Rensink. Towards Model Checking Graph Grammars. In Michael Leuschel, S. Gruner, and S. Lo Presti, editors, *Workshop on Automated Verification of Critical Systems (AVoCS)*, Technical Report DSSE–TR–2003–2, pages 150–160. University of Southampton, 2003.

[38] Wilhelm Schäfer and Heike Wehrheim. The Challenges of Building Advanced Mechatronic Systems. In *FOSE 0́7: 2007 Future of Software Engineering*, pages 72–84. IEEE Computer Society, 2007.

[39] Andy Schürr. Specification of graph translators with triple graph grammars. In *Graph-Theoretic Concepts in Computer Science 20th International Workshop*, volume 903 of *Lecture Notes in Computer Science*, pages 151–163. Springer, 1994.

[40] Giovanna Di Marzo Serugendo, Marie-Pierre Gleizes, and Anthony Karageorgos. Self-Organisation in MAS. *Knowledge Engineering Review*, 20(2):165–189, 2005.

[41] Michael Spijkerman. Monitoring gemischt struktureller und temporaler Eigenschaften von UML Modellen. Master's thesis, University of Paderborn, 2007. german.

[42] Janos Sztipanovits, Gabor Karsai, and Ted Bapty. Self-Adaptive Software for Signal Processing. *Communications of the ACM*, 41(5):66–73, 1998.

[43] Gabriele Taentzer, Michael Goedicke, and Torsten Meyer. Dynamic change management by distributed graph transformation: Towards configurable distributed systems. In *TAGT 1998*, pages 179–193. Springer-Verlag, 2000.

[44] Dániel Varró. Automated formal verification of visual modeling languages by model checking. *Software and System Modeling*, 3(2):85–113, 2004.

[45] Wang Yi, Paul Pettersson, and Mats Daniels. Automatic verification of real-time communicating systems by constraint-solving. In Dieter Hogrefe and Stefan Leue, editors, *Proc. of the 7th International Conference on Formal Description Techniques*, volume 6 of *IFIP Conference Proceedings*, pages 243–258. Chapman & Hall, 1994.

[46] Jian Zhang and Betty H.C. Cheng. Model-based development of dynamically adaptive software. In *ICSE 0́6: Proceeding of the 28th international conference on Software engineering*, pages 371–380, New York, NY, USA, 2006. ACM.

On a Model for a Service Database

Mohammed AbuJarour
mohammed.abujarour@hpi.uni-potsdam.de

As a consequence of the wide adoption of the Service-Oriented Architecture (SOA) in businesses, the role of service brokers has become vital, especially to manage services and their metadata. Most standard features of service brokers, i.e., Service Registries and Repositories (SRR), are similar to those of Database Management Systems (DBMS), but in different contexts and environments. In this work, we propose a new approach to extend the features of the DBMS's to provide the standard features needed in a service broker.

1 Introduction: Service Brokers vs. DBMS

The focus of my research activities has shifted, from Information Retrieval and Information Integration to Service Management. Service Management is a vital field in Service-Oriented Computing. In this field, Information Retrieval techniques and concepts can be adapted or extended to handle many issues in Service Management. For example, XML Information Retrieval is so close to manage service artifacts. As shown in Table 1, current service brokers do not provide any means of service data quality and integration. I considered this problem in my first technical report [13], and I emphasize it here again because of its importance.

One of the most common terms used for service brokers is Service Registries and Repositories (SRR), which play a basic role in systems based on the Service-Oriented Architecture (SOA) [18]; they manage, track, and classify services, both their data and the metadata. Organizational polices are also enforced through service brokers in SOA systems. Service consumers find the needed service(s) and establish the necessary binding with the help of the service registry.

A service registry, like UDDI [10] or ebXML [5], contains only links or pointers to service metadata or artifacts; it does not contain the artifacts themselves. On the other hand, a service repository stores the metadata and artifacts itself [4]. One example of service brokers is the ebXML Registry-Repository [4]. Table 1 lists some of the key features of the ebXML Registry-Repository, and emphasizes the fact that a traditional database does not provide such features. In our work, we try to integrate these missing features into the DBMS itself, in order to get rid of any added complexity associated with building a service broker on top of a DBMS.

Enterprise systems based on the *n-tier* [16] software architecture rely on the data-

Standard Feature	**Database**	**ebXML Registry-Repository**
Service discovery	Inadequate query languages	Yes
Evolution and version management	No	Yes
Standard protocols for subscription and notification	Database Change Notifications (DCN) not enough	Yes
SOA security	No	Yes
Integration	No	No
Service data quality	No	No
Entailed complexity	None	A new layer of complexity

Table 1: The main features of the ebXML Registry Repository [4].

tier to manage application data. In traditional software and programing paradigms, e.g., structural programing, the managed data includes numerical, date-time, or string values. For such data, relational databases [15] are sufficient. As software architectures and applications developed to the object-oriented paradigm, databases evolved correspondingly to the object-oriented databases [19] where the main entity is an object.

Recently, software architectures have moved to the Service-Oriented Architecture *(SOA)*. A software application under this architecture is a collection of services interacting over a network, such as the Internet, in order to achieve a business task or process. A logical response to this recent trend in software development is the introduction of *Service Databases*. The main entity in this model is the service, which encapsulates the necessary features that builds up a service, including the data, metadata, polices, and functionalities.

The description of the addressed problem follows this introductory section. Before we introduce our proposed approach in Section 4, we refer to the main contributions that have been made in this field in Section 3. Further implementation issues are discussed in Section 5. We close with a conclusion in Section 6.

2 The Problem: The Rising Complexity of Services

Managing services and the information about them is no longer an easy job that could be achieved using a traditional middleware solution. The main reason for this is the increasing complexity of service information [12]. The types of service metadata are versatile, e.g., XML, BPEL, XSLT, WSRP .. etc. Moreover, a service could be described in more than one type.

There are several important issues to consider when a service broker is designed (as shown in Table 1). Among these issues are:

- **Service discovery**: the primary functionality of a service broker is to help service consumers find the needed services. Automatic cataloging of services and their metadata is a necessary feature to achieve this requirement.

- **Evolution and version management**: service providers usually try their best to provide the optimal service for their consumers. This is reflected in releasing new versions of the previously published services. A service broker should be able to handle such evolutions. It could simply replace the old service with the new release. It could also keep the previous versions and let the consumer decide which version to use taking into consideration compatibility and roles issues.

- **Subscription and notification protocols**: because a service broker is a meeting point for many systems, it wold not be feasible for a service consumer to browse the whole system to find a particular service, that has been added recently. This could be time-consuming. As an alternative, the broker itself could provide protocols for subscription and notification, that notify the interested user about the interesting services or changes.

- **Security**: the nature of SOA systems pushes the need for security to the surface, e.g. Identity Management, Organization and federated Policy Enforcing.

- **Integration**: a service broker is a meeting point for several systems (from different organizations or organizational divisions) and hence should be engineered as a SOA integration platform. It should consider different data types and legacy systems.

- **Service data quality**: the amount of information about services is usually so huge [12]. This requires automatic content validation to ensure that the stored information is valid and useful. Another related issue to consider here is duplicates management and redundancy because a service could be described using more than one data type and a service could be provided by several service providers.

Existing service brokers have some limitations. For example, in [14], the authors showed that about 53% of the links discovered in UDDI Business registries were inactive. In our model, all information should be verify to be up-to-date. Old information and inactive links should be always discarded.

3 Related Work: Service Brokers

IBM's WebSphere Service Registry and Repository (WSRR) "is the master metadata repository for service interaction endpoint descriptions" [11]. WSRR is not limited to web services (WSDL), but it deals also with SOA services described using SCDL, XSD and policy decorations; other service metadata, e.g., WSRP , could not be managed by WSRR. Additionally, WSRR does not manage service metadata across the whole SOA life cycle; it interacts and federates with other metadata stores that do this. Another limitation of WSRR is that it does not support service data quality.

Sun's Service Registry is a Registry and Repository for web services [9]. Sun's Service Registry is based on ebXML Registry 3.0 with added support for UDDI 3.0. This SRR is limited to web services only; it does not deal with SOA services in general.

Centrasite [2] is also a registry and repository for web services, which is based on UDDI 3.0. It is a useful tool for SOA architects, SOA developers, and business analysts. According to its information model, Centrasite stores metadata about services, as well as the files that relate to those artifacts. Centrasite is limited to the web services inside the organization only. This means that it integrates different web services from different divisions within the organization.

FreebXML [4] is an open source, functionally complete reference implementation for the OASIS ebXML Registry specifications as defined by the OASIS ebXML Registry Technical Committee [6]. FreebXML is a general purpose registry and repository. Several organizations and associations have already deployed the FreebXML Registry and Repository, such as United Nations CEFACT ICG, Government of Canada, France Telecom and others. Service data quality is not one of the features of the FreebXML Registry and Repository.

