Proceedings of the 4th 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. 31

des Hasso-Plattner-Instituts für
Softwaresystemtechnik
an der 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 | 31

Proceedings of the 4th 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 2010
http://info.ub.uni-potsdam.de/verlag.htm

Am Neuen Palais 10, 14469 Potsdam
Tel.: +49 (0)331 977 4623 / Fax: 3474
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**.**

Das Manuskript ist urheberrechtlich geschützt.

Online veröffentlicht auf dem Publikationsserver der Universität Potsdam
URL http://pub.ub.uni-potsdam.de/volltexte/2010/4083/
URN urn:nbn:de:kobv:517-opus-40838
http://nbn-resolving.de/urn:nbn:de:kobv:517-opus-40838

Zugleich gedruckt erschienen im Universitätsverlag Potsdam:
ISBN 978-3-86956-036-6

Contents

Context-aware Reputation in SOA and Future Internet

Rehab Alnemr
rehab.alnemr@hpi.uni-potsdam.de

In this report I continue to elaborate different constructs of my work in order to reach the final goal of designing and modeling context-aware reputation systems. Throughout the report, I illustrate different goals and benefits of developing such systems.

1 Introduction

Reputation is used in our social communities as a tool of regulating society by the mean of circulating evaluations. It has been also an important tool in computer science, especially after the emergence of e-markets. In these markets, consumers get product information from the auction sites as well as weblogs. Thus mobilizing and combining the experience of many consumers. Currently, it became even more crucial with the increased deployment of Service Oriented Architectures (SOA), Internet of Service (IoS), and recently cloud computing. In the near future, reputation will be a factor in negotiating about computer resources in cloud computing. It plays even a bigger role in mitigating risks, when the price involved is higher, and in conveying cooperation and increasing trust among different entities in these systems.

Transferring the internet from a network of information to a network of knowledge and services requires more complex and cognitive view on reputation. Existing reputation based systems and models are not cognitive enough to reflect the real nature of reputation notions. Understanding reputation concepts and constructs is essential specially if we are dealing with humans and artificial intelligence agents. In my previous work [4] [5], I have introduced a context-aware reputation framework that enables reputation portability between different virtual communities. The basic constructs of this framework are: Reputation Object (holds the contexts to be evaluated along with the corresponding values and their calculation models), Trust Reputation Centers (reputation providers), and RRTM (categorization of reputation models to be used as reference). Reputation in this framework addresses different contexts to be evaluated, and is generic enough to be transfered among communities(platforms), which fits in the SOA and cloud vision.

Later, I have introduced in [3] the use of Attention Allocation Points (AAPs) techniques -used in economics- to address the problem of having an abundance of information and solve it by pinpointing the most important information to the current transaction. The output pieces of information are used later to build reputation of entities involved in the process. Thus, the process of ”rating” is transformed into a process

of ”evaluation”. The building blocks to this evaluation process construct at the end a new concept of Reputation-as-a-Service (RaaS). The relation between reputation and quality is explored (in the same reference), since understanding reputation requires a correct understanding of the notions of quality to enable the evaluation process. Understanding -and later measuring- Service Reputation requires distinguishing it from other concepts. This distinction helps in differentiating what is being measured, which is a critical factor in conguring Service Level Agreements (SLAs). This separation of concepts helps also in evaluating quality processes along each phase of the service life cycle, and among the different roles of service parties, e.g., service provider.

Following this line of work, I am working with the POSR [1] team in studying the quality notions and processes to have the right information to build the RaaS and assign Reputation Objects to services. In parallel, SLA is being investigated, since they include performance measurement information needed in the evaluation process. For allocating the right kind of information to be used in reputation evaluation, I am working with a team in Frei University in Berlin, specialized in Complex Event Processing, to allocate attention points. This is done by using their RuleResponder [8] project.

In the following section I start by introducing new definitions for quality notions and service reputation along with the proposed model-driven approach to service reputation [2]. Section 3 illustrates our work in analyzing SLAs and introduces a new hybrid approach for distributing trust in SLA choreography [6]. The work with Frei University team [9] is introduced in Section 4. This is followed by the conclusion and next steps.

2 Model-driven approach to Service Reputation

This section elaborates on our study of the quality notions that was done to get the right kind of information for the reputation evaluation process. The investigation revealed several misconceptions and a diversity of terminologies usage. Moreover, it led us to the fact that current approaches use limited sources to evaluate quality. Therefore, we try to provide concrete denitions of Quality of Service (QoS), Quality of Results (QoR), and Service Quality and to differentiate between them. Quality attributes associated with these concepts vary from subjective to objective measures (depending on the context in which the service used). The discussion in this section includes: the proposed new meta-model, the combined sources for service’s quality assessment, and the new definition of Service Reputation Object.

2.1 Quality notions in SOA

The requirements specified by the service consumer in the process of service selection include both functional and non-functional features of the service. A critical requirement for service selection is quality, but assessing the quality of a certain service is not straightforward. Gathering sufficient information for such assessment incorporates determining what to ask for and from which source to get it. Using service description provides information about functional features of the services, but this is not enough to select a service.

We propose an extensible meta-model that specifies the definitions, relations, and differences between quality notions. In our approach, quality attributes of a specific service are derived from the proposed meta-model by means of *model-to-model* transformation. A service call corresponds to a *model-to-text* transformation, where the values of service quality attributes are specified. This is accomplished by using a combined approach to collect the required information from different sources: users ratings, service past interactions, user preferences and invocation analysis. Using these sources, a *Service Reputation Object* is derived from the model for each service. The object holds values of different quality attributes and represents a context-oriented reputation for a service. In the literature, reputation is considered as a quality attribute. In our work, we view Service Reputation as an upper concept that encapsulates quality notions not the other way around.

2.1.1 New Definitions

The terminologies associated with quality and reputation concepts lack unied denitions in the literature, e.g.,QoS is sometimes confused with QoR, and there is no concrete denition for the notion of Service Quality (SQ), yet. Our definitions are:

Quality of Results describes the extend of how the outputs of a service met or exceeded the service consumer's expectations (degree of satisfaction), in terms of results' soundness, completeness, and correctness.

Service Quality describes the extend of how the service has been invoked in a convenient manner (from the service consumer's point of view).

Quality of Service describes the general features that are used to evaluate service efficiency in its context.

Reputation describes the notion of "profiling" an entity to evaluate the expectation of its performance in several contexts.

2.1.2 Quality Attributes

The above denitions classify the attributes into three categories: general attributes (applicable to any service), domain based attributes (applicable to a specic domain only), and service specic attributes (applicable to a single service only). Some approaches categorize quality attributes as strict separation of *general attributes*, such as price and delivery time, and *domain based attributes* (also called business related attributes), such as how many days left for the flight in a travel service website. Others categorize them as user-centeric (where quality attributes stem from user needs and preferences only) and non-user-centric attributes. In our approach we describe the nature of the attributes as dynamic (general or domain-based according to the context), subjective (affected by several evaluation sources like user preferences, history, invocation analysis), and service specific (applicable to a single service only)

In the quantification phase of service's quality profiling, each attribute should be defined in terms of: domain, value, measuring method, weight relative to domain,

potential gain of rating , and temporal characteristic (decaying or increasing). Several attributes are used in the study of quality of service and information quality, such as: completeness, reliability, accuracy, accessibility, consistency, timeliness, availability, relevancy, efficiency, usability, security, and trust.

2.2 A meta model to Service Reputation

A system model abstracts system's specifications from any specific implementation platform; which is referred to as Platform-Independent Model (PIM). Adjusting the abstract model (PIM) to the chosen implementation platform to get a Platform-Specific Model (PSM) is achieved through a set of model-to-model transformations. Finally, software artifacts, e.g., code, reports, are generated from the PSM by means of model-to-text transformations. Our model describes different quality attributes independently of any environment or preference settings. This corresponds to the PIM in Software Engineering. To derive a more concrete service quality model which describes a specific service in certain environments and settings and conforming to specific user preferences, we apply model-to-model transformation techniques. The result is a model that corresponds to a PSM in Software Engineering. In the PSM of service quality, service's specific domain attributes are considered and dynamically categorized according to the PIM. Eventually, we apply model-to-text transformation techniques to the PIM to get the software artifacts for the considered service. In our approach, the resulting software artifacts are concrete values for the service quality attributes of the considered service. Our proposed meta-model to the quality notions associated with a service is depicted

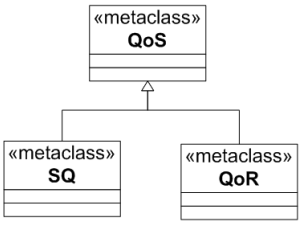


Figure 1: A Meta-model for quality notions in SOC

in Figure 1. In this model, we present a classification of quality attributes in three different abstract categories, which are QoS, SQ, and QoR. Moreover, we emphasize the generalization-specification relationship between QoS from one side, and SQ and QoR from the other side. This classification is based on the quality attributes used to assess a service call. General quality attributes fall under the QoS category, domain based quality attributes fall under the QoR category, and service-specific quality attributes under the SQ category. Extensibility is one of the main features of our proposed model. Quality attributes are not limited to a predefined set of attributes, but rather a service consumer can adapt the model to his own definition of quality by adding more quality attributes that are, in most cases, domain-specific and require domain experts.

The model has a dynamic feature, which is reflected in the dynamic categorization of quality attributes (generic, domain based, and service specific). In one application,

cost can be considered as a QoR attribute, but when it comes to *cost vs. purpose*, it falls in the SQ category. Figure 2 depicts our approach through an example.

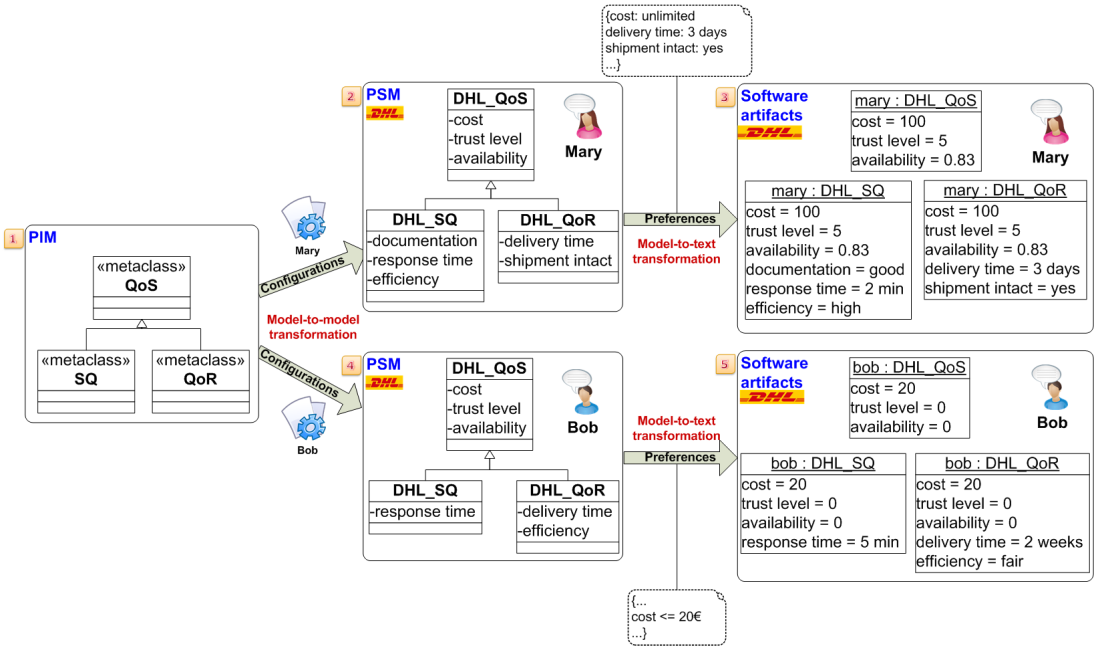


Figure 2: Model-to-model and model-to-text transformations

2.3 Discussion: From Quality Attributes to Service Reputation

In this section, we discuss specific aspects of our approach; namely: evaluation sources, invocation analysis and Service Reputation Object. One of the uses of our work is to enhance service selection by investigating its reputation. This is done by providing enough information about several quality aspects of a service (*profiling*) to be able to select the most appropriate service. Previous work in the field use *only* one or two of the following evaluation sources:

1. feedback from users: ratings per service are aggregated as a reputation value.
2. service history: some or all of the service past interactions.
3. service provider's reputation: service reputation is a direct *inheritance* from the service provider's reputation.
4. advertisement: the service provider advertise his services himself.

In our approach we combine several sources to get a full service profile, so that we construct service reputation objects. Instead of using binary reports from service consumers, we use a feedback report detailing the rating of each quality attributes involved in the process in the light of service's configurations and user's preferences. Invocation analysis is introduced as an automatic source of information to enrich the process of deriving service quality.

2.3.1 Service Reputation Object

Our approach led us to conclude a new reputation concept for a service. Giving a service a reputation value is not a novel idea, some researchers already addressed this idea, but most of them either collects binary reports from users and aggregate the result in a single rating value or they use only the service provider's reputation, which is again a single value. Being a single value, reputation is usually thought of as a quality attribute. In our work, we do not only consider reputation as an object, but we also consider *Service Reputation* as an upper concept that encapsulates quality notions not the other way around. Referring to reputation definition in Section 2.1.1, profiling an entity describes recording its behavior and analyzing its characteristics in order to predict or assess its ability in a certain sphere or to identify a certain pattern. These characteristics are various quality attributes that are tested and calculated per service call creating at the end a list of values that are used to construct a service profile. Hence, we redefine service reputation as:

- *Service Reputation*: describes the notion of profiling a single service by collecting its performance information using different quality attributes to construct the service profile.

- *Service Reputation Object*: is an object that holds service reputation information, and consists of a set of collective values calculated by functions that depend on the nature of the corresponding quality attribute, aggregated from several service calls.

A Service Reputation Object (SRO), which has detailed values of quality attributes, can be visualized by addressing Figure 3. A single quality attribute semantic description is formalized as: $QA_i \in \{QA_1,\ldots,QA_n\}$, an algorithm used to calculate one or more quality attribute is $Alg_k \in \{Alg_1,\ldots,Alg_y\}$, a single service call is $SC_j \in \{SC_1,\ldots,SC_m\}$. The functions used to calculate the quality value are not necessarily aggregation or averaging. Moreover, it does not have to be the same function used to compute the elements of the array, can be different for each attribute; according to its nature, i.e., average, maximum, multiplication, etc. , where $F_w \in \{F_1,\ldots,F_x\}$. The final calculated value for a single attribute from all the service calls is $CalcValue_i \in \{CalcValue_1,\ldots,CalcValue_n\}$.

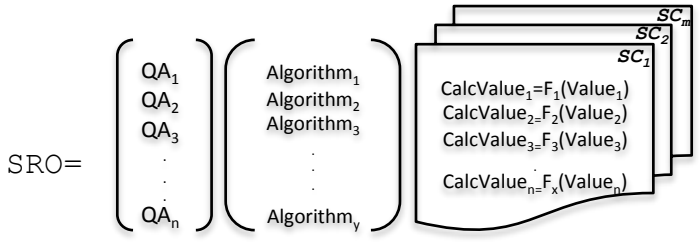


Figure 3: Constructing Service Reputation Object by analyzing quality notions

3 Distributed Trust Management for Validating SLA Choreographies

For business workflow automation in a service-enriched environment services scattered across heterogeneous Virtual Organisations (VOs) can be aggregated in a producer -consumer manner, building hierarchical structures of added value. To preserve the supply chain, the Service Level Agreements (SLAs) corresponding to the underlying choreography of services should also be incrementally aggregated. This cross-VO hierarchical SLA aggregation requires validation, for which a distributed trust system becomes a prerequisite.

This section elaborates on the proposed hybrid approach of using reputation systems and PKIs to distribute trust in SOA, and to identify violation-prone services at service selection stage. It also actively contributes in breach management at the time of penalty enforcement. This work is part of the phase which aims to use SLAs to get quality information that leads later to constructing the reputation service. The hybrid distributed trust system enable previously introduced rule-based runtime validation framework for hierarchical SLA aggregations. The discussion includes: the justification and significance of a hybrid trust model for the validation of hierarchical SLA aggregations, and the conceptual elements of our hybrid PKI and reputation based trust model.

3.1 A Framework for Validation of Hierarchical SLA Aggregations

A Service Level Agreement (SLA) is a formally negotiated contract between a service provider and a service consumer to ensure the expected level of a service. In a service enriched environment such as Grid, cooperating workflows may result into a service choreography spun across several Virtual Organisations and involving many business partners. Service Level Agreements are made between services at various points of the service choreography. Service choreography is usually distributed across several Virtual Organizations and under various administrative domains. The complete aggregation information of the SLAs below a certain level in the chain is known only by the corresponding service provider and only a filtered part is exposed up towards the immediate consumer. This is the reason why during the validation process, the composed SLAs are required to be decomposed in an incremental manner down towards the supply chain of services and get validated in their corresponding service providers's domain. A validation framework for the composed SLAs, therefore, faces many design constraints and challenges: a trade-off between privacy and trust, distributed query processing, and automation to name the most essential ones.

In our proposed model, the privacy concerns of the partners are ensured by the SLA View model [7], whereas the requirements of trust and security can be addressed through a reputation-based trust system built upon a distributed PKI (Public Key Infrastructure) based security system. Additionally, we use Rule Responder [8] to weave the outer shell of the validation system by providing the required infrastructure for the automation of role description of partners as well as steering and redirection of the distributed validation queries. Every service provider is limited only to its own view. The

whole SLA Choreography is seen as an integration of several SLA Views.(details found in [7]). SLA-views can be implemented by using Rule Responder architecture.

Rule Responder adopts the approach of multi agent systems. There are three kinds of agents, namely: Organisational Agents (*OA*), Personal Agents (*PA*), and External Agents (*EA*). An *EA* is an entity that invokes the system from outside. A virtual organization is represented by an *OA*, which is the main point of entry for communication with the ”outer” world. A *PA* corresponds to the SLA View of a service provider. Each individual agent is described by its syntactic resources of personal information, the semantic descriptions that annotate the information resources with metadata and describe the meaning with precise business vocabularies (ontologies) and a pragmatic behavioral decision layer to react autonomously. The flow of information is from external to organizational to personal Agent.

During service choreography, services may form temporary composition with other services, scattered across different VOs. The question of whose parent VO acts as the root CA in this case is solved by including third party trust manager like the case for dynamic ad hoc networks. In case of SLA violation, in addition to enforcing penalty, the affected party is likely to keep a note of the violating service in order to avoid it in future. Moreover, a fair business environment demands even more and the future consumers of the failing service also have a right to know about its past performance. Reputation-based trust systems are widely used to maintain the reputation of different business players and to ensure this kind of knowledge. Our hybrid trust model based on PKI and reputation- based trust systems to harvest advantages from both techniques. The main points of the model are first, the PKI based trust model has a third party trust manager that will act as a root CA and authenticate member VOs. These VOs are themselves CAs as they can further authenticate their containing services. Second, Selection of services at the the pre-SLA stage is done by using reputation to prevent SLA violation. Services reputation are updated after each SLA validation. third, While the trust model promises trust and security, the SLA views protect privacy.

3.2 Single Sign-On and Reputation Centers

In the proposed model, a third party acts as a root CA and authenticates member VOs. These VOs are themselves CAs as they can further authenticate their containing services. Each member is given a certificate. With Single Sign-On, the user does not have to bother to sign in again and again in order to traverse along the chain of trusted partners (VOs and services). This can be achieved by the Cross-CA Hierarchical Trust Model where the root CA provides certificates to its subordinate CAs and these subordinates can further issue certificates to other CAs (subordinates), services or users.

In our previous work [4], the Trust Reputation Center (TRC) acts as a trusted third party. As depicted in figure 4, this reputation-based trust model has direct correspondence with Rule Responder’s agents and their mutual communication. The PAs consult OAs and OAs in return consult the TRC which is equivalent to the third party CA in PKI based system.

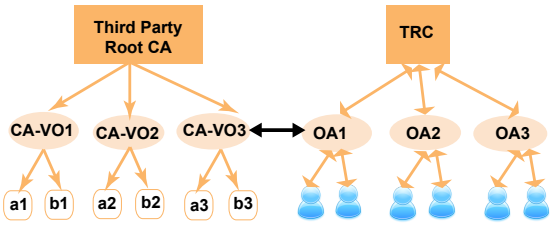


Figure 4: Correspondence bet.: PKI, reputation based systems, and Rule Responder architecture

3.3 Proposed Trust Management Model

The processes involved in our model are:

- *Validation of complete SLA aggregation*: to do this the validation query is required to traverse through all the SLA views lying across heterogeneous administrative domains and get validated locally at each SLA view. The multi-agent architecture of Rule Responder provides communication middle-ware to the distributed stake-holders namely the client, the VOs, and various service providers.

- *Use of reputation in the selection phase*: reputation transfer is required at two stages: at service selection stage and at penalty enforcement stage. In the process of service selection, the reputation transfer helps to select the least violation-prone services, taking into account proactive measures to avoid SLA violations. Out of all the available services, the client first filters the best services complying its ”happiness criteria”. Then the client compares the credentials from reputation objects of the services. The reputation object is traced . Then the client can select the best service in accordance to its already devised criteria. We assume that out of redundant services which fulfil client’s requirements, the service with the highest reputation is selected.

- *Use of PKI and reputation in breach management*: this hybrid Trust is used in the breach management after an occurrence of SLA violation. [7]

Refer to [6] to find more details on the use case scenario and about this paper.

4 Towards Semantic Event-Driven Systems

This section elaborates on the work of allocating the right kind of information to be used in reputation evaluation, using Complex Event Processing (CEP) techniques and semantic web technologies. The discussion in this section includes: description of a new conceptual proposal for a semantic event-driven Internet, and contribution with an architectural design artefact consisting of two essential levels of enhancements. A common denominator is the use of Semantic Web technologies on each level, i.e.,

for processing data (tied to events) and for semantically formalizing attention allocation points which are applied on the detected and predicted situation instances via a semantic match making process (ontology and rule-based constraint inference logic). Here, we introduce our work-in-progress on *semantic, proactive, quality assured, event-driven information processing approach* towards the vision of a semantic event-driven systems, which comprises tow levels of proposed enhancements:

1. Semantic Complex Event Processing (SCEP), which is the combination of CEP and and Semantic Web technologies, to achieve better machine understandability of (event) data and highly automated real-time processing of large and heterogenous event sources

2. Semantic quality, trust and reputation assessment of the produced information to support precise information dissemination and focused user attention allocation

we are currently working on grounded integration of the two layers and on the implementation and evaluation of the approach with industrial relevant use cases.

4.1 Semantic Complex Event Processing

Real-world occurrences can be defined as events that are happening over space and time. An event instance is an concrete semantic object containing data describing the event. An event pattern is a template that matches certain sets of events. One of the critical success factors of event-driven systems is the capability of detecting complex events from simple event notifications. The promises of the combination of event processing and semantic technologies, such as rules and ontologies, which leads to semantic event processing (SCEP), are that the event processing rule engines can *understand* what is happening in terms of events and (process) states and that they will *know* what reactions and processes they can invoke and what events it can signal.

Semantic (meta) models of events can improve the quality of event processing by using event metadata in combination with ontologies and rules (knowledge bases). Event knowledge bases can represent complex event data models which link to existing semantic domain knowledge such as domain vocabularies / ontologies and existing domain data. Semantic inference is used to infer relations between events such as e.g. transitivity or equality between event types and their properties. Temporal and spatial reasoning on events can be done based on their data properties, e.g. a time ontology describing temporal quantities.

In our semantic event model an event instance is a set of RDF triples and an event pattern is a graph pattern. More complex events are formed by combining smaller patterns in various ways. A complex conjunctive event filter is a complex graph pattern of the RDF graph. A map of the attribute/value pairs to a set of RDF triples (RDF graph) can be used as event instance. The interesting part of this event data model is the linking of the existing knowledge (non-event concepts) to the event instances, for example the name of the stock which the event is about is not identified by a simply string name, but by an URI which links to the semantic knowledge about the stock. This

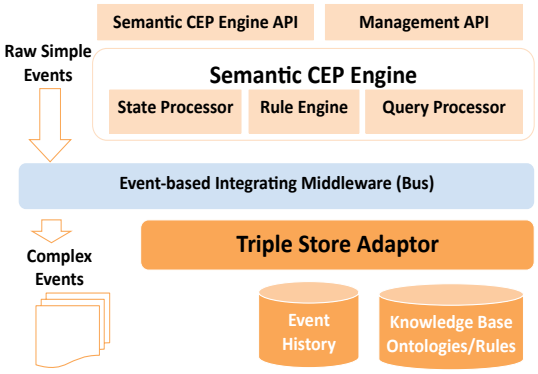


Figure 5: Architecture of Semantic Complex Event Processor

knowledge can be used later for the processing of events, e.g. in the condition part of a Event-Condition-Action (ECA) reaction rule.

Figure 5 shows our architectural vision for semantic complex event processing. It combines a knowledge base which includes ontologies and rules, and an incoming event stream which comes from event producers, e.g. sensors or event adapters. The system has to combine static and real-time knowledge references and generate new knowledge. Having such a semantic event pattern/filter makes it possible to detect complex event which is derived from the already happened events in combination with the knowledge of the processing system. In the first step the raw-level events are classified and mapped to a secondary level of events. The sequence and syntactic checks are also processed in this step. Next step, the SCEP engine processes the events more expressively based on their semantic relationships to other events and other none-event concepts existing in the knowledge-base. The knowledge-based can be seen as TBox (assertions on concepts) and the event object stream as ABox (assertions on individuals)

4.2 Proposed Approach through Stock Market Use Case

Semantic event-driven systems can have different use cases in different fileds such as e-health, business activity monitoring, fraud detection, logistics and cargo, etc . In this example, a broker-agent in the financial stock market (who studies price movements, and patterns), has to take the decision whether to buy Company_XY stocks for his customers. Several factors contribute to his decision: recent trends seen in the market movements, events detected in the company itself, and the reputation of the sources from which the broker gets his intelligence. We assume that the broker has information from two sources: *source 1* has high reputation in the real-estate market movements while *source 2* has high reputation in commodity stocks. Attention to these two particular markets can form the final decision. Next step is to formalize this info into a correspondent complex event. During this process, a new intel appears: the elimination of several agricultural lands to be converted into construction areas due to change of political administration in the city. This new intel is entered as a new event in the process, with the source reputation and his area of expertise. Then, the trend mining

engine process information about trends in the political section to get the possibility of another change in the administration.

5 Conclusion and Future work

Correct understanding and later modeling of reputation concept helps to: enhance the e-market dynamics to reflect real-life cognitive models of interactions, connect reputation models to other elements in the architecture (eg.agent memory), base the decision making process on correct and context-related parameters, and construct SLAs in e-contracts that are: context related, customized, and realistic. I will continue working on the three previously mentioned parts of my vision in parallel by cooperating with different teams.

References

[1] M AbuJarour, M Craculeac, F Menge, T Vogel, and J Schwarz. Posr: A Comprehensive System for Aggregating and Using Web Services. *Proceedings of the 2009 IEEE Congress on Services*, 2009.

[2] Rehab Alnemr, Mohammed AbuJarour, and Christoph Meinel. Model-driven Approach to Service Reputation. 2009.

[3] Rehab Alnemr, Justus Bross, and Christoph Meinel. Constructing a context-aware service-oriented reputation model using attention allocation points. 2009.

[4] Rehab Alnemr and Christoph Meinel. Getting More from Reputation Systems: A ContextAware Reputation Framework Based on Trust Centers and Agent Lists. *Computing in the Global Information Technology, ICCGI'08*, pages 137–142, 2008.

[5] Rehab Alnemr, Matthias Quasthoff, and Christoph Meinel. Taking trust management to the next level. 2009.

[6] Irfan Ul Haq, Rehab Alnemr, Adrian Paschke, E. Schikuta, H. Boley, and C. Meinel. Distributed Trust Management for Validating SLA Choreographies. *SLAs in Grids workshop, CoreGRID Springer series (to appear)*, 2009.

[7] Irfan Ul Haq, Altaf Huqqani, and Erich Schikuta. Aggregating hierarchical service level agreements in business value networks. *Business Process Management Conference (BPM2009)*, 2009.

[8] A. Paschke, H. Boley, A. Kozlenkov, and B. Craig. Rule responder: Ruleml-based agents for distributed collaboration on the pragmatic web. *Proceedings of the 2nd international conference on Pragmatic web Tilburg, The Netherlands*, 2007.

[9] Kia Teymourian, Rehab Alnemr, Olga Streibel, Adrian Paschke, and Christoph Meinel. Towards semantic event-driven systems. 2009.

Abstraction of Process Specifications

Artem Polyvyanyy
Artem.Polyvyanyy@hpi.uni-potsdam.de

Software engineers constantly deal with problems of designing, analyzing, and improving process specifications, e.g., source code, service compositions, or process models. Process specifications are abstractions of behavior observed or intended to be implemented in reality which result from creative engineering practice. Usually, process specifications are formalized as directed graphs, where edges capture temporal relations between decisions, synchronization points, and work activities. Every process specification is a compromise between two points: On the one hand engineers strive to operate with less modeling constructs which conceal irrelevant details, while on the other hand the details are required to achieve the desired level of customization for envisioned process scenarios. In our research, we approach the problem of varying abstraction levels of process specifications. Formally, developed abstraction mechanisms exploit the structure of a process specification and allow the generalization of low-level details into concepts of a higher abstraction level. The reverse procedure can be addressed as process specialization.

Keywords: Process abstraction, process structure, process modeling

1 Introduction

Process specifications represent exponential amounts of process execution scenarios with linear numbers of modeling constructs, e.g., service compositions and business process models. Nevertheless, real world process specifications cannot be grasped quickly by software engineers due to their size and sophisticated structures making a demand for techniques to deal with the complexity. The research topic of process abstraction emerged from a joint research project with a health insurance company. Operational processes of the company are captured in about $4\,000$ EPCs. The company faced the problem of information overload in the process specifications when employing the models in use cases other than process execution, e.g., process analysis by management. To reduce the modeling effort, the company requested to develop automated mechanisms to derive abstract, i.e., simplified, process specifications from the existing ones. The research results derived during the project are summarized in [6].

Abstraction is the result of the generalization or elimination of properties in an entity or a phenomenon in order to reduce it to a set of essential characteristics. Information loss is the fundamental property of abstraction and is its intended outcome. When working with process specifications, engineers operate with abstractions of real world concepts. In our research, we develop mechanisms to perform abstractions of formal process specifications. The challenge lies in identifying what the units of process

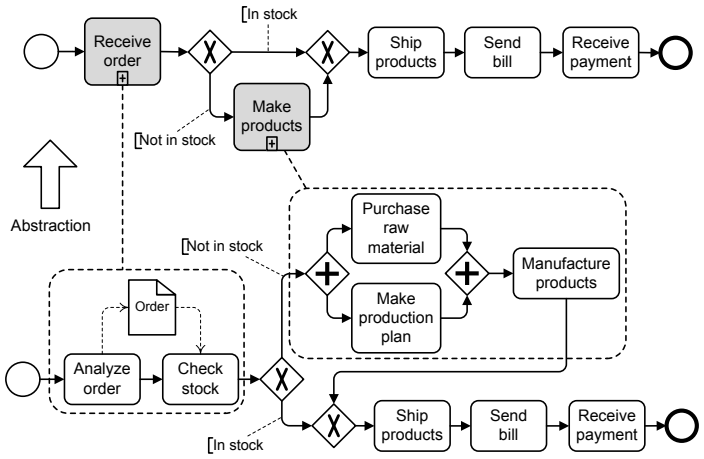


Figure 1: Process abstraction (BPMN notation)

logic suitable for abstraction are and then afterwards performing the abstraction. Once abstraction artifacts are identified, they can be eliminated or replaced by concepts of higher abstraction levels which conceal, but also represent, abstracted detailed process behavior. Finally, individual abstractions must be controlled in order to achieve an abstraction goal—a process specification that suits the needs of a use case.

Figure 1 shows an example of two process specifications (given as BPMN process models) which are in the abstraction relation. The model at the top of the figure is the abstract version of the model at the bottom. Abstract tasks are highlighted with a grey background; the corresponding concealed fragments are enclosed within the regions with a dashed borderline. The fragments have structures that result in an abstract process which captures the core process behavior of the detailed one. The abstract process has dedicated routing and work activity modeling constructs and conceals detailed behavior descriptions, i.e., each abstracted fragment is composed of several work activities. The research challenge lies in proposing mechanisms which allow examining every process fragment prior to performing abstraction and suggesting mechanisms which coordinate individual abstractions, i.e., assign higher priority to abstracting certain fragments rather than the others.

The rest of the paper is organized as follows: The next section presents the connectivity-based framework designed to approach the discovery of process fragments suitable for abstraction. Sect. 3 discusses issues relevant to the control of process abstraction. Sect. 4 discusses a technique which aids validation of process correctness and is found on ideas from Sect. 2. The paper closes with conclusions which summarize our findings and ideas on next research steps.

2 Discovery of Process Fragments

A necessary part of a solution for process abstraction are the mechanisms for the discovery of process fragments, i.e., parts of process logic suitable for abstraction. The chances of making a correct decision on which part of a process specification to abstract from can only be maximized if all the potential candidates for conducting an abstraction are considered. To achieve this completeness, we employ the *connectivity* property of

process graphs—directed graphs used to capture process specifications.

Connectivity is a property of a graph. A graph is k-connected if there exists no set of $k-1$ elements, each a vertex or an edge, whose removal makes the graph disconnected, i.e., there is no path between some pair of elements in a graph. Such a set is called a separating $(k-1)$-set. 1-, 2-, and 3-connected graphs are referred to as connected, biconnected, and triconnected, respectively. Each separating set of a process graph can be addressed as a set of boundary elements of a process fragment, where a boundary element is incident with elements inside and outside the fragment and connects the fragment to the main flow of the process. Let m be a parameter, the discovery of all separating m-sets (graph decomposition) of the process graph leads to the discovery of all process fragments with m boundary elements—potential abstraction candidates.

In general, one can speak about (n, e)-*connectivity* of process graphs. A graph is (n, e)-*connected* if there exists no set of n nodes and there exists no set of e edges, whose removal makes the graph disconnected. Observe, an (n, e)-connected graph is $(n+e+1)$-connected. A lot of research was carried out by the compiler theory community to gain value from the triconnected decomposition of process specifications, i.e., the discovery of triconnected fragments in process graphs. The decompositions which proved useful were $(2, 0)$-decomposition, or the tree of the triconnected components, cf., [9], and $(0, 2)$-decomposition, cf., [2]. Triconnected process graph fragments form hierarchies of single-entry-single-exit (SESE) fragments and are used for process analysis, process comparison, process comprehension, etc. For these decompositions, linear-time complexity algorithms exist [1, 3]. Recently, these techniques were introduced to the business process management community [10, 11]. We employed triconnected process graph fragments to propose mechanisms of process abstraction [7, 8]; we discover and generalize the triconnected process fragments to tasks of a higher abstraction level.

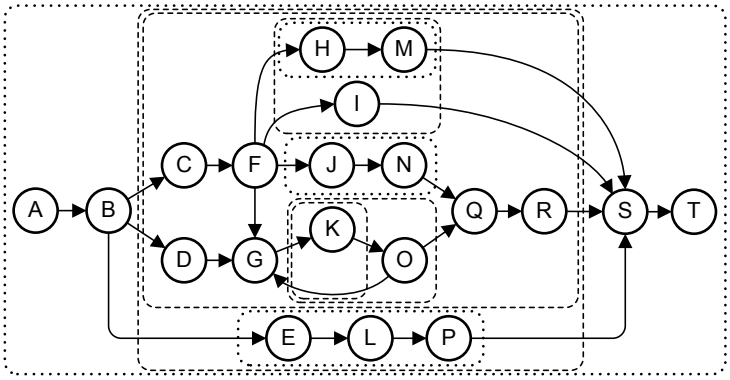


Figure 2: Process graph, its SESE fragments

Figure 2 shows an example of a process graph. Routing decisions and synchronization points can be distinguished by the degree of the corresponding vertex, i.e., the number of incident edges, and orientation of the incident edges, i.e., incoming or outgoing. Process starts (ends) have no incoming (outgoing) edges. Moreover, Figure 2 visualizes the triconnected fragments of the graph (SESE fragments). Each triconnected fragment is enclosed in the region and is formed by edges inside or intersecting the region. Fragments enclosed by regions with dotted borderlines can be discovered after performing $(0, 2)$-decomposition of the process graph,

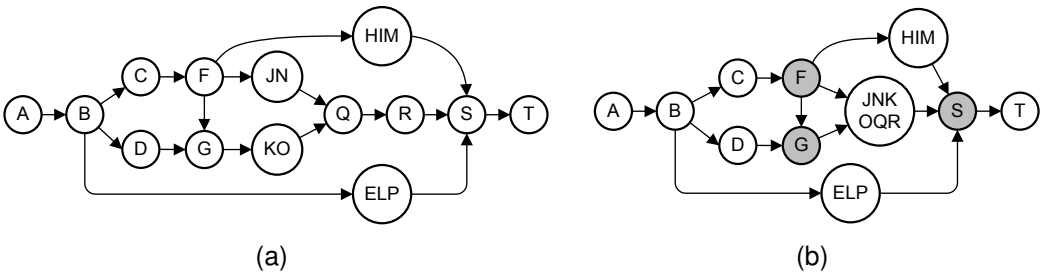


Figure 3: (a) Abstract process graph, (b) $(3,0)$-connected fragment abstraction

whereas regions with dashed borderlines define fragments additionally discovered by $(2,0)$-decomposition. Observe that trivial fragments composed of a single vertex of the process graph are not visualized.

The abstract process graph, obtained from the graph shown in Figure 2, is given in Figure 3(a). The abstraction is performed following the principles from [8], i.e., by aggregating the triconnected fragments into concepts of a higher abstraction level. The graph from Figure 3(a) has a single separating pair B, S. Next abstractions of the triconnected fragments of the graph will result in aggregation of either the unstructured fragment composed of nodes $\{C, ..., R\} \setminus \{E, L, P\}$, or the fragment with entry node B and exit node S.

In order to increase the granularity of process fragments that are used to perform abstractions in [10, 11], one can start looking for multiple-entries-multiple-exits (MEME) fragments within the triconnected fragments. Figure 4 visualizes a connectivity-based process graph decomposition framework, i.e., the scheme for process fragment discovery. In the figure, each dot represents a connectivity property of the process graph (process fragment) subject to decomposition, e.g., $(0,0)$ means that the graph is connected if no nodes and no edges are removed. Edges in the figure hint at which decomposition can be performed for a graph with a certain connectivity level. For instance, one can decompose a $(0,0)$-connected graph by looking for single nodes or edges which make it disconnected. Similarly, the triconnected graphs can be decomposed into $(3,0)$-, $(2,1)$-, $(1,2)$-, or $(0,3)$-connected fragments. In general, an (n,e)-connected process graph (process fragment) can be $(n+1,e)$- or $(n,e+1)$-decomposed. Observe that in this way highly connected process fragments get gradually decomposed.

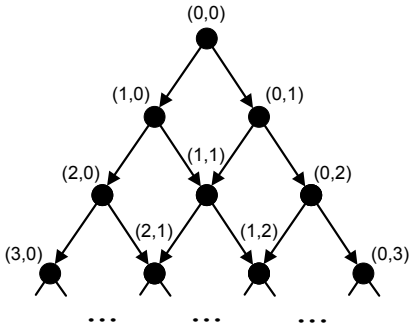


Figure 4: Connectivity-based process graph decomposition framework

An (n,e)-decomposition ($n+e>3$) allows to decompose unstructured process graphs into MEME fragments with $n+e$ entries and exits. A process graph in Figure 3(b) is obtained by abstracting a $(3,0)$-connected fragment defined by a separating set $\{F, G, S\}$ (F and G are the entries and S the exit of the fragment, highlighted with grey background). For reasonable $n+e$ combinations, it is possible to perform decomposition in low polynomial-time complexity. For example, the $(3,0)$-decomposition of a $(2,0)$-connected graph can be accomplished by removing a vertex from the graph and then

running the triconnected decomposition [1]. Each discovered separation pair together with the removed vertex form a separating triple of a 4-connected fragment. The procedure should be repeated for each vertex of the original graph. Hence, a square-time complexity decomposition procedure is obtained. Following the described rationale, one can accomplish $(k, 0)$-decomposition in $O(n^{k-1})$ time.

By following the principles of the connectivity-based decomposition framework, we not only discover process fragments used to perform process abstraction, but also learn their structural characteristics. Structural information is useful at other stages of abstraction, e.g., when introducing control over abstraction. Initial work on classifying and checking the correctness of process specifications based on discovered process fragments was accomplished in [4].

3 Abstraction Control

The task of adjusting the abstraction level of process specifications requires intensive intellectual work and in most cases can only be accomplished by process analysts manually. However, for certain use cases it is possible to derive automated or to support semi-automated abstraction control mechanisms. The task of abstraction control lies in telling significant process elements from insignificant ones and to abstract the latter. In [6], work activities are classified as insignificant if they are rarely observed during process execution. We were able to establish the abstraction control as investigated processes were annotated with information on the average time required to execute work activities. Process fragments which contain insignificant work activities get abstracted. Hence, we actually deal with the significance of fragments which represent detailed work specifications.

Significant and insignificant process fragments can be distinguished once a technique for fragment comparison is in place, i.e., a partial order relation is defined for the process fragments. The average time required to execute work activities in the process provides an opportunity to derive a partial order relation, i.e., fragments which require less time are considered insignificant. Other examples of criteria and the corresponding abstraction use cases are discussed in [5].

Once an abstraction criterion, e.g., the average execution time of work activities, is accepted for abstraction, one can identify a minimal and a maximal value of the criterion for a given process specification. In our example, the minimal value corresponds to the most rarely observed work activity of the process and the maximal value corresponds to the average execution time of the whole process. By specifying a criterion value from the interval, one identifies insignificant process fragments—those that contain work activities for which the criterion value is lower than the specified value. Afterwards, insignificant fragments get abstracted. In [5], we proposed an abstraction slider as a mechanism to control abstraction. An abstraction slider is an object that can be described by a slider interval defined by a minimal and a maximal allowed value for an abstraction criterion value, has a state—a value which defines the desired abstraction level of a process specification, and exposes behavior—an operation which changes its state.

4 Correctness of Process Specifications

Process specifications define allowed process execution scenarios. As a result of creative modeling practices, models can contain errors, e.g., have scenarios with improper termination or contain activities that can never become enabled and, hence, executed. The basic correctness criterion for process specifications, originally defined for WF-nets, is behavioral *soundness*. In a sound process, for each activity from a specification there exists an execution scenario (a process instance) which contains this activity. Moreover, one can uniquely recognize the events of starting and finalizing a process instance. SESE fragments have proven useful when decomposing the task of analyzing behavioral correctness of process specifications. For instance, if a SESE fragment of a process specification was shown to be sound, in the context of overall model correctness it can be addressed as a single edge passing control flow from its entry to its exit [4, 10].

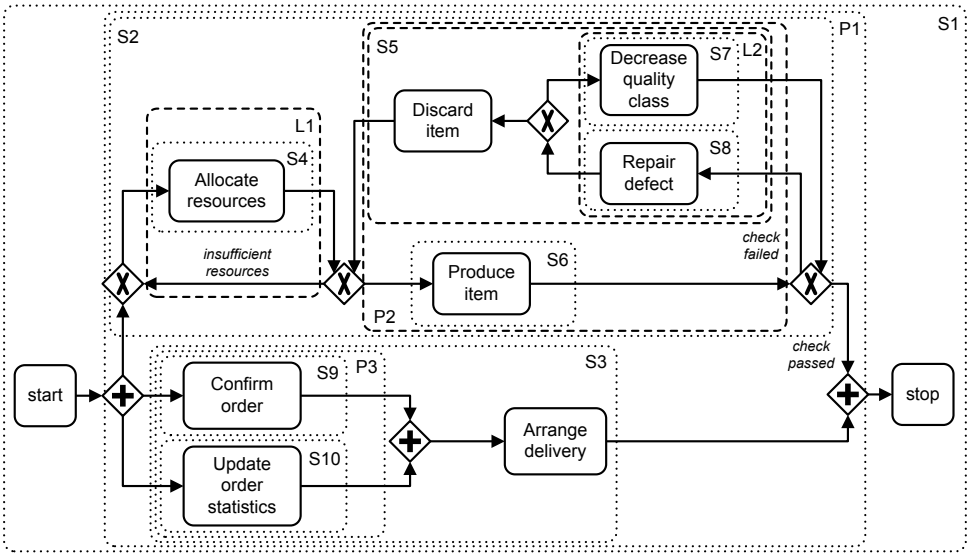


Figure 5: $(2, 0)$-decomposition of the process specification (BPMN notation)

The connectivity property of process graphs, in particular $(2, 0)$- and $(0, 2)$-decompositions of process specifications, are extensively studied in literature for the purpose of SESE fragments discovery [2, 7, 9, 11]. However, these techniques cannot be applied to an arbitrary structural class of process specifications in a straight-forward manner. Figure 5 shows the $(2, 0)$-decomposition of a process specification. Observe that not all $(2, 0)$-fragments form SESE fragments, the ones enclosed into the regions with a dashed borderline do not have a dedicated entry and/or exit nodes, whereas fragments enclosed in the regions with a dotted borderline are SESE fragments. The primary reason for this is that the process specification contains “mixed” gateways, i.e., control flow routing nodes with multiple incoming and multiple outgoing edges. Despite the fact that the process from Figure 5 appears to be block structured it can expose complex execution scenarios with control flow entering and leaving structured block patterns through both gateways which mark the block. In [4], we show that such complex behavior can be localized by examining structural patterns in “hidden” unstructured

regions of control flow. As an outcome, the correctness of the behavior of process specifications within these regions can be validated in linear time.

Figure 6 provides an alternative view on $(2,0)$-decomposition of the process specification from Figure 5. Here, each node represents a $(2,0)$-fragment and edges hint at containment relation of fragments. For instance, fragment $P1$ is contained in fragment $S1$, and contains fragments $S2$ and $S3$. Observe that one obtains a tree structure and that fragment names hint at their structural class, e.g., S for sequence, P for parallel block (structurally, but not semantically), and L for loops, cf., [4].

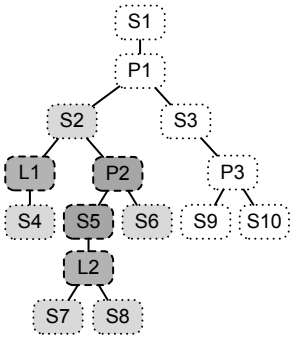


Figure 6: The tree of the triconnected components of the process from Figure 5

In the process specification from Figure 5, fragments $S5$, $L1$, $L2$, and $P2$ are non-SESE fragments, the corresponding fragment nodes are highlighted with dark grey background in Figure 6. Together with fragments that are represented by adjacent nodes in Figure 6 the control flow region constitutes a SESE fragment of hidden complex behavior with entry and exit nodes of fragment $S2$. The region is highlighted with grey background in Figure 6.

In [4], we observe that process behavior within above descried regions is determined by loop fragments, i.e., by fragments $L1$ and $L2$ in the running example. A $(2,0)$-fragment is a non-SESE fragment if and only if it has a boundary node that is a mixed gateway which has at least one incoming and at least one outgoing edge both among internal and external fragment edges. Moreover, a non-SESE fragment is either a loop fragment, or it shares a boundary node with a loop fragment. In the class of free-choice process specifications, the boundary nodes of a loop fragment cannot introduce concurrency to the process instance of a sound process specification. Hence, the boundary nodes of $L1$ and $L2$ fragments in process specification from Figure 5 must implement exclusive-or semantics. Furthermore, as loops share boundary nodes with other non-SESE fragments they imply behavioral constraints on all $(2,0)$-fragments of the region. For further information, cf., [4].

5 Conclusions

In our research we develop methods which allow the derivation of high abstraction level process specifications from detailed ones. In order to discover fragments suitable for abstraction, we employ structure of process specifications, which are usually formalized as directed graphs. As an outcome, developed techniques can be generalized to any process modeling notation which uses directed graphs as the underlying formalism.

It is a highly intellectual task to bring a process specification to a level of abstraction that fulfills emergent engineering needs without a single perfect solution. By employing the technique for the discovery of abstraction fragments, one can approach the problem as a manual engineering effort. Besides, when it is sufficient to fulfill certain use case, cf., [6], one can define the principles for the semi-automated or fully automated composition of individual abstractions.

As future steps aimed at strengthening the achieved results, we plan to validate the applicability of the connectivity-based process graph decomposition framework for the purpose of process abstraction with industry partners and to look for process abstraction use cases for which automated control mechanisms can be proposed. Finally, studies regarding the methodology of abstractions need to complement technical results.

References

[1] Carsten Gutwenger and Petra Mutzel. A Linear Time Implementation of SPQR-Trees. In *GD*, pages 77–90, London, UK, 2001. Springer Verlag.

[2] Richard Johnson. *Efficient Program Analysis using Dependence Flow Graphs*. PhD thesis, Cornell University, Ithaca, NY, USA, 1995.

[3] Richard Johnson, David Pearson, and Keshav Pingali. The Program Structure Tree: Computing Control Regions in Linear Time. pages 171–185. ACM Press, 1994.

[4] Artem Polyvyanyy, Luciano García-Bañuelos, and Mathias Weske. Unveiling Hidden Unstructured Regions in Process Models. In *CoopIS*, Vilamoura, Algarve-Portugal, November 2009. Springer Verlag.

[5] Artem Polyvyanyy, Sergey Smirnov, and Mathias Weske. Process Model Abstraction: A Slider Approach. In *EDOC*, pages 325–331, Munich, Germany, September 2008. IEEE Computer Society.

[6] Artem Polyvyanyy, Sergey Smirnov, and Mathias Weske. Reducing Complexity of Large EPCs. In *MobIS: EPK*, pages 195–207, Saarbruecken, Germany, November 2008. GI.

[7] Artem Polyvyanyy, Sergey Smirnov, and Mathias Weske. On Application of Structural Decomposition for Process Model Abstraction. In *BPSC*, Leipzig, Germany, 2009. GI.

[8] Artem Polyvyanyy, Sergey Smirnov, and Mathias Weske. The Triconnected Abstraction of Process Models. In *BPM*, Ulm, Germany, September 2009. Springer Verlag.

[9] Robert E. Tarjan and Jacobo Valdes. Prime Subprogram Parsing of a Program. In *POPL*, pages 95–105, New York, NY, USA, 1980. ACM.

[10] Jussi Vanhatalo, Hagen Völzer, and Frank Leymann. Faster and More Focused Control-Flow Analysis for Business Process Models Through SESE Decomposition. In *ICSOC*, pages 43–55. Springer Verlag, September 2007.

[11] Jussi Vanhatalo, Hagen Völzer, and Jana Koehler. The Refined Process Structure Tree. In *BPM*, pages 100–115, Milan, Italy, September 2008.

Information Integration in Services Computing

Mohammed AbuJarour
mohammed.abujarour@hpi.uni-potsdam.de

Information integration has been the typical approach to data-driven applications in several domains, such as, enterprise applications. Applying information integration techniques in SOA is not straightforward, because of the lack of adequate resources. In this work, we propose a novel approach and platform to alleviate this problem and investigate the benefits of information integration in SOA applications. This idea has been partially incorporated in the PoSR project successfully [2].

1 Introduction

Service-Oriented Architecture (SOA) has changed the way in which IT serves business needs. Major service providers tend to provide Web Service interfaces to their services over the Web, e.g., Amazon, flickr, Yahoo!, eBay. This increasing number of Web Services poses challenges and creates opportunities in both business and IT. Business bodies need to utilize each possible opportunity to increase their profits and market shares. Moreover, business bodies need flexible, adaptive and scalable solutions. SOA has emerged as a solution for such challenging requirements.

Service Management has been a vital issue in SOA. This is handled by Service Registry and Repository, which also plays the role of broker between a service provider and a service consumer. However, managing Web Services and their associated information is no longer an easy job that could be achieved using a traditional middleware solution, such as a UDDI registry. This is due to the increasing complexity of services and their associated information [11]. On the one hand, the types of service metadata are versatile, e.g., XML, BPEL, XSLT, WSRP, etc, and a single service could be described in more than one type. On the other hand, Web Services usually lack adequate formal descriptions.

This increasing complexity of service information influences service discovery and selection in SOA applications, where a service consumer queries a service registry for a required service that achieves a specific task. A service registry needs to consider as many service description artifacts as possible to provide a precise result list to the service consumer. When the service consumer, such as human, application, service, etc., receives a result list for its query, it would like to select the most suitable service from the returned result list efficiently and effectively. Making the right choice is not straightforward because several factors should be considered, e.g., price, quality, reputation, etc. Service description that is gathered from service providers and consumers is the

main source to determine which service to select. We view this task as an information integration task, as we explain in Section 3.

The remainder of this report is organized as follows. Section 2 introduces the research context and gives further details about the research problem and challenges. We explain our proposed approach in Section 3. Then, we highlight the significant research works in this field in Section 4. After that, we summarize this report and show our future steps in Section 5.

2 Research Context

In this section, we describe our research context. We start by introducing our research problem, then outline major research challenges, in addition to our expected contributions.

2.1 Research Problem

The available tools and frameworks that help service providers deploy their systems as Web Services and the popular *Software-as-a-Service* trend have helped increasing the number of services provided in the form of Web Services. Nevertheless, this easiness in service creation has complicated the problem of service reusability because those services usually lack enough description artifacts that help service consumers select the most appropriate service for their needs.

The complexity of SOA systems hinders employing SOA in several domains, in particular in real-time applications. Ideally, a service consumer issues a service call to a service provider, in order to invoke a specific service. The service provider returns the result of that query to the service consumer. In real word scenarios, this is not usually the case. A service might be no longer available, unavailable at the moment of its call, or unable to serve this request. Reacting to this situation pushes complexity inside the SOA application. In our approach, this complexity is hidden inside the service broker component, as shown in Section 3.2.

SOA has been proposed to bridge the gap between business and IT environments. But, SOA environments (both business and IT) are dynamic by nature. Typically, services appear and disappear often. To respond to such a situation – while achieving maximum business benefits – a service consumer needs to use a reliable service registry, use several service registries, or deploy a complex application, which reacts to expected failures or exceptions. Unexpected failures or exceptions remain threats to application's sustainability and reliability.

In our research, we aim at enriching service descriptions, maximizing the benefits of existing descriptions, and integrating all available resources to provide the highest precision for service discovery and selection. Our research statement is summarized in this question: How to enrich, integrate, and manage service descriptions efficiently and what are the benefits of enriching service descriptions in SOA?

2.2 Research Challenges

The key challenges that drive our research include:

Dynamic SOA and business environments: Business needs and conditions are always changing by nature. To respond to these changes, services and service providers have to adapt by modifying their services or providing new services in a time-effective manner.

Increasing complexity: This complexity of services is due to the increasing number of tools, frameworks, domains, and business needs that support or require Web Services. In most cases, this is reflected in the lack of adequate service description artifacts.

Heterogeneity: Different techniques and formats are used to describe and provide services. Different parties in SOA environments may have different notions of the same concept, e.g., reputation, business objects. Furthermore, different scales could be used to describe the same notion, e.g., trust level.

Inadequate criteria for service discovery and selection: Full-text search is usually used by service registries as a means of service lookup, but the quality of the result list depends on the information used to lookup a service. Furthermore, non-functional requirements of the services should be considered in the lookup process to make service discovery more comprehensive.

The expected contributions of this work include:

- A set of techniques to enrich service descriptions.
- Model of service data quality.
- A novel approach to enhance service selection and discovery.
- Assessment methods of service discovery by service consumers.
- An integration environment for SOA applications.
- A smart service registry and repository, coined *"Deposr"*.

3 Our Approach: Non-traditional Information Integration

In our approach, there is a substantial difference between information integration in SOA and traditional information integration. This difference leis in the heterogeneous types of data, on which we apply our information integration techniques. For example, WSDL files have structured and semi-structured data, whereas, a category (metadata) is a simple label. On the other hand, service's quality measures can be represented as a complex matrix. Integrating all these versatile types of information requires more

than traditional information integration techniques, as we explain in this section, where we give more details about our proposed approach to information integration in SOA. We start with an overview of the proposed environment and its expected features, then we show further implementation details.

3.1 Deposr: An Information Integration Environment

Our SOA information integration system is coined *"Deposr"*. The three layers of *Deposr* are depicted in Figure 1. The storage layer (Layer 1) manages all relevant information about the services, such as services' data, metadata, community annotations. Several internal functions are required to acquire and manage this amount of data, such as validate and update service data, integrate community annotations. This set of functions is provided via Layer 2. On top of the storage and internal functions layers, Layer 3 provides several useful features for service consumers, such as service lookup, service recommendation, service quality assessment. In the sequel, we explain these layers in a bottom-top order.

The **Storage Layer** controls several classes and types of information about the services that are managed by *Deposr*. Data class includes structural and semi-structural data, such as WSDL and WADL files. This class is typically provided or acquired from service providers. The second class, metadata, includes information about the managed services, such as invocation frequency, information about service providers, category. This class is derived from services' data and their calls. Invocation metadata is gathered by analyzing services invocations, which helps us get more information about each service, its inputs and outputs, in which context it is being invoked, and in conjunction with which other services it is being invoked. Additionally, users who would like to help *Deposr* improve the quality of its features and assess the quality of service discovery could also provide explicit feedback and annotations about the services that they have used. This information falls in the community annotations class. Furthermore, a consumer's profile tracks service usage history where information about previous services' invocations by each consumer is tracked and gathered. This information is helpful for the consumer and for other similar consumers. We can make use of this history of invocations to rank the list of retrieved services according to consumer's preferences (tastes) that we derive from its previous service usage. Other similar consumers – in terms of preferences and tastes – could also get more accurate results based on the behavior of similar consumers, which is derived from usage history.

To manage, control, and integrate the different types of storage information, and to provide the essential and value-added features of *Deposr*, several internal functions are required. This set of internal functions represents the **Internal Functions Layer**, Layer 2 in Figure 1. Part of these internal functions control the different types of information in the Storage Layer, such as validate and update service data, manage service information and user interface and description. Other internal functions create or extract new information about services from services invocations and community annotations. This latter set of internal functions provides a set of techniques to enrich service descriptions. Bringing all these classes of information require a service data quality assurance internal function that performs required information quality techniques on the level of

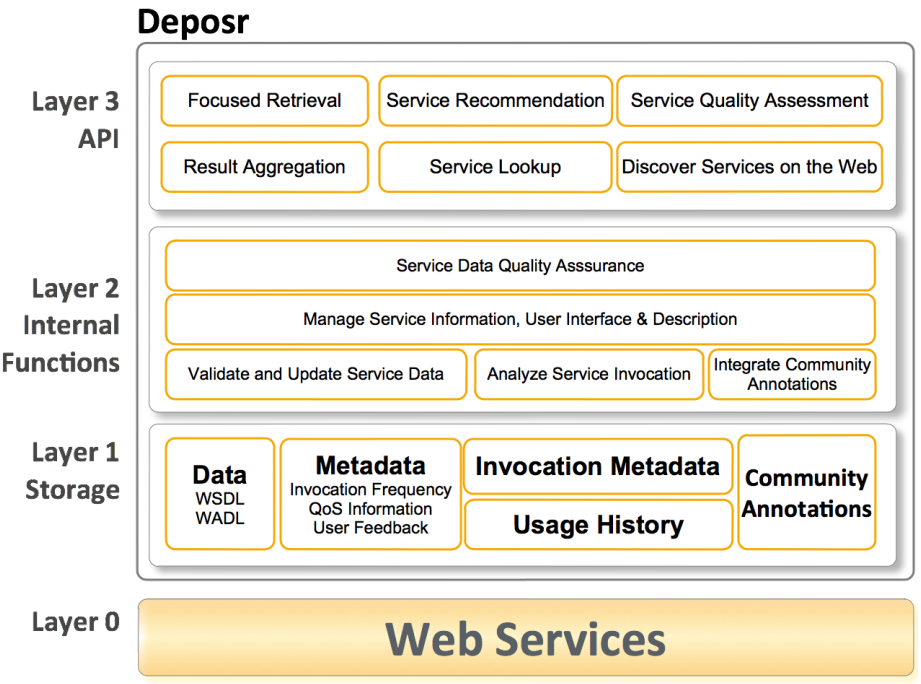
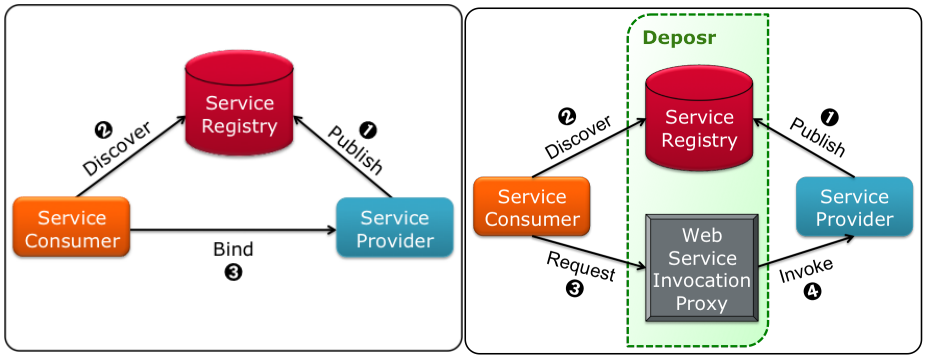


Figure 1: The Architecture of Deposr

the information in the Storage Layer.

Fundamental and value-added features of *Deposr* are made available via the **API Layer**; Layer 3. Service lookup is a fundamental feature that enables service consumers find services, which fulfill their needs. As we use enriched service descriptions and employ new models to fulfill non-functional requirements, this enhances service selection and discovery. Result aggregation has been a popular research trend: search results from several systems are aggregated in one list, to obtain more accurate and comprehensive results. For instance, Metacrawler (`http://www.metacrawler.com`) submits user queries to several search engines, e.g. Google, Yahoo!, Windows Live (Bing), and Ask, and aggregates the different individual result lists, it gets back from those engines, into a single result list. This principle can also be adapted to *data* Web Services, where the results of several similar services' invocations are aggregated into a single result list, e.g., news items. Nevertheless, the aggregation of several result lists is not practically useful if the resulting aggregated list is simply too long. This limitation is handled by adapting the principles of focused retrieval. The task of focused retrieval is to specify the most relevant parts of each item in the result list and report them instead of reporting the entire items to the user. By tracking consumers usage history, we can apply a service recommendation feature, where service consumers get a list of recommended services or services that were used by similar consumers. Deposr allows service providers to add their services to the system manually. Additionally, we employ crawling techniques and other heuristics that we have developed to automatically gather Web Services on the Web. This set of features help service consumers respond to the dynamic changes in SOA and business environments by keeping their



(a) Traditional service registries (b) The role of Service Invocation Proxy

Figure 2: The traditional role of service registry and the extended role of the Service Invocation Proxy

information up-to-date.

The unified service description, which we get from the different types of service information, helps us provide and evaluate more quality attributes to provide an enhanced set of service quality assessment. Service discovery is usually based on functional requirements, e.g., the operation of each service. Nevertheless, non-functional requirements, e.g., reliability, accuracy, accessibility, consistency, timeliness, availability, relevancy, efficiency, usability, security, trust, etc., are vital criteria, especially in business. Several approaches have been proposed to handle this issue. Some approaches use information provided by the service provider to evaluate Quality of Service (QoS) attributes for that specific service provider [6]. Other approaches make use of service consumers' ratings for the used services [7]. In our approach, we combine both approaches and augment them with a third source of information to evaluate QoS attributes for a service provider, a service, and a service consumer (e.g., reputation). This source is the invocation metadata, which we derive from service invocation analysis.

3.2 Web Service Invocation Proxy

Traditionally, a Service Registry and Repository (see Figure 2 (a)) acts as a broker between service consumer and service provider. The role of the service registry ends when the suitable service is discovered by the service consumer. Afterwards, the service consumer contacts the service provider to request achieving the required task.

In our research, we propose an extended architecture where a service invocation proxy acts as a mediator between service consumers and service providers. This extended architecture is depicted in Figure 2 (b). A service consumer queries the service registry to find an appropriate service. After that, the service consumer sends a request to the service invocation proxy to use the selected service. The service invocation proxy issues a service call to the service provider to invoke the service and sends the result back to the service consumer. If the service is no longer available, the service invocation proxy notifies the service consumer with a list of alternative services.

If the selected service is not available at the moment, or the issued service call takes much time, then the service broker can suggest making another call to a similar service transparently.

The proposed architecture has several advantages. It hides the complexity of SOA applications by moving some complex tasks, e.g., fault tolerance, which would be achieved by each SOA application separately, into the service broker. Moving such tasks into the service broker enables it to achieve them more efficiently because it can access more service calls and their associated invocation information. Another advantage of this architecture would be caching. Suppose that hundreds of service consumers issue separate service calls to a few weather forecast services. If the service invocation proxy is deployed in this scenario, then the number of service calls will be reduced by providing a cached copy of the service result from previous service calls. This will reduce network traffic and load on services.

The storage layer of Deposr, as depicted in Figure 1, is served by the service registry component and the service invocation proxy. The data and metadata are managed by the registry, whereas the invocation metadata, usage history, and community annotations are managed by the service invocation proxy (Figure 2 (b)). Both components, service registry and service invocation proxy, are the main source of data used to provide internal functions and API's.

3.3 Invocation Analysis

Invocations are either automatic invocations or invocations done by service consumers. For new services, automatic invocations would be necessary to generate information about the new service, so that, for instance, quality measures of that particular service can be carried out. Invocation analysis are achieved through Web Service Invocation Proxy, as we described in Section 3.2. On the other hand, tracking users' invocations of services through the system can be a useful source of information. Each service call is tracked and "sampled" to extract metadata about the invoked service.

Invocation analysis are especially beneficial for new services which have not yet been evaluated by service consumers and lack enough information, which is necessary to achieve a precise service discovery. Moreover, for services that have been already used and evaluated by service consumers, our approach is still beneficial because a combined source of information is used to measure service quality and enhance the quality of service discovery. Additionally, invocation analysis helps alleviate the increasing complexity in SOA by sampling so many service calls issued by several service consumers in diverse domains by gathering more information that can be used to understand these services better.