The QWS Dataset is a collection of web services gathered over the web. It includes about 5,000 web services - so far. These web services were discovered using the Web Service Crawler Engine (WSCE) from UDDI registries, search engines and service portals [8]. We plan to use this list as an initial test case.

4 Approach: A DBMS-based Service Broker

"ebXML Registry-Repository is the database for Web Services" [4]. Based on this assumption and abstracting it one step further,we could say that a registry-repository is the database for services in general.

Traditional service brokers build up a new layer on top of a database layer, which increases the complexity of such systems. The basic idea of our approach, as shown in Figure 1, is to combine both concepts - SRR and Database - into a single model coined Service Database Model, which is basically a database model aimed at managing services in SOA systems like a service broker does. This model is designed to deal with both contents and metadata, which facilitates metadata-based discovery.

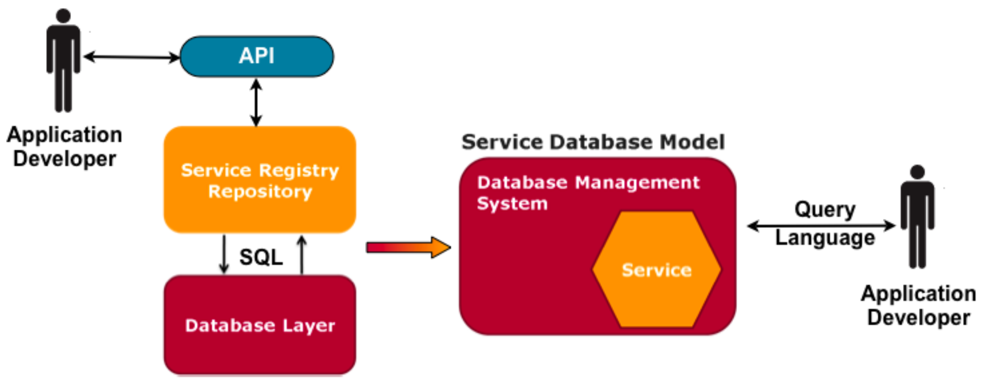


Figure 1: The Basic Idea of the Service Database Model.

In our approach, the client deals with an abstract type called *service*. The system, which implements this model, deals then with the management of the underlying and technical parts of the service. This includes usual information about the service, e.g., name, provider, version . . . , as well as actual files that describe the service, e.g. WSDL ,BPEL . . . . This hides the the complexity of the managed services and their metadata, and simplifies the communication with the broker. A client could simply ask for services using SQL commands, or an appropriate UI on top of SQL.

Cataloging services in a service broker and indexing tables in a relational database are very similar in concept. Publishing new information involves automatic generation of metadata that is used to discover the published information efficiently [12].

Most current databases already have a component called Database Change Notifications (DCN), which provides server-side primitives that allow clients or applications to subscribe to query results, and the server notifies them as soon as these query results have been changed [17]. We extend this approach in our Service Database Model to provide standard methods for subscription and notifications in services and their metadata. It would be valuable for a client to receive a notification when a new version of a service it uses has become available.

Major relational and object database systems, such as Oracle and DB2, provide XML support. Either by mapping XML to existing concepts or by providing *native* XML support. DB2 falls into the latter category [20]. In our service database model, a similar approach with some extensions can be used to handle service data and metadata, as they are based on XML. PostgreSQL does not support all XML capabilities and features [7]. For example, it supports indexing and search XML documents, but it does not have an XML data type. Upcoming releases of PostgreSQL will include more and more of the missing features. In [21], for example, the authors provide a roadmap for the development of native XML type support in PostgreSQL.

Other standard features of service brokers are also already implemented in DBMS's, e.g., role-based access control through users and groups, and allowing information to be linked and searched securely across and outside organizational boundaries using distributed features of DBMS's.

5 Implementation Issues: Service Management

Search engines and crawling techniques could be employed to discover services within an environment, e.g., web services on the web. Special attention should be payed here because of the nature of web services, where we usually lack enough descriptions. We are going to test the best techniques in this field to automate service discovery and validation on the web. This is part of a current master thesis. This will provide us continuously with many services that we will try to manage under our model.

As soon as we gather many services, we need to have a common abstraction model to describe a service. This model facilitates the interaction between the searching and crawling part and the broker part. This step is still in progress.

As a next step, we have to choose an existing DBMS and try to introduce our model inside it. There are adequate open-source DBMS's, e.g., Apache Derby [1]. DBMS's are either relational -like MySQL-, object-oriented - like PostgreSQL-, or mix of both - like DB2. From an implementation perspective for our model, a DBMS has a advantage over others if it is an object one because a service is an object, with special features.

In Table 2, we compare between five important DBMS's [3]. Some of the addressed DBMS's are commercial, but they are included here because they are important and common, like Oracle and DB2. This comparison nominates PostgreSQL because it is open source and object-oriented DBMS. It has many key features, such as ACID, Transactions, and Blobs and Clobs support. On the other hand, Apache Derby has an important feature, which is the ability to get it embedded into Java programs [1]. In the upcoming months, we will test potential alternatives and choose among them the most suitable one.

	Apache Derby	DB2	MySQL	Oracle	PostgreSQL
Maintainer	Apache	IBM	Sun Microsystems	Oracle Corporation	PostgreSQL Global Development Group
First public release	2004	1982	1996	1979	1989
Latest stable version	10.4.1.3	9.5	5.0.67	11g Release 1	8.3.4
Software license	Apache License	Proprietary	GPL or proprietary	Proprietary	BSD
Fundamental Features	ACID Referential integrity Transactions Unicode SQL interface	ACID Referential integrity Transactions Unicode GUI & SQL interface	ACID Referential integrity Transactions Unicode-Partial SQL interface	ACID Referential integrity Transactions Unicode SQL interface	ACID Referential integrity Transactions Unicode SQL interface
Database Capabilities	Union Inner joins Outer joins Blobs and Clobs	Union Intersect Except Inner joins Outer joins Inner selects Merge Blobs and Clobs	Union Inner joins Outer joins Inner selects Merge Blobs and Clobs	Union Intersect Except Inner joins Outer joins Inner selects Merge Blobs and Clobs	Union Intersect Except Inner joins Outer joins Inner selects Merge Blobs and Clobs
Classification	Relational	Object-relational	Relational	Object-relational	Object-relational

Table 2: Comparing some major DBMS's.

6 Conclusion: Service Management with DBMS

In this work we propose a new database model, the Service Database Model, that helps SOA systems manage their services efficiently. We borrow the ideas used in DBMS's to solve the problem of managing services and their metadata and to provide the necessary standard features to run a SOA system.

Existing service registries and repositories could be classified in two classes based on purpose. One category includes those which include the registry component only, like Centrasite, and the other one includes those which include both the registry and the repository components, like WebSphere Service Registry and Repository. Another possible classification is the classification by domain; some SRR's deal *only* with web services, e.g., Sun's Service Registry, others deal with web services and SOA services, e.g., freebXML.

Some service broker vendors tend to provide functionalities that are vendor-specific [12]. By adopting our approach, there is always a common underlying platform, which can be used by different implementations to communicate and cooperate. Service discovery will be straightforward; it will be just like issuing an SQL statement to a Database Management System. The complexity of having descriptions in many, different formats will not be visible any more since the user or business-logic tier is using the same query language.

References

[1] Apache Derby. http://db.apache.org/derby/.

[2] Centrasite community. http://www.centrasite.org.

[3] Comparison of relational database management systems. http://en.wikipedia.org/wiki/Comparison_of_relational_database_management_systems.

[4] ebXML registry-repository. http://ebxmlrr.sourceforge.net.

[5] Electronic business using eXtensible markup language. http://www.ebxml.org.

[6] Organization for the advancement of structured information standards. http://www.oasis-open.org.

[7] PostgreSQL 8.2.10 documentation. http://www.postgresql.org/docs/8.2/static/datatype-xml.html.

[8] Quality of web services. http://www.uoguelph.ca/~qmahmoud/qws/index.html.

[9] SUN's service registry. http://www.sun.com/products/soa/registry.

[10] UDDI 3.0. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uddi-spec.

[11] Websphere service registry and repository. www.ibm.com/software/integration/wsrr.

[12] Effective SOA deployment using an SOA registry repository. 2005. White paper.

[13] Mohammed AbuJarour. Data integration in web services. Technical report, Hasso-Plattner-Institut, Potsdam, Germany, April 2008.

[14] Eyhab Al-Masri and Qusay H. Mahmoud. Investigating web services on the world wide web. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 795–804, New York, NY, USA, 2008. ACM.

[15] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970.

[16] Wayne W. Eckerson. Three tier client/server architecture: Achieving scalability, performance, and efficiency in client server applications. *Open Information Systems*, 10(1), 1995.

[17] Cesar Galindo-Legaria, Torsten Grabs, Christian Kleinerman, and Florian Waas. Database change notifications: primitives for efficient database query result caching. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 1275–1278. VLDB Endowment, 2005.