To illustrate the value of invocation analysis, we introduce a simple example to service quality. Buying an item online includes two broad activities: ordering the item and shipping the ordered item. Usually, different service providers are in charge of each activity. In most cases, more than one service provider can achieve the same task. See Figure 3. Two service providers (B and C) can ship the ordered item, but the buyer decides which one to choose. Suppose that service provider (A) and service provider (B) have a business partnership, which is reflected in handling the item from (A) to (B)

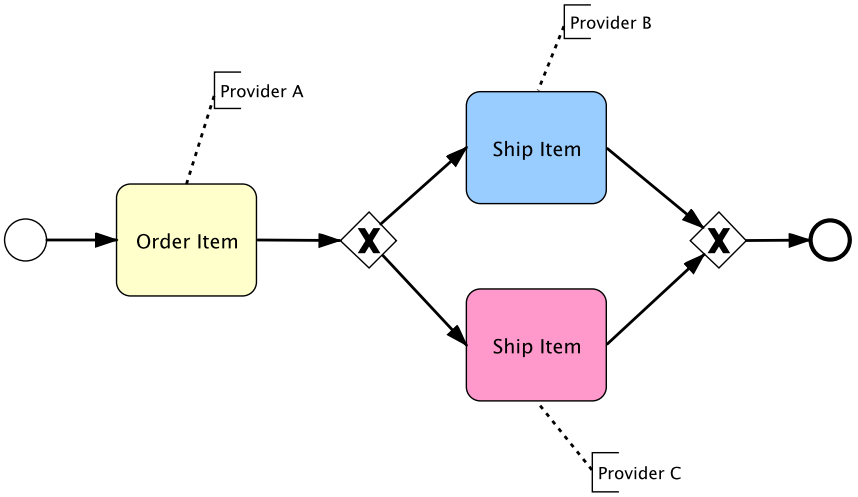


Figure 3: A composite service for ordering and shipping an item online.

in less time, say in the same day. On the other hand, delivering the item from (A) to (C) takes 2-3 days. Furthermore, suppose that both providers (B) and (C) take the same time to deliver the item to the buyer, say 3 days. Shipping the item through (B) takes around 3 days, whereas, shipping the same order via (C) takes 5-6 days. Following the traditional SOA approach where the entire composite service is viewed as a single service, the total time required to execute this service call will be 3 days, in the case of (B) and 5-6 days in the case of (C). This gives (B) advantage over (C) with respect to time, but this is not accurate, because both providers (B and C) need 3 days to deliver the item.

Applying our approach gives different results. By analyzing each individual service call, we can evaluate quality attributes for each service separately. Additionally, analyzing several service invocations issued by service consumers help us provide more accurate quality measures for each service.

4 Related Work: Service Brokers

In this report, we only highlight most relevant projects and papers in this field. For a detailed overview of related work, please, refer to [1].

Public UDDI Business Registries were shutdown, because they did not prevail in the domain of public Web Services [9]. *ebXML* registries, e.g., FreebXML [4], were proposed by OASIS (`http://www.oasis-open.org`) to handle this issue. Existing service registries and repositories still have limitations, e.g., inadequate functions, supporting limited Web Service types, etc. IBM's WebSphere Service Registry and Repository does not support all service metadata, e.g., WSRP [5]. SUN's Service Registry and Repository is based on both ebXML and UDDI, but is limited to Web Services only [10]. Centrasite is based on UDDI and limited to Web Services inside the organization only [3]. FreebXML is an open source registry, which is based on ebXML [4].

The notion of proxies in SOA was introduced in [8] as an agent-oriented, integrated component in service consumers to evaluate the reputation, and level of trustworthiness of services and service providers, respectively.

5 Summary and Future Work

The increasing number of web services and the easiness of creating Web Services from existing software applications have been increasing the complexity of SOA systems and making service discovery and selection, evaluating service quality, and providing fundamental and value-added features more and more challenging. The main reason for these limitations is the lack of enough information and descriptions associated with these services. In addition to other reasons that help complicate the problem further, such as the dynamic nature of SOA and business environments. In this report, we introduced a new integration environment for SOA applications, where we integrate different sources of information about services to provide the required features and value-added features in SOA.

All parties involved in a SOA application are taken into consideration and information from these parities is gathered. This information includes *data* from service providers, e.g., WSDL, *metadata* which is stored in the registry, e.g., category, *community annotations and consumers feedback, invocation metadata*, and *usage history*. All these different types of information are then used to create a unified service description for each service using non-traditional information integration techniques.

One important aspect in this context is privacy and information security. Therefore, this architecture fits better for SOA applications inside the boundaries of a single organization. Further schemes for privacy and information security are part of our future work.

References

[1] Mohammed AbuJarour. On a Model for a Service Database. Technical report, Hasso-Plattner-Institut, Potsdam, Germany, October 2008.

[2] AbuJarour M. et al. Posr: A Comprehensive System for Aggregating and Using Web Services. In *Proceedings of the 2009 IEEE Congress on Services*, USA, 2009.

[3] Centrasite Community. Centrasite Registry. `http://www.centrasite.org`.

[4] FreebXML. ebXML Registry-Repository. `http://ebxmlrr.sourceforge.net`.

[5] IBM. WebSphere Service Registry and Repository. `http://www.ibm.com/software/integration/wsrr`.

[6] Yutu Liu, Anne H. Ngu, and Liang Z. Zeng. QoS Computation and Policing in Dynamic Web Service Selection. In *WWW Alt. '04*, pages 66–73, New York, NY, USA, 2004. ACM.

[7] Zaki Malik and Athman Bouguettaya. RATEWeb: Reputation Assessment for Trust Establishment among Web services. *VLDB J.*, 18(4):885–911, 2009.

[8] E. Michael Maximilien and Munindar P. Singh. Toward Autonomic Web Services Trust and Selection. In *ICSOC '04*, pages 212–221, New York, NY, USA, 2004. ACM.

[9] Steinmetz N. et al. D1.1 Requirement Analysis and Architectural Plan. Technical report, Service Finder, October 2008.

[10] SUN Microsystems. SUN's Service Registry. `http://www.sun.com/products/soa/registry`.

[11] Sun Microsystems. Effective SOA Deployment Using an SOA Registry Repository. *http://www.sun.com/products/soa/registry/soa_registry_wp.pdf*, 2005. White paper.

Declarative and Event-based Context-oriented Programming

Malte Appeltauer
Software Architecture Group
Hasso-Plattner-Institut
malte.appeltauer@hpi.uni-potsdam.de

The original context-oriented programming approach does not provide dedicated means for the representation of event-specific context-dependent behavior. In this report, we motivate both the need for event-specific context-dependent behavior and appropriate constructs provided with our JCop programming language extension in its support. We present both a reimplementaion of an example discussed previously to show the improvement achieved based on the application of our newly introduced language constructs and a service-based application which we set up to evaluate JCop in a distributed environment.

1 Introduction

With the increasing demand of personalization and mobility of applications, context awareness gets a distinguishing feature for software systems. Context-aware applications consider context-information, which can be any information computationally accessible, for individual computation. Depending on their context, actors regard a software system from different perspectives. Actors can be objects of the system itself, software developers, or end-users. More formally, a context is constituted by predicates that evaluate its presence and a set of behavioral variations that reflect the context-specific behavior. Context-specific behavior variations are often crosscutting concerns whose implementation is scattered over a decomposition. Therefore, a major task for context representation is the modularization of these crosscutting concerns. Besides modularization, context-specific concerns require means for fine-grained expression of their composition.

Context-oriented programming [5] (COP) is an approach to represent context-specific concerns that focuses on dynamic composition of control flows. COP allows for the definition of layers, cross-cutting modules that encapsulate behavioral variations. Layers are composed at run-time to the system behavior that is required in a certain context. In general, a context can appear at any point in an execution graph. Moreover, at the same time, several clients can regard an object from within different contexts. For these control-flow specific contexts, COP provides the *with* statement, a construct that allows for the composition of layers for the dynamic extent of a specific computation.

In this paper, we report on our recent research on context-oriented programming. We present JCop, a COP-based programming language with explicit support for event-

specific context. Moreover, we discuss the development of context-aware service-based systems and their implementation using COP.

Section 2 gives an overview of COP and ContextJ. Section 3 motivates the need for explicit support of event-based context that is addressed by JCop, which is presented in Section 4. Section 5 reports on another thread of our research, the adoption of COP in service-based applications. A summary and next steps are provided by Section 6.

2 Context-oriented Programming for Java

Our research is based on the context-oriented programming approach. In the following, we give an overview of COP and ContextJ [2–4], our earlier COP language extension to Java.

2.1 Context-oriented Programming

Context-oriented programming [5] (COP) addresses the development of systems, whose behavior varies depending on their context of use. In most cases, a behavioral variation is not implemented by a single object; instead, it is distributed over a team of collaborating objects. Such distributed functionality is denoted as *crosscutting concern* [6]. The modularization and composition of crosscutting concerns requires additional language abstractions beyond object-oriented programming. COP allows for the convenient expression of behavioral variations that cut across a system's dominant decomposition. Context-dependent functionality is explicitly represented and can be dynamically composed at run-time. Context that requires a composition of behavioral variations can be *everything that is computationally accessible* [5], such as state, control flow, or properties of the system's environment.

Layers. Behavioral variations are implemented by *layered methods* that consists of a *base method definition* and at least one *partial method definition*, which is defined in a layer. A base method denotes the Java method definition that is executed when no active layer provides a corresponding partial method. A partial method definition implements the functionality of a behavioral variation that extends or overrides a base method definition for the time the layer is active.

Dynamic composition. Layers can be composed at run-time. Invocations of layered methods are first send to their active partial method definitions. During its execution, a behavioral variation can proceed to a corresponding partial method in another active layer or, if such method does not exist, to the base method definition. If more than one active layer provides a partial definition for a layered method, the order of layer activation defines the proceed chain, in which the layer activated last is accessed first. Per default, layer activation is scoped per thread and to the dynamic extent of a block of statements.

2.2 ContextJ

The ContextJ[1] [2–4] programming language is an extension to Java and features the *layers-in-class* style [5]: each class affected by a layer L contains a declaration of L that contains partial methods for its class. Thus, classes contain their own context-specific variations.

A layer definition consists of an identifier and a list of method definitions. These definitions specify either new methods that are visible in the scope of their layer, or partial method definitions, whose signature must correspond to a base method in the hierarchy of the enclosing class. During layer activation, invocations of this method are dispatched to the definition provided by this layer.

The built-in pseudo method `proceed` can be used to explicitly invoke the next partial method definition (or the base method). Both the return type and the expected arguments of `proceed` conform to the method's signature.

To control scoped layer activation, ContextJ provides the `with` block statement that can be used in method bodies. It consists of list of expressions of the built-in type `contextj.lang.Layer`. The `without` block construct, as counterpart to `with`, is used for explicitly disabling layer execution for a certain control flow.

3 Event-based Context-specific Behavior

In this section, we present a case study in which we first apply ContextJ, our earlier COP language, and discuss its strength and weaknesses in the domain of event-based systems. In Section 4 we will revisit this example using our new language abstractions provided by JCop.

3.1 Motivation

Besides control-flow specific contexts, which is addressed by COP's with statement, *event-specific context* can influence a system's behavior. We observe two key properties that distinguish event- and control-flow specific context: First, context enter and exit is control-flow independent. For instance, a mobile phone could change its behavior, i.e, changing display brightness and disconnecting network connections, when its battery power decreases 10% and as long as the power stays lower than 20%. For this period of time, any computation of the cell phone should use the respective layers. Second, events can constitute a new context asynchronously. Such asynchronous context changes that cause immediate system adaption may lead to inconsistent computations.

With the abstractions of state-of-the-art COP languages, event-specific context cannot be represented without scattering layer composition statements over the program. Furthermore, for event-specific layer activation we need a save mechanism that does not lead to inconsistent state.

[1] ContextJ is available for download at `http://www.hpi.uni-potsdam.de/swa/cop`

```
1  import layer RTFWidgets;
2  import layer CodeWidgets;
3
4  public class CJEditWindow
5    extends QMainWindow {
6    ...
7    layer RTFWidgets {
8      private QMenu formatMenu;
9      private FormatToolBar formatToolBar;
10
11     after private void showMenus() {
12       ...
13     }
14     after private void showToolBars() {
15       ...
16     }
17     after private void hideWidgets() {
18       ...
19     }
20     after private void showWidgets() {
21       ...
22     }
23   }
24 }
```

```
1  public class CJEditWindow
2    extends QMainWindow {
3    ...
4    void onCursorPositionChanged() {
5      with (getLayersOfPreviousBlock()) {
6        hideWidgets();
7      }
8      with (getLayersOfCurrentBlock()) {
9        showWidgets();
10   }}
11   void onPrint() {
12     with (getLayersOfPreviousBlock()) {
13       ...
14   }}
15   void onSave() {
16     with (getLayersOfPreviousBlock()) {
17       ...
18   }}
19   void onFileNew() {
20     with (getLayersOfPreviousBlock()) {
21       ...
22   }}
23 }
```

Figure 1: *left:* Layered specification of task-dependent GUI Widgets. *right:* Layer compositions within event handlers.

3.2 Case-Study: CJEdit

In a case study, we develop an event-based context-dependent GUI application using ContextJ. CJEdit is a programming environment that supports rich text comments within ContextJ programs.

3.2.1 CJEdit

The CJEdit editor is equipped with syntax highlighting, an outline view, and a compilation/execution toolbar. In addition, it allows to format ContextJ compilation units with rich text comments. For this task, the editor provides *rich text formatting features*, such as font family, size, style, and color modifications. Through the combination of rich text and source code, CJEdit documents are single-source, executable representations of code and documentation. Both activities require different functionality, therefore our application supports focusing on the actual task at hand by offering only relevant tools, menus, and widgets. A context switch between text editing and programming features is either directly triggered by the user, or on text cursor change: While writing new text, the user can enter the programming mode by pushing a toolbar button. Whenever the text cursor is moved through the document from text to code and vice versa, the GUI elements are changed accordingly.

3.2.2 Implementation

CJEdit is implemented using ContextJ and the *Qt Jambi GUI Framework*. The editor consists of approximately 1400 lines of code, where most parts are written with plain Java constructs and the help of the Qt GUI Designer. The overlay of task-specific user interfaces and behavior is implemented in ContextJ. The system contains layers that encapsulate rich text and programming widgets such as toolbars and their corresponding behavior.

Figure 1 (left) shows the implementation of a layer that encapsulates widgets for the rich text edit-specific user interfaces. It provides partial methods that are executed after the execution of their base methods. On a task switch, the system hides specific GUI elements by calling `hideWidgets` and invokes the `showWidgets`, `showMenus`, and `showToolBars` methods. Thus, the layers extend `hideWidgets` with a partial method that removes the layer-specific widgets.

The editor's underlying document tree represents each text line as a text block node. Each block holds a list of layers that should be activated when it is focused. By default, blocks refer to the layers responsible for rich text behavior. If the user switches to the programming activity (by pressing the 'code environment' button in the toolbar), the following text blocks are linked with programming environment-specific layers.

Layers are recomposed whenever the type of the focused block changes from rich text to code block, and vice versa. This change is explicitly activated by entering or leaving the programming activity (by pressing the code button) or on moving the text cursor between blocks of different types.

For the dynamic extent of the recomposition, the layer list of the current block is activated. The composition is triggered by the `onCursorPositionChanged` event, as depicted in Figure 1 (right). First, `hideWidgets` is invoked in the context of layers of the previous block to remove their specific widgets. The GUI elements are then recomposed with the layer composition of the current block.

3.3 Lessons Learned

The two main behavioral variations implemented in our example, namely *rich text editing* and *program development* have been implemented using layers. The layers contain partial method definitions that implement the variations of the default behavior of certain methods. The user-based behavioral switch can be mapped directly to dynamic layer composition.

Besides these benefits, we had to consider some characteristics of GUI-based programming that led to additional challenges for the ContextJ-based implementation. The two most important findings are explained in the following.

First, user interaction with GUI behavior is less control flow-centric but rather event-driven. This complicates dynamic extent-based layer composition as proposed originally by COP. On user interaction – such as printing the document, writing new text, or moving the text cursor through the document – the layers of the current block must be activated in their respective control flows. In the source code, this issue is manifested as repeated `with` statements in the event callback methods, which itself is a

crosscutting concern. Another issue is the intrinsic difference between declarative GUI specifications and dynamic behavioral variations. The exclusive specification of layer-specific widgets is not sufficient, we also need auxiliary methods such as `showWidgets` and `hideWidgets`, and explicitly trigger their execution after layer activation. In an more declarative solution, we would only need to specify the GUI variations and add them to the internal structure. With the activation of a layer, the layered state of this structure would be activated, too.

Based on these findings, we developed a programming language that is capable of the original COP features and additional abstractions for event-based context.

4 Event-based Context Representation with JCop

Based on the experiences discussed in the previous section, we present the JCop language that extends ContextJ with new composition mechanisms for a more declarative expression of event-specific context activation. In the following, we revisit our CJEdit application using the JCop's new language constructs.

4.1 Modularization

Layers can either be defined within the classes for which they provide behavioral variations (*layer-in-class*), or in a dedicated top-level layer similar to an aspect (*layer-in-class*) [1, 5]. Besides the structural differences of the two declaration styles, layer-in-class can access and extend the host object's internal state and methods, while class-in-layer are restricted to public interfaces. Developers can decide per situation if they prefer to define a layer within it's host object, allowing private member access, or to declare all partial definitions of a layer as one layer module to reduce scattering.

Besides partial methods known from ContextJ, partial fields override the state of their base definition and persist these state over layer deactivations. In additions, layers can contain auxiliary methods and fields that are only visible within the scope of their layer.

4.2 Declarative Layer Composition

JCop adopts control-flow specific layer activation from ContextJ. As we identified in our case-study, some layer activations have multiple entry points, requiring repetition of identical with statements. To avoid this cross-cutting concern, JCop supports *declarative with statements*. Using a pointcut-like construct adopted from aspect-oriented programming, a layer composition can be bound to multiple (join-)points. The following listing depicts the use of declarative with statements in CJEdit.

```
1 on(* *.onPrint(..)) ||
2 on(* *.onSave(..)) ||
3 on(* *.onFileNew(..)) {
4   with(getLayersOfCurrentBlock());
5 }
```

The statement declares that the dynamic extent of the executions `onPrint`, `onSave`, and `onFileNew` is executed with the layer composition returned by `getLayersOfCurrentBlock()`.

4.3 First-class Event-based Context

Using declarative with statements solves the issue of scattered with statements, but although we have reduced code repetition, we still need to handle the context change event within the business logic of our text editor. For instance, the editor provides methods to access the layers of the current and previous text node, which need to be stored by the application. For a better separation of concerns and an even more declarative description of context change, JCop provides event-based layer composition which is based on declarative layer composition. The next listing presents two declarative *with* statements using `when` to express the event that triggers context activation.

```
1 ( on(* *.onPrint(..)) || on(* *.onSave(..)) || on(* *.onFileNew(..)) ) &&
2 when(TextEditor.getInstance().getBlock() == BlockType.Programming) {
3   with(CodeWidgetes, SyntaxHighlighting, Outline);
4 }
5
6 ( on(* *.onPrint(..)) || on(* *.onSave(..)) ||  on(* *.onFileNew(..)) ) &&
7 when(TextEditor.getInstance().getBlock() == BlockType.Commenting) {
8   with(RTFWidgets);
9 }
```

The `on` expressions specify the method executions whose dynamic extents use a layer composition. The `when` predicate specifies a runtime condition that must be fulfilled. In many cases we can omit the declaration of the start of our dynamic extent. A shorthand notation that only describes the event predicate for our CJEdit example would be:

```
1 when(TextEditor.getInstance().getBlock().getType() == BlockType.Programming) {
2   with(CodeWidgetes, SyntaxHighlighting, Outline);
3 }
4
5 when(TextEditor.getInstance().getBlock().getType() == BlockType.Commenting) {
6   with(RTFWidgets);
7 }
```

JCop provides a first-class context construct as a dedicated location for the composition statements shown above. In addition to composition statements, contexts can contain auxiliary methods and fields The following listing shows a first-class context specification for the programming context.

```
1 context Programming {
2   when(getBlockType() == BlockType.Programming) {
3     with(CodeWidgetes, SyntaxHighlighting, Outline);
4   }
5
6   private BlockType getBlockType() {
7     TextEditor te = TextEditor.getInstance();
8     return te.getBlock().getType();
9   }
10 }
```

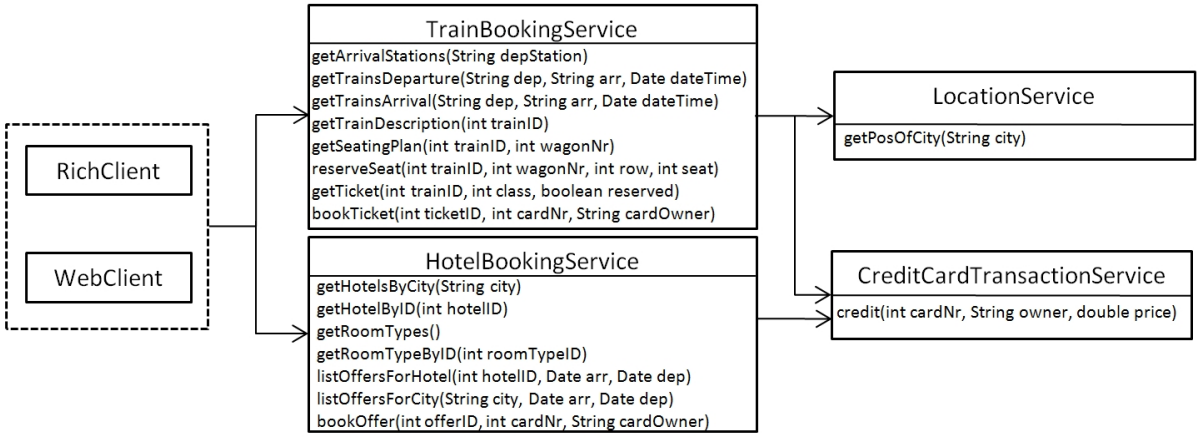


Figure 2: Architecture of the ticket shop system.

5 Using JCop for the Development of a Service-based Ticket Shop

As in any software system, the behavior of a service component can depend on context information. However,

Therefore, we investigate, weather our language abstractions are applicable for service-based systems. In the following, we report on our first results in developing a service-based application using JCop.

5.1 Motivation

Service-based systems are mostly heterogeneous environments, including services that are implemented in different languages and run on machines with different properties. In addition, different types of client applications, such as rich clients, mobile clients, and Web-based clients, can use the same service infrastructure from different execution contexts. Especially mobile clients often provide context information that is relevant for computation.

To evaluate the properties of COP language abstractions in a service-oriented scenario, we implement a service-based ticket shop that adopts a ticket shop use-case by W3C [8][2]. A travel agency offers to book vacation packages including train tickets and hotels. Service providers are providing Web services to query their offerings and perform reservations. `TrainBookingService` employs `LocationService` for the computation of train connections. To purchase tickets, credit card companies are providing services to guarantee payments made by consumers.

To create a heterogeneous service environment, we implement our application in Python and Java using RPC libraries, and provide two different clients. Our Web

[2]The system has been developed in collaboration with Michael Perscheid. We will both share this infrastructure as a base for our research experiments.

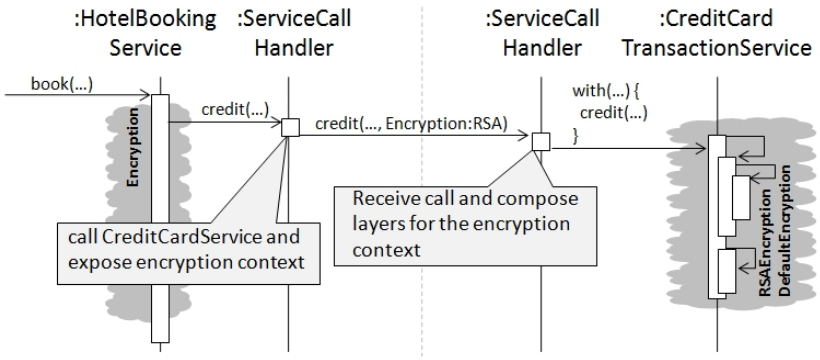


Figure 3: Layer compositions during a service call.

client is based on Smalltalk/Seaside [7], the rich client is a Java/Swing application. Figure 2 presents the application's key components. `TrainBookingService` and `HotelBookingService` offer methods to lookup and book the their respective products. Both services use `CreditCardTransactionService` to accomplish bookings.

5.2 Context-aware Behavior

The ticket shop example contains several context-dependent concerns that we implement using COP. Context can either affect only one module or larger parts of the system, including several services. In the first case, the scope of a layer composition begins and ends within the same module without crossing distribution boundaries.

In the second case, the dynamic extent of a layer composition includes calls to other services. Due to loose coupling, we cannot anticipate that another service provides the same layers as the consumer. Thus, passing the current layer composition to the service is inappropriate. Instead, the service should be informed about the context of the service call in an abstract way. The service can then decide how it varies its response according to this information.

An example for the latter case is `CreditCardTransactionService` can be called using *RSA* or *Elgamal* cryptography, see Figure 3. Which of them is used depends on context information of the consumer that is exposed to the service. The encryption context is passed as an additional parameter. The credit card service composes its layers according to this context information.

6 Summary and Next Steps

In this report, we motivate the explicit support of event-based behavioral variations in context-oriented programming language extensions. We discuss the benefits and some shortcomings of this implementation, which led us to the design of JCop, a COP-based language that adopts ideas of aspect-oriented programming for a more declarative

representation of event-based context. Furthermore, we report on the implementation of a service-based ticket-shop scenario using COP.

In future work, we will extend the event-based context specification presented in this report to support asynchronous context enter and exit predicates. In addition to control-flow- and event- specific context, we will develop a semantics of object-specific context that is relevant, for instance, in role-based scenarios. Furthermore, we will evaluate the concepts developed in the course of this thesis. As a part of this evaluation, we will extend our ticket shop scenario and develop a framework for context propagation over distribution boundaries.

References

[1] Malte Appeltauer, Robert Hirschfeld, Michael Haupt, Jens Lincke, and Michael Perscheid. A Comparison of Context-oriented Programming Languages. In *COP '09: International Workshop on Context-Oriented Programming*, pages 1–6, New York, NY, USA, 2009. ACM Press.

[2] Malte Appeltauer, Robert Hirschfeld, Michael Haupt, and Hidehiko Masuhara. ContextJ - Context-oriented Programming for Java. 2009. submitted.

[3] Malte Appeltauer, Robert Hirschfeld, and Hidehiko Masuhara. Improving the Development of Context-dependent Java Applications with ContextJ. In *COP '09: International Workshop on Context-Oriented Programming*, pages 1–5, New York, NY, USA, 2009. ACM Press.

[4] Malte Appeltauer, Robert Hirschfeld, and Tobias Rho. Dedicated Programming Support for Context-aware Ubiquitous Applications. In *UBICOMM 2008: Proceedings of the 2nd International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, pages 38–43, Washington, DC, USA, 2008. IEEE Computer Society Press.

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

[6] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-oriented Programming. In *Proceedings 11th European Conference on Object-Oriented Programming*, volume 1241, pages 220–242. Springer-Verlag, 1997.

[7] Michael Perscheid, David Tibbe, Martin Beck, Stefan Berger, Peter Osburg, Jeff Eastman, Michael Haupt, and Robert Hirschfeld. *An Introduction to Seaside*. Software Architecture Group, Hasso-Plattner-Institut, April 2008.

[8] W3C. Web service use case: Travel reservation, March 2002. http://www.w3.org/2002/06/ws-example.

Towards Service-Oriented, Standards-Based, Image-Based Provisioning, Interaction with and Styling of Geovirtual 3D Environments

Dieter Hildebrandt
dieter.hildebrandt@hpi.uni-potsdam.de

In this paper, I present an approach for the service-oriented, standards-based, image-based provisioning, interaction with and styling of 3D geovirtual environments (3DGeoVE). The proposed approach allows a human user to explore, interact with, and define the visual appearance of a remote 3DGeoVE through the Internet and by using lightweight clients such as web-based clients and Smartphones. The use of service-orientation and standards facilitates designing a distributed system that is interoperable and can easily be adapted to changing requirements. The image-based provisioning of visual representations allows for providing high-quality visual representations by dedicated services and consuming them on lightweight clients.

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 [13]. Massive amounts of distributed and heterogeneous geospatial information and geospatial computing functionality are increasingly available as distributed resources that can be accessed through the Internet. This increased availability has created the demand and feasibility to build distributed systems that leverage these resources for visualizing and interacting with geospatial information. For the implementation of such distributed systems, the application of the architectural concept *service-oriented architecture* (*SOA*) [9, 11] and standardization proposals by the *Open Geospatial Consortium* (*OGC* [1]) are commonly proposed. The primary potential of the application of the SOA paradigm in the geospatial domain is that it supports the uniform access, exploitation, integration and reuse of distributed geodata and geospatial functionality [8]. The application of the SOA paradigm for designing geovisualization systems implies the functional decomposition of the geovisualization process into reusable services accessible through a network. The OGC has approved various standards for service interfaces, data models and data encodings in the geospatial domain. For the presentation of information to humans, the OGC proposes portrayal services. For 2D portrayal the *Web Map Service* (*WMS*) [2] is proposed as an approved standard, whereas for 3D portrayal the *Web 3D Service* (*W3DS*) [16] and the *Web Perspective View Service* (*WPVS*) [20] are proposed as dif-

ferent approaches that are both still in the early stages of the standardization process.

In this paper, I present an approach for the service-oriented, standards-based, image-based provisioning, interaction with and styling of 3D geovirtual environments (3DGeoVE). The proposed approach allows a human user to explore, interact with, and define the visual appearance of a remote 3DGeoVE through the Internet and by using lightweight clients such as web-based clients and Smartphones. The use of service-orientation and standards facilitates designing a distributed system that is interoperable and can easily be adapted to changing requirements. The image-based provisioning of visual representations allows for providing high-quality visual representations by dedicated services and consuming them on lightweight clients. This paper is structured as follows. In Section 2, I present my *research questions* and give an overview of my *proposed approach*. In Section 3, I describe the WPVS++ service that is capable of *providing* service consumers with 2D images of perspective views of 3DGeoVE (e.g., virtual 3D city models). In Section 4, I present the WPVS++ client that offers facilities for human users to display 3D visual representations and *interact* with these representations. In Section 5, I present an approach for pre-rendering and post-rendering *styling*. Section 6 concludes with a summary, current status and future work.

2 Research Questions and Approach

2.1 Research Questions

This research focuses on the higher-level research question of how to apply service-oriented principles in the geospatial domain in an effective and value-adding way. More precisely, this research focuses on the question of which basic and high-level services are appropriate to enable and support the service-oriented 3D geovisualization of 3DGeoVE in an effective and value-adding way.

For this, my approach is to identify and investigate various research threads in the domain, to conceive and design new, concrete services in each thread that help solving open problems, and to implement exemplary, selected services for case studies.

In this paper, I focus on the thread of *server-side, service-oriented rendering*. For this thread, the research question that I investigate is as follows: How do we implement and expose the functionality of

- providing,
- interacting with, and
- styling (i.e., what and how to portray)

visual representations of 3DGeoVE within a geospatial, service-oriented architecture while meeting the following requirements:

- Access to remote, massive virtual 3D city models
- High quality visual representations

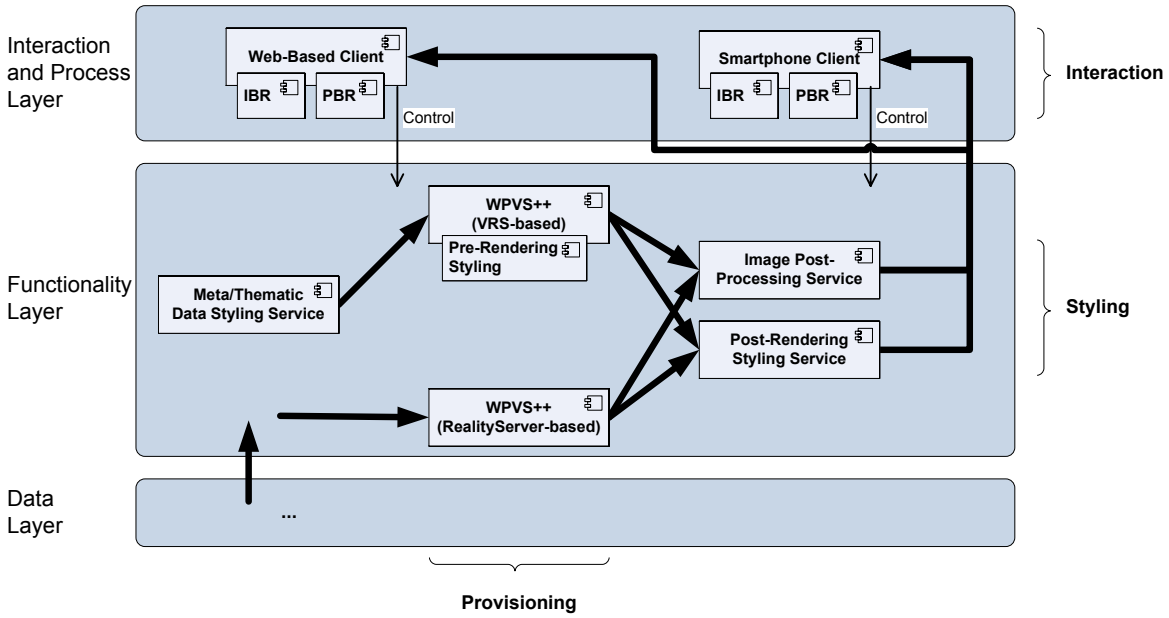


Figure 1: Overview of the service-oriented, standards-based architecture of the proposed approach. Bold arrows indicate the major data flow. Thin arrows indicate flow of control data.