[18] Nicolai Josuttis. *SOA in Practice: The Art of Distributed System Design*. O'Reilly Media, Inc., 2007.

[19] Won Kim. *Introduction to object-oriented databases*. MIT Press, Cambridge, MA, USA, 1990.

[20] Matthias Nicola and Bert van der Linden. Native XML support in DB2 universal database. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 1164–1174. VLDB Endowment, 2005.

[21] Nikolay Samokhvalov. XML support in PostgreSQL. In *SYRCoDIS*, volume 256 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

Towards the Automatic Generation of Effective, Map-Like Visual Representations from Heterogeneous Geodata in a Service-Oriented Infrastructure

Dieter Hildebrandt
dieter.hildebrandt@hpi.uni-potsdam.de

Visual representations of geospatial information proved to be valuable means to facilitate thinking, understanding, and knowledge construction about human and physical environments, at geographic scales of measurement. In particular, maps and map-like 2D and 3D visual representations are prominent and common examples of forms of visual representations that are leveraged in various application areas. For creating map-like representations, the Internet has become the prominent medium through which geodata is distributed and accessed.

In this paper, we address the design and implementation of standards-based systems that facilitates the automatic generation of effective map-like 2D and 3D visual representations from heterogeneous geodata in a service-oriented infrastructure. We identify basic requirements for such systems, identify concepts and methods that support implementing such systems and discuss their strengths and weaknesses. Furthermore, we outline the high-level design of a configurable, standards- and service-based visualization pipeline that can be applied for their implementation. Particular focus is placed on the integration of heterogeneous geodata and generalization as an essential operator for the generation of map-like visual representations.

1 Introduction

Visual representations of geospatial information proved to be valuable means to facilitate thinking, understanding, and knowledge construction about human and physical environments, at geographic scales of measurement [26]. In particular, maps and map-like 2D and 3D visual representations are prominent and common examples of forms of visual representations that are leveraged in various application areas. For creating map-like representations, the Internet has become the prominent medium through which geodata is distributed and accessed. In modern environments, map-like representations have the potential to literally use the Internet as its “database”, acting as integrators for and dynamic portals to interconnected, distributed, heterogeneous geodata resources [29]. Within various application areas, fully automating the process of generating map-like visual representations from distributed, heterogeneous geodata sources and specified visualization requirements is desirable and required in order to maximize cost reduction, flexibility, acceptance and applicability. Furthermore, generating visual representations that are generally effective and meet specified visualization requirements is required for maximizing the usability of the representations.

For the design and implementation of distributed systems exhibiting the aforementioned characteristics, the architectural concept *service-oriented architecture* (SOA) is commonly proposed [17, 25]. In order to improve the interoperability of distributed, geospatial services, the Open Geospatial Consortium (OGC) [1] approves standards commonly accepted in research and industry.

Presently, the implementation of a standards-based system that is required to facilitate the automatic generation of effective map-like visual representations from heterogeneous geodata in a service-oriented infrastructure faces several issues including the following. No proposals by the OGC or other parties exist for services that perform integration of heterogeneous geodata or automated generalization, a central operation for creating map-like representations. The data models proposed by the OGC do not support integration of heterogeneous geodata. The portrayal services proposed by the OGC (WMS, W3DS and WPVS) do not generate effective map-like visual representations in general.

In this paper, we address the design and implementation of standards-based systems that facilitates the automatic generation of effective map-like 2D and 3D visual representations from heterogeneous geodata in a service-oriented infrastructure. We identify basic requirements for such systems, identify concepts and methods that support implementing such systems and discuss their strengths and weaknesses. Furthermore, we outline the high-level design of a configurable, standards- and service-based visualization pipeline that can be applied for their implementation. Particular focus is placed on the integration of heterogeneous geodata and generalization as an essential operator for the generation of map-like visual representations.

The remainder of this paper is organized as follows. Section 2 briefly describes general requirements for visual representations of geospatial information and presents the conceptual process of generating map-like visual representations. In Section 3, we briefly present the state of the art of standards for service-oriented geovisualization systems as defined by the OGC and present extensions and supporting concepts. Section 4 presents a functional decomposition of a proposed visualization pipeline into a set of operators. In Section 5, we discuss some weaknesses of current proposals presented in Section 3. Section 6 outlines the high-level design of a configurable, standards- and service-based visualization pipeline. Finally, Section 7 concludes the paper and outlines future work.

2 Generating Effective Map-Like Visual Representations

In this Section, we briefly describe general requirements for visual representations of geospatial information and present the conceptual process of generating map-like visual representations.

2.1 Requirements for Visual Representations

General requirements for visual representations of geospatial information are expressiveness, effectiveness and appropriateness [46]. A visual representation is considered *expressive* if only the information contained in the data is represented and the information is represented unaltered. A visual representation is considered *effective* if it supports the goal for its creation in an optimal way taking into account the context of the application. Finally, if the tradeoff between efforts required for creating the visual representation and the benefits yielded by it is balanced, a visual representation is considered *appropriate*.

For a differentiated discussion, the three dimensions of semiotics [7] – syntactics, semantics and pragmatics – can be applied to the requirement of effectiveness.

Syntactic dimension The syntactic dimension comprises the relation of signs to each other in formal structures. For instance, a visual representation is not effective on the syntactic level if overlap of graphical elements impairs the legibility of the representation or too many graphical elements clutter a small display of an output medium.

Semantic dimension The semantic dimension comprises the relation between signs and the things they refer to. For instance, a visual representation is not effective on the semantic level if a user is hindered in understanding graphical elements due to an unfamiliar mapping of data to graphical variables.

Pragmatic dimension The pragmatic dimension comprises the relation of signs to their impacts on those who use them. For instance, a visual representation is not effective on the pragmatic level if it does not contain the information that a specific user needs in a specific situation.

In Sections 4 and 5, the requirements for a system generating map-like visual representations and the discussion of existing concepts and methods are presented in terms of the three introduced dimensions of effectiveness.

2.2 Map-Like Visual Representations and Generalization

Map-like 2D and 3D visual representations use the spatial dimensions of data as a standard frame of reference and represent a part of geographical reality in an abstracted way. They can be regarded as the outcome of a system that implements and executes a specific visualization process and pipeline [46]. Depending on the design of the system, the execution of the process is done manually by an expert user, semi-automatic

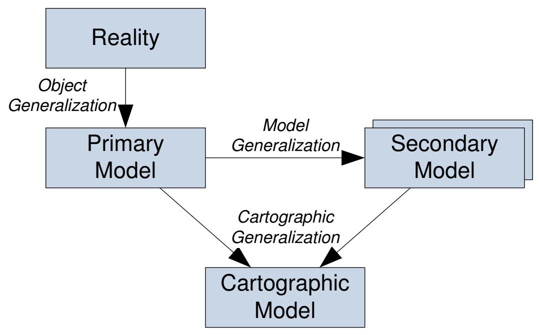


Figure 1: Abstract model of the generalization process as a set of model transformations [13].

or fully automatic. This process transforms a given set of geodata and a specified set of visualization requirements into a map-like visual representation. It is the system's responsibility to produce a representation from the given geodata that meets the specified requirements. These visualization requirements contain a specification of the targeted application area of the representation, the spatial area of interest and general requirements such as accuracy, information content and legibility. In general, the process of producing map-like visual representations is complex and sophisticated and requires a significant amount of expertise.

A central part of the process of generating maps and map-like representations is the generalization subprocess. *Generalization* can be defined as the selection and simplified representation of detail appropriate to the scale and/or the purpose of a map [18]. An established abstract model of the generalization process is given by Grünreich [13] (see Figure 1). Generalization is modeled as a set of model transformations that allow transforming aspects of reality into data sets or maps. The *object generalization* transformation selects, captures, abstracts, and reduces aspects of reality and stores these aspects digitally as the primary model. The *model generalization* transforms the primary model into a model that is optimized for performance (memory and processor requirements) and further processing in GIS or geovisualization systems but does not address any visualization aspects. The *cartographic generalization* transformation processes the primary or secondary model in order to produce a cartographic model that represents a map.

With the intention to support the automation of the generalization process, the overall process can be decomposed into a set of *generalization operators*. In the literature, various propositions for operators and the classification of operators exist that differ in both number and terminology. McMaster and Shea [30] introduced the first set of operators that was applicable to digital cartography. They proposed twelve operators: simplification, smoothing, aggregation, amalgamation, merging, collapse, refinement, exaggeration, enhancement, displacement, classification and symbolization. Because of the complexity of the problem, automated generalization that allows for unrestrictedly effective geocommunication in the general case is not yet feasible. Nevertheless, automation is already feasible for specific parts of the generalization process and for generalization processes in specific application areas. We expect substantial progress in this field in the medium term.