- High degree of interactivity
- High degree of control over what and how to portray
- Low requirements on end user resources
- Moderate requirements on service provider
- Standards-based

2.2 Approach

In this Subsection, I present an overview of a preliminary approach for the service-based, image-based provisioning of, interaction with and styling of visual representations of 3DGeoVE. This approach aims at answering aspects of the research question stated in Section 2.1.

Figure 1 gives an overview of the architecture of the proposed approach. The *Meta/Thematic Data Styling Service* receives as input a reference to a model of a 3DGeoVE, and a high-level description of what metadata and/or thematic data is to be styled in what way within the 3DGeoVE. The service outputs a styling description document that is based on the *Styled Layered Descriptor* (*SLD*) [12]. The WPVS++ service (VRS-based implementation [3])) receives as input a virtual camera specification and SLD document and generates a 2D image of a 3D perspective view of a 3DGeoVE. The SLD document specifies what geodata contained within the WPVS++ and what additional geodata drawn from external sources is to be portrayed in what way. The WPVS++ service based on the RealityServer [14] does not support SLD documents.

As a commercial, closed-source, third-party product, the RealityServer does not support SLD-based styling as a feature and cannot be extended to support SLD documents by external developers. The *Image Post-Processing Service* (*IPPS*) receives 2D images and a processing specification as input, applies image post-processing filters to the images (e.g., non-photorealistic rendering effects) and outputs 2D images. The *Post-Rendering Styling Service* (*PRSS*) service receives 2D images and a post-render styling (PRS) specification as input. The PRS specification defines what features of the 3DGeoVE are portrayed in what way similar to a SLD document. The difference is that SLD specifications are evaluated and applied in the mapping stage of the geovisualization pipeline [5], *before* the rendering stage. In contrast, processing specifications for the PRSS are evaluated and applied *after* the rendering stage. Thus, the styling is performed on already-rendered images. The *Web-Based Client* is operating inside a web browser and requests 2D images of the 3DGeoVE from the IPPS and/or the PRSS. The client allows a user to explore, interact with, and define the visual appearance (i.e., styling) of visual representations of 3DGeoVE. To control what is portrayed in what way, the client calls services from the functionality layer and includes information such as the position and viewing direction of a virtual camera, a high-level styling description for metadata and/or thematic data, a SLD document, and a PRS document. The *Smartphone Client* is a port of the Web-Based Client with a similar feature set that operates on a Smartphone (e.g., the Apple iPhone). Both the Web-Based Client and the Smartphone Client support two different techniques for reconstructing the remote 3DGeoVE locally on the client. The first is based on *image-based rendering* (*IBR*) [19], and the second on *point-based rendering* (*PBR*) [10].

In the following Sections, I briefly discuss the proposed concepts and services. Since this paper focuses on the research thread of server-side, service-oriented rendering of complex 3D models, I do not discuss the Meta/Thematic Data Styling Service.

3 Provisioning: WPVS++ Service

The *Web Perspective View Service* (*WPVS*) [20] generates 2D images of perspective views of 3DGeoVE (e.g., virtual 3D city models) and represents one fundamental class of portrayal services. As key advantage, this image-based approach can be deployed across arbitrary networks due to server-side 3D rendering and 3D model management. However, restricted visualization and interaction capabilities of WPVS-based applications represent its main weaknesses. To overcome these limitations, we propose the WPVS++ [6], a WPVS extension, which provides a) additional thematic information layers for generated images and b) additional service operations for requesting spatial and thematic information. Based on these functional extensions, WPVS++ clients can implement various 3D visualization and interaction features without changing the underlying working principle, which leads to an increased degree of interactivity and is demonstrated by prototypic web-based client applications. A WPVS++ client can retrieve thematic information layers storing information such as color, depth, object id, surface normal and mask encoded in standard 2D image formats for a specified virtual camera location, orientation and perspective or orthographic projection (Figure 2). This concept is based on the G-buffer [17] concept from 3D computer graphics.

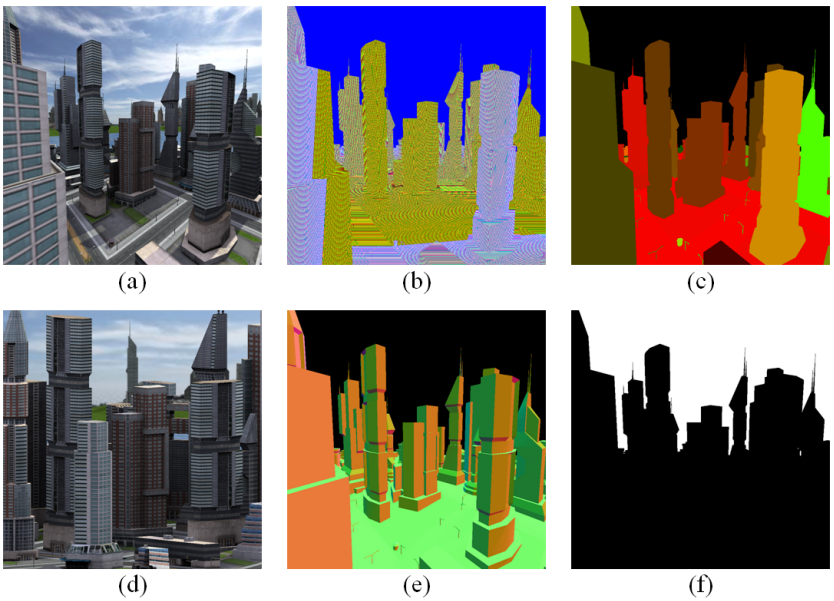


Figure 2: Examples of image layers that are provided by WPVS++: (a) color layer, (b) depth layer, (c) object id layer, (e) normal layer, and (f) mask layer. Subfigure (d) depicts an orthographic projection for the same location of the virtual camera and point-of-interest. [6]

4 Interaction: WPVS++ Clients

4.1 Image-Based Rendering Client

To enable a user to explore and interact with a remote, massive 3DGeoVE stored and rendered by the WPVS++ through the Internet, I developed a first concept and prototype implementation for a lightweight, web-based client application that utilizes the WPVS++. Figure 3 depicts a screenshot of the implemented client application running inside a web browser. Figure 4 depicts a screenshot of the client application ported to the Smartphone Apple iPhone.

The underlying concept of the WPVS++ client application uses ideas from the domains of image-based rendering [19] and smart navigation. The client requests cube maps for user specified virtual camera locations in several information dimensions (e.g., color, depth, object id) from the WPVS++ (Figure 5). The cube maps are transferred as image sequences to the client. The client performs an image-based reconstruction of the 3D environment from the cube maps. The reconstruction is then used by the client to enable a user to interact with and navigate in the 3D environment. The client supports common direct navigation techniques (e.g., rotate virtual camera, pan, zoom, and orbit). To render novel views from arbitrary virtual camera view points, a 3D point cloud is constructed from the cube map and rendered with point or triangle primitives. Smart navigation concepts are employed where the user states navigation intentions (e.g., “I want to inspect this building”) and lets the client application create and execute plans to satisfy the user’s intention. Smart navigation concepts are employed for two reasons. First, as a hypothesis, they allow for more efficient and effective navigation in a wide range of applications than direct navigation techniques. Second, they allow masking the latency of transmitting data from the WPVS++ to the client, by pro-

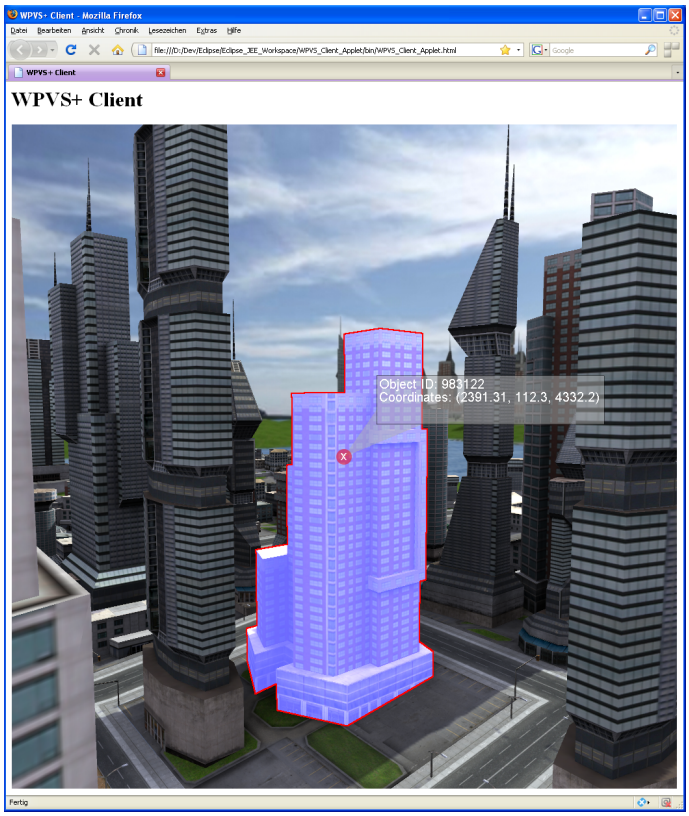


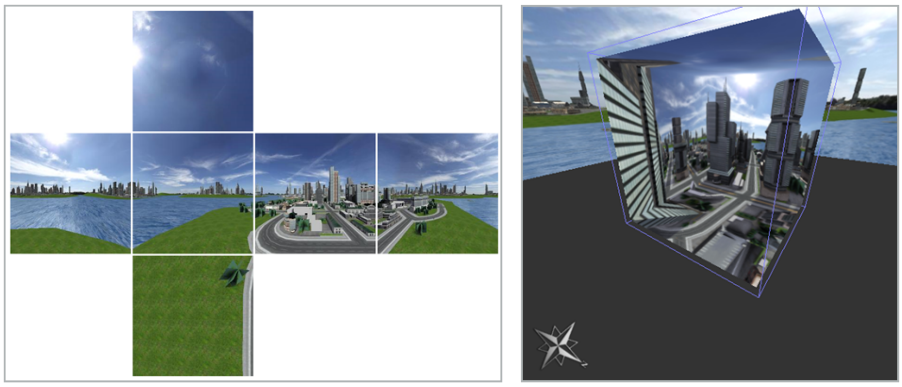
Figure 3: Screenshot of the web-based WPVS++ client application running inside a web browser. An object feature that was selected by the user is highlighted and annotated with additional information.

gressively transferring new cube maps in the background while the client application executes navigation tasks or the user interacts with the environment. The client application supports additional features including field of view-based zoom with selective refinement of the cube map, selection and highlighting of object features (e.g., buildings) within the 3D virtual environment, retrieving and displaying additional thematic information for object features from the WPVS++, and measuring Euclidean distance between arbitrary spatial positions.

4.2 Point-Based Rendering Client

The PBR client differs in the way that the 3DGeoVE is reconstructed and rendered locally from 2D images received from a remote image-based service. Image layers containing spatial depth information can be interpreted as a 3D representation of the 3DGeoVE. Each pixel can be interpreted as a patch in 3D space that covers a part of the visible surface of a 3DGeoVE. Furthermore, these patches can be interpreted as simplified 3D points, surfels [15] or voxels. These elements are then added to a hierarchical spatial data structure (e.g., an octree). The spatial information can be complemented with additional attributes such as color, object id, and normal if the respective image layers are retrieved. The spatial data structure is used for aggregating data, storing data in multi-resolutions, querying data, and rendering data. Data received from multiple calls to image-based services can be integrated in a unified manner. Multiple-

Figure 4: Screenshot of the web-based WPVS++ client application running on the Smartphone Apple iPhone.



(a) (b)

Figure 5: Cube maps as primitives for image-based modeling and rendering. (a) Six faces of a cube map as requested by the client from the WPVS++ service. (b) The cube map assembled and rendered by the client. When the virtual camera is placed in the center of the cube and the virtual camera is rotated, the user experiences a perfect illusion of being in the 3D virtual environment.

resolutions can be stored at different levels of the data structure, whereby each node stores a coarser representation of its child nodes. The data structure supports querying spatial data in logarithmic complexity.

Furthermore, the data structure supports rendering the multi-resolution representation with view-frustum culling, level-of-detail control, and control of a trade-off between performance and quality. A rendering algorithm could traverse the data structure. When reaching a leaf node, the node is rendered. An inner node is rendered, as soon as its projected screen space area falls below a given threshold, e.g., one pixel. Nodes can be rendered with a technique called “splatting”.

I assume that in practical cases the amount of data needed for visualizing 3DGeoVEs exceeds the main memory capacities of average computers that are used for executing the client application. For this, I propose to utilize appropriate out-of-core techniques [4] for implementing the client application. A general strategy could be to first retrieve the data in form of images from the service and dynamically add the data to a spatial data structure (e.g., octree). Then, when the amount of the locally accumulated data

exceeds the main memory capacity, parts of the spatial data structure are stored on the local hard disk and are removed from main memory. Subsequently, when data is needed that does not reside on main memory but on the hard disk, it is retrieved from that location instead of from the remote service. For improved efficiency, when adding 3D patches from the retrieved images to the spatial data structure, the spatial coherence of adjacent pixels in the image can be exploited. Pixels that are close in the image are very likely mapped to spatially close parts of the spatial data structure. Furthermore, an image can be interpreted as a multi-resolution representation of a 3DGeoVE. Each pixel in the image covers an area of the visible surface of the 3DGeoVE that is different in size. Generally, the farther an associated depth value for a given pixel is, the bigger the covered world space area is. This fact can be exploited for directly mapping pixels representing 3D patches to specific nodes of the spatial data structure. In order to reduce latency, the tasks of retrieving data from a service, decoding and encoding the retrieved data (e.g., images), and uploading data to the memory of the GPU, should be parallelized.

In this context, the PBR client has the potential to produce visual representations more efficiently and with higher quality than the IBR client. However, research needs to be carried out to test this hypothesis and to explore the respective advantages and disadvantages of the IBR client in comparison to the PBR client.

5 Styling

5.1 Image Post-Processing Service

The server-side, service-oriented rendering of massive 3DGeoVE has the potential to enable users with lightweight clients to access high-quality, image-based visual representations of the environments without having to download massive amounts of geodata. Image post-processing is a well known concept that allows for decoupling the process of generating an image from applying enhancement processes to already generated images. Commonly applied image post-processes are for example image compositing, or high dynamic range filters. Moreover, several visual effects are implemented most efficiently and effectively as image post-processes, e.g., non-photorealistic rendering effects. In a SOA, providing the functionality of image post-processors as dedicated, loosely coupled services as an application of the principle of separation of concerns offers several advantages. These services easily can be reused and recomposed with further services to form different distributed applications, the increased modularity has the potential to improve the maintainability and flexibility of the resulting systems, and existing applications and services can be extended to take advantage of image post-processing functionality without having to implement the functionality themselves. However, so far no proposals for providing the functionality of image post-processors as dedicated services exist.

For this, I propose a concept and implementation for the provisioning of image post-processing functionality as dedicated, loosely coupled Image Post-Processing Services (IPPS). I place a particular focus on processing images that are enriched with spatial depth information and that are generated by image-based 3D portrayal services portraying virtual 3D city models. I propose an interface for the IPPS that is based on the

Web Processing Service (*WPS*) [18] standard by the OGC. As a proof of concept, I plan to implement four exemplary image post-processors (screen-space ambient occlusion, non-photorealistic rendering, image integration and compositing, and projective texturing). For each processor I plan to carefully choose an algorithm from the literature that best meets the requirements of our specific application. I plan to implement each processor in a GPU-based and a server deployment-friendly CPU-based version and to evaluate and compare the performance of the implementations. The utility and characteristics of the IPPS will be evaluated by extending our service-oriented, image-based remote visualization system with the functionality of the IPPS. Furthermore, I plan to evaluate the performance overhead of interposing an IPPS between a portrayal 3D service and the portrayal service consumer.

5.2 Post-Rendering Styling

The *Post-Rendering Styling Service* (*PRSS*) service receives 2D images and a post-render styling (PRS) specification as input. The PRS specification defines what features of the 3DGeoVE are portrayed in what way similar to a SLD document. The difference is that SLD specifications are evaluated and applied in the mapping stage of the geovisualization pipeline [5], *before* the rendering stage. In contrast, processing specifications for the PRSS are evaluated and applied *after* the rendering stage. Thus, the styling is performed on already-rendered images.

In comparison to the pre-rendering styling, the post-rendering styling has some advantages. The post-rendering styling can be applied to any 2D color image (when required complemented with a depth image) regardless of the source. Image-generating services such as the WPVS++ do not have to be equipped each with styling functionality. Implementing such functionality may be very complex and even unfeasible as is the case with the WPVS++ based on the RealityServer as described in Section 2.2. This advantage can be attributed to the fact that the functionality is moved within the geovisualization pipeline to a latter place. Implementing this functionality within a client has a similar effect but has the disadvantage of not offering the functionality as a reusable service. As a disadvantage, the post-rendering styling is estimated to be less powerful than the pre-rendering styling since at this stage in the pipeline less information might be available.

5.3 Pre-Rendering Styling

As already discussed in the previous Section, pre-rendering styling must be available in order to facilitate more powerful styling. Pre-rendering styling can be implemented as part of a WPVS++ and controlled by passing SLD documents. However, the current SLD specification only supports 2D portrayal. Extensions of this specification to 3D exist (e.g., [7]) but are still limited. There is still need to research how effective styling can be specified for visual representations of 3DGeoVEs and how it can be implemented efficiently.

6 Summary, Current Status, and Future Work

In this paper, I presented an approach for the service-oriented, standards-based, image-based provisioning, interaction with and styling of visual representations of 3DGeoVE. I presented the WPVS++ service for the provisioning, the WPVS++ clients for display and interaction. Furthermore, I presented the concept of pre-rendering styling that can be implemented in the WPVS++, and the concept of the post-rendering styling that can be implemented in the IPPS and PRSS.

Currently, the work on all variations of the WPVS++ clients (i.e. IBR and PBR, web-based and Smartphone-based) is in progress. The web-based IBR client is developed the farthest. Work on IPPS as preparative study for the PRSS is in progress and a publication is being prepared. The basic WPVS++ is the contribution of Benjamin Hagedorn and is in a deployable state.

After finishing the work in progress, the next step is to further develop, implement and investigate the concept of post-render styling. At the same time, a WPVS++ implementation based on the RealityServer will be pursued. Finally, I plan to investigate the pre-rendering styling, its implementation as part of a WPVS++, and the concept and implementation of a Meta/Thematic Data Styling Service that produces inputs for a WPVS++.

References

[1] Open Geospatial Consortium. Open Geospatial Consortium (OGC) Website. URL, `http://www.opengeospatial.org/`, 2009. Accessed 28.8.2009.

[2] Jeff de la Beaujardiere, editor. *OpenGIS Web Map Server Implementation Specification, Version 1.3.0*. Open Geospatial Consortium Inc., March 2006.

[3] Jürgen Döllner and Klaus Hinrichs. A Generic Rendering System. *IEEE Transactions on Visualization and Computer Graphics*, 8(2):99–118, April 2002.

[4] Enrico Gobbetti, Dave Kasik, and Sung-eui Yoon. Technical Strategies for Massive Model Visualization. In *SPM '08: Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pages 405–415, New York, NY, USA, 2008. ACM.

[5] R.B. Haber and D. A. McNabb. Visualization Idioms: A Conceptual Model for Scientific Visualization Systems. In B. Shriver, G. M. Nielson, and L. Rosenblum, editors, *Visualization in Scientific Computing*, pages 74–93, Los Alamitos, 1990. IEEE Computer Society Press.

[6] Benjamin Hagedorn, Dieter Hildebrandt, and Jürgen Döllner. Towards Advanced and Interactive Web Perspective View Services. In *4th International 3D GeoInfo Workshop*, Lecture Notes in Geoinformation and Cartography. Springer, November 2009.

[7] 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*, pages 121–131, October 2007.

[8] Dieter Hildebrandt and Jürgen Döllner. Service-Oriented, Standards-Based 3D Geovisualization: Potential and Challenges. *Journal on Computers, Environment and Urban Systems, special issue on GeoVisualization and the Digital City*, 2010. Invited submission.

[9] Dieter Hildebrandt, Oliver Holschke, Philipp Offermann, and Ulrike Steffens. Entwurf serviceorientierter Architekturen. In Ralf Reussner and Wilhelm Hasselbring, editors, *Handbuch der Software-Architektur*, pages 123–149. dpunkt Verlag, January 2009.

[10] Leif Kobbelt and Mario Botsch. A Survey of Point-Based Techniques in Computer Graphics. *Computers & Graphics*, 28(6):801–814, 2004.

[11] Dirk Krafzig, Karl Banke, and Dirk Slama. *Enterprise SOA: Service-Oriented Architecture Best Practices*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[12] Markus Lupp, editor. *Styled Layer Descriptor Profile of the Web Map Service Implementation Specification, Version 1.1.0*. Open Geospatial Consortium Inc., June 2007.

[13] Alan M. MacEachren and Menno-Jan Kraak. Research Challenges in Geovisualization. *Cartography and Geographic Information Science*, 28(1):3–12, 2001.

[14] mental images. Reality Server – 3D Web Services Application Platform. URL, `http://www.mentalimages.com/products/realityserver.html`, 2009. Accessed 13.9.2009.

[15] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus H. Gross. Surfels: Surface Elements as Rendering Primitives. In *SIGGRAPH*, pages 335–342, 2000.

[16] Udo Quadt and Thomas H. Kolbe, editors. *Web 3D Service, Version 0.3.0*. Open Geospatial Consortium Inc., February 2005.

[17] Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3-D shapes. *SIGGRAPH Computer Graphics*, 24(4):197–206, 1990.

[18] Peter Schut, editor. *OpenGIS Web Processing Service, Version 1.0.0*. Open Geospatial Consortium Inc., June 2007.

[19] Heung-Yeung Shum, Shing-Chow Chan, and Sing Bing Kang. *Image-Based Rendering*. Springer, 2007.

[20] Raj R. Singh, editor. *OGC Web Terrain Server, Version 0.3.2*. Open Geospatial Consortium Inc., August 2001.

Reliable Digital Identities for SOA and the Web

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

This report summarizes my latest research activities towards a more effective and trustworthy management of digital identities in service-oriented architectures and the Internet, and draws an outline of planned research activities for the upcoming year. While my former work was focussed on theoretical concepts and ideas, during the last months I put emphasis on the implementation of a small prototype scenario which makes use of my latest research findings.

As a basis for my research, I consider open identity management models with a particular focus on the claim-based identity management. Claim-based identity management allows services and applications to specify their identity data requirements as a list of attributes (claims). In my research, I argue that for a relying party (such as a service) to assess the value of received identity information in terms of correctness and integrity, trust should be defined on the same granular level as the required identity data. As a solution to integrate such trust-related information, I proposed a data model which adds identity meta data to the notion of claims and allows identity providers and relying parties to distinguish different qualities of the same identity attribute. This report presents this model as well as its implementation in a small demo use case.

1 Introduction

Looking at the current online world, performing transactions as online banking, online shopping or communicating in social networks has become an inherent part of life. Hereby, personal, identity-related data plays a major role, since for many activities a service provider requires details about the identity of a user. Traditional approaches for identity management like the isolated model (cf. [2]) require users to register with every single service and to re-authenticate each time they use a service in another trust domain. The reasons for the pre-dominance of the isolated model are comprehensible. Isolation allows organizations to retain control over their identity management systems. As organizations usually have different legal and technical requirements for identity management, they find it difficult to give up this control. However, with regard to the Internet, we can find many identity attributes which do not require strong verification. It really depends on what a digital identity is used for. If the user logs on to a site to prove on repeat visits that it is the same user, it does not matter whether his digital identity matches with his ”real-life identity” as long as it is always the same digital identity he uses to log on. Only if critical transactions are performed, as ordering an item or paying

for a service, the integrity of provided user data is required to hold the user liable in case anything bad happens.

Therefore, an identity management for the Internet should be able to deal with attributes with a strong verification beneath attributes managed by the user himself. In addition, it should consider who (i.e. which identity provider) can assert a certain claim based on a strong verification and which identity provider issues a certain claim which is ”only” managed by the user himself.

While claim-based identity management allows applications to specify required claims on a per-attribute basis, trust is still mostly established as a whole between two partners. In order to close this gap, I extend the notion of a claim by a *credibility value* indicating the assumed correctness or integrity of a claim in dependence of its issuer. To be specific, this report

- presents a data model as an extension to the claim-based approach which
  1. allows an application or service to define the required credibility for a claim and
  2. allows an application or service to state whom to trust on a per-claim basis
- describes a prototype implementation making use of the model to decide based on a claim, the required credibility of the claim and the issuer of the claim whether to trust a received claim value or not.

2 Motivating Use Case

2.1 Open Identity Management Models

In order to address the open and decentralized nature of the Internet, open identity management models emerged, which are based on the idea of having several places to manage a users identity data (so called identity providers). Claim-based identity management denotes such an open identity model which uses the notion of claims to describe identity attributes. A claim is an identity attribute named with an abstract identifier (e.g. a URI), which applications and services can use to specify the attributes they need. Open and extensible formats for the exchange of identity attributes ensure interoperability among different identity systems. For this reason, claim-based identity management lays the ground for Identity Metasystems [5], which provide an identity layer on top of existing identity systems and promise an easier management of digital identities among the Internet.

2.2 Example

Basically, we can make two observations with regard to the storage and administration of identity information on the Internet, today. The first observation is that basically every service provider on the Internet manages information which is specific to its domain, namely the information which was created during the interaction with a customer

and the system, such as a customer number. A second observation is that information stored in independent domains is often redundant, because certain pieces of a subject's identity are required by every service or Web site provider. Examples include: the name and address of a person or its birthday.

Hence, basically every service or Web site provider has identity information, i.e. information about its user's digital identities, which he could provide to other participants (given the user's consent) and basically every service or Web site provider also consumes certain information which it requests from the user and which it does not necessarily need to manage itself. A possible solution towards a more effective management of identity information is demonstrated in figure 1. Instead of entering the same information into different user accounts, the user could reference to another account which already contains this information. For example, the newspaper publisher would receive the assertion that its customer is a student directly from the users university and the information about the user's banking account information directly from the bank.

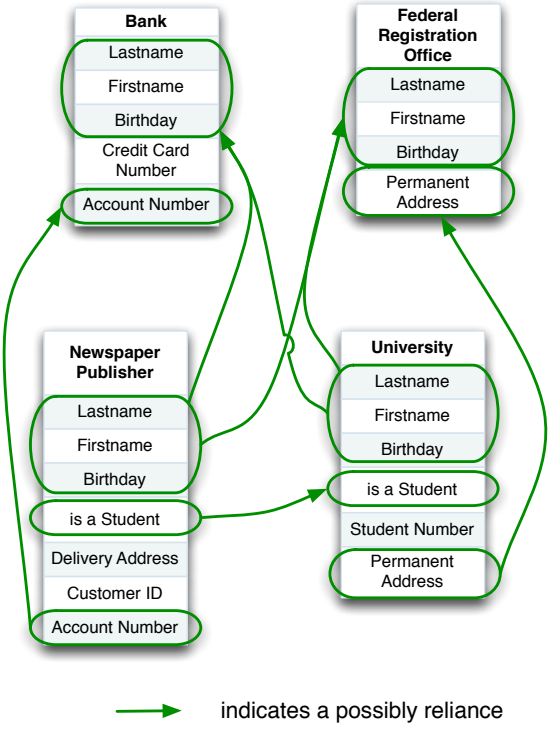


Figure 1: Usecase showing independent identity domains and possible reliance on other identity providers

3 Metadata for the Credibility of a Claim

This section presents my approach for a trust-aware claim-based identity management. Starting with a short requirements analysis to point out which additional concepts are necessary to implement a scenario as described in section 2.2, this sections presents my solution consisting of a data model to store additional trust-related information about claims and a proposal to fill and use the model.

3.1 Requirements analysis

Claims are subject to verification by an Identity Provider prior to using them. However, verification processes can differ tremendously in their strength and reliability of the result. Often users can enter any information to a registration form when registering for a new service. Sometimes little checks are performed as for example the sending of an email to verify that an entered email address is valid. Only when it comes to more critical transactions as paying for a service, stronger verification processes are

employed such as checking with the customers bank whether an entered credit card is valid.

Obviously, it is not necessary for applications to have all required user attributes verified in the most secure way. In fact, only attributes which involve a risk for the Relying Party require further verification.

Summing up, an identity management system needs to capture that *(a)* user attributes differ in their strength of verification and *(b)* applications have different requirements concerning the strength of verification of identity attributes.

In a closed environment, these requirements are usually considered when setting up a system and reflected in the decision which technology is used for identification and authentication of users. In a decentralized environment as service-oriented architectures or the Internet, however, this knowledge needs to become explicit.

Therefore, using claim-based identity management, we require a notion to express the following information:

Requirement 1 In addition to stating, which identity attributes a Relying Party requires (= claims), it is required to express also the quality in terms of validity of an asserted claim value, that a Relying Party requires.

Requirement 2 In addition to the information which Identity Provider is trusted, it should also be known for which claims. To put is simple: Deciding not only whom to trust, but for what.

3.2 Proposed Data Model

This section introduces the data model to fulfill the requirements identified before. It builds upon the notion of claims and the entities involved in the Identity Metasystem [5]. The data model presented as an ER-diagram is shown in figure 2.

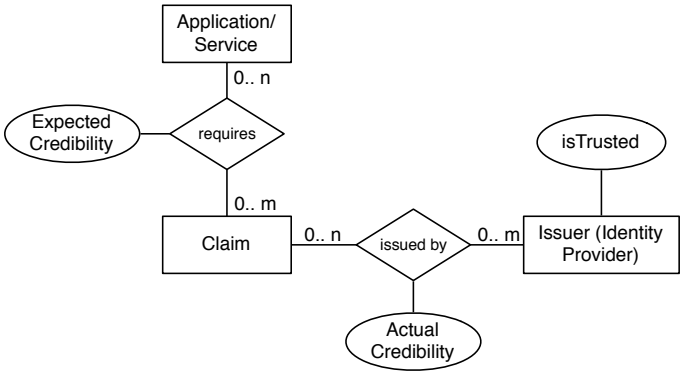


Figure 2: Proposed Data Model to enhance the Notion of Claims by a Credibility Value.

Looking at current implementations of the Identity Metasystem, a Relying Party usually specifies a list of Identity Provider it trusts to make right assertions. Using

the notion of claims, the Relying Party can express for a list of claims the issuer(s) it will accept tokens from. When receiving a security token, the Relying Party verifies the issuer of the token by checking whether the signature of the token matches the certificate of one of his trusted Identity Providers. Only upon correct verification, the Relying Party will process the claims in the token.

It is important to note that in spite of giving up control over the management of a user's identity attributes, the decision how to proceed with received identity information is at the entity which is in control of the requested resources. It is in the responsibility of the Relying Party to decide whom to trust and from whom to accept security tokens.

In our model, we aim at defining this trust in a more fine-granular way. In addition to state whom to trust, we want to be more specific and state on a per-claim basis the believe that a certain claim issued by a certain issuer is correct. Therefore, we introduce the notion of the *credibility* of a claim. The credibility of a claim is defined as the expected or assumed integrity of a claim. It reflects the belief that a certain claim value is correct and corresponds with the ”real-life identity” of a user.

Remark: Even though the credibility of a claim depends a lot on trust and trustworthy behaviour of the party asserting a claim, we use the term *credibility* in order to put emphasis on the integrity of the claim value rather than the process of trusting that the other party makes right assertions.

In our model, we can distinguish two different kinds of credibility values for a claim, that is *(a)* the credibility value an applications expects (*Expected credibility*) and *(b)* the actual credibility value the Relying Party assigns to a claim (*Actual credibility*).

The expected credibility basically is the extension we need to solve Requirement 1, while the second requirement is fulfilled by adding the actual credibility to the model.

3.3 Using the data model

Since the usability of a data model rises and falls with the values stored in it, this section discusses how one can fill the model with reasonable data. Emphasis is put on finding values for the credibility of a claim.

Identity Provider Metadata Usually, an Identity Provider provides an endpoint address under which a requesting party can retrieve metadata about this Identity Provider. Such metadata includes a list of claims this Identity Provider can assert for its users. In our model, the Relying Party stores for each Identity Provider, it communicates with

- the list of supported claims which it can get from this endpoint (e.g. by using the WS-MetadataExchange protocol)
- and whether the Identity Provider is trustworthy. This value refers to the overall quality of the trust relationship between the Relying Party and the Identity Provider. Factors as past experience, the reputation of an Identity Provider, whether the Identity Provider is managed by the government, a research or a commercial organization are important criteria to assess its trustworthiness. (cf. the notion of Organizational Trust introduced in [4])

The credibility of a claim Choosing the right settings for the credibility of a claim is by no means easy. With regard to current online systems, it seems reasonable to distinguish two different qualities of user attributes, that is *(a)* low-value information entered by the user and remaining unverified and *(b)* critical information which has been verified by the Identity Provider. In this case, we would have a binary credibility value. In order to put this observation in a more formal context, we define the following relations. *isTrusted* is a function which returns for a given issuer whether it is on the list of trusted issuers. (cf. Identity Provider Metadata)

$$\textit{isTrusted} : \textit{Issuer} \mapsto \{\textit{trusted}, \textit{untrusted}\}$$

In addition, we define *isVerified* to be a function which returns whether an identity attribute/claim was verified by the Identity Provider.

$$\textit{isVerified} : (\textit{Issuer}, \textit{Claim}) \mapsto \{\textit{verified}, \textit{unverified}\}$$

To derive the credibility, we combine both results. The way in which both results are combined shall be defined by a function h, which can be application-specific or globally defined. The function h describes, in which way the fact whether an identity attribute has been verified is combined with the fact whether this has been done by a trusted Identity Provider. To follow our observation, we would define the credibility of a claim to be 1 only if the claim was verified and issued by a trusted issuer. In all other cases, it is 0. A mathematical definition for h is given below.

$$\begin{aligned}\textit{credibility}(\textit{issuer}, \textit{claim}) \mapsto & h(\textit{isTrusted}(\textit{issuer}), \\ & \textit{isVerified}(\textit{issuer}, \textit{claim}))\end{aligned}$$

with h e.g. defined as

$$h : \{\textit{trusted}, \textit{untrusted}\} \times \{\textit{verified}, \textit{unverified}\} \mapsto \{1, 0\}$$

$$h : (b_1, b_2) \mapsto \begin{cases} 1, & \text{if } b_1 = \textit{trusted} \text{ and } b_2 = \textit{verified} \\ 0, & otherwise \end{cases}$$

Regarding the cases definition of h, the case of having a verified claim asserted by an issuer that we do not know and which is therefore untrusted, seems interesting. Having such a situation, we could imagine an application which is willing to rely on such an assertion given that it gets the same assertion from multiple untrusted sources. For example, a user could prove that he is over 18 by providing assertions from two different online stores he has purchased goods recently, which are not known by the third online store, from which he is going to buy a movie which is rated ”Restricted”. In order to reflect such scenarios, another possible definition for the function h could be as follows:

$$h : \{\textit{trusted}, \textit{untrusted}\} \times \{\textit{verified}, \textit{unverified}\} \mapsto \{2, 1, 0\}$$

$$h : (b_1, b_2) \mapsto \begin{cases} 2, & \text{if } b_1 = \textit{trusted} \text{ and } b_2 = \textit{verified} \\ 1, & \text{if } b_1 = \textit{untrusted} \text{ and } b_2 = \textit{verified} \\ 0, & \text{otherwise} \end{cases}$$

Extensions Depending on the needs of an application, our model can be extended in several ways. One possible extension is to adapt the definition of the function *isTrusted* by enlarging their range of values. This way, we could express several qualities of trust into an Identity Provider, for example to rate a Business-to-Business (B2B) relationship differently than a Business-to-Consumer relationship.

Another suitable extension concerns the verification of an identity attribute. While the function *isVerified* distinguishes only two states (*unverified* and *verified*), a more fine-granular differentiation of verification states might also be an option. For example, taking the quality of the user authentication into account might be desirable to assert how much trust a Relying Party can put into the identity of a user. With the SAML Authentication Context Classes [6], a standard to convey this authentication meta information between an Identity Provider and a Relying Party already exists and can be used to substantiate the verification state of identity attributes. Also, previous research work exists [7], which describes a suitable mapping between the quality of an authentication process and a trust level.

4 Prototype implementation

This section presents the prototype which makes use of the model, and describes the underlying architecture.

4.1 Use Case Scenario

Our demo application is a classical Web site for an online store selling music files which is shown in figure 4 (appendix). In order to populate our demo online store, we use real data retrieved by calling the Amazon web services [1]. To complete the purchase, several personal attributes are requested from the user, such as his name, address and payment information (cf. screenshot in figure 5). The Web application defines for each attribute or claim requirements regarding the credibility, e.g. the address is rated as low-value information (corresponds to level 0 in our model in section 3.3), while the name and payment information is rated as highly important in terms of validity and integrity.

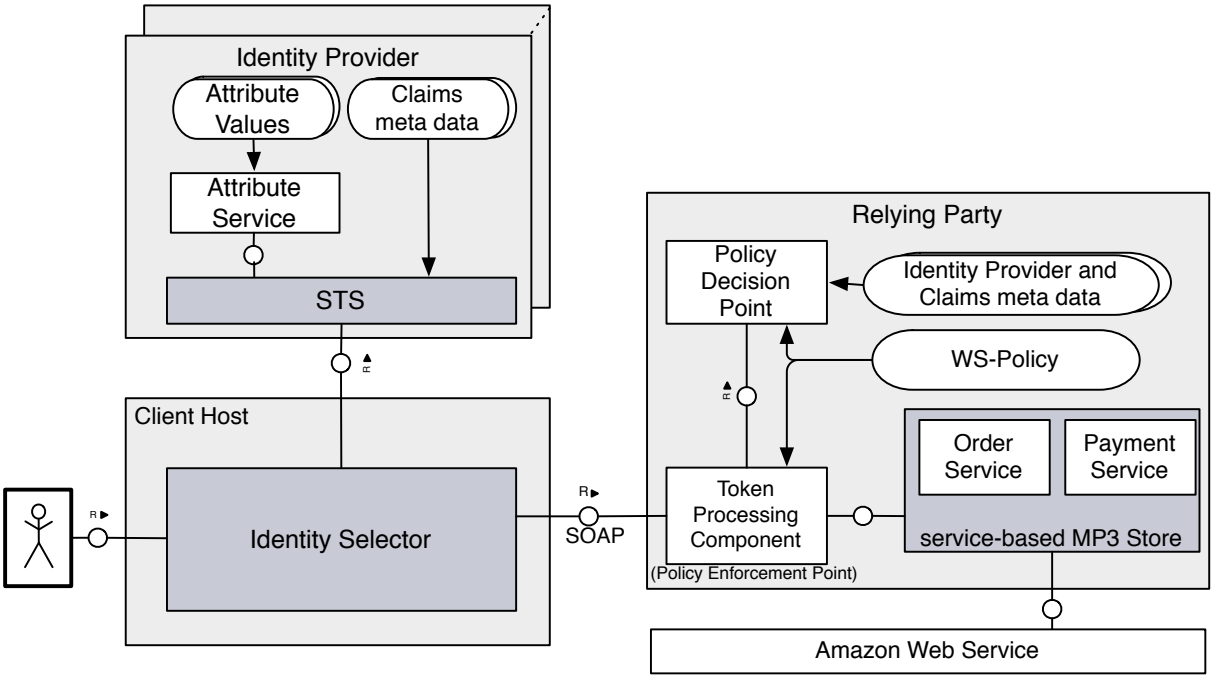


Figure 3: Architecture based on the Identity Metasystem to implement our approach for a trust-aware claim-based identity management

4.2 Architecture

In order to implement this scenario, we implemented and extended the classical architecture of an Identity Metasystem as described in [5]. Our architecture comprises the main entities involved in such a metasystem: Identity Provider, Relying Party, Identity Selector and the User.

Figure 3 shows the overall architecture. As can be seen from the figure, the user is dealing with a number of Identity Providers of his choice. When the user accesses a service, the user can ask an Identity Provider of his choice to issue a security token. Once triggered, e.g. by clicking on the CardSpace icon (cf. screenshot in figure 5), the Identity Selector on the user's machine pops up to support the user in requesting and returning an appropriate token. In a first step, it will find out the policy of the Relying Party by sending a request for meta data using the WS-MetadataExchange protocol. By analyzing the meta data document, the Identity Selector gets to know the required claims and accepted token types of the Relying Party. It can align this information with the Identity Providers the user is registered with and presents the user with a choice of possible Identity Providers. The user selects an Identity Provider and a request is performed by sending a special kind of message, which is described by the WS-Trust standard, called a Request for Security Token (RST). If everything went well, the Identity Provider will answer with another WS-Trust-defined message, called Request for Security Token Response, which contains the requested claim values in the requested token format. This token can be used by the user to deliver the requested attribute values to the application. Upon receipt, the Relying Party extracts the token,

and analyses its validity. In order to decide whether the information in the token is trusted, the Relying Party applies its trust model to the received token.

5 Future Work and Next Steps

Based on the model and the implementation done so far, I am planning to investigate into the following directions for further research.

5.1 Facilitating Identity Metadata in Identity Federations

Identity Federation is a concept for the controlled sharing of identity information based on a *Circle of Trust* which is established beforehand between trusted parties. Identity federation enables the propagation of identity information including the authentication of users to services located in different trust domains. If a user has accounts at two or more of the trusted parties, he can decide to connect (”federate”) these accounts. This enables him to log in at one identity provider and retrieve identity information located at a federated partner. In [3] we defined the general trust requirements to establish such a federation and proposed extensions to federation metadata in order to take different trust requirements into account when establishing such a federation. Building on this work, we can further enhance identity federations with identity meta data to allow for a more flexible and reliable propagation of identity information across domains. If a relying party such as a service or web application requires identity information which the users has stored at different identity providers such as his bank and the university, he would need to authenticate with each identity provider separately. Using identity federation, scenarios are possible in which the user authenticates only with one provider and retrieves further identity information facilitating the identity federation. To allow such scenarios, it is necessary to define the trust requirements of the involved parties and put them into context with the risk of the transaction to be carried out. Here, information about the quality of exchanged identity information is essential to decide whether the relying party can rely on the received information.

5.2 User-Control and Attribute Release Policies

To put the user into full control of his identity data is essential to achieve his acceptance for any identity management system. In the context of the claim-based identity management, this is achieved by requesting a so called digital information card from the user, which comprises the identity attributes that he wants to share, each time when a user accesses a service. Instead of requesting a decision from the user each time, a possibly extensions towards a more effective identity management could be the introduction of attribute release policies based on the quality of identity attributes. For example, a user could define for a relying party and a certain attribute and its quality that this attribute value is released without further questioning.

6 Conclusion

Past experiences have shown that there would be no single center to the world of information. In order to get from the isolated model, in which each consumer of identity information manages this information himself to an identity management which takes the decentralized nature of the Internet into account, we argue that consumers of identity information need to be able to assess and distinguish the quality of the information they receive. In particular, with regard to the launch of electronic ID cards as fostered by several european governments, different sources of identity information will have a different quality in terms of correctness and integrity. To have this information integrated into current identity management models is the essence of my latest research work. Following this track, my vision is to help to open up currently pre-dominating identity islands and to create a better user experience as well as to decrease security risks by reducing the number and associated credentials of user accounts.

7 List of Publications

2009

- Martin Wolf, Ivonne Thomas, Michael Menzel, and Christoph Meinel: *A Message Meta Model for Federated Authentication in Service-oriented Infrastructures* In Proceedings of the 2009 IEEE International Conference on Service-Oriented Computing and Applications (Los Alamitos, CA, USA, 2009).

- Ivonne Thomas and Christoph Meinel: *Enhancing Claim-Based Identity Management by Adding a Credibility Level to the Notion of Claims* In Proceedings of the 2009 IEEE International Conference on Services Computing (SCC-09) (Bangalore, India, Sept 21 - 25).

- Michael Menzel, Ivonne Thomas, Benjamin Schueler, Maxim Schnjakin, and Christoph Meinel: *Security Requirements Specification in Process-aware Information Systems*, Highlights of the Information Security Solutions Europe (ISSE) 2009 Conference, vol. , Vieweg-Verlag, 2009.

- Regina N. Hebig, Christoph Meinel, Michael Menzel, Ivonne Thomas and Robert Warschofsky: *A Web Service Architecture for Decentralised Identity- and Attribute-based Access Control*, In Proceedings of the 2009 IEEE International Conference on Web Services (ICWS-09)(L.A., USA, July, 2009).

- Uwe Kylau, Ivonne Thomas, Michael Menzel, and Christoph Meinel: *Trust Requirements in Identity Federation Topologies* In Proceedings of the 2009 IEEE International Conference on Advanced Information Networking and Applications (AINA-09)(Bradford, UK, May 26 - 29, 2009).

- Michael Menzel, Ivonne Thomas, and Christoph Meinel *Security Requirements Specification in Service-oriented Business Process Management* In Proceedings

of the 2009 IEEE International Dependability Conference (ARES-09) (Fukuoka, Japan, March 16 - 19, 2009).

2008

- Ivonne Thomas, Michael Menzel, and Christoph Meinel *Using Quantified Trust Levels to describe Authentication Requirements in Federated Identity Management* In Proceedings of the 2008 ACM Workshop on Secure Web Services (Alexandria, Virginia, USA, October 31 - 31, 2008). SWS '08. pp. 71 - 80, ISBN:978-1-60558-292-4
- Ivonne Thomas, Michael Menzel, and Christoph Meinel *Quantified Trust Levels for Authentication*, Highlights of the Information Security Solutions Europe (ISSE) 2008 Conference, vol. , Vieweg-Verlag, 2008.

2007

- Michael Menzel, Ivonne Thomas, Christian Wolter, and Christoph Meinel *SOA Security - Secure Cross-Organizational Service Composition* Proc. Stuttgarter Softwaretechnik Forum (SSF), Fraunhofer IRB-Verlag, Stuttgart, Germany, November 2007. pp. 41 - 53, ISBN 978-3-8167-7493-8

References

[1] Amazon. Amazon Web Services. http://aws.amazon.com/, September 2009.

[2] Audun Jøsang, John Fabre, Brian Hay, James Dalziel, and Simon Pope. Trust requirements in identity management. In *ACSW Frontiers*, pages 99–108, 2005.

[3] Uwe Kylau, Ivonne Thomas, Michael Menzel, and Christoph Meinel. Trust requirements in identity federation topologies. *Proceedings of the 2008 IEEE International Conference on Advanced Information Networking and Applications (AINA-09)*, 2009.

[4] Michael Menzel, Ivonne Thomas, and Christoph Meinel. Security Requirements Specification in Service-oriented Business Process Management. In *IEEE International Dependability Conference (ARES-09)*. IEEE, 2008.

[5] Microsoft. Microsoft's Vision for an Identity Metasystem, May 2005.

[6] OASIS. Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.02 . *OASIS Standard Specification*, March 2005.

[7] Ivonne Thomas, Michael Menzel, and Christoph Meinel. Using quantified trust levels to describe authentication requirements in federated identity management. In *SWS '08: Proceedings of the 2008 ACM workshop on Secure Web Services*, pages 71–80, New York, NY, USA, 2008. ACM.

A Appendix

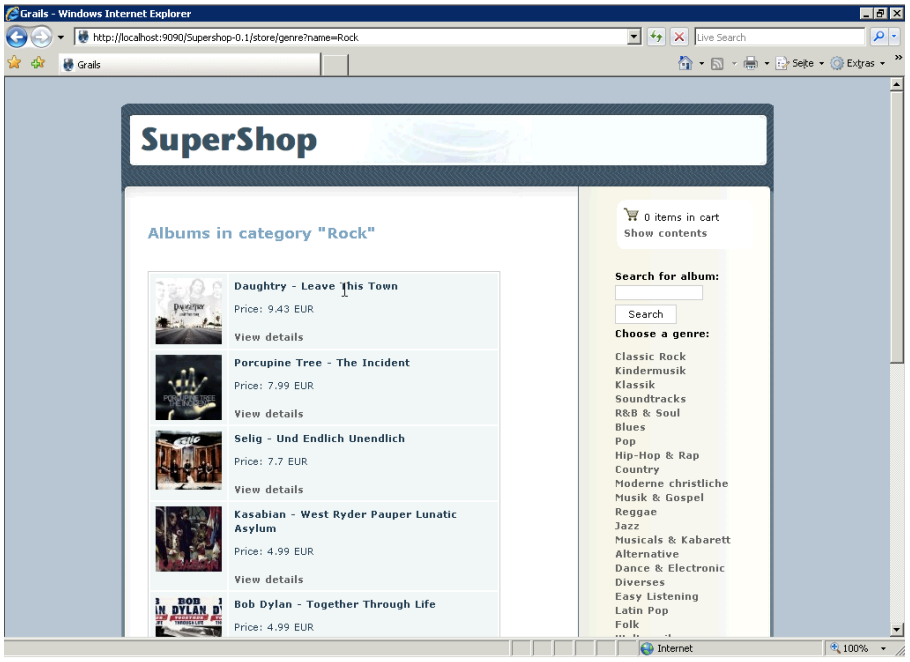


Figure 4: The MP3 Store Szenario

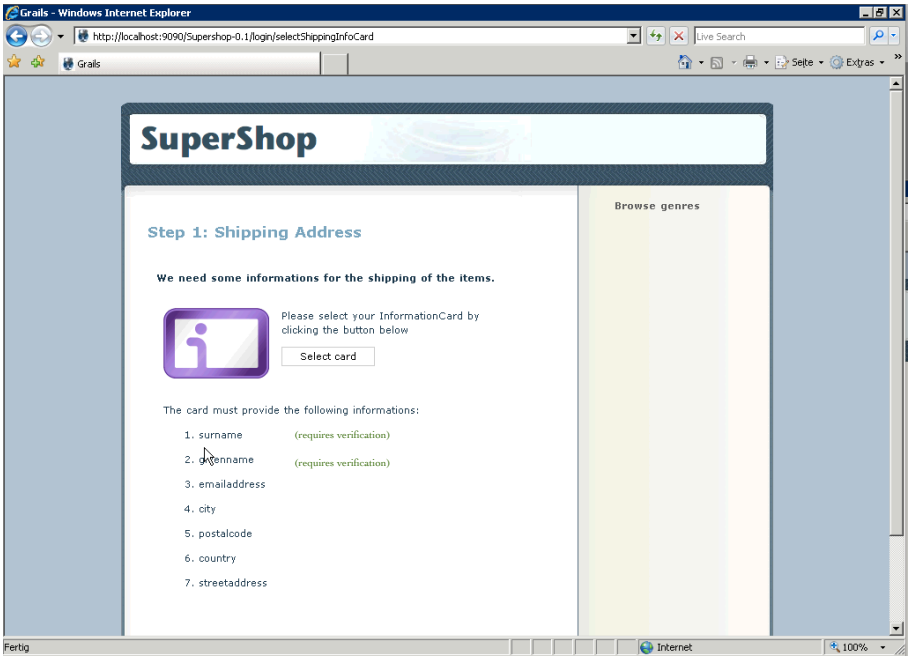


Figure 5: The MP3 Store Scenario: Retrieving identity information using information cards.

Introducing the Model Mapper Enactor Pattern

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

1 Introduction

In the beginning of the web, little thought was spent on resources and representations. People were thinking about “web pages” and such web pages were simply mapped to files in a directory structure. Consequently, the web was static and read-only. Quickly, the need for dynamic content arose, from trivial such as displaying the current time to complex such as completely personalized content out of recommendation system analyzing the users’ interactions.

To aid the developer, frameworks evolved. A framework is the combination of code libraries and development methodology to handle the reoccurring parts of an application, allowing the developer using a framework to focus on the parts of the application that are specific. Consequently, a framework reduces bugs by reuse of common parts, increases development speed by providing a methodology, and improves maintainability by making the specific code follow the framework’s convention and thus ease the initial training of new developers on existing code.

In the course of this paper, we will introduce a new pattern for web frameworks to implement resources. The rest is of the paper is structured as follows: In section 2, a short introduction to the concept of resources is given. In section 3, the *model view controller* pattern (MVC) is outlined, as it is frequently used to by today’s frameworks to implement resources. In section 4, the problems of MVC are discussed. Section 5 discusses related work. In section 6, the *model mapper enactor* pattern is introduced as an evolution to the MVC pattern. Finally, section 7 provides a conclusion and outlook.

2 About Resources

From a conceptual perspective, resources—as described by Fielding [2]—are active components in a global namespace, i.e. the Internet. Within this namespace, they are uniquely identified via Unique Resource Identifiers (URI) [9] and may interact by exchanging messages. However, their life cycle is independent of the medium. In an FMC block diagram, the conceptual design of a resource may be visualized as shown in Figure 1(a). To the outside, a resource is an active system component operating on an internal serialization. As the resource is in control of its on life cycle, the resource can be drawn as an self-controlling structure variance. Notice, that there may be many resources for a single serialization exposing various aspects and behaviors.

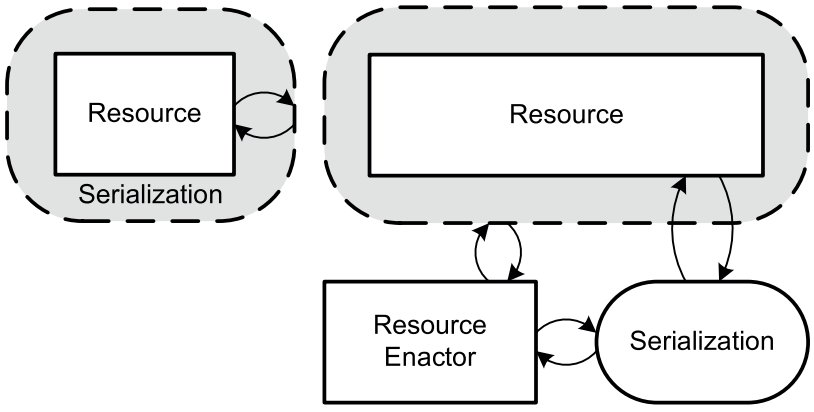


Figure 1: Resources: From Concept to Implementation

As valuable as Figure 1(a) may be from a conceptual point of view, it is neither valid FMC nor is the implementation obvious. It is best to regard Figure 1(a) to be a shortcut notation for Figure 1(b). A *resource enactor* reads the *serialization* to create a structure variance acting as the actual resource implementation. This implementation may modify the serialization including ending the life cycle by destroying the serialization. There are more aspects to resources than this, but as their relevance is not immediate for the discussion in this paper, we will neglect them for the time being.

To improve the discussion, let us introduce a concrete example of a resource: a single online auction. This auction qualifies as a resource. As resources are identified by URIs, the auction should be accessible through a single URI for all its life cycle. However, during its life cycle, the auction's behavior changes dramatically. Before the auction is started, it will be editable by the auction owner. Once it is started, the auction is not editable anymore, but all registered users except the owner may place bids. When the auction finishes, all further bids must be rejected. Keep this example in mind when we outline the MVC pattern in the next section. We will come back to this example in section 4.

3 The Model View Controller Pattern

The MVC pattern [7] is very useful for architecting interactive software systems. The pattern isolates business logic from input and presentation, permitting independent life-cycles of each aspect. It is very common in today's web frameworks.

Models represent the persistent data of the application and the rules to manipulate that data, i.e. the business logic. Models are very frequently an abstraction on top of a relational database [6].

Views embody the user interface of the application. In the web context, the view's role is to generate the representation returned in an interaction. The common approach here is to provide a choice of one or more template engines taking the current context as input.

Controllers are the glue between models and views. Controllers are responsible for handling the request. Based on request URI and request header only, a controller is responsible for creating a request context. In this request context, the controller instantiates the correct model, calls the appropriate business logic methods on this model, then instantiates a suitable view, and finally returns the view's result as the interaction response.

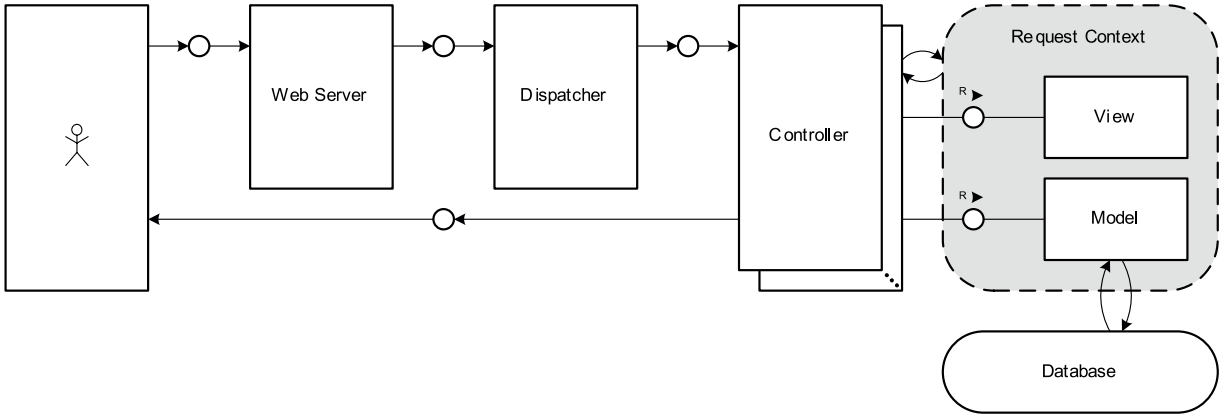


Figure 2: Model-view-controller based web frameworks

This architecture is visualized in Figure 2. Compared to Figure 1(b) model, view, and controller are all internal components of the resource. The resource enactor is implemented as a dispatcher, but how does a dispatcher of a modern web framework actually work?

Listing 1: Example URI dispatching

```
1 require "sinatra"
2
3 get "/say/*/to/*" do
4   # matches /say/hello/to/world
5   what, who = params[:splat] # => ["hello", "world"]
6
7   content_type "test/plain"
8   "Saying #{what} to #{who}" # "Saying hello to world"
9 end
```

Discussing URI dispatching is difficult to visualize, hence we use actual source code as illustration (see Listing 1). The example given is complete, i.e. it is executable without any further source code or configuration.

Before discussing URI dispatching, let us fully understand this application. First of

all, it is written in Ruby[1], but should be understandable even without in-depth Ruby comprehension. In Ruby, the character # starts a comment to the end of the line. In line 1, the *Sinatra* framework [2] is loaded. This single line also starts a web server and dispatcher implicitly (compare Figure 2). Lines 3-9 define a controller. In line 3, a HTTP verb and an URI pattern are given. Line 5 extracts parts of the URI pattern defined in line 3 and these parts are used instead of a model to keep the example short. Line 7 sets the response's content type. This reflects the controller's decision logic which view to selected based on the request and executed business logic. In this example there is only one view, and it's inlined in line 8. In Ruby, the last statement of a block is implicitly returned. The Sinatra framework treats this return value as the view. Everything else is addressed by convention, Listing 1 is a complete and executable examples.

The Sinatra framework was chosen for this illustration, simply because it reflect the state of the art and allows such compact examples. So how does URI dispatching work in a modern framework? Line 3 gives the answer: Pattern matching. Here, each request is match for a verb and an URI pattern. More complex examples may also include request header matching.

4 Problems arising with MVC

From the discussion in section 3 it becomes apparent that there is no actual resource orientation inside of MVC-based frameworks. The dispatcher has a static set of controllers, upon receiving a request it will simple pattern match the request and pass it to the matching controller. Looking at Figure 1(b), reaching the controller implies reaching the resource instance. But from Figure 2 it is obvious that instantiating the resource is left to the responsibility of the controller. Hence, all controller implementations will have to start with logic to lookup the internal resource serialization—in the form of instantiating the correct model—and deal with error handling in case no matching resource serialization exists.

Returning to the auction example in section 2, all auction URIs will have to align to the pattern matcher of the auction controller. URIs will match regardless if the resource actually exists. The auction controller will have to lookup the correct model and report and error if failing to do so. But let us assume the auction controller was able to instantiate the correct auction for this request. What will happen next? This depends on the life cycle of the auction as we discussed when introducing the example. Resource are allowed and expected to change their behavior dramatically throughout their life cycle. As the MVC pattern has a static controller lookup, this implies a set of condition checks to be executed to figure out the state of the model within its life cycle. Only then, the specific business logic for the current state of the targeted instance can be triggered. Out of the author's personal experience, getting the condition checks right in all edge cases is not a trivial task.

Both aspects increase the amount of boiler plate code to be written and increase the likelihood of introducing logic errors into the application. Resource orientation is a very

[1] See `http://ruby-lang.org/`
[2] See `http://www.sinatrarb.com/`

powerful design approach, but given the MVC pattern it disappears almost completely at the implementation level. This can be shown in yet an another aspect of Listing 1, we have not discusses yet. The URI pattern given in line 3 is not matching against the complete URI, but only against the path. *Scheme*, *host*, and *port* are ignored by all major web frameworks. The underlying assumption is any request reaching the web framework to be meant to be handled by the application.

In essence, using the MVC pattern to implement resources violates the spirit of resource orientation in many ways. The MVC pattern is much older than the web and adopting it to a resource-oriented facade allowed programmers to write applications to the new environment with familiar tools. However, resource orientation is not new and unfamiliar anymore, it is time for the internal architecture of web applications be more resource-oriented. In the next section, the *model mapper enactor* pattern is proposed to transform the internal architecture of web application to be more resource-oriented.

5 Related work

Proposing a new design pattern should always reflect on related trends. Regarding the model, there is a broad trend to simplification of storage. As mentioned before, storing models in relational databases is still very popular. These mappings can be quite complex, but lately the active record pattern is gaining popularity [3, p. 160ff]:

> An active record is an object that wraps a row in a database table or view, encapsulates the database access, and adds domain logic on that data.

In other words, the mapping of relational storage to application objects becomes much more simple and straight forward. Each model object maps to exactly one row in the underlying relational database.

For the sake of scalability, explicit and distinct models are suggested [4]. Dealing with concurrency means giving up the illusion of omnipotence, such as arbitrary serializability of operations. Explicit and distinct models give clean boundaries to the scope of serializability.

In the industry, this trend is topped by a strong move towards key-value stores or document-oriented databases. Explicit boundaries are becoming more important and data schemes. The best example is Google [5] facing the challenge of storing and indexing petabytes of crawled web content.

In [1] the *map reduce* pattern is introduced to processing large sets of data. Map functions are context free, idempotent functions taking a single model as input at a time and producing key value pairs for an associated index. Context free means, that a map function can not keep state between calls. Idempotent means, that a map function will always produce the same output given the same input. These characteristics make parallelized execution of map function on arbitrary amounts of models trivial. Reduce function may be used to merge all emitted values with the same key, however they are less relevant in the context of this paper.

6 The Model Mapper Enactor Pattern

The goal of the model mapper enactor pattern (MME) is to provide the flexibility of the model view controller pattern, yet be more resource-oriented and avoid the problems discussed in section 4.

The first assumption this pattern makes is that a model represents an internal serialization of a resource as visualized in Figure 1(b). This assumption works best, if we assume a document store instead of a relational database. All models are processed by map function producing an index with full URI as key and a reference to a state specific enactor. The whole architecture is visualized in Figure 3.

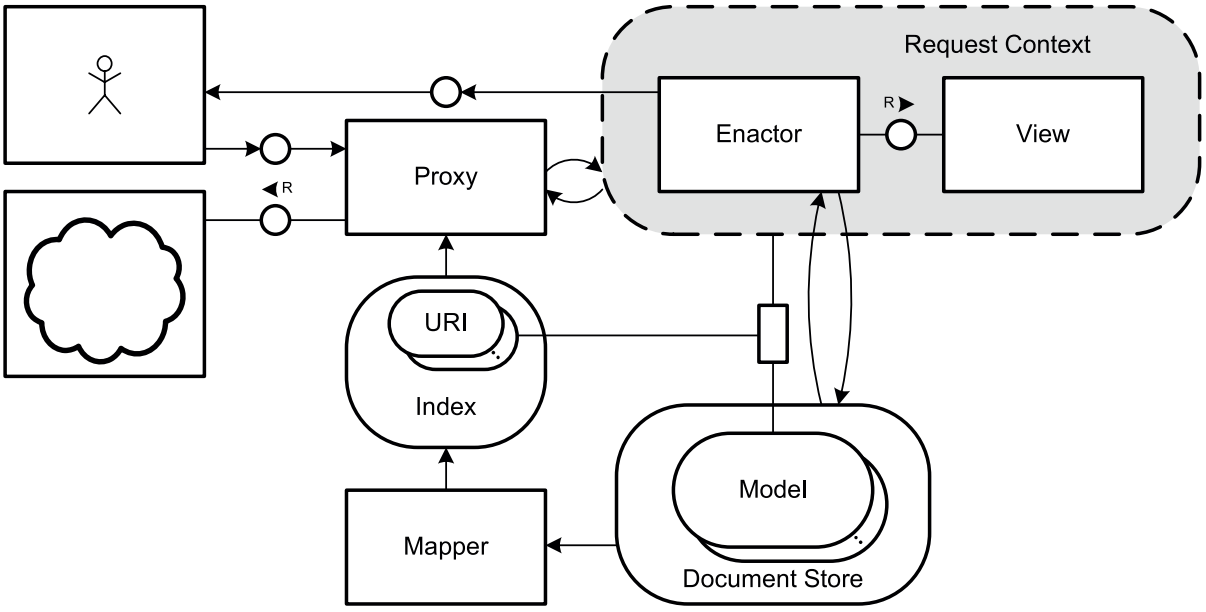


Figure 3: Model Mapper Enactor Pattern

This simple change eliminates several weaknesses of MVC: First, mappers can only emit if a model exists. In MVC, the controller is called first and needs to look up the model, which is an instance mismatch. In MME, the model comes first as the mapper can only emit a *behavior* for an existing model. The need for boiler plate code inside a controller to instantiate the model and do error handling if there is none disappears.