3 State of the Art in Service-Oriented Geovisualization Systems

In this paper, we address the generation of map-like visual representations by standardized services within a *spatial data infrastructure (SDI)*. *Services* are the key components of SDIs. They represent modular units, providing specific capabilities through an explicit interface. They allow for accessing, managing, processing, combining, and visualizing various complex and massive geoinformation sources. On the one hand, this geoinformation is widely distributed and varies, e.g., in semantics, format, and scale. On the other hand, this diverse geoinformation is a keystone in a growing number of applications and systems. The overall vision is to combine this diverse geoinformation according to actual tasks and needs, to integrate it into workflows, and thereby to support various processes by specialized services.

In SDIs, the standardization of data formats and service interfaces is a crucial issue as it provides a common basis for communication, the reuse of existing capabilities, and interoperability between system components. The standard and specification family of the OGC focuses on geospatial interoperability. They are publicly available and form a foundation for manifold activities in research and industry.

In this Section, first, we briefly present the state of the art of standards for service-oriented geovisualization systems as defined by the OGC. Then, we name relevant proposals for extending OGC standards and additional concepts that are presently not reflected by OGC standards and that we have identified as valuable for supporting the design and implementation of systems with requirements as discussed in Section 1.

3.1 OGC Standards

Figure 2 gives an overview of a selection of OGC standards and services that are relevant for geovisualization and are applied in today's SDIs. In the following, they are explained in more detail.

3.1.1 Information Management

Web Feature Service and Web Coverage Service (and the original Web Map Service) provide interfaces for accessing distributed geodata and transferring them in a standardized way, e.g., encoded in *Geography Markup Language* or as image.

The *Geography Markup Language (GML)* [39] is the fundamental OGC standard for describing geospatial features. It is an XML-encoding and supports geometry and geometric complexes, topology, coordinate reference systems (CRS), temporal and dynamic information, units, measures, values, and dictionaries, directions, observations, and coverages. GML profiles describe a subset of the GML types, attributes, and elements and can be extended to GML application schemas describing, e.g., domain-specific feature types. A prominent example of a GML application schema is CityGML, a standard for the description and exchange of 3D city models. Since version 3.0 GML provides 3D features.

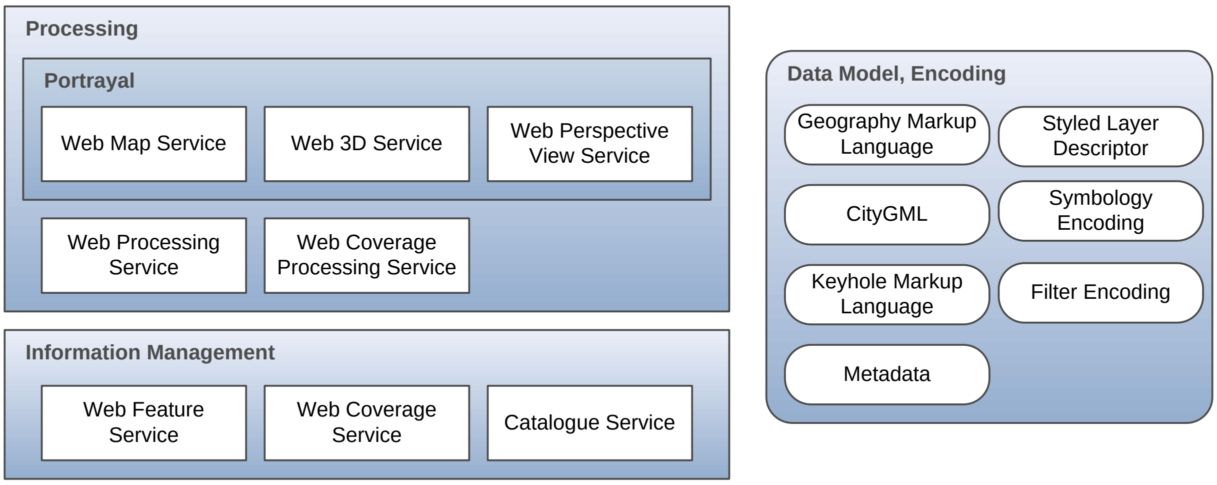


Figure 2: OGC Standards for services, data models and encodings with relevance to service-oriented geovisualization.

The *Web Feature Service (WFS)* [35] allows for accessing feature data encoded at least in GML. By the definition of a bounding box or applying filters, the retrieved data set can be restricted to a subset. The *DescribeFeatureType* operation returns a schema description for every feature type that can be retrieved from the WFS by *GetFeature* operation. The optional *Transaction* allows for data manipulation (insert, update, delete). The *Web Coverage Service (WCS)* [42] provides geospatial data as coverages, i.e., “digital geospatial information representing space-varying phenomena”. The transferred raster data sets can represent, e.g., elevation data or fume data and have to be interpreted by the service consumer.

In the domain of spatial data infrastructures, *metadata* describes geospatial information resources, regarding its discovery, evaluation and use. ISO 19115 [19] and ISO 19119 [20] propose basic metadata elements for geographic information and spatial services respectively. ISO 19139 [21] defines an applicable XML schema implementation for ISO 19115.

Catalogues are essential elements in complex and flexible service-oriented architectures, as they provide the functionality for publishing available resources and thereby make them findable. The OGC *Catalogue Services* [38] contain metadata for geospatial information, for available services, and further information needed for publishing and accessing this data. It provides operations, e.g., for finding resources, retrieving metadata, and managing the catalogue.

3.1.2 Generic Processing

The *Web Processing Service (WPS)* [40] provides so called “geospatial processes” which include any processing of geospatial data (e.g., calculations), describes its functionality (e.g., in WSDL[1] format) and makes it available to the service user.

[1]Web Service Description Language

3.1.3 Portrayal Processing

According to ISO 19119, portrayal services also represent a type of processing services. They produce image data or in the case of W3DS intermediate rendering artefacts, which are transferred as a computer graphical representation. WTS, WPVS and W3DS represent first approaches for 3D portrayal.

The *Web Map Service (WMS)* [36] is capable of generating map-like 2D images. WMS typically offers geoinformation as layers together with supported styles. A *GetMap* request defines a set of styled layers (i.e., a set of layers and specific styles to apply), which are rendered and combined at the server-side and returned to the consumer. A styling-enabled WMS is described later. The *Web Perspective View Service (WPVS)*, which was originally defined as Web Terrain Service (WTS) [43], generates perspective views of a three-dimensional scene and transfers them as image to the service consumer. The client may select from supported layers and which styling to apply to the data. Depending on the server capabilities and functionality, the WPVS can generate high-quality images, which can be displayed even with simple clients. The *Web 3D Service (W3DS)* [34] generates scene graphs, which represent computer graphical descriptions of a scene and have to be rendered by the client. On the one hand, this requires for rich clients, on the other hand the W3DS allows for interactivity as known from typical desktop applications.

Symbology Encoding (SE) [37] represents a language for defining rendering parameters for specific features and coverages. SE describes the symbolizer (line, polygon, point, text, raster) to use for rendering the feature geometry, which appearance parameters to consider, and for which scale this styling is applicable. The *Styled Layer Descriptor (SLD) Profile for WMS* [41] allows for user-defined styling: Together with the *GetMap* request, an SE-encoded SLD description is transmitted inline or as URL reference. Therefore, the WMS interface is extended for retrieving the feature types of a layer.

3.2 Proposals for OGC Extensions and Supporting Concepts

3.2.1 Information Management

A multiple representation database (MRDB) can be described as a spatial database which can be used to store representations relating to the same real world phenomena in different themes and at different levels of precision, accuracy and resolution with explicit links between the representations [6, 48]. In the context of cartographic generalization, MRDBs were successfully applied for creating adapted generalizations by exploiting existing representations within a MRDB [6, 15], storing created generalized representations for later reuse [10] and storing auxiliary data that is generated within a generalization process for later reuse [32]. Hampe et al. [16] propose an extension of the WFS standard for multiple representation data.

The integration of heterogeneous geospatial data is still an area of active research (e.g., [4,24]). Geospatial data integration can be differentiated into a purely geometrical integration and a semantical integration that includes the former. Another classification scheme is concerned with the level of heterogeneity of the data: data sets that are to be integrated can conform to the same data model or to different data models. As an example, Regnauld [45] proposes an approach for the integration of data with heterogeneous data models that is based on ontologies. A system that aims at generating map-like visual representations of data from distributed, heterogeneous sources has to incorporate semantic data integration methods that can bridge different data models.

3.2.2 Generalization Processing

The automation of the generalization process was the focus of extensive research within the last few decades. Research mainly targeted three levels of abstraction: generalization operators and implementing algorithms (e.g., [9]), the automated control of the generalization process and the application of the operators [23] and algorithms and data structures that assist the generalization process [32].