Also the mapper is free to emit a completely different enactor for each model state, hence the boiler plate for deciding on the state within the controller disappears as well. Notice, both code blocks just described were executed by the controller within the request cycle. Pushing them to the mapper removes them from the request cycle, which ultimately will reduce request latency. Request latency has a strong influence on perceived performance [2], reducing it is always beneficial. Applying this these changes to our example auction, each phase in the auction's life cycle will have it's own mapper emitting the correct behavior only when the underlying model matches.

Moreover, the MVC pattern must rely on URI pattern matching to find the correct controller to forward the request to. While human-readable URIs are nice to have, a

machine should always be allowed to treat a given URI as blackbox. Pattern matching is a violation of this rule. In contrast, using the MME pattern, the lookup starts at the model, the mapper is free to emit any URI, no restriction are given to its format.

Emitting full URIs allows the web server and dispatcher from Figure 2 to be replaced by a proxy, as shown in Figure 3. All user requests are passed to the proxy. The proxy will check the index. If it doesn't match, it will forward the request to the Internet, just as a normal proxy would do. However, if the index does have a matching entry, the index value is used to instantiate the correct enactor and view in the request context. This provides the basis for an alternative approach to offline capabilities of web applications by moving the resource implementation into the user's proxy.

In the MME pattern, the model is assumed to be stored in a document store. Loading and storing is relatively trivial. Also the correct model is related to the request context by design. At the same time, the controller is freed of most of it's responsibilities. The only thing left unassigned to a component is business logic—the model is loaded by a generic mechanism and does not include business logic any longer—and choosing the correct view. To make this differentiation clear, the MME pattern calls this component responsible for these task an enactor. Also, this correlates nicely with Figure 1(b).

7 Conclusion and Outlook

The model mapper enactor pattern is an evolution of the model view controller pattern. It builds on the assumption, that models can be processes by map functions, which coincidently is a valid approach for large-scale data processing. MME is more resource-oriented than MVC, as resources are only exposed, if an internal serialization in a specific phase of it's life cycle exists. The controller is freed of boiler plate code necessary to construct the right request context. Also the execution of this code is moved out of the request pipeline, improving request latency.

However, the MME pattern puts much burden on the mapper. This is the point, where framework builders should focus their effort on guiding the developer, when incorporating the MME pattern instead of the MVC pattern. Another critical aspect is how the value of the URI index is turned into executable code.

The documentation of a use case implemented using the MME pattern was accepted to the *Mashups'09 3rd International Workshop on Web APIs and Services Mashups at OOPSLA 2009* [8]. The paper was written in cooperation with Matthias Kunze, Alexander Großkopf, and Matthias Weidlich. The prototypical implementation of the MME pattern are based on node.js[3], an asynchronous server-side Javascript environment. As Javascript is capable of eval'ing code, the value of the URI index is directly eval'able by the proxy implemented via node.js. The framework used to generate the mappers and its design are beyond the scope of this paper, but will be included in the finally Ph.D. thesis scheduled to be completed by year's end.

[3]See `http://nodejs.org/`

References

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

[2] 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`.

[3] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[4] 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`.

[5] Wilson Hsieh, Jayant Madhavan, and Rob Pike. Data management projects at google. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 725–726, New York, NY, USA, 2006. ACM.

[6] Wolfgang Keller. Mapping objects to tables - a pattern language. In *Proc. Of European Conference on Pattern Languages of Programming Conference (EuroPLOP)"97*, 1997.

[7] Glenn E. Krasner and Stephen T. Pope. A cookbook for using the model-view controller user interface paradigm in smalltalk-80. *J. Object Oriented Program.*, 1(3):26–49, 1988.

[8] Matthias Kunze, Alexander Großkopf, Hagen Overdick, and Matthias Weidlich. Lightweight Collaboration Management. In *Proceedings of the Mashups'09 3rd International Workshop on Web APIs and Services Mashups at OOPSLA'09*, 2009.

[9] L.Masinter T.Berners-Lee, R.Fielding. Uniform Resource Identifiers (URI): Generic Syntax. Technical report, The Internet Engineering Task Force, 1998. `http://www.ietf.org/rfc/rfc2396.txt`.

A Runtime Environment for Online Processing of Operating System Kernel Events

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

This report summarizes my research work at the HPI Research School on Service-Oriented Systems Engineering, done at the operating systems and middleware group supervised by Prof. Dr. Andreas Polze.

A review of the different steps leading to the main focus of my work - *online processing of operating system kernel events* - is given.

1 Introduction

Each member of the research school is assigned to a specific research group at the HPI. Therefore, the initial members of the research school tried to find the elements that connect the area of research of the particular group and the topic of the research school: service-oriented systems engineering.

A common understanding and definition of terms such as *service* or *SOA* was hard to find due to the different perspectives of the research school members. A poll of people working at IT departments of different companies and evaluating research papers and other online resources also did not lead to concise definitions.

From an operating system point of view [7] a service definition can be reduced to: *a service is provided by a server application with an interface that can be utilized by external client applications*. The (important) meta-aspects of service provisioning such as interface description, message format definition, stateful/stateless behavior, or non-functional properties are ignored on the operating system level and are handled by the service container, e.g. by an application server. Nevertheless, the operating system is the central component determining the performance of server applications.

The main task of an operating system is resource management by assigning processing capabilities (e.g. the CPU or different kinds of IO bandwidth) to concurrent activities. The optimal way of resource assignment depends on different system characteristics such as the current workload (e.g. number of active requests, number of concurrently provided services) and service meta-information (e.g. the required quality of service level). To support the efficient execution of components of a service-based environment, the operating system should provide interfaces to allow optimal resource assignment in different scenarios. In a perfect world, the operating system (or the platform in general) would be able to process high-level target descriptions in terms of

throughput and response time and derive the optimal resource assignment automatically.

The initial research question therefore was: How can the operating system support the efficient execution of server applications? The natural first step was to investigate resource assignment in the context of service-oriented systems. Additionally, monitoring of activities in the operating system kernel was investigated.

This work lead to the conclusion that an important aspect is the connection between the observation of the system state (= monitoring activities in the operating system kernel) and the adaptation of the system behavior (= resource assignment policy). System adaptation in this sense is not limited to the resource aspects and can also be used for other purposes, such as re-configuration in general, system analysis, debugging, or logging.

The remaining sections of this report summarize the work done in the different areas. The result is a runtime environment for online processing of operating system kernel events which can be used to detect specific event constellations and to react to such constellations in an (almost) arbitrary way.

The described work was implemented and tested in context of the Windows Research Kernel (WRK) [4], the core parts of the Windows Server 2003 Enterprise Edition. The WRK comes with a build-environment which allows to build custom versions of the Windows kernel.

2 Ressource assignment

To ensure certain service levels for concurrent activities, predictable resource assignment is a necessary precondition: only if the environment guarantees that each activity gets the required amount of processing capabilities, the service provisioning goals can be achieved. In general, the goals are defined in terms of maximal tolerable response time or minimal throughput of the system.

For fine-grained CPU cycle assignment, the scheduling server concept [9] may be used. The scheduling server is integrated into the Windows kernel and provides a programming interface to assign specific percentages of the CPU capacity to activities. The kernel scheduler ensures that each activity can use its assigned amount of CPU time.

The high level scheduling server concept is depicted in figure 1: a dispatcher thread, running at the highest possible priority, lifts the controlled tasks for a specific timespan to the second highest priority. In this way, a fine-grained CPU assignment can be implemented by using the round-robin scheduler implementation of the regular operating system kernel.

By modifying the clock interrupt handler of the the kernel, the dispatcher thread can be removed and the kernel can directly manage specific CPU shares for different activities. This approach leads to low overhead (around 1%) and allows CPU assignment in steps <1%. More details can be found in [9].

With the scheduling server infrastructure in place, the remaining question is: How to determine the required shares of CPU time for each concurrent activity? Different ap-

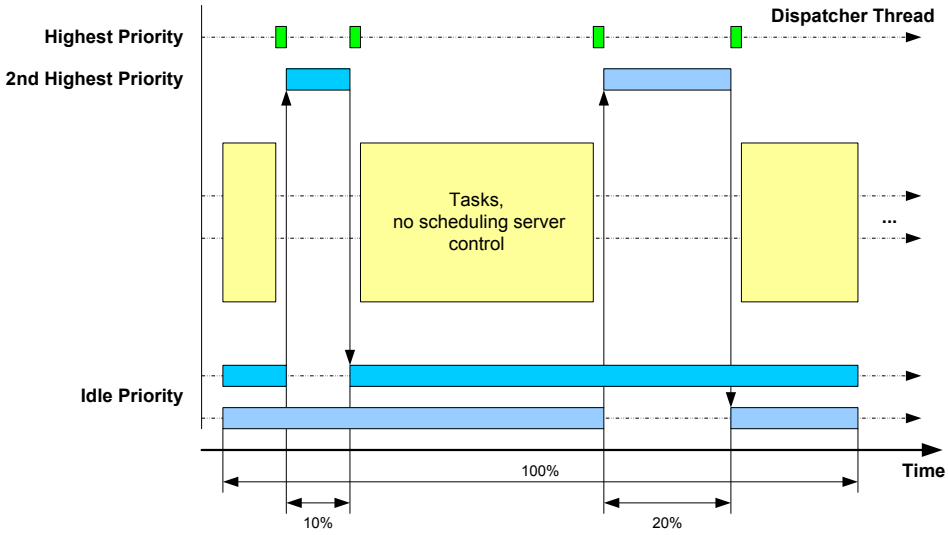


Figure 1: Scheduling server concept

proaches were proposed in related research work - concepts of autonomic computing or control theory. A central requirement for resource assignment is system monitoring. By determining the current state of request processing in the system, necessary actions can be derived to optimize system behavior.

The next section describes an approach for monitoring events in the operating system kernel.

3 Operating system monitoring

Monitoring events in a system with low overhead and high reliability was investigated and implemented for a large number of platforms and operating systems. Due to the closed-source nature of the Windows operating system, such an implementation dedicated for research purposes did not exist previously[1]. The Windows Monitoring Kernel (WMK) [6, 8] closes this gap and provides an efficient monitoring infrastructure for the Windows kernel.

The whole WMK approach is comparable to KLogger [3], an event logging infrastructure for the Linux kernel. Figure 2 shows the WMK architecture. Unmodified user-mode applications are executed on top of a modified Windows kernel. These applications communicate with the kernel via system service calls. At different places in the kernel instrumentation points indicate events, e.g. the creation of a new thread, synchronization operations, or context switches. The WMK provides event allocate and commit operations for event data. Gathered events are stored in an event buffer and periodically written to disk. The WMK uses double buffering for event storage and lock-free synchronization of concurrent buffer accesses.

[1] Event Tracing for Windows (ETW) can not be extended with new kernel events in the way required for our work.

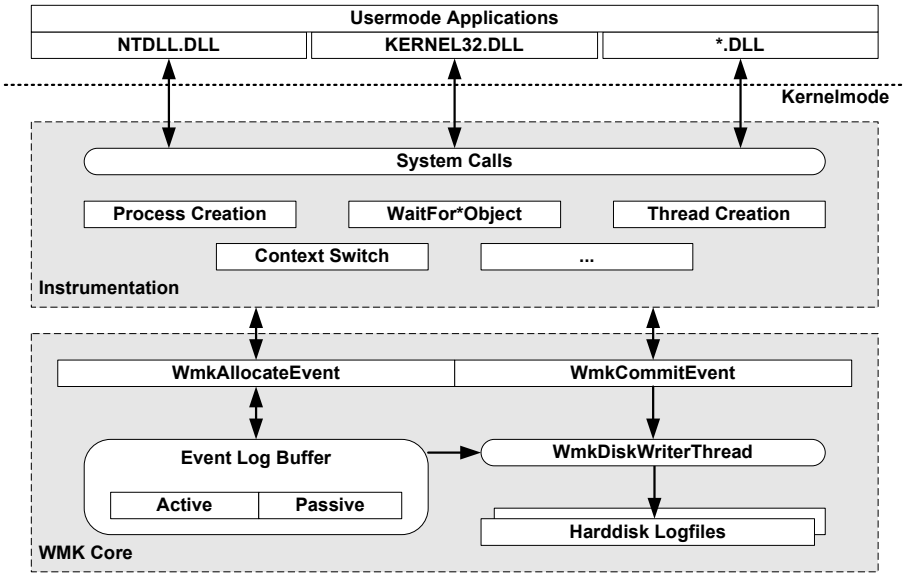


Figure 2: Overview: WMK components

The WMK is able to record events at a rate of 60000 events per second with an overhead (= system slow down) around 5%. More details can be found in [6, 8].

The WMK offers the possibility to record events for almost every aspect in the operating system kernel. The basic infrastructure can lead to several thousand events per second which are stored in logfiles. The logfiles are analyzed after the event recording is finished. This shows the two most important disadvantages of tracing-based approaches for system analyses: a huge amount of data and the time lag between event recording and reaction.

On the other hand, tracing based system analysis offers several unique advantages: special system states (e.g. failing components or deadlocks) can be detected easily while scanning a complete event trace. Furthermore, request processing in server applications can be described as sequence/stream of steps which can be monitored in an event-based way to detect divergent behavior.

The next section describes a runtime environment which tries to alleviate the problems of tracing based system analysis by processing ocurring events at the time they happen.

4 Online event processing

The proposed system [5] allows analyzing event streams by searching for specific event constellations. A pattern specification language allows describing complex event patterns and (optional) reactions to these patterns. A programming interface enables applications to register patterns in the system and to react if such patterns occur. The system can be used for the dynamic runtime analysis of operating system kernel events without writing the events to a logfile, i.e. the events are processed *online* as opposed to the more common offline analysis.

Concept		Description
EPC	=	epcstatement {epcstatement}
epcstatement	=	(eventdefinition|ruledefinition)
eventdefinition	=	"EVENTS" <file name>
ruledefinition	=	["SYNCHRONOUS"|"ASYNCHRONOUS")] "RULE" <rule name> patterndefinition {rulemodifier}
rulemodifier	=	(wheredefinition|whithindefinition|resultdefinition)
patterndefinition	=	["STRICTSEQUENCE"|"STRICTPARTITION"|"SKIPTILLNEXT")] "PATTERN" "{"eventstructure"}"
eventstructure	=	(eventspecification|eventsequence| eventalternative|eventnegation)
eventsequence	=	"["eventstructure {"," eventstructure}"]"
eventalternative	=	"("eventstructure"|"eventstructure { "|" eventstructure}")"
eventnegation	=	"~" (eventspecification|eventsequence|eventalternative)
eventspecification	=	<event type> [eventlength] [":" <event name>]
eventlength	=	"["[(<start> ".." <end>|("<"|"="|">") <number>)]"]"
wheredefinition	=	"WHERE" "{"wherecondition {"," wherecondition}"}"
wherecondition	=	(fieldjoin|relation)
fieldjoin	=	"["<event field name>"]"
relation	=	value relationop value
value	=	fieldspec [operation fieldspec]
fieldspec	=	(<integer const>|<float const>|<event name> (fieldspecevent|fieldspecarray))
fieldspecevent	=	"." <event field name>
fieldspecarray	=	"[" "]" "." ("len"|("avg"|"max"|"min") "." <event field name>)
operation	=	("+"|"-"|"*"|"/"|"&"|"|")
relationop	=	("<"|"<="|"=="|">="|">"|"!=")
whithindefinition	=	"WITHIN" <time>
resultdefinition	=	"RETURN" ("SEQUENCE"|'{' value { ',' value } '}')

Table 1: Language for event pattern specification

The Windows Monitoring Kernel provides the basic event logging infrastructure. Instead of writing events to a logfile, the runtime environment handles events as they occur. The event processing is based on deterministic finite automata which are derived from a formal description of event constellations.

Event constellations can be specified using the grammar shown in table 1. The language uses concepts from EventScript [1] and SASE+ [2]. An *EPC* (event processing) file contains a sequence of `epcstatements`, which are either an `eventdefinition` or a `ruledefinition`.

An `eventdefinition` is used to define the available event types and the structure of the event data. Each event contains event header information (such as timestamps) and event data fields (arbitrary information about the occurring event). Currently, the EPC compiler parses a regular C header file with Windows Monitoring Kernel event

definitions and extracts the event structure information. The datatypes of the event data fields are saved to ensure typesafe usage of event data fields in other parts of the EPC file.

After defining available event types, a set of rules can be specified in an EPC file. Each rule definition contains at least a name and a pattern definition. Additionally, the mode of rule processing (either synchronous or asynchronous) and additional rule modifiers can be specified. The mode of a rule defines whether events are processed in the execution context in which they occur (synchronous processing, blocks the activity which causes the event) or not (asynchronous processing, no blocking). Rule modifiers are used to fine-tune the rule definition, e.g. by specifying additional relations between certain events or by setting a time limit for the pattern occurrence.

The basic elements of each pattern definition are either single events or arrays of events. Single events are specified by giving an event type name, arrays are specified by an event type name and "`[]`". Optionally, length limitations can be specified in the brackets, e.g. `a[<5]` defines that less than five events of type `a` are expected to match the pattern.

Single events and arrays can be named by using a colon: `type:name` specifies a named single event and `type[]:name` specifies a named array of events. These names can be used e.g. in rule modifiers to compare specific event data fields.

Event patterns can be constructed analogous to regular expressions:

- An event **sequence** `[a,b,c]` defines that events of types `a`, `b` and `c` have to occur subsequently in an eventstream to match the pattern.
- An **alternative** `(a|b)` defines that either an event of type `a` or an event of type `b` is expected.
- A **negation** `~X` specifies that a certain pattern `X` does *not* occur in the eventstream.

These structure elements can be nested in arbitrary ways, e.g. `[a,~[b,~c],(d|e)]` is a valid pattern specification. Repetitions of composed sub-patterns (kleene star operators) are currently not supported - for repetitions of single events arrays can be used.

Additionally to the event pattern structure, the pattern semantic has to be specified. There are three different semantic modes for a pattern specification:

- `STRICTSEQUENCE`: all specified pattern elements occur in a strict sequence in the combined eventstream containing all occurring events in the system.
- `STRICTPARTITION`: all pattern elements occur in a strict sequence in a sub-eventstream (partition) containing only events of a specific execution context.
- `SKIPTILLNEXT`: no strict sequence of pattern elements is required, irrelevant events are skipped.

After specifying the event pattern, additional conditions can be defined. These conditions allow to define relations between event data fields or to define join fields. In the current prototypical implementation a single event data field or a pair of event data fields can be combined and compared to another single event data field or pair of fields. The common arithmetic and relation operators can be used. Special array operators provide access to the number of array elements and the minimum value, the maximum value, or the average value of the array elements. These basic operations are sufficient for our proof of concept implementation. Support for arbitrary calculations might be added in future versions.

Event data fields can be used as join fields by using square brackets, e.g. `WHERE {[name]}`. This example specifies that *all* events used in the `PATTERN` statement must contain a data field `name` and all matching events must have the same value assigned to this field.

Join fields which are part of the execution context define a eventstream partition that is considered in the `STRICTPARTITION`-semantic.

A rule can have a timeframe which defines the maximal difference between the timestamp of the first detected event of a pattern and the final event. In our prototype, the timeframe can be defined based on the CPU cycle counter which is used for event timestamping. Alternatively, time can be specified in seconds. The runtime environment calculates the corresponding value for the CPU cycle counter.

The last element in a rule description is the `RETURN`-statement. There are two possible results a rule can provide: (1) the rule can return the whole event sequence which matched the pattern, or (2) the rule can return a set of event data fields.

Concepts that are planned for extensions of the language are: (1) A `DO`-statement for action specification which are executed immediately after a defined pattern is detected. The set of useful actions and security aspects has to be investigated. (2) String operations, e.g. for file name comparisons.

Following, an example for a pattern specification is given: The rule, shown in listing 1, can be used to **detect lock contention**, i.e. concurrent acquire operations for the same synchronization object.

Listing 1: Detect lock contention

```
1 SYNCHRONOUS RULE lockcontention
2   SKIPTILLNEXT PATTERN { [lock:a, ~lock:b, lock:c] }
3   WHERE { [ObjAddress],
4           a.Operation == 1,
5           b.Operation == 0,
6           c.Operation == 1 }
```

This rule analyzes `lock` events which belong to the same synchronization object (= the event field `ObjAddress` has the same value). The first `lock:a` event tries to acquire the synchronization object (`Operation` is 1), the second `lock:b` event releases the object (`Operation` is 0), and the third event also tries to acquire the object.

By negating the second event (i.e. the event must not occur in the eventstream) the pattern can detect lock contention: a synchronization object can only be acquired by one single thread, therefore a release operation can only be generated if the syn-

chronization object was acquired successfully. If there is no contention on the synchronization object, event `lock:a` and event `lock:b` are generated by the same thread and the pattern detection aborts. If the release is not detected and another acquire event occurs, it must be from a different thread and the lock is contented.

The exclusion of recursively acquired locks would require a more complex rule.

The rule is executed in a synchronous way, therefore the execution of the activity that generates the `lock:c` event is blocked if the described pattern occurs. In such cases the locking strategy might be adapted or an entry might be logged for later analysis.

The runtime environment consists of the following components: the compiler for EPC specifications and a user-mode application which loads compiled rule definitions into the system.

The compiler was implemented using the Coco/R[2] framework. It compiles a rule specification (compliant with the described grammar) into a deterministic finite automata. Due to space restrictions we omit details about the compilation process. Converting regular expressions into automata is a well investigated topic.

Conditions and actions are assigned to automata transitions. Conditions are checked to determine if the transition is valid at the current state of the automata, considering the current event. E.g. the expected event type and the `WHERE` conditions are checked via conditions. Actions are executed if the transition is actually chosen. Actions might save specific data fields of the current event for later reference or modify certain aspects of the runtime environment for the specific rule.

The binary representation of the automata is loaded by the runtime system. If an event occurs, the runtime system identifies rules that are interested in the specific event. The transition from the initial automata state to the next state (i.e. when the first event relevant for the pattern occurs) starts an *automata run*.

Each run is represented in memory by a runtime state representation. The data structure for runtime state representation is generated by the compiler. It contains the current automata state, event field information (which is initialized by transition actions), and information about results that is returned if a pattern is detected. These items can be derived from the EPC script.

When a new event occurs, the event information is stored in an execution context local buffer. That means e.g. that each thread has its own buffer for occurring events. The event information is then fed into the event processor. The event processor manages a global buffer for event data. This buffer contains event data that is relevant for rule results and event data that does not require immediate processing.

First, the event processor checks if there are any synchronous rules waiting for the new event. If this is the case, the event processor evaluates the specific rule automata. Only if the event data is relevant for rule results (e.g. if the rule has to return the whole event sequence), the event information is copied to the global event buffer. Secondly, if there are any asynchronous rules waiting for the new event, the event data is copied to the global event buffer (if not already done as a result of processing a synchronous rule) and a reference is stored in the asynchronous event queue.

[2]http://www.ssw.uni-linz.ac.at/Research/Projects/Coco/

Afterwards, the execution context local event buffer can be reused for the next event. Events from the asynchronous event queue are processed by a special thread.

As already described, the runtime state of a specific rule encapsulates all information relevant for a potential event pattern match. Each rule has its own memory pool for such runtime state information. The pool size can be configured to either contain a specific number of state information or to have a specific size. Then, the actual pool size determines how many event pattern matches can be detected parallely.

This concept of memory pooling was chosen for the following reasons: (1) the amount of memory required to process a specific rule is predictable, (2) allocating and deallocating memory for runtime state information (which in general requires only a few byte) is inefficient, and (3) allocating the memory before the start of rule processing ensures that the required memory is indeed available.

The event processor-managed global buffer is subdivided into smaller buffer units. Each of these units maintains a reference counter which describes how many event data entries are still referenced by runtime state entries. If the reference counter reaches zero, the particular buffer can be re-used to store event data.

Different heuristics are investigated to improve the performance of the runtime environment: (1) Adaptive event activation and deactivation. The runtime environment can determine the set of events that is relevant for the loaded rules and for current automata runs. By interacting with the instrumentation framework, the generation of irrelevant events can be turned off to reduce system disturbance. (2) Evaluation ordering. The compiler generates *deterministic* automata, i.e. there is always exactly one valid transition for a specific state and a specific event. Transitions and conditions are evaluated sequentially. Based on probabilities of evaluation results, reordering the evaluation sequences can lead to increased performance of rule processing.

The online processing of event streams offers the advantage that the eventstream itself must not be stored to disk. Synchronous event processing allows blocking activities if they generate “bad” patterns. Furthermore, the runtime environment API provides functions which can be used to build self-aware applications, and to implement concepts from autonomic computing. More details and examples can be found in [5].

5 Summary

Online processing of operating system kernel events can be used to detect specific event constellations in the stream of events ocurring in the system. The possibility to react to detected constellations (synchronously or asynchronously) can be utilized to implement a variety of system adaptation policies - e.g. resources may be reassigned, malicious activities can be blocked or deadlocks can be prevented.

The design of the operating system kernel integrated runtime environment for online processing of event streams is the main contribution of my work. The proof of concept implementation shows the feasibility and demonstrates different usage scenarios.

References

[1] Norman H. Cohen and Karl Trygve Kalleberg. Eventscript: An Event-processing Language based on Regular Expressions with Actions. In *LCTES '08: Proceedings of the 2008 ACM SIGPLAN-SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems*, pages 111–120, New York, NY, USA, 2008. ACM.

[2] Yanlei Diao, Neil Immerman, and Daniel Gyllstrom. Sase+: An agile Language for Kleene Closure over Event Streams, 2007.

[3] Yoav Etsion, Dan Tsafrir, Scott Kirkpatrick, and Dror G. Feitelson. Fine Grained Kernel Logging with KLogger: Experience and Insights. In *Proceedings of the EuroSys 2007*, pages 259–272, March 2007.

[4] 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.

[5] Michael Schöbel and Andreas Polze. A Runtime Environment for Online Processing of Operating System Kernel Events. In *Proceeding of the Seventh International Workshop on Dynamic Analysis*, 2009.

[6] Alexander Schmidt and Michael Schöbel. Analyzing System Behavior: How the Operating System Can Help. In *Proceedings of INFORMATIK 2007, LNI Nr. 110, Band 2*, 2007.

[7] Michael Schöbel. Operating System Abstractions for Service-Based Systems. In *Proceedings of the Fall 2006 Workshop of the HPI Research School on Service-Oriented Systems Engineering*, 2006.

[8] 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.

[9] Michael Schöbel and Andreas Polze. Kernel-mode scheduling server for cpu partitioning: a case study using the windows research kernel. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 1700–1704, New York, NY, USA, 2008. ACM.

Computational Analysis of Virtual Team Collaboration in the Early Stages of Engineering Design

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

This report briefly outlines and motivates the content of my thesis, which is drawing close to completion.

1 Summary of the Dissertation

The early stages of engineering projects are considered as the most critical phase of a product lifecycle. They mark a vibrant, creative, and dynamic process at the outset of conceptual design. Decisions made at this phase determine the rest of the engineering process and any misconceptions and omissions have significant impact on the project success [32]. Researchers are exploring factors for successful design collaboration that encourage innovative potential, but appropriate observation methods are needed to respond to the augmented virtualization and dispersion of team environments. How can computer-mediated team interactions be efficiently captured and explored? Can we further use this information to identify collaboration signatures that may indicate beneficial or detrimental movements in the creation of new concepts?

The dissertation describes how virtual, unstructured, and heterogeneous design team interactions can be captured to support the assessment and comparison of process characteristics at real-time. It presents models, software technologies, and methods, which automate the collection of online collaboration activities and establishes a central interface to actor and asset relationships over the course of time. A case study in eleven distributed design projects suggests that this approach can point out significant team performance indicators during the early stages of concept creation and prototyping.

My research is structured into three phases. In the first phase, the work develops a theoretical foundation and a service-oriented software implementation to provide a platform for the computational capture of distributed online team activities. *Team collaboration networks* are introduced as a model to establish a generic foundation for the graph-based, semantic representation of associations between actors and shared information objects over time. A resource-oriented software system (*d.store*) provides a service interface to describe and access team collaboration networks, and to explore trends and signatures in the communication behavior of design teams.

In a second phase, the system is integrated into the concept development phases of

eleven distributed, inter-disciplinary engineering projects over a period of eight month in each case. The information sharing activities scanned in email archives, wiki spaces, and public team folders are translated into team collaboration networks and provide a basis for the visual and quantitative examination of communication structures and evolving trends.

In a final research phase, the generated models are taken into a case study analysis of potential team performance indicators. Significant dependencies are revealed between patterns in the email correspondence of the observed teams and self-reported team satisfaction ($R^2 = 0,48$). Overall, the results of the analysis substantiate the beneficial impact of design thinking principles in the early stages of engineering projects. The findings suggest that those teams generally perform better, which put emphasis on external communication (to end-users, etc.), internal information sharing (through documentation), and diversity in the solution space (through iterative prototyping).

The dissertation merges recent trends in information system technology with prevailing research questions in engineering design and design thinking. With the digital footprint of team communication steadily growing, computational data acquisition represents a promising approach to improve empirical observation techniques in the study of conceptual design. The contribution of this work is a flexible service platform to conduct unobtrusive design process analysis and to create insights into virtual collaboration practice during the front–end of innovation. Team collaboration networks provide a configurable, semantic data layer for the temporal and structural analysis of communication signatures in distributed teams.

The results of the case study reveal a closer understanding of factors in computer-mediated communication that are critical for success. In particular, it is shown that communication structures in virtual design collaboration can serve as a surrogate for qualitative aspects of the team performance. This is significant because it demonstrates that objective collaboration metrics may indicate design team performance live and in situ. With that, this work provides a foundation for present and future research in design management and collaboration support.

2 Motivation

2.1 A Window on Conceptual Design Practice

Despite previous efforts to improve the assessment and transparency of performance-relevant process characteristics at project run-time, a deeper understanding of the requirements (i.e., *how* to monitor) and the relevant metrics (i.e., *what* to monitor) is required. This work aims to further promote the field of automated design observations by suggesting answers to both, the *how* and the *what*: it develops a novel approach to the IT-supported measurement, processing, and analysis of online collaboration activities and scrutinizes the expressiveness of the obtained data structures with regard to indicating team performance. With the instantiation of appropriate sensors during

the early stages of conceptual design, the research tries to improve the real-time[1] evaluation capabilities for engineering processes and to generate new insights from computational team observations.

2.2 Challenges for Design Research

The scientific exploration of what designers do when they do design relies on data that is collected either directly by researchers observing the design process or indirectly through a design proxy. Methods for direct data collection comprise ethnographic studies such as the classification of visual or verbal design activity recordings and transcripts [11,21]. Indirect data collection implies, e.g, the conduction of ex-post interviews and the study of logbooks, documentation or designed artifacts to get insight into the design process [19].

While both approaches to data collection introduce considerable costs and efforts to the research, each comes with additional distinct drawbacks. Direct data collection usually relies on a controlled environment, design laboratories, or artificial design tasks that support thorough real-time observations of the design process. Indirect analysis, on the other hand, can be criticized for its loss of accuracy and objectiveness that is introduced by the mediating proxy. In distributed design scenarios, the costs of data collection methods are even more aggravated, as multiple design locations and concurrent, individual activities need to be considered. The acquisition of data is ultimately decelerated, allowing only for a retrospective analysis of the process, rather than taking a live view on the researched subject.

As a result, field observations and experiments in design research are often coarse or limited to a small number of samples. More efficient data collection approaches are required to increase the resolution of embedded design observations. In this context, the increasing proliferation of online information handling and digital design knowledge repositories has been identified to provide worthwhile sources for design research [22]. How can this information be efficiently leveraged and what can we learn from it?

2.3 Challenges for IT Systems Engineering

Challenges in the design of IT systems to support design research stem from the complexities in the targeted multi-user environment. These are defined by a framework of requirements related to human behavior, organizations, and technology [15]. Conceptual design is inherently unstructured and ad-hoc, rendering the computational measurement, comparison, and prediction of complex process variables idiosyncratic [8]. The possibilities for automated data collection and interpretation of team behavior are naturally limited.

A second dimension of complexity is spanned by a constantly transforming tool landscape in engineering design, which is and will remain heterogeneous. The unstruc-

[1] In respect of the ambivalent meaning of this notion, especially in computer science, *real-time* refers here to the ability to collect and prepare data from user observations steadily and almost instantaneously, i.e., within the range of minutes.

tured and varying nature of team interaction in the different phases of design demands for a diverse set of support tools with different strengths and characteristics. Currently, it is common to evaluate technology-mediated team communication based on the observation of a single domain or communication channel only. However, any observation and analysis focussed on a single communication tool or collaboration suite can comprise only fractional parts of the information handling. In order to complete the picture, capabilities to centrally monitor and access a broader range of distributed and concurrent communication activities are needed. What are the characteristics of a computer system to record, formalize and leverage those activities?

3 Studying Team Communication to Improve Innovation Potential

The long-term objective of this research is ultimately shaped by the ambition to extend knowledge about the conceptual design process and ways to increase their innovation potential. The dissertation aims to contribute to this goal by improving design research methodology with an instrument for the incessant capture and evaluation of online communication activities.

3.1 Rationale

Building on previous works in design theory and shifting paradigms in IT-supported collaboration, three outside factors predominantly provide the underlying rationale for this study. In the following, I argue that the way design project teams work, communicate, and interact in a computer-mediated environment ultimately affects the potential of an organization to grow and innovate. Therefore, IT-based communication demonstrates a supplemental, yet important subject for design research.

3.1.1 Economic Growth Needs Innovation

The dynamics of economic life has always been influenced by wave-like movements, many times triggered by new technology and innovation entering the market [18]. It is generally accepted that innovation is a motor for economic growth and should therefore be maximized [1]. Companies are constantly challenged with what has been described by Joseph A. Schumpeter as *creative destruction*: the incessant generation of viable solutions, products, or services, which is imperative to stay ahead and survive in a global market [27].

It is necessary to understand how innovation occurs in order to systematically increase innovative potential. The foundation for innovation is laid through new concept creation during the ”fuzzy front-end of innovation” [16]. This front-end of innovation is poorly understood and presents one of the greatest opportunities for improving the innovation process [17, 25]. Iterative methodologies such as *user-centered design* [23]

and *design thinking* [3] have been shown to provide successful approaches to encourage the generation of new concepts during the front-end of innovation [10, 33]. At the same time, there is little understanding of how design thinking or user-centeredness can be measured and evaluated.

3.1.2 Innovation Needs Communication

Schoen [26] describes design as a reflective conversation with the materials of a design situation. He sketches design as the process of seeing-moving-seeing: interpreting and making sense of the world (seeing), performing actions to affect a desired change (moving), and assessing the effects in the changed environment (seeing), which causes further actions or the completion of the design cycle [4]. In a team-based environment, the design situation is constantly manipulated in a social, interacting, and collaborative manner. This perspective on design reveals the importance of communicating and sharing a precise picture of relevant design information to other process participants. Having a holistic view on the fluctuating design situation in a team, i.e., achieving common ground among the stakeholders, is a crucial, yet challenging task in engineering design [24].

The correlation between communication and the outcome of a design process has been subject of many previous works. Essays such as Mythical Man-Month [2], Peopleware [7], but also other more research oriented studies [9, 31] have shown over and over that efficient team communication is a dominant factor for the well-being and success of engineering projects. Recent studies of the design process [6, 30] further suggest that communication within design teams is instrumental to successful design activity. While it is generally accepted that communication is a critical factor for design, only little is known about how certain communication behavior influences the outcome quality.

3.1.3 Communication Needs IT

The increasing proliferation of software services in computer-supported co-operative work (CSCW) and virtual collaboration has changed work environments and the way people communicate and share information. Groupware applications have moved to the Internet and the World Wide Web, providing an open interface to connecting people and information. In fact, communicating over the Internet has become standard and indispensable, especially in global organizations. About 62% of all employed Americans have Internet access and virtually all of those (98%) use email on the job [13]. Web 2.0 services and other communication methods such as instant messaging are increasingly moving into the workplace [29]. Storing and distributing project data in online services creates an unequaled level of information availability. This expansion of online instruments in the information handling and negotiation activities of design teams can be leveraged to better understand communication structures during the early phases of innovation. With the digital footprint of communication growing, computational observation and analysis methods for team processes become potentially more feasible and powerful.

3.2 Digital Traces as Surrogates for Communication Behavior

While the connections between economic growth, innovative potential, and communication are undisputed in literature, the correlations between the *technical* encoding of online communication and its *semantic* value in a collaborative environment are relatively unexplored. It is probably the most fundamental question in the computational analysis and interpretation of unstructured communication, to what extent the technical representation can serve as a surrogate in the interpretation of the original intent of a message. This is a problem deeply rooted in the nature of communication itself. Shannon and Weaver [28] count it to the category of semantic problems: how precisely do the observable symbols convey the desired meaning?

Recent studies give first evidence that data at the technical level of communication might be an observable surrogate for the semantic intent [22]. If and how data in the form of recorded interaction events could ultimately serve as an indicator of differences in design team activity and innovative performance remains largely an open question. If dependencies between observable patterns in team communication and the objective qualities of a design process can be identified, the envisioned instrument would qualify itself as an improvement to design research methodology. The case study and data analysis conducted later in this work sought to explore and interpret such correlations. The question if and how this kind of observation itself effects the behavior of the observed designers is interesting and relevant, but is outside the scope of this work.

4 A Service Approach to Real-time Communication Analysis

The dissertation sought to expand the exploration window in empirical design research by providing new capabilities for automated data collection and analysis. The focus is on technology-enabled, distributed engineering design spaces, in which computer-mediated team interactions are common and widespread. Responding to the diversity of virtual collaboration environments, the work introduces a distributed approach to capturing and measuring team interactions from miscellaneous online communication streams. A set of software services provides the functionality to incrementally capture and query detailed communication properties. This establishes a central point of access to the communication records and meta-information about how design teams virtually communicate information over the course of a project.

4.1 Research Questions and Hypothesis

In the effort to construct, apply, and evaluate an automated, service-based approach to design team observation, this work raises and answers three principal research questions.

Research Question 1. How can the chronological appearance of design team interactions be modeled and represented in a computer-processable format?

The computer-supported analysis of team collaboration requires the formalization and recording of activities under observation. The first research question is asking for an appropriate data structure, which is able to maintain a temporal representation of the identified collaboration properties. The requirements for the formalization of generally informal activities are complex. To ensure applicability in the myriad of different scenarios, the data structure cannot be designed for a particular predetermined environment or project, nor must it interfere with the natural creative modes of the subjects under study. This raises demand for a generic data schema, which supports the flexible configuration of extensible, yet unambiguous semantics.

Research Question 2. What are the structural and dynamic properties of a software system that facilitates the integration of concurrent communication capture and analysis into dispersed design environments?

The second question addresses the architectural layout of a software system to handle the formalization process and to provide capabilities for data inspection. The system design clearly needs to respond to the peculiarities of prevailing design environments without interfering with their existing setups. Multiple workspaces distributed in time and space, concurrent team interactions and a diversity of media and groupware for virtual collaboration demand for a scalable solution. Service-orientation defines an architectural paradigm to construct distributed and loosely coupled software systems that promote integrability, flexibility, and reusability [12]. The instantiation of a service-based software system to conduct computational observations and analyses in distributed and heterogeneous design environments is new and unprecedented. What services does this system need to provide and how can it be integrated into the design process?

Research Question 3. Which communication patterns in conceptual design may serve as observable surrogates for applied design thinking principles?

To scrutinize the value of the collected data for design research, the dissertation tries to identify recurring characteristics that may stand for ‘designerly ways’ of interacting during the creation of new concepts [5]. Design thinking is a methodology to stimulate creativity and innovation in earl-stage engineering by integrating a set of principles, attitudes, and methods into the design process [3]. How is design thinking reflected in the online communication behavior of teams? Can we observe structural properties (patterns) that indicate the observance of design thinking principles? Focusing on potential design thinking proxies in the captured communication activities provides a meaningful starting point for the analysis of performance correlations. Revealing such correlations through computational observation and analysis would endorse the presented approach.

In the context of this research question, the following hypothesis is being formulated:

Hypothesis. The computational analysis of online communication in conceptual design processes can reveal quantitative characteristics that correlate with independent team performance measures.

Identifying a significant correlation between properties of the formalized communication behavior and independently measured team performance is a necessary, but not sufficient requirement for the computational evaluation of design process qualities. Too many indeterminable factors are influencing team performance, preventing a complete and definite assertion by means of IT. However, correlations could provide indicators for what is relevant and worth to observe in a design process. The presented solution would suggest itself for the implementation of a design management dashboard to monitor those indicators at real-time.

4.2 Step 1: Development of a Model for Team Collaboration Capture

The development of an appropriate data model to describe the individual communication activities of design teams is a necessary first step for the analysis. While the observation of technology-mediated design activities has been the subject of many previous works, a more flexible and generic approach is required, which supports application in the diversity of existing and future IT-supported design environments. The work introduces *team collaboration networks* (*TCN*), a graph structure in which the occurrence, attributes, and relationships of heterogeneous actors and information objects are represented over the course of a project. A system of team collaboration networks (*TCN-S*) combines multiple network instances to support the conduction of parallel team observations. A TCN-S is organized by a set of domain- and network-specific ontologies to support integration, re-usability, and comparison of individual network structures.

4.3 Step 2: Service Implementation & Testbed Integration

In a second step, the work introduces *d.store*, a resource-based implementation of a TCN-S. The platform provides a rich API for the continuous recording and evaluation of virtual collaboration activities in heterogeneous groupware applications. The services were applied in an eight-month period of early stage concept creation in eleven engineering design projects and used to collect data from email archives, wiki pages, and online shared folders. The generated team collaboration networks provide the basis for the upcoming exploration of the communication structures.

4.4 Step 3: Case Study in Team Performance Evaluation

Finally, a case study is conducted to evaluate the captured online activities in design collaboration with the developed tools and to test the hypothesis. In the focus of this analysis are patterns, which may reflect applied design thinking principles in the collected data sets. In particular, the work seeks to identify structures in email and wiki-based communication that correlate with the different, independent team performance measures. Correlations would reveal potential indicators for beneficial or detrimental process characteristics. With relevant, observable metrics in place, the *d.store* plat-

form would legitimate itself as a foundation for team observations continuing research in real-time design process evaluation and management.

5 Research Methodology

The three illustrated phases of this work are guided by the principles of research in information systems (IS). The IS research framework developed by Hevner and March [14] forms a symbiosis of *design science* and *behavioral science*, defining a bilateral process to create knowledge *and* to contribute to a socio-technological environment. This synergetic approach to information systems research is visualized in Figure 1.

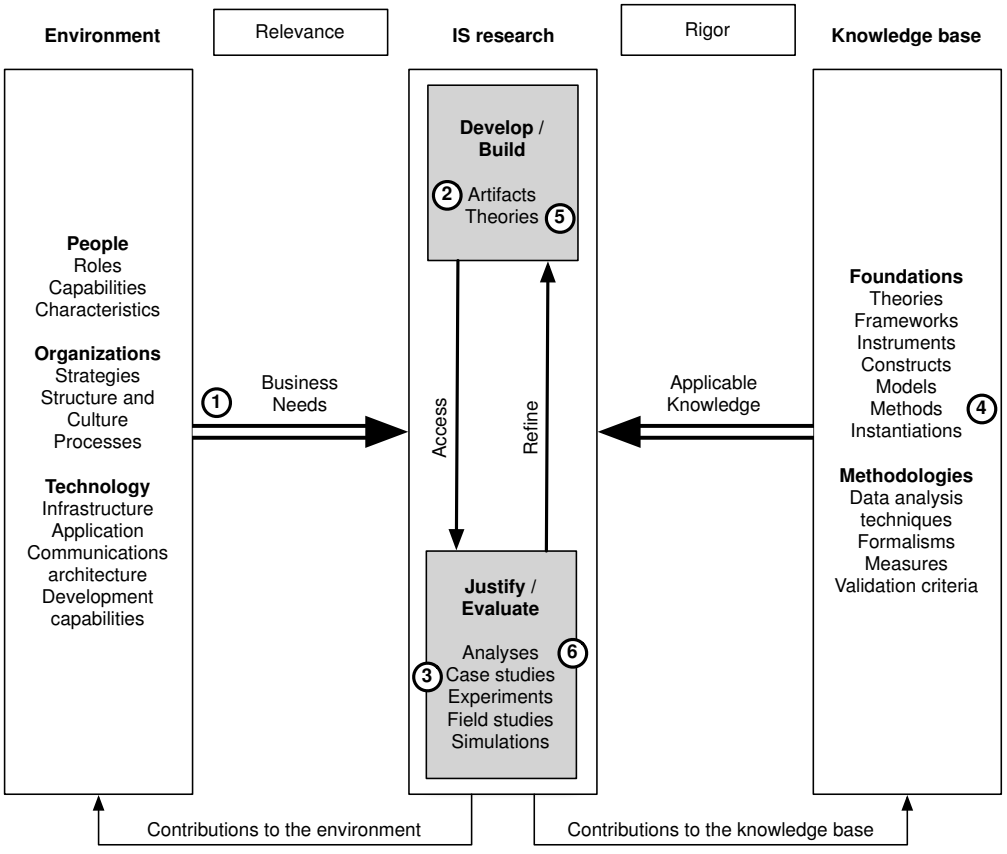


Figure 1: Research framework by Hevner and March [14].

Design science seeks to extend the boundaries of human and organizational capabilities by creating new and innovative artifacts [14]. It attempts to create things that serve human purpose [20]. This work's purpose is to create a tool that supports design researchers by enabling new observation and analysis capabilities. In this context, the work starts with a detailed exploration of the environment in design research and practice. It identifies needs and requirements in the IT-based conduction, observation, and analysis of innovation processes in conceptual design (1, Fig. 1). Building on existing

instruments, models, and formalisms, the work continues with developing software artifacts to better address the identified needs in design research (2). A service-based solution is presented, which provides a flexible system for capturing and leveraging observed communication structures of design teams. The created models and tool instantiations are evaluated in a case study with engineering design teams (3).

Behavioral science has its roots in natural science research methods. It seeks to develop hypotheses and empirically justify theories that explain or predict organizational and human phenomena surrounding the use of information systems [14]. This work conducts an analysis of the communication behavior of engineering teams (4, Fig. 1). Addressing the organizational and technological environment of conceptual engineering design processes, theories about the explanatory power of online communication structures are developed (5). With the data collected during the project case studies, it is tested whether these structures can be used to predict independent team performance measures (6).

6 Contribution

The contribution of this work to the field of design research is expected to entail beneficial input to both, theory and praxis. This section shall briefly highlight the results of the research process in terms of input to a common knowledge base and to the research environment (cf. Figure 1).

6.1 Contributions to the Knowledge Base

With team collaboration networks, the work presents a solution to a research problem rooted in the formal requirements of computational models and the informality of design processes. How can the unstructured collaboration activities in conceptual design be captured for a computer-based observation and real-time analysis? A system of networks constitutes a configurable semantic model for representing and exploring meta-information about the temporal relationships between actors and arbitrary information assets in multiple design projects.

An analysis of communication artifacts that were scanned in different groupware activities of real-live design teams shows that this approach is expedient. Correlations between the observed online communication behavior and supporting performance measurements could be demonstrated for the analyzed use cases, suggesting beneficial impact of design thinking principles. In particular, a significant interdependency between the results of a team performance diagnostic [34] and two independent variables in the email-based communication could be identified ($R^2 = 0,48$).

Other recent design research studies already build on the results of this work. Skogstad [30] has developed new theory about how designers gain the insights needed to create novel solutions and how reviewers can have both positive and negative effects on the design process. Parts of the hypotheses have been tested with team collaboration networks. Overall, the developed constructs for the observation of team communication provide a basis for achieving new insights into design aspects and performance-

relevant properties of virtual collaborative processes. Future research projects may build up on these foundations and refine the results through additional case studies and improved software instantiations.

6.2 Contributions to the Environment

The work provides tools and techniques to design researchers that help to study the ever-widening variety of technologies and phenomena that arise within distributed engineering team activity. The *d.store* platform largely decouples the data collection process from the actual analysis and allows for consistent and continuous observation. The service-oriented design simplifies the automated recording of distributed communication data and expedites an unobtrusive integration of team work analysis into existing and future research projects. A novel approach to real-time team diagnostics is established. The service platform also provide new capabilities for supporting design teams. With a more precise understanding of performance-relevant indicators, real-time awareness for potential drawbacks and process impediments is created. New starting points for the design and implementation of improved tools and dashboards for the management of design processes are created.

References

[1] D.B. Audretsch. Innovation, growth and survival. *International Journal of Industrial Organization*, 13(4):441–457, 1995.

[2] F.P. Brooks. *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley Reading, MA;, 1995.

[3] Tim Brown. Design thinking. *Harvard Business Review*, pages 85–92, June 2008.

[4] W.J. Clancey. *Situated cognition: On human knowledge and computer representations*. Cambridge University Press, 1997.

[5] N. Cross. Designerly ways of knowing: design discipline versus design science. *Design Issues*, 17(3):49–55, 2001.

[6] N. Cross and A. Clayburn. Observations of teamwork and social processes in design. *Design Studies*, 16(2):143–170, 1995.

[7] T. DeMarco and T. Lister. *Peopleware: Productive Projects and Teams*. Dorset House, 2 edition, 1999.

[8] K.J. Dooley and A.H. Van de Ven. Explaining complex organizational dynamics. *Organization Science*, pages 358–372, 1999.

[9] J.E. Driskell, P.H. Radtke, and E. Salas. Virtual teams: Effects of technological mediation on team performance. *Group Dynamics: Theory, Research, and Practice*, 7(4):297–323, 2003.

[10] C.L. Dym, A.M. Agogino, O. Eris, D.D. Frey, and L.J. Leifer. Engineering design thinking, teaching, and learning. *IEEE Engineering Management Review*, 34(1):65–92, 2006.

[11] O. Eris. *Perceiving, Comprehending, and Measuring Design Activity Through the Questions Asked while Designing*. PhD thesis, Stanford University, Stanford, CA, 2002.

[12] T. Erl. *Service-oriented architecture: concepts, technology, and design*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[13] Deborah Fallows. Email at work. *Pew Internet & Americal Life Project*, 2002.

[14] A.R. Hevner and S.T. March. The information systems research cycle. *IEEE Computer*, 36(11):111–113, 2003.

[15] A.R. Hevner, S.T. March, J. Park, and S. Ram. Design Science in Information Systems Research. *MIS Quarterly*, 28(1):75–106, 2004.

[16] J. Kim and D. Wilemon. Strategic issues in managing innovation's fuzzy front-end. *European Journal of Innovation Management*, 5(1):27–39, 2002.

[17] P.A. Koen, G. Ajamian, S. Boyce, A. Clamen, E. Fisher, S. Fountoulakis, A. Johnson, P. Puri, and R. Seibert. Fuzzy front end: Effective methods, tools, and techniques. *The PDMA toolbook for new product development*, 2002.

[18] ND Kondratieff and WF Stolper. The long waves in economic life. *The Review of Economics and Statistics*, 17(6):105–115, 1935.

[19] Ade Mabogunje. *Measuring Conceptual Design Process Performance in Mechanical Engineering: A Question Based Approach*. PhD thesis, Stanford University, Stanford, CA, 1997.

[20] S.T. March and G.F. Smith. Design and natural science research on information technology. *Decision Support Systems*, 15(4):251–266, 1995.

[21] A. J. Milne and L. Leifer. Information Handling and Social Interaction of Multi-Disciplinary Design Teams in Conceptual Design: A Classification Scheme Developed from Observed Activity Patterns. In *Proc. of the ASME Design Theory & Methodology Conference*, 2000.

[22] Andrew J. Milne. *An Information-theoretic Approach to the Study of Ubiquitous Computing Workspaces Supporting Geographically Distributed Engineering Design Teams as Goupusers*. PhD thesis, Stanford University, Stanford, CA, 2005.

[23] J. Nielsen. Usability Engineering. *Morgan Kaufmann*, 1994.

[24] 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.

[25] DG Reinertsen. Taking the fuzziness out of the fuzzy front end. *Research technology management*, 42(6):25–31, 1999.

[26] D.A. Schön. Designing as reflective conversation with the materials of a design situation. *Research in Engineering Design*, 3(3):131–147, 1992.

[27] Joseph A. Schumpeter. *Capitalism, Socialism, and Democracy*. Harper and Brothers, New York, 1942.

[28] C.E. Shannon and W. Weaver. The mathematical theory of information. *Urbana: University of Illinois Press*, 97, 1949.

[29] Eulynn Shiu and Amanda Lenhart. How Americans use instant messaging. *Pew Internet & Americal Life Project*, 2004.

[30] Philipp Leo Skogstad. *A Unified Innovation Process Model For Engineering Designers and Managers*. PhD thesis, Stanford University, Stanford, CA, 2009.

[31] L.F. Thompson and M.D. Coovert. Teamwork online: The effects of computer conferencing on perceived confusion, satisfaction, and postdiscussion accuracy. *Group Dynamics: Theory, Research, and Practice*, 7(2):135–151, 2003.

[32] S. Vosinakis, P. Koutsabasis, M. Stavrakis, N. Viorres, and J. Darzentas. Supporting conceptual design in collaborative virtual environments. In *Proc. of 11th Panhellenic Conference on Informatics, PCI 2007*, 2007.

[33] K. Vredenburg, J.Y. Mao, P.W. Smith, and T. Carey. A survey of user-centered design practice. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves*, pages 471–478. ACM New York, NY, USA, 2002.

[34] Ruth Wageman, J. R. Hackman, and Erin V. Lehman. Team diagnostic survey: Development of an instrument. *The Journal of Applied Behavioral Science*, 41(4):373, 2005.

Handling of Closed Networks in FMC-QE

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

The following report summarizes the investigations for the handling of closed networks in FMC-QE. In a closed network, the population of service requests is fixed and the external service time is zero. While the standard FMC-QE load generation model is a good representation of many other real world scenarios, it is imprecise for this class of models. In order to cope this problem, some investigations on solution methods for closed networks have been done. As a result, the summation method, an iterative solution method for closed queuing network models, has been adapted to FMC-QE. In this report, this adaption is explained. In examples, the summation method in FMC-QE is compared to the standard FMC-QE M/M/m model and M/M/1/K models. For comparative performance values, the investigated examples are also evaluated using classical solution methods for closed queuing networks.

1 Introduction

In the following report, the performance predictions for closed queuing networks, modeled and evaluated with Queuing Theory approaches are compared to the predictions of the corresponding FMC-QE model and Tableau. Therefore, first, the networks are modeled and analyzed with Queueing Theory techniques and later on, an FMC-QE model and FMC-QE Tableaux are set up. It will be seen, that the standard FMC-QE performance evaluation techniques are not a good approximation for this class of models. Therefore, FMC-QE will be extended in order to address this class of problems.

2 Closed Tandem Network

As a simple, but significant example, first, a tandem network consisting of two servers connected to each other in a closed network is examined. First, the example is calculated using standard Queuing Theory approaches, calculating the global balance equations and solving of the linear equation system. Then the example is calculated using the standard FMC-QE load model with an external service time of zero time units. It will be seen, that there are too large approximation errors in this model. In a second FMC-QE model, the performance values are calculated using an M/M/1/K model. This model leads to correct solutions for this special case of a closed network, but in order to compute the performance values, some results of the calculations had to be known in advance. In the third FMC-QE model, the summation method [1] is adapted to FMC-QE.

2.1 Original Example

The original model, shown in figure 1 is defined in [2].

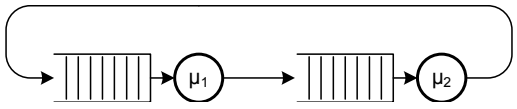


Figure 1: Closed Tandem Network - Original Model [2]

In this example, the two servers have exponentially distributed service times with mean values of 5s ($\mu_1 = \frac{1}{5}\left[\frac{SRq_1}{s}\right]$) and 2,5s ($\mu_1 = \frac{1}{2,5}\left[\frac{SRq_1}{s}\right]$) and a FCFS service discipline. There are 3 service requests in the network ($n_{ges} = 3$), which leads to the state transition diagram shown in figure 2 [2]:

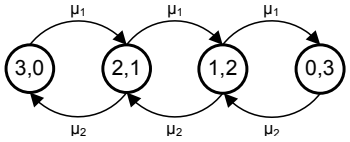


Figure 2: Closed Tandem Network - State transition diagram [2]

In [2], the global balance equations are set up as:

$$\begin{aligned} p(3,0)\mu_1 &= p(2,1)\mu_2, \\ p(2,1)(\mu_1+\mu_2) &= p(3,0)\mu_1 + p(1,2)\mu_2, \\ p(1,2)(\mu_1+\mu_2) &= p(2,1)\mu_1 + p(0,3)\mu_2, \\ p(0,3)\mu_2 &= p(1,2)\mu_1. \end{aligned} \tag{1}$$

This leads to the steady state probabilities [2]:

$$p(3,0) = 0,5333,\ p(2,1) = 0,2667,\ p(1,2) = 0,1333,\ p(0,3) = 0,0667. \tag{2}$$

Using this steady state probabilities, the marginal probabilities are computed as [2]:

$$\begin{aligned} p_1(0) = p_2(3) = p(0,3) &= 0,0667, \\ p_1(1) = p_2(2) = p(1,2) &= 0,1333, \\ p_1(2) = p_2(1) = p(2,1) &= 0,2667, \\ p_1(3) = p_2(0) = p(3,0) &= 0,5333. \end{aligned} \tag{3}$$

After computing these probabilities, the performance values, starting with the utilization ρ_i, are derived as [2]:

$$\rho_1 = 1 - p_1(0) = 0,9333,\ \rho_2 = 1 - p_2(0) = 0,4667. \tag{4}$$

The arrival rates (throughput in the closed network) are then derived as [2]:

$$\lambda = \lambda_1 = \lambda_2 = \frac{rho_1}{\mu_1} = \frac{rho_2}{\mu_2} = 0,1867\left[\frac{SRq}{s}\right]. \tag{5}$$

The mean number of service requests n_i are [2]:

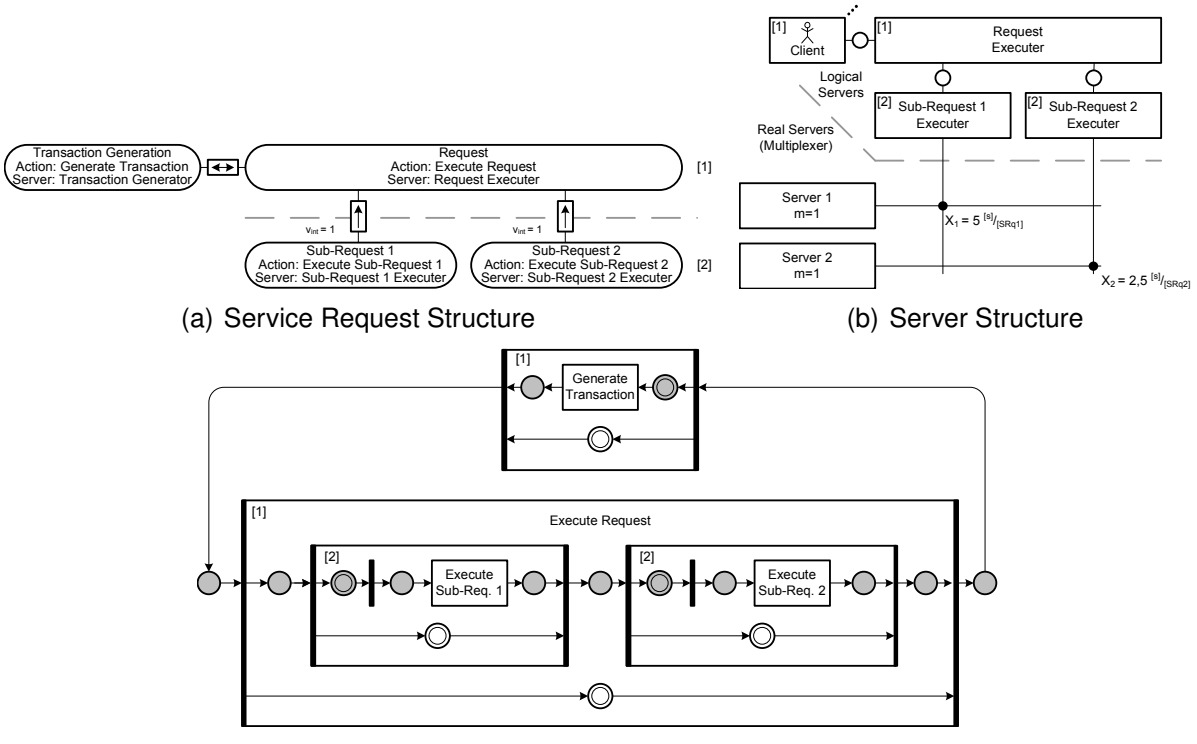
$$n_1 = \sum_{k=1}^{3} k * p_1(k) = 2,2667\,[SRq_1]\,,\ n_2 = \sum_{k=1}^{3} k * p_2(k) = 0,7333\,[SRq_2]\,. \tag{6}$$

The mean response times R_i are [2]:

$$R_1 = \frac{n_1}{\lambda_1} = 12,1429[s],\ R_2 = \frac{n_2}{\lambda_2} = 3,9286[s]. \tag{7}$$

2.2 M/M/1 Approximation

The corresponding FMC-QE model is shown in figure 3.



(a) Service Request Structure (b) Server Structure

(c) Dynamic Behavior

Figure 3: Closed Tandem Network - FMC-QE Model

In a first approximation for this closed network, the performance values are derived in an FMC-QE Tableau with M/M/1 servers:

$$\begin{aligned} n_i, q &= \frac{\rho^2}{1-\rho} \\ n_i, s &= \rho \\ n_i &= \frac{\rho}{1-\rho} \end{aligned} \tag{8}$$

and an external time $X_{ext} = 0$ (the arrival rate λ was adjusted till $X_{ext} = 0$ for $n_{ges} = 3$). This tableau is shown in table 1.

Table 1: Closed Tandem Network - M/M/1 Tableau (also in Appendix)

Experimental Parameters	
$n_{ges}^{[1]}$	3
$\lambda_{bott}^{[1]}$	0,2000
f	0,7101
$\lambda^{[1]}$	0,1420

Service Request Section								Server Section						Dynamic Evaluation Section							
[bb]	i	$SRq_i^{[bb]}$	$p_{parent(i),i}$	$v_{parent(i)}^{[bb-1]}$	$v_{i,int}^{[bb]}$	$v_i^{[bb]}$	$\lambda_i^{[bb]}$	$Server_i^{[bb]}$	$m_{parent(i)}^{[bb-1]}$	$m_{i,int}^{[bb]}$	$m_i^{[bb]}$	Mpx_i	$X_i^{[bb]}$	$m_{i,mpx}^{[bb]}$	$\mu_i^{[bb]}$	$\rho_i^{[bb]}$	$n_{i,q}^{[bb]}$	$n_{i,s}^{[bb]}$	$n_i^{[bb]}$	$R_i^{[bb]}$
3	1	Sub-Request 2	1,00	1	1	1	0,1420	Executer 2	1	1	1	2	2,5000	1,0000	0,4000	0,3551	0,1955	0,3551	0,5505	3,8763
3	2	Sub-Request 1	1,00	1	1	1	0,1420	Executer 1	1	1	1	1	5,0000	1,0000	0,2000	0,7101	1,7394	0,7101	2,4495	17,2474
1	3	Request	1,00	1	1	1	0,1420	Executer	1	1	1				0,2000		1,9348	1,0652	3,0000	21,1237
1	4	Request Generation	1,00	1	1	1	0,1420	Client	1	1	1		0,0000		######			0,0000	0,0000	0,0000

Multiplexer Section			
j	$Name_j$	m_j	$X_j^{[1]}$
1	Server 1	1	2,5000
2	Server 2	1	5,0000

The approximation errors (relative error $\delta x_i = \frac{\Delta x_i}{x}$ [3]) in the prediction of the overall arrival rate:

$$\begin{aligned} \lambda_{GlobalBalanceCalculation} &= 0,1867,\ \lambda_{M/M/1\ Approx.} = 0,1420 \\ \delta_\lambda &= \frac{|\lambda_{GlobalBalanceCalculation} - \lambda_{M/M/1\ Approx.}|}{|\lambda_{GlobalBalanceCalculation}|} = 0,239 \end{aligned} \tag{9}$$

and the response times of the servers (especially R_1):

$$\begin{aligned} R_{1,GlobalBalanceCalculation} &= 12,1429,\ R_{1,M/M/1\ Approx.} = 17,2474, \\ R_{2,GlobalBalanceCalculation} &= 3,9286,\ R_{2,M/M/1\ Approx.} = 3,8763 \\ \delta_{R_1} &= \frac{|R_{1,GlobalBalanceCalculation} - R_{1,M/M/1\ Approx.}|}{|R_{1,GlobalBalanceCalculation}|} = 0,296 \\ \delta_{R_2} &= \frac{|R_{2,GlobalBalanceCalculation} - R_{2,M/M/1\ Approx.}|}{|R_{2,GlobalBalanceCalculation}|} = 0,013 \end{aligned} \tag{10}$$

are very high, because in this solution, the number of service requests in the system is only a mean number derived by calculations for open networks and not a constant value as usually for closed networks.