In recent years, we observed several efforts to offer generalization operators and processes as distinct services on the Internet and to standardize these services. In the WebPark project [11], a specific WMS implementation is presented that generalizes requested map layers and styles them as specified by a SLD parameter. Burghardt et al. [3] propose offering generalization functionality through dedicated generalization services and present a hierarchical categorization for these services. *Generalization support services* assist the generalization process by providing auxiliary measures, procedures and data structures. *Generalization operator services* deliver the functionality of standalone generalization operators and may use support services for their implementation. *Generalization process services* implement complete generalization processes and may use operator and support services. In addition, researchers demonstrate utilizing a workflow management system for the orchestration of generalization services [44], wrapping generalization functionality with a WPS [12] and providing prototypical implementations. Förster et al. [12] report on the early results and current work in progress of a working group that targets at the standardization of generalization services.

3.2.3 Portrayal Processing

As already discussed, the styling of the 2D portrayal from a WMS can be specified using SLD. In contrast, until now no OGC standard exists that defines the styling of 3D portrayal from a W3DS or WPVS. Nevertheless, Haist et al. [14] and Neubauer et al. [31] propose separate extensions for the SLD and SE for 3D portrayal.

4 Analysis of the Geovisualization Pipeline

In this Section, we present a functional decomposition of the visualization pipeline for generating map-like 2D and 3D visual representations from heterogeneous geodata and visualization requirements into a set of operators. Particular focus is placed on the integration of heterogeneous geodata and generalization. Furthermore, we relate

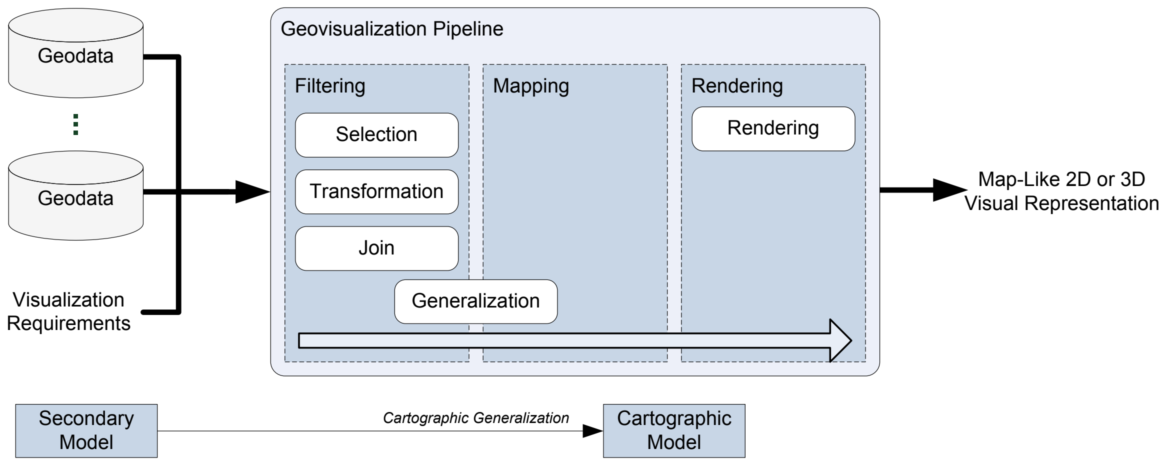


Figure 3: Instance of a geovisualization pipeline mapping geodata and visualization requirements to map-like visual representations. The proposed operators and the generalization model are related to the pipeline.

the visualization pipeline and the identified operators to the generalization model introduced in Section 2. The presented decomposition is a result of an analysis of the visualization process for the stated purpose. This analysis is based on proposals from the OGC for service-oriented geospatial systems [33], common proposals for visualization pipelines [5] and common practice for producing maps [2, 30, 47].

Figure 3 depicts the process of generating map-like visual representations from geodata visualization requirements. The process incorporates at least five operators: *selection*, *transformation*, *join*, *generalization* and *rendering*. These operators represent model transformations that transform one or more sets of input geodata into a resulting set of (geo)data. The concrete application of an operator on given data is guided by the set of given visualization requirements that act as additional input to the operator. In order to design a concrete system, a set of operators is chosen, combined and *chained* in such a way that the designed system meets given systems requirements. Each operator can be associated with a stage of the visualization pipeline, i.e., filtering, mapping and rendering. The generalization operator is an exception. Some parts of its functionality are related to filtering (e.g., data abstraction) while others are related to mapping (e.g., assignment of visual variables). In general, it is not possible to split the functionality of the operator into a filtering and a mapping stage that are executed in sequence without iteration. This is because of complex interrelations that exist between the parts. Regarding the generalization model introduced in Section 2, the input geodata corresponds to the secondary model, the intermediate output of the mapping stage corresponds to the cartographic model and the generalization operator corresponds to the cartographic generalization mapping.

Figure 4 depicts an example of a concrete process design that contains an interaction of the presented operators. This design describes a system that selects and retrieves data from two different data sources, transforms each disjoint data set into a common model, joins the transformed data and then generalizes and renders it into a 2D image that constitutes the map-like visual representation.

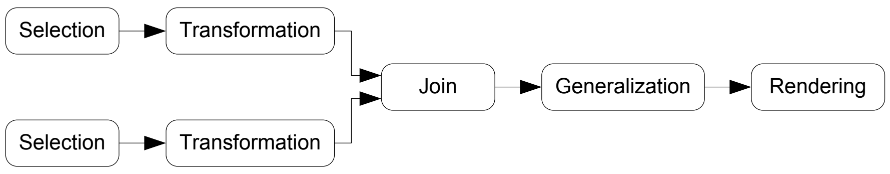


Figure 4: Example design of a concrete process that contains an interaction of the proposed operators.

4.1 Visualization Requirements

Visualization requirements are part of the input for the process. They describe on a high level of abstraction what is expected of a map-like visual representation from the perspective of a user. In order to be applicable in an automated system, the visualization requirements have to be transformed into a visualization specification. This specification describes in a formal and more concrete manner the properties of the visual representation that is to be generated. The visualization requirements should contain a description of the real world section that the visual representation should contain information on (e.g., spatial bounding box and map scale for a 2D representation and view frustum for a 3D representation), properties of the output medium (e.g., screen resolution) and information on the intended application area of the representation. The information on the intended application area of the map should include information on the user (e.g., task, professional domain, emotional condition, age, interests, skills, limitations) and context (e.g., type of device, conditions of light and sound) [22].

4.2 Operators

In order to support the generation of effective map-like visual representations, the presented operators and the data that they operate on have to satisfy several requirements. In the following, we list some basic requirements that we have identified.

Selection The selection operator must be capable of retrieving geodata and at the same time their respective metadata from data sources. The geodata has to conform to a semantic data model that clearly and explicitly relates the data to real world phenomena. The metadata has to contain and make formally explicit semantic and pragmatic aspects of the geodata: The metadata has to contain a self-description of the semantic data model (semantic aspect). Moreover, it has to contain information regarding the intended application area of the geodata, the acquisition scheme, quality, scale, former processing, identification of representation and so on (pragmatic aspect). This information is required for successive operators, for instance, in order to determine the suitability of certain geodata for a specific application or to integrate geodata from different sources. Here, we assume that geodata and metadata from all sources are exchanged using a common encoding and that metadata adheres to a common data model.

Furthermore, the operator has to incorporate the concept of multi representational geodata and must be capable of retrieving specified representations from a MRDB.

This capability is required for other operators that want to exploit existing representational geodata. In addition, storing different representations of the same phenomena at the same time is common in practice.

Transformation On the semantic level, the transformation operator must be capable of transforming a geodata set from its original data model into a different data model. This includes that a consistent identification scheme is applied to all features within the geodata set. On the syntactic level, the operator must be capable of transforming a geodata set from its original CRS to a different one. These capabilities are required in order to homogenize geodata from different sources by transforming them to a common semantic and syntactic model.

Join As a precondition, the input geodata that the binary join operator merges must be compatible and *homogeneous* on the pragmatic and to a certain extent on the semantic and syntactic level. For two geodata sets to be homogeneous on the pragmatic level, they are required to be equal, similar or consistent in properties such as intended application area of the geodata, quality and scale. On the semantic and syntactic level, the input geodata sets have to conform to a common data model and CRS. The join operator must be capable of further homogenizing two input geodata sets. On the semantic level, the operator has to match and link features if the geodata sets refer to a shared set of phenomena. This may lead to geometrical adjustments of the features on the syntactic level.

Integrating incompatible geodata will most probably lead to ineffective map-like visual representations. For example, when integrating two geodata sets with different and specialized application areas, e.g., airplane navigation and pedestrian navigation, most probably there will be no application area that the resulting geodata set would be useful for (pragmatic level). As another example, when integrating two geodata sets with non disjoint feature sets that are not labeled with feature entity identifications consistently, feature duplicates might not be detected in later stages and appear on the final representation (e.g., syntactic level).

Generalization Process The generalization process operator performs a complex cartographic generalization process in terms of Grünreich's model (see Section 2.2). It generates cartographic geodata for rendering a map-like representation from input geodata and given visualization requirements according to the purpose of the whole system as specified in the systems specification. The input geodata must be homogeneous on the semantic, syntactic and pragmatic level. The output consists of generalized geodata and a visualization specification for the geodata. This is an application of the general principle of *separation of concerns* that allows for improved reusability and maintainability of the individual components.

For the internal design of the operator, as already noted, several frameworks exist, that decompose the generalization process into a set of generalization operators. Though practice and research did not settle on just one established framework, they usually include at least functionality to create an aggregated, simpler geometric object from a set of objects (e.g., aggregation generalization operator [30]) and to define the set of visual variables [2] for a geometric object (e.g., symbolization generalization operator [30]). In general, in order to be generic and to achieve the expressiveness of a specific framework, the generalization process operator has to implement all the operators that a chosen framework includes.

Furthermore, generalization can be considered as a "holistic" process. This implies that for the duration of the process, it generally has to have access to all the operators of the chosen framework and all the features that are considered for inclusion into a representation. This is because several generalization operators are contextual [6]: they depend on and may influence their spatial context. Therefore, in general, generalizing isolated subsets of features and then simply merging the results of the individual generalizations does not yield effective results [4]. The symbolization operator serves as an example of an operator that cannot be applied isolated from other operators. The results of this operator may have impact on the execution of other operators. For example, the symbolization of a particular feature might have the effect that the required space for the feature is enlarged and that nearby features must be displaced in order to solve overlapping conflicts. Hence, arbitrarily applying symbolizations to geodata sets without applying further generalization operators does not yield effective results.

The symbolization operator has to take into account the type and characteristics of the output medium (syntactic aspect). The characteristics include the total display area for the visual representation on the output medium, the size and shape of the picture elements and the range of available colors.

Rendering The rendering operator produces a 2D raster image from a given geodata set and a visualization specification. The operator applies the visualization specification to the geodata set. Within this process, an implementation may produce an intermediate representation of the targeted image that combines both components and contains less of the original semantics of the geodata. This representation depends on the dimensionality of the geodata to be displayed: for 2D, a vector-based representation may be produced, for 3D, a representation that is based on vectors and scene graphs.

Finally, the operator has to generate the output 2D image by rasterizing the original data or intermediate representation. On the syntactic level, the rasterization has to take into account the type and characteristics of the output medium. For example, depending on the relative size of the picture elements and display technology (CRT or LCD), different antialiasing strategies should be applied.

5 Discussion of the State of the Art of Service-Based Geovisualization

In this Section, we relate the state of the art in service-oriented geovisualization systems as stated in Section 3 to the proposed operators of the visualization pipeline as stated in Section 4. We discuss some weaknesses of current proposals and show that

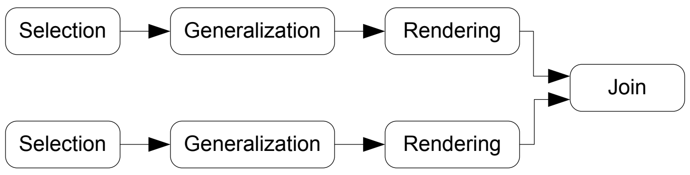


Figure 5: WMS-SLD design in terms of the proposed operators.

they are not appropriate for designing and implementing a system that automatically generates map-like visual representations from heterogeneous geodata.

5.1 Information Management

The proposed standard for geoinformation, ISO 19115 [19], does not contain information regarding the intended application area of the geoinformation, conducted generalization processing of the data or identification of representation. Explicit information on the intended application area is required in order to be able to select only geodata sets that adequate for a given visualization requirement. Information on conducted generalization processing is required in order to determine if a geodata set offers a certain a level of detail. Finally, information on the identification of representation is required to exploit effectively the presence of different representations from on or even multiple data sources. Moreover, concepts of multiple representations are not supported by the WFS.

GML enables domain communities to model specific domain models, called application schemas. Service requester s can obtain geodata along with its data model in shape of an application schema from a WFS. However, there are no proposed concepts or services for mapping between the data models of different communities in the general case. Ontologies have been suggested for solving this issue [45] but no results were reported yet. For a solution based on ontologies, the GML must be extended to support ontologies (e.g., tagging of features with concepts) in a standardized way.

Finally, no proposals exist for the higher-level task of integrating heterogeneous geodata.

5.2 Generalization Processing

In general, WMS, W3DS and WPVS including the proposals for extensions mentioned in Section 3.2 do not contain sufficient generalization functionality and thus do not generate effective map-like visual representations if the input geodata comes from different sources.

To illustrate this, we exemplarily describe a WMS-SLD in terms of the operators introduced in Section 4.2 (see Figure 5). First, the WMS executes a selection operator by choosing a specific WFS from a range of WFS as specified in the `RemoteOWS` element of a SLD. Additionally, features from the chosen WFS are selected according to the `FeatureTypeName` and `Rule` elements of the SE section of the SLD. Subsequently, the WMS executes a generalization process operator as specified in the SE section of the SLD. For this purpose, the WMS classifies, assesses and enlarges the selected features as specified in the `Symbolizer` element of the SE. Additionally, the `Symbolizer` element specifies how the generalized features are rendered by the WMS. Finally, the WMS can join several feature sets that have been selected, generalized and rendered individually by stacking the individual results on top of each other in image space ("painter's algorithm"). Since the WMS requires that requested features from different WFS are all available in a common CRS, and the join of the requested feature sets takes place in image space, the WMS does not require a transformation operator.

This analysis indicates that a WMS-SLD applies generalization operators individually on each input geodata set. Moreover, the WMS-SLD does not implement indispensable generalization operators such as aggregation. Because of these characteristics, in general, a WMS-SLD does not generate effective map-like visual representations, as pointed out in Section 4.2. For W3DS and WPVS the discussion is analog to the discussion of the WMS-SLD. The proposed extensions of the SLD (Section 3.2) do not change the exposed characteristics. Consequently, for producing effective map-like visual representations, the input geodata of a WMS, W3DS or WPVS must have already been generalized as a whole. This requires additional functionality. Within a SOA, this functionality can be provided in the shape of services. OGC publications (e.g., [33]) hint at the meaningfulness of the existence of generalization services but so far, no further effort was committed.

The central issue regarding the provision of automated generalization functionality as standardized services is that it is not feasible yet to automate fully the cartographic generalization process in the general case. Even if this issue is solved, agreeing on a set of common abstractions for the standardization of generalization services might prove challenging. Reported work on standardization efforts [12] is promising but is still in an early stage.

Another minor issue of the WMS-SLD is that it does not sufficiently accounts for the characteristics of the output medium. For instance, the WMS-SLD offers the functionality of a specific *selection* generalization operator. A layer is only displayed by the WMS-SLD if the current scale is within the specified valid scale range of the layer. However, the calculation of the current scale does not take into account the actual pixel size of the output medium, resulting, e.g., in the same visual representation on mobile devices as well as video projectors.

Regarding 3D portrayal of map-like visual representations, another issue is that common approaches for 3D portrayal do not generally produce effective visual representations. For example, Figure 6 presents a 3D portrayal produced with common techniques. The highlighted area reveals "pixel clutter" (artifacts on the syntactic level), i.e., dead areas that contain little useful or even confusing information (impacts on the semantic level). This exemplifies the need for generalization techniques that are specific for map-like 3D visual representations.

5.3 Portrayal Processing

For 3D portrayal of map-like visual representations, extensions for 3D of OGC's SLD and SE standards have been proposed [14, 31] that act as visualization specifications

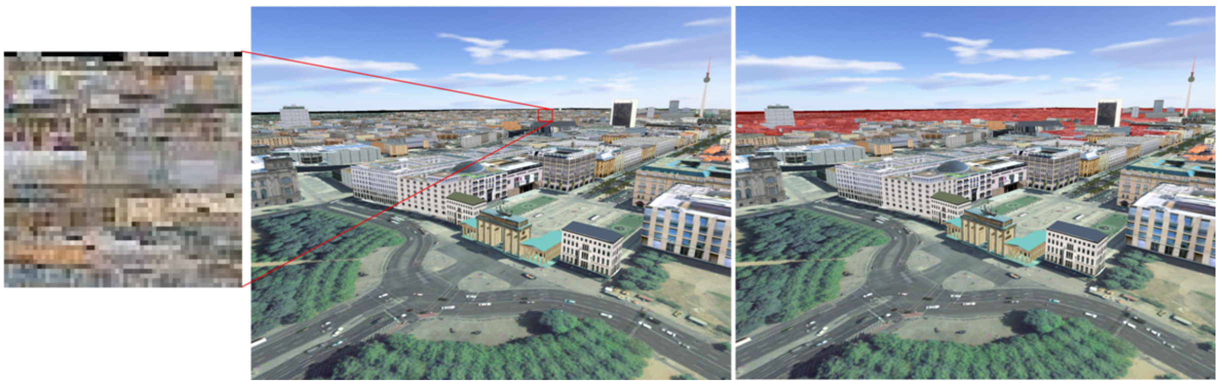


Figure 6: “Pixel clutter” as a symptom for insufficient generalization generated by common approaches for 3D portrayal.

for separate geodata. However, these proposals do not support specific visualization requirements and do not exploit the full potential of 3D portrayal. For example, if photorealistic representations are required, enabling realistic global illumination, water and cloud rendering and controlling its properties supports this task. On the other hand, if abstracted, non-photorealistic representations are required, enabling and controlling specific techniques supports this task.

6 Design of a Configurable Service-Based Geovisualization Pipeline

In this Section, we outline the high-level design of a configurable, standards- and service-based visualization pipeline that can be applied for implementing a system required to generate automatically map-like 2D and 3D visual representations from heterogeneous geodata and visualization requirements.

For this purpose, we map the proposed operators (see Section 4) to existing OGC standards and proposed new services. Figure 7 depicts the relevant services in the context of an exemplary architecture. In the following, we briefly explain the four new services and modifications to existing services and data models:

Geovisualization Process Service This service computes an output map-like visual representation from the input visualization requirements and references to geodata sources. The formalized visualization requirements include information on the application area, the real world area to be visualized and characteristics of the output medium. The service transforms the requirements into an internal visualization specification. It implements an orchestration of processing services and acts as a facade to these services. The orchestration is affected by the visualization specification. A reference to geodata is expressed as a reference to an instance of a WFS, WCS or WMS with an optional filter expression.

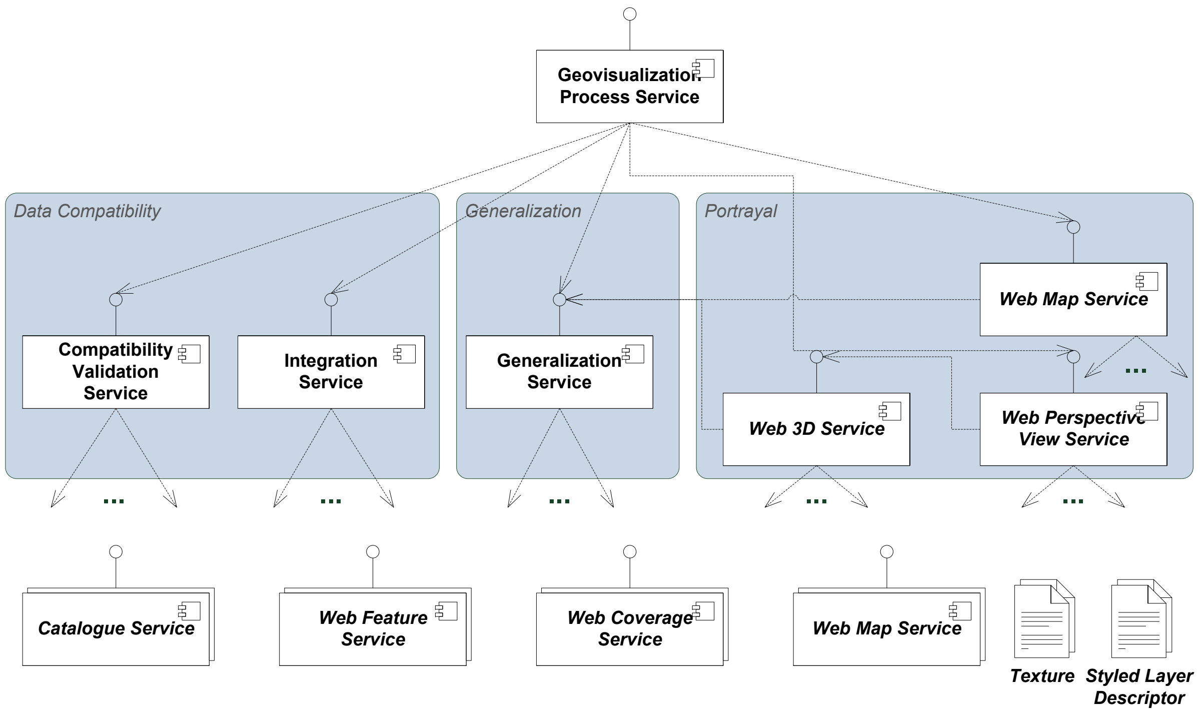


Figure 7: Exemplary architecture of a service-oriented subsystem utilizing existing and newly proposed services that implements a process for the on-the-fly integration, generalization and portrayal of map-like 2D and 3D visual representations.

Compatibility Validation Service This service computes from the input validation criteria and references to geodata sources if the referenced geodata sets are valid and homogeneous. The validation can be performed on the syntactic level (e.g., CRS must be the same or compatible, different geodata sets must spatially overlap, check if spatial overlap of objects is plausible), on the semantic level (e.g., all geodata objects have a type and this type is known and understood) and on the pragmatic level (e.g., intended application areas of the geodata sets are compatible, geodata sets are still marked as valid, quality and resolution is sufficient). This service implements the selection operator.

Integration Service This service computes an output integrated, homogeneous geodata set from the input integration specification and references to geodata sources. If the referenced geodata sets cannot be homogenized, the service throws an exception or returns undefined results. The output geodata set can either be written to a set of specified geodata storage services (e.g., transactional WFS and WCS) or kept in local storages for direct retrieval by other services. The implementation of this service and the Compatibility Validation Service might be based on ontologies. This service implements the selection, transformation and join operators.

Generalization Service This service computes an output generalized geodata set and a SLD visualization specification from the input generalization specification and references to homogeneous geodata sources. The output geodata sets and SLD can either be written to a set of specified storage services or kept in local storages for direct retrieval by other services. This service implements the generalization process operator.

GML, Metadata, SLD The visualization specification SLD must be extended for 3D and advanced styling techniques as discussed in section 5. The metadata specification must be extended by additional as discussed in section 5.

WMS, W3DS, WPVS These services compute output 2D images or intermediate vector representations from input geodata sets and a SLD visualization specification. W3DS and WPVS must be extended to support the extended SLD. These services implement the rendering operator.

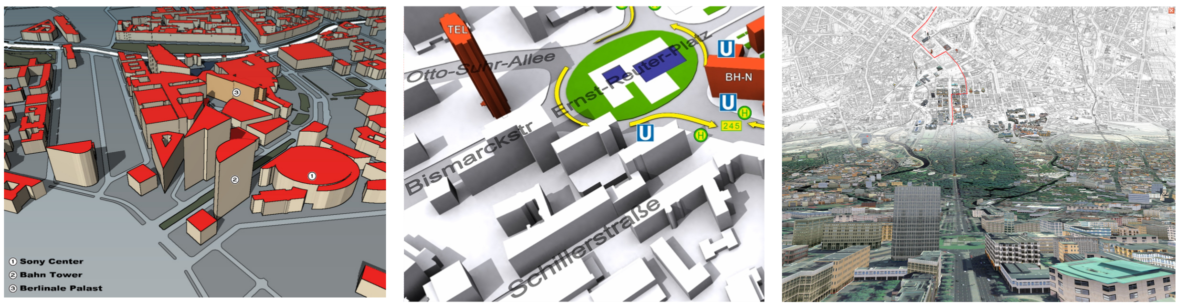
The services mentioned above can be combined to implement various concrete processes meeting different system requirements. For instance, a process is conceivable for the on-the-fly compatibility validation, integration, generalization and portrayal of map-like 2D and 3D visual representations. A variation of this process could use WFS and WCS as storages to cache persistently homogenized and generalized geodata as they are computed for later reuse. This reuse could be for speeding up the process. As another example, a different process could completely decouple the mass generation of persistent generalizations and on-demand rendering of visual representations utilizing the precomputed data. This reuse could be for implementing focus and context techniques based upon the persistent storage of different generalization levels of the same geodata [15].

7 Summary and Future Work

In this paper, we addressed the design and implementation of standards-based systems that facilitate the automatic generation of effective map-like 2D and 3D visual representations from heterogeneous geodata in a service-oriented infrastructure. We identified basic requirements for such systems and identified concepts and methods that support implementing such systems and discussed their strengths and weaknesses.

One important insight is that in general current proposals for portrayal, .i.e, WMS, W3DS and WPVS, do not generate effective map-like visual representations if the input geodata comes from different sources. Generating map-like representations demands that the complex relationships between features are made explicit, analyzed and adjusted in order to meet the visualization requirements.

In Section 6, we outlined the high-level design of a geovisualization pipeline for map-like visual representations. Several areas touched are still actively researched, in particular the automated integration of heterogeneous geodata, automated generalization and 3D generalization. Further areas for future work include formalizing



(a) Non-photorealistic, illustrative visualization of 3D city models [8]. (b) Integration of labels into virtual 3D environments [28]. (c) Multi-perspective views of virtual 3D landscape and city models [27].

Figure 8: Examples for advanced 3D styling techniques.

the visualization requirements and transforming them into a visualization specification and investigating and integrating advanced styling techniques for 3D portrayal into a service-oriented geovisualization infrastructure. Examples for promising advanced styling techniques that can be utilized for generating effective map-like visual representations but that are not yet integrated include the following (see Figure 8): non-photorealistic, illustrative visualization of 3D city models [8], integration of labels into virtual 3D environments [28] and multi-perspective views of virtual 3D landscape and city models [27].

References

[1] Open Geospatial Consortium (OGC) Homepage. URL, `http://www.opengeospatial.org/`. Accessed 15.4.2008.

[2] Jaques Bertin. *Grafische Semiologie*. Walter de Gruyter, Berlin/New York, 1974.

[3] Dirk Burghardt, Moritz Neun, and Robert Weibel. Generalization Services on the Web – A Classification and an Initial Prototype Implementation. In *Cartography and Geographic Information Science*, volume 32, pages 257–268, 2005.

[4] Matthias Butenuth, Guido von Gösseln, Michael Tiedge, Christian Heipke, Udo Lipeck, and Monika Sester. Integration of heterogeneous geospatial data in a federated database. In *ISPRS Journal of Photogrammetry & Remote Sensing*, volume 62, pages 328–346, 2007.

[5] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman, editors. *Readings in information visualization: using vision to think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[6] Alessandro Cecconi. *Integration of Cartographic Generalization and Multi-Scale Databases for Enhanced Web Mapping*. PhD thesis, University Zürich, 2003.

[7] Daniel Chandler. *Semiotics - the basics*. Routledge Taylor&Francis Group; New York, 2002.

[8] Jürgen Döllner, Henrik Buchholz, Marc Nienhaus, and Florian Kirsch. Illustrative Visualization of 3D City Models. In *Proceedings of Visualization and Data Analysis 2005 (Electronic Imaging 2005, SPIE Proceedings)*, pages 42–51, 2005.

[9] David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. In *The Canadian Cartographer*, volume 10, pages 112–122, 1973.

[10] M. Dunkars. *Multiple representation databases for topographic information*. PhD thesis, Royal Institute of Technology, Stockholm, Sweden, 2004.

[11] Alistair Edwardes and Dirk Burghardt. Project WebPark Report –Generalisation Services D4.4.2. Technical report, Department of Geography, University of Zürich, October 2003.

[12] Theodor Foerster, Dirk Burghardt, Moritz Neun, Nicolas Regnauld, Jerry Swan, and Robert Weibel. Towards an Interoperable Web Generalisation Services Framework – Current Work in Progress. 2008.

[13] Dietmar Grünreich. Computer-Assisted Generalisation. In *Papers CERCO Cartography Course*, Frankfurt am Main, 1985. Institut für angewandte Geodäsie.

[14] Jörg Haist, Hugo Miguel Figueiredo Ramos, and Thorsten Reitz. Symbology Encoding for 3D GIS - An Approach to Extending 3D City Model Visualization to GIS Visualization. In *Urban Data Management Symposium*, October 2007.

[15] Mark Hampe, Lars Harrie, and Monika Sester. Multiple Representation Databases to Support Visualization on Mobile Devices. In *Proceedings of the XXth ISPRS Congress*, volume B4, pages 135–140, 2004.

[16] Mark Hampe and Sebastian Intas. Extension ofthe ogc web feature service standard for multiple representation data. In W. Kainz and A. Pucher, editors, *Proceedings of the ISPRS Technical Commission II Symposium*, volume XXXVI of *ISPRS Archives*, pages 49–54, Vienna, Austria, 2006.

[17] Dieter Hildebrandt, Oliver Holschke, Philipp Offermann, and Ulrike Steffens. Entwurf serviceorientierter Architekturen. In Wilhelm Hasselbring and Ralf Reussner, editors, *Handbuch der Software-Architektur*, page 575. dpunkt Verlag, September 2008.

[18] International Cartographic Association ICA. *Multilingual Dictionary of Technical Terms in Cartography*. Franz Steiner Verlag, Wiesbaden, 1973.

[19] ISO/TC 211. *ISO 19115:2003 Geographic Information – Metadata*, 1st edition, May 2003.

[20] ISO/TC 211. *ISO 19119:2005 Geographic Information – Services*, 2005.

[21] ISO/TC 211. *ISO 19139:2007 Geographic information – Metadata – XML schema implementation*, 1st edition, May 2007.

[22] Anthony Jameson. Modelling both the Context and the User. *Personal and Ubiquitous Computing*, 5(1):29–33, February 2001.

[23] C.B. Jones and J.M. Ware. Map Generalization in the Web Age. In *International Journal of Geographical Information Science*, volume 19, pages 859–870, 2005.

[24] Andreas Koch. *Semantische Integration von zweidimensionalen GIS-Daten und Digitalen Geländemodellen*. PhD thesis, Fakultät für Bauingenieurwesen und Geodäsie der Universität Hannover, 2006.

[25] Dirk Krafzig, Karl Banke, and Dirk Slama. *Enterprise SOA: Service-Oriented Architecture Best Practices*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[26] P.A. Longley, M.F. Goodchild, D.J. Maguire, and D.W. Rhind. *Geographical Information Systems and Science*. Wiley, Chichester, 2. edition, 2005.

[27] Haik Lorenz, Matthias Trapp, Markus Jobst, and Jürgen Döllner. Interactive Multi-Perspective Views of Virtual 3D Landscape and City Models. In *Proceedings of the 11th AGILE International Conference on GI Science*. SPRINGER, May 2008.

[28] Stefan Maass and Jürgen Döllner. Seamless Integration of Labels into Interactive Virtual 3D Environments Using Parameterized Hulls. In *4th International Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*, June 2008. to appear.

[29] Alan M. MacEachren and Menno-Jan Kraak. Research Challenges in Geovisualization. *Cartography and Geographic Information Science*, 28(1):3–12, 2001.

[30] Robert B. McMaster and Stuart K. Shea. *Generalization in Digital Cartography*. Association of American Geographers, Washington, D.C., USA, 1992.

[31] Steffen Neubauer and Alexander Zipf. Suggestions for Extending the OGC Styled Layer Descriptor (SLD) Specification into the third Dimension - An Analysis of possible Visualization Rules for 3D City Models. In *Urban Data Management Symposium*, Stuttgart, Germany, October 2007.

[32] Moritz Neun, Dirk Burghardt, and Robert Weibel. Web service approaches for providing enriched data structures to generalisation operators. In *International Journal of Geographical Information Science*, volume 22, pages 133–165, 2008.

[33] Open Geospatial Consortium Inc. *OGC Reference Model, Version 0.1.3*, September 2003.

[34] Open Geospatial Consortium Inc. *Web 3D Service, Version 0.3.0*, February 2005.

[35] Open Geospatial Consortium Inc. *Web Feature Service Implementation Sepecification, Version 1.1.0*, May 2005.

[36] Open Geospatial Consortium Inc. *OpenGIS Web Map Server Implementation Specification, Version 1.3.0*, March 2006.

[37] Open Geospatial Consortium Inc. *Symbology Encoding Implementation Specification, Version 1.1.0*, July 2006.

[38] Open Geospatial Consortium Inc. *OpenGIS Catalogue Services Specification, Version 2.0.2*, February 2007.

[39] Open Geospatial Consortium Inc. *OpenGIS Geography Markup Language (GML) Encoding Standard, Version 3.2.1*, August 2007.

[40] Open Geospatial Consortium Inc. *OpenGIS Web Processing Service, Version 1.0.0*, June 2007.

[41] Open Geospatial Consortium Inc. *Styled Layer Descriptor Profile of the Web Map Service Implementation Specification, Version 1.1.0*, June 2007.

[42] Open Geospatial Consortium Inc. *Web Coverage Service (WCS) Implementation Specification, Version 1.1.2*, March 2008.

[43] Open GIS Consortium Inc. *OGC Web Terrain Server, Version 0.3.2*, August 2001.

[44] Ingo Petzold, Dirk Burghardt, and Matthias Bobzien. Workflow Management and Generalisation Services. In *Workshop of the ICA Commission on Map Generalisation and Multiple Representation*, 2006.

[45] Nicolas Regnauld. Evolving from automating exsting map production systems to produce maps on demand automatically. In *Proceedings of the 10th ICA Workshop on Generalisation and Multiple Representation*, August 2007.

[46] Heidrun Schumann and Wolfgang Müller. *Visualisierung: Grundlagen und allgemeine Methoden*. Springer-Verlag, Berlin, 2000.

[47] Robert Weibel. *Generalization of Spatial Data: Principles and Selected Algorithms*, volume 1340 of *Lecture notes in computer science*, chapter 5, pages 99–152. Springer, 1997.

[48] Robert Weibel and G. Dutton. Generalising spatial data and dealing with multiple representations. In *Geographic Information Systems – Principles and Technical Issues*, volume 1, pages 125–155, 1999.

ISBN 978-3-940793-81-2
ISSN 1613-5652