2.3 M/M/1/K Model

In a second calculation, the two servers are represented by M/M/1/K servers with a capacity of $K = 3$ and so the server formulas are:

$$
\begin{aligned}
\rho_i &= \frac{\lambda i}{\mu_i} \\
n_{i,q} &= \begin{cases} \frac{\rho_i}{1-\rho_i} - \frac{\rho_i\left(K\rho_i^K+1\right)}{1-\rho_i^{K+1}} & \rho_i \neq 1 \\ \frac{K(K-1)}{2(K+1)} & \rho_i = 1 \end{cases} \\
n_{i,s} &= \begin{cases} 1 - \frac{1-\rho_i}{1-\rho_i^{K+1}} & \rho_i \neq 1 \\ 1 - \frac{1}{K+1} & \rho_i = 1 \end{cases} \\
n_i &= \begin{cases} \frac{\rho_i}{1-\rho_i} - \frac{K+1}{1-\rho_i^{K+1}}\rho_i^{K+1} & \rho_i \neq 1 \\ \frac{K}{2} & \rho_i = 1 \end{cases}
\end{aligned}
\tag{11}
$$

In this tableau, shown in table 2, the different arrival rates λ_i are adjusted using the overall arrival rate λ and the traffic flow coefficients v_i in order to fit with the value $0,1867$ of the original example [2], calculated via the global balance equation.

Table 2: Closed Tandem Network - M/M/1/K Tableau (also in Appendix)

Experimental Parameters	
$n_{ges}^{[1]}$	3
$\lambda^{[1]}$	0,4000

Service Request Section									Server Section						Dynamic Evaluation Section							
[bb]	i	$SRq_i^{[bb]}$	$p_{parent(i),i}$	$v_{parent(i)}^{[bb-1]}$	$v_{i,int}^{[bb]}$	$v_i^{[bb]}$	$\lambda_i^{[bb]}$	$\lambda_{i,eff}^{[bb]}$	$Server_i^{[bb]}$	$m_{parent(i)}^{[bb-1]}$	$m_{i,int}^{[bb]}$	$m_i^{[bb]}$	Mpx_i	$X_i^{[bb]}$	$m_{i,mpx}^{[bb]}$	$\mu_i^{[bb]}$	$\rho_i^{[bb]}$	$n_{i,q}^{[bb]}$	$n_{i,s}^{[bb]}$	$n_i^{[bb]}$	$R_i^{[bb]}$
3	1	Sub-Request 2	1,00	1	0,5	0,5	0,2000	0,1867	Executer 2	1	1	1	2	2,5000	1,0000	0,4000	0,5000	0,2667	0,4667	0,7333	3,9286
3	2	Sub-Request 1	1,00	1	1	1	0,4000	0,1867	Executer 1	1	1	1	1	5,0000	1,0000	0,2000	2,0000	1,3333	0,9333	2,2667	12,1429
2	3	Request	1,00	1	1	1	0,4000	0,1867	Executer	1	1	1				0,2000		1,6000	1,4000	3,0000	16,0714
1	4	Request Generation	1,00	1	1	1	0,4000		Client	1	1	1		0,0000		#####			0,0000	0,0000	0,0000

Multiplexer Section			
j	$Name_j$	m_j	$X_j^{[1]}$
1	Server 1	1	5,0000
2	Server 2	1	1,2500

In this special case of this tandem network, the performance values are exactly the same for the M/M/1/K model and the calculation via the global balance equations, but this in not true for every closed network (in the second example of this section the values are not correct) and for the calculation of this model, the results (especially the arrival rates) had to be known in advance in order to adjust the traffic flow coefficients for this model or a more complex equation system had to be solved in order to retrieve the results. In this closed tandem model, the effective arrival rates:

$$
\lambda_{i,eff} = \begin{cases} \lambda\left(1 - \frac{1-\rho_i}{\frac{1}{\rho_i^K}-\rho_i}\right) & \rho_i \neq 1 \\ \lambda\left(1 - \frac{1}{K+1}\right) & \rho_i = 1 \end{cases}
\tag{12}
$$

had to be the same for every server. So the arrival rates λ_i had to be adjusted through the traffic flow coefficient $\lambda_i = v_i * \lambda$. Also the overall number of service requests in the system n_{ges}, with:

$$
n_i = \begin{cases} \frac{\rho_i}{1-\rho_i} - \frac{K+1}{1-\rho_i^{K+1}}\rho_i^{K+1} & \rho_i \neq 1 \\ \frac{K}{2} & \rho_i = 1 \end{cases}
\tag{13}
$$

and

$$n_{ges} = \sum_{i=1}^{2} n_i \tag{14}$$

had to be 3[SRq] ($n_{ges} = 3[SRq]$). For this proof-of-concept tandem network, the M/M/1/K model was of course calculable, but for larger networks, this model and calculation is not feasible.

2.4 Summation Method

In FMC-QE, the arrival rate is an input parameter and the number of service requests in the system are a result. In closed systems, this is normally the opposite, so classical approaches do not fit. The summation method [1] are an exception, while there, the arrival rate is also an input parameter in a closed model.

In the summation method, the mean number of service requests in each node are a function of the throughput of the node [2]:

$$n_i = f_i(\lambda_i). \tag{15}$$

[1] propose the following formulas for $f_i(\lambda_i)$ [2]:

$$f_i(\lambda_i) = \begin{cases} \frac{\rho_i}{1-\frac{K-1}{K}\rho_i}, & Type-1,2,4\ (m_i = 1), \\ m_i\rho_i + \frac{\rho_i}{1-\frac{K-m_i-1}{K-m_i}\rho_i} p_i(m_i), & Type-1\ (m_i > 1), \\ \frac{\lambda_i}{\mu_i}, & Type-3. \end{cases} \tag{16}$$

with the utilization [2]:

$$\rho_i = \frac{\lambda_i}{m_i\mu_i} \tag{17}$$

and the waiting probabilities (for Type-3 Server, $m_i > 1$) [2]:

$$p_i(m_i) = \left(\frac{m_i!\,(1-\rho_i) \sum_{k=0}^{m_i-1} \frac{(m_i\rho_i)^k}{k!}}{(m_i\rho_i)^{m_i}} + 1 \right)^{-1} \tag{18}$$

The function $f_i(\lambda_i)$ is correct for Type-3 servers (infinite servers) and an approximation for Type-1,2 and 4 [2].

If f_i is given for every basic server station (number of basic server stations = I) in the network, the overall number of service requests in the system is given by [2]:

$$\sum_{i=1}^{I} n_i = \sum_{i=1}^{I} f_i(\lambda_i) = K \tag{19}$$

and including the traffic flow coefficients v_i, the overall number of service requests in the system is a function of the arrival rate [2]:

$$\sum_{i=1}^{I} f_i(v_i\lambda) = g(\lambda) = K. \tag{20}$$

For the usage of the summation method in the Tableau, the basic server formulas are substituted by the summation formulas (16) and then the solution is derived in an iterative calculation, modified from [2]:

While the desired bottleneck utilization f ($\lambda = f * \lambda_{bott}$) is 0 for the lower bound of the arrival rate and 1 for the upper bound (arrival rate λ = bottleneck throughput λ_{bott}), the bounds are $f_l = 0$ and $f_u = 1$ in the first step. (desired bottleneck utilization $f \neq f_i$)

Then in the second step, $f = \frac{f_l + f_u}{2}$ and the tableau is solved. If the overall number of service requests in the system are $K \pm \epsilon$, then the solution is found, else the bounds are set to

- $f_u = f$ if the overall number of service requests in the system is greater than K and
- $f_l = f$ if the overall number of service requests in the system is smaller than K.

and then the next iteration starts with the second step.

Table 3 shows the corresponding Tableau of the closed tandem network example.

Table 3: Closed Tandem Network - Summation Method Tableau (also in Appendix)

Experimental Parameters	
$n_{ges}^{[1]}$	3
$\lambda_{bott}^{[1]}$	0,2000
f	0,9144
$\lambda^{[1]}$	0,1829

Service Request Section								Server Section						Dynamic Evaluation Section						
[bb]	i	$SRq_i^{[bb]}$	$p_{parent(i),i}$	$v_{parent(i)}^{[bb-1]}$	$v_{i,int}^{[bb]}$	$v_i^{[bb]}$	$\lambda_i^{[bb]}$	$Server_i^{[bb]}$	$m_{parent(i)}^{[bb-1]}$	$m_{i,int}^{[bb]}$	$m_i^{[bb]}$	Mpx_i	$X_i^{[bb]}$	$m_{i,mpx}^{[bb]}$	$\mu_i^{[bb]}$	$\rho_i^{[bb]}$	$n_{i,q}^{[bb]}$	$n_{i,s}^{[bb]}$	$n_i^{[bb]}$	$R_i^{[bb]}$
3	1	Sub-Request 2	1,00	1	1	1	0,1829	Executer 2	1	1	1	2	2,5000	1,0000	0,4000	0,4572	0,2005	0,4572	0,6577	3,5961
3	2	Sub-Request 1	1,00	1	1	1	0,1829	Executer 1	1	1	1	1	5,0000	1,0000	0,2000	0,9144	1,4279	0,9144	2,3423	12,8078
1	3	Request	1,00	1	1	1	0,1829	Executer	1	1	1				0,2000		1,6284	1,3716	3,0000	16,4039
1	4	Req. Generation	1,00	1	1	1	0,1829	Client	1	1	1		0,0000		#####			0,0000	0,0000	0,0000

Multiplexer Section			
j	$Name_j$	m_j	$X_j^{[1]}$
1	Server 1	1	2,5000
2	Server 2	1	5,0000

The approximation errors in the prediction of the overall arrival rate and the response times of the servers are:

$$\begin{aligned}
\lambda_{GlobalBalanceCalculation} = 0,1867,\ \lambda_{SUM\ Approx.} = 0,1829 \\
\delta_\lambda = \frac{|\lambda_{GlobalBalanceCalculation} - \lambda_{SUM\ Approx.}|}{|\lambda_{GlobalBalanceCalculation}|} = 0,0204 \\
R_{1,GlobalBalanceCalculation} = 12,1429,\ R_{1,SUM\ Approx.} = 12,8078, \\
R_{2,GlobalBalanceCalculation} = 3,9286,\ R_{2,SUM\ Approx.} = 3,5961 \\
\delta_{R_1} = \frac{|R_{1,GlobalBalanceCalculation} - R_{1,SUM\ Approx.}|}{|R_{1,GlobalBalanceCalculation}|} = 0,0548 \\
\delta_{R_2} = \frac{|R_{2,GlobalBalanceCalculation} - R_{2,SUM\ Approx.}|}{|R_{2,GlobalBalanceCalculation}|} = 0,0846
\end{aligned} \tag{21}$$

3 Closed Central Server Example

The second example is a classical CPU - Disk(s) closed model (Closed Central Server). In this part, a solution following the Theorem of Gordon and Newell [4] is compared to the FMC-QE summation method.

3.1 Original Model

The original example is the example 4.3-2 in [5], re-engineered in figure 4.

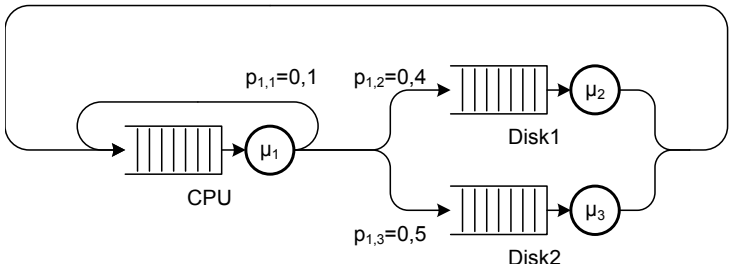


Figure 4: Closed Central Server Example - Original Model Reengineered

In [5] the model was calculated using the Theorem of Gordon and Newell [4]. For a system with three Service Requests inside ($K = 3[SRq]$) the calculations lead to the following results [5] (recalculated manual and using WinPEPSY [6][1] for higher precision):

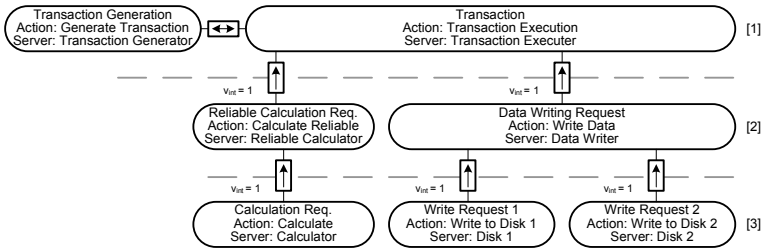
$$\begin{array}{llllll} e_1 & = 1,000, & e_2 & = 0,400, & e_3 & = 0,500, \\ \rho_1 & = 0,6939, & \rho_2 & = 0,3469, & \rho_3 & = 0,6939, \\ D_1 & = 0,3469, & D_2 & = 0,1388, & D_3 & = 0,1735, \\ n_{1,q} & = 0,5714, & n_{2,q} & = 0,1224, & n_{3,q} & = 0,5714, \\ n_{1,s} & = 0,6939, & n_{2,s} & = 0,3469, & n_{3,s} & = 0,6939, \\ n_1 & = 1,2653, & n_2 & = 0,4693, & n_3 & = 1,2653, \\ R_1 & = 3,6471, & R_2 & = 3,3824, & R_3 & = 7,2941, \\ & & R & = 8,6471 & & \end{array} \tag{22}$$

3.2 FMC-QE Model

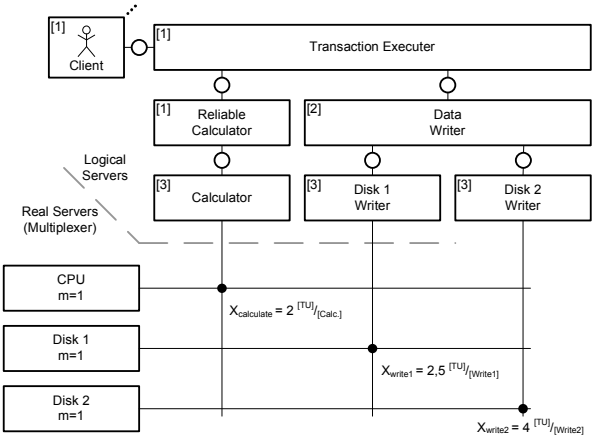
In the corresponding FMC-QE model, shown in figure 5, some transformations have been done. The different steps will not be shown here in detail, but these were a transformation to a Petri Net, the integration of the external load generation, a feed-forward - feed-backward transformation for the loop around the CPU, a hierarchy for the disk branch and another hierarchy for the whole service request.

The corresponding Tableau in table 4 shows the performance predictions of the FMC-QE model.

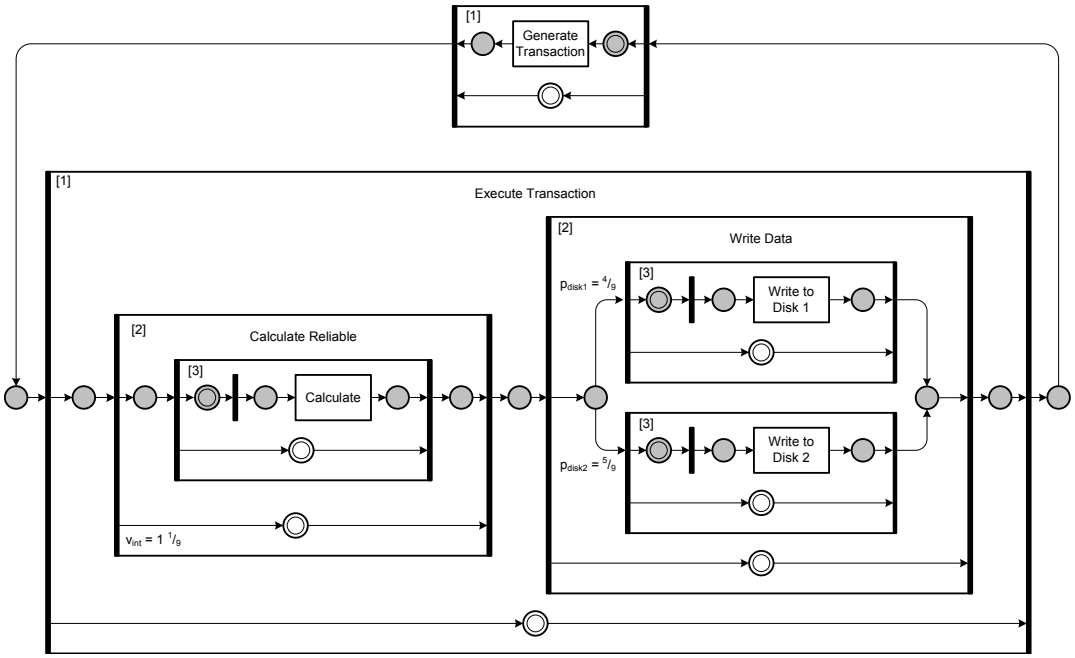
[1] http://www7.informatik.uni-erlangen.de/ prbazan/pepsy/



(a) Service Request Structure



(b) Server Structure



(c) Dynamic Behavior

Figure 5: Closed Central Server Example - FMC-QE Model

Table 4: Closed Central Server Example - Tableau (also in Appendix)

Experimental Parameters	
$n_{ges}^{[1]}$	3
$\lambda_{bott}^{[1]}$	0,4500
f	0,6895
$\lambda^{[1]}$	0,3103

Service Request Section								Server Section						Dynamic Evaluation Section						
[bb]	i	$SRq_i^{[bb]}$	$p_{parent(i),i}^{[bb-1]}$	$v_{parent(i)}^{[bb-1]}$	$v_{i,int}^{[bb]}$	$v_i^{[bb]}$	$\lambda_i^{[bb]}$	$Server_i^{[bb]}$	$m_{parent(i)}^{[bb-1]}$	$m_{i,int}^{[bb]}$	$m_i^{[bb]}$	Mpx_i	$x_i^{[bb]}$	$m_{i,mpx}^{[bb]}$	$\mu_i^{[bb]}$	$\rho_i^{[bb]}$	$n_{i,q}^{[bb]}$	$n_{i,s}^{[bb]}$	$n_i^{[bb]}$	$R_i^{[bb]}$
3	1	Write Request 2	0,56	1,000	1,000	0,556	0,1724	Disk2 Writer	1	1	1	3	4,0000	1,0000	0,2500	0,6895	0,5866	0,6895	1,2762	7,4031
3	2	Write Request 1	0,44	1,000	1,000	0,444	0,1379	Disk1 Writer	1	1	1	2	2,5000	1,0000	0,4000	0,3448	0,1029	0,3448	0,4477	3,2461
2	3	Data Writing Request	1,00	1,000	1,000	1,000	0,3103	Data Writer	1	1	1				0,4500		0,6895	1,0343	1,7238	5,5556
3	4	Calculation Request	1,00	1,000	1,111	1,111	0,3448	Calculator	1	1	1	1	2,0000	1,0000	0,5000	0,6895	0,5866	0,6895	1,2762	3,7016
2	5	Reliable Calculation Req.	1,00	1,000	1,000	1,000	0,3103	Reliable Calculator	1	1	1				0,4500		0,5866	0,6895	1,2762	4,1128
1	6	Transaction	1,00	1,000	1,000	1,000	0,3103	Transaction Executer	1	1	1				0,4500		1,2762	1,7238	3,0000	9,6684
1	7	Transaction Generation	1,00	1,000	1,000	1,000	0,3103	Client	1	1	1		0,0000		######		0,0000	0,0000	0,0000	0,0000

Multiplexer Section			
j	$Name_j$	m_j	$X_j^{[1]}$
1	CPU	1	2,2222
2	Disk1	1	2,2222
3	Disk2	1	1,1111

The approximation errors (relative error $\delta x_i = \frac{\Delta x_i}{x}$ [3]) at the server level are very small (except the runaway value $n_{2,q}$ for which the error is still under 16%):

$$\begin{array}{lll} \delta_{\rho_1} = 0,0063, & \delta_{\rho_2} = 0,0061, & \delta_{\rho_3} = 0,0063, \\ \delta_{D_1} = 0,0061, & \delta_{D_2} = 0,0065, & \delta_{D_3} = 0,0063, \\ \delta_{n_{1,q}} = 0,0266, & \delta_{n_{2,q}} = 0,1593, & \delta_{n_{3,q}} = 0,0266, \\ \delta_{n_{1,s}} = 0,0063, & \delta_{n_{2,s}} = 0,0061, & \delta_{n_{3,s}} = 0,0063, \\ \delta_{n_1} = 0,0086, & \delta_{n_2} = 0,0460, & \delta_{n_3} = 0,0086, \\ \delta_{R_1} = 0,0149, & \delta_{R_2} = 0,0403, & \delta_{R_3} = 0,0149. \end{array} \tag{23}$$

On the net level (R and D), the approximation error seems very high at the first moment, but in the FMC-QE model the repetition of the calculation (loop around the *CPU*) is transformed into a higher traffic flow within the *Reliable Calculation*. The arrival rate at the highest level is $\frac{1}{1\frac{1}{9}}$ lower than in the original model, which is just another interpretation of the model and not an error. So with this ”normalization”, the approximation errors are:

$$\delta_D = 0,0061, \delta_R = 0,0063. \tag{24}$$

4 Conclusions

The first two approaches of the integration of closed Queuing Networks into FMC-QE, more precisely the approximation using M/M/m servers and an external service time of zero time units, as well as the second model of M/M/m/K servers, were either to imprecise or inapplicable. But the third approach, the integration of the summation method [1, 2] into FMC-QE, explained and exemplified in this report, extends FMC-QE to the class of closed Queuing Networks, providing an iterative algorithm and good approximation for closed systems. The approximative errors in the examples, evaluated using the summation method, are, except for the runaway value $n_{2,q}$ in the second example, within the maximal error of 15% as descibed in [1] and often even better than the average error of 5% for the number of service requests and the response time also described in [1]. The values for the throughput deliver also precise values as descibed in [1].

References

[1] Gunter Bolch, Georg Fleischmann, and R. Schreppel. Ein funktionales Konzept zur Analyse von Warteschlangennetzen und Optimierung von Leistungsgrößen. In Ulrich Herzog and Martin Paterok, editors, *Messung, Modellierung und Bewertung von Rechensystemen, 4. GI/ITG-Fachtagung, Erlangen, 29. September - 1. Oktober 1987, Proceedings*, volume 154 of *Informatik-Fachberichte*, pages 327–342. Springer, 1987.

[2] Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor Shridharbhai Trivedi. *Queueing Networks and Markov Chains : Modeling and Performance Evaluation With Computer Science Applications*. John Wiley & Sons, Inc., 1998.

[3] Ilja N. Bronstein, Konstantin A. Semendjajew, Gerhard Musiol, and Heiner Muehlig. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, Thun, Frankurt am Main, 2 edition, 1995.

[4] W. J. Gordon and G. F. Newell. Closed queueing systems with exponential servers. *Operations Research*, 15:254–265, 1967.

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

[6] Matthias Kirschnick. The Performance Evaluation and Prediction SYstem for Queueing NetworkS PEPSY-QNS. Technical Report TR-I4-18-94, Computer Science Department Operating Systems - IMMD IV, Friedrich-Alexander-University , Erlangen-Nürnberg, Erlangen, Germany, June 1994.

Appendix

Table 5: Closed Tandem Network - Tableaux

(a) M/M/1

Experimental Parameters	
$n_{ges}^{[1]}$	3
$\lambda_{bott}^{[1]}$	0,2000
f	0,7101
$\lambda^{[,]}$	0,1420

Service Request Section									Server Section							Dynamic Evaluation Section								
[bb]	i	$SRq_i^{[bb]}$	$p_{parent(i),i}$	$v_{parent(i)}^{[bb-1]}$	$v_{i,int}^{[bb]}$	$v_i^{[bb]}$	$\lambda_i^{[bb]}$	$\lambda_{i,eff}^{[bb]}$	$Server_i^{[bb]}$	$m_{parent(i)}^{[bb-1]}$	$m_{i,int}^{[bb]}$	$m_i^{[bb]}$	Mpx_i	$X_i^{[bb]}$	$m_{i,mpx}^{[bb]}$	$\mu_i^{[bb]}$	$\rho_i^{[bb]}$	$n_{i,q}^{[bb]}$	$n_{i,s}^{[bb]}$	$n_i^{[bb]}$	$R_i^{[bb]}$
3	1	Sub-Request 2	1,00		1	1	0,1420		Executer 2		1	1	2	2,5000	1,0000	0,4000	0,3551	0,1955	0,3551	0,5505	3,8763
3	2	Sub-Request 1	1,00		1	1	0,1420		Executer 1		1	1	1	5,0000	1,0000	0,2000	0,7101	1,7394	0,7101	2,4495	17,2474
1	3	Request	1,00		1	1	0,1420		Executer		1	1				0,2000		1,9348	1,0652	3,0000	21,1237
1	4	Request Generation	1,00		1	1	0,1420		Client		1	1		0,0000		#######			0,0000	0,0000	0,0000

Multiplexer Section			
j	$Name_j$	m_j	$X_j^{[1]}$
1	Server 1	1	2,5000
2	Server 2	1	5,0000

(b) M/M/1/K

Experimental Parameters	
$n_{ges}^{[1]}$	3
$\lambda^{[1]}$	0,4000

Service Request Section									Server Section							Dynamic Evaluation Section								
[bb]	i	$SRq_i^{[bb]}$	$p_{parent(i),i}$	$v_{parent(i)}^{[bb-1]}$	$v_{i,int}^{[bb]}$	$v_i^{[bb]}$	$\lambda_i^{[bb]}$	$\lambda_{i,eff}^{[bb]}$	$Server_i^{[bb]}$	$m_{parent(i)}^{[bb-1]}$	$m_{i,int}^{[bb]}$	$m_i^{[bb]}$	Mpx_i	$X_i^{[bb]}$	$m_{i,mpx}^{[bb]}$	$\mu_i^{[bb]}$	$\rho_i^{[bb]}$	$n_{i,q}^{[bb]}$	$n_{i,s}^{[bb]}$	$n_i^{[bb]}$	$R_i^{[bb]}$
3	1	Sub-Request 2	1,00		1	0,5	0,2000	0,1867	Executer 2		1	1	2	2,5000	1,0000	0,4000	0,5000	0,2667	0,4667	0,7333	3,9286
3	2	Sub-Request 1	1,00		1	1	0,4000	0,1867	Executer 1		1	1	1	5,0000	1,0000	0,2000	2,0000	1,3333	0,9333	2,2667	12,1429
2	3	Request	1,00		1	1	0,4000	0,1867	Executer		1	1				0,2000		1,6000	1,4000	3,0000	16,0714
1	4	Request Generation	1,00		1	1	0,4000		Client		1	1		0,0000		#####			0,0000	0,0000	0,0000

Multiplexer Section			
j	$Name_j$	m_j	$X_j^{[1]}$
1	Server 1	1	5,0000
2	Server 2	1	1,2500

(c) Summation Method

Experimental Parameters	
$n_{ges}^{[1]}$	3
$\lambda_{bott}^{[1]}$	0,2000
f	0,9144
$\lambda^{[,]}$	0,1829

Service Request Section									Server Section							Dynamic Evaluation Section								
[bb]	i	$SRq_i^{[bb]}$	$p_{parent(i),i}$	$v_{parent(i)}^{[bb-1]}$	$v_{i,int}^{[bb]}$	$v_i^{[bb]}$	$\lambda_i^{[bb]}$	$\lambda_{i,eff}^{[bb]}$	$Server_i^{[bb]}$	$m_{parent(i)}^{[bb-1]}$	$m_{i,int}^{[bb]}$	$m_i^{[bb]}$	Mpx_i	$X_i^{[bb]}$	$m_{i,mpx}^{[bb]}$	$\mu_i^{[bb]}$	$\rho_i^{[bb]}$	$n_{i,q}^{[bb]}$	$n_{i,s}^{[bb]}$	$n_i^{[bb]}$	$R_i^{[bb]}$
3	1	Sub-Request 2	1,00		1	1	0,1829		Executer 2		1	1	2	2,5000	1,0000	0,4000	0,4572	0,2005	0,4572	0,6577	3,5961
3	2	Sub-Request 1	1,00		1	1	0,1829		Executer 1		1	1	1	5,0000	1,0000	0,2000	0,9144	1,4279	0,9144	2,3423	12,8078
1	3	Request	1,00		1	1	0,1829		Executer		1	1				0,2000		1,6284	1,3716	3,0000	16,4039
1	4	Req. Generation	1,00		1	1	0,1829		Client		1	1		0,0000		#####			0,0000	0,0000	0,0000

Multiplexer Section			
j	$Name_j$	m_j	$X_j^{[1]}$
1	Server 1	1	2,5000
2	Server 2	1	5,0000

Table 6: Closed Central Server Example - Tableau

Experimental Parameters	
$n_{ges}^{[1]}$	3
$\lambda_{tcat}^{[1]}$	0,4500
f	0,6895
$\lambda^{[1]}$	0,3103

Service Request Section									Server Section						Dynamic Evaluation Section							
[bb]	i	$SRq_i^{[bb]}$	$p_{parent(i),i}$	$v_{parent(i),i}^{[bb-1]}$	$v_{i,int}^{[bb]}$	$v_i^{[bb]}$	$\lambda_i^{[bb]}$	$Server_i^{[bb]}$	$m_{parent(i),i}^{[bb-1]}$	$m_{i,int}^{[bb]}$	$m_i^{[bb]}$	Mpx_i	$X_i^{[bb]}$	$m_{i,mpx}^{[bb]}$	$\mu_i^{[bb]}$	$\rho_i^{[bb]}$	$n_{i,q}^{[bb]}$	$n_{i,s}^{[bb]}$	$n_i^{[bb]}$	$R_i^{[bb]}$
3	1	Write Request 2	0,56	1,000	1,000	0,556	0,1724	Disk2 Writer	1	1	1	3	4,0000	1,0000	0,2500	0,6895	0,5866	0,6895	1,2762	7,4031
3	2	Write Request 1	0,44	1,000	1,000	0,444	0,1379	Disk1 Writer	1	1	1	2	2,5000	1,0000	0,4000	0,3448	0,1029	0,3448	0,4477	3,2461
2	3	Data Writing Request	1,00	1,000	1,000	1,000	0,3103	Data Writer	1	1	1	1			0,4500		0,6895	1,0343	1,7238	5,5556
3	4	Calculation Request	1,00	1,000	1,111	1,111	0,3448	Calculator	1	1	1	1	2,0000	1,0000	0,5000	0,6895	0,5866	0,6895	1,2762	3,7016
2	5	Reliable Calculation Req.	1,00	1,000	1,000	1,000	0,3103	Reliable Calculator	1	1	1	1			0,4500		0,5866	0,6895	1,2762	4,1128
1	6	Transaction	1,00	1,000	1,000	1,000	0,3103	Transaction Executer	1	1	1	1			0,4500		1,2762	1,7238	3,0000	9,6684
1	7	Transaction Generation	1,00	1,000	1,000	1,000	0,3103	Client	1	1	1	1	0,0000		######		0,0000	0,0000	0,0000	0,0000

Multiplexer Section			
j	$Name_j$	m_j	$X_j^{[1]}$
1	CPU	1	2,2222
2	Disk1	1	2,2222
3	Disk2	1	1,1111

Modelling Security in Service-oriented Architectures

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

Service-oriented Architectures (SOA) facilitate the provision and orchestration of business services to enable a faster adoption to changing business demands. Web services provide a technical foundation to realize this paradigm and support a variety of different security mechanisms and approaches. Security requirements are codified in Web Service policies that control the service's behavior in terms of secure interactions with other participants in an SOA.

To facilitate and simplify the generation of enforceable security policies, I foster a model-driven approach based on the modelling of security intentions in system design models. These security intentions are translated to my security meta-model for SOA that is used to generate Web Service policies.

This report summarizes my research work of the past six month that has been focused on the modelling of security requirements. I defined SecureSOA as a security design language that enables the definition of security intentions for SOA. In this report, the abstract syntax and notion of SecureSOA are introduced. In addition, the schema is described that has been chosen to integrate SecureSOA in any system design language. As an example, I will demonstrate the integration of SecureSOA in Fundamental Modelling Concept (FMC) Block Diagrams.

1 Introduction

IT-infrastructures have evolved into distributed and loosely coupled service-based systems that are capable to expose company's assets and resources as business services. To implement this paradigm, the Web Service specification provide a technical foundation based on XML-messaging. In addition, Web Services facilitate the usage of different security pattern, mechanisms and algorithms to secure the interaction between the participants in an Service-oriented Architecture. These specifications enable the enforcement of security goals such as confidentiality, integrity, authentication and identification in a Web Service-based system.

Web Service policies (WS-Policy and WS-SecurityPolicy) are used to state these security requirements on a technical layer concerning the usage of Web Service specifications such as WS-Security. They are deployed at Web Services that have to enforce these requirements. In addition, polices enables services to communicate requirements to its service consumers in order to inform about e.g. required identity

information, trusted parties or required mechanisms to secure exchanged information.

However, Web Service policies are complex, hard to understand and even harder to codify. To simplify security configurations of Web Services, tool support is offered by all major Web Service platforms. These platforms provide preconfigured policies that can be selected using profiles or bindings. However, this approach requires developers to have some security knowledge, since an appropriate binding has to be choosen and additional security related configurations might still be necessary. In addition, these tools do not take the overall system architecture into consideration.

To overcome this limitation and to enable a simplified generation of security policies, I foster a model-driven approach that integrates simple security intentions in SOA system models. SOA system models provides an abstract view on different aspects such as participants, information, and workflows. The integration of security intentions enables a modeller to state basic requirements on a technically independent level. For instance, specific information can be annotated as confidential or services might require a specific set of trustworthy identity information, e.g. credit card information.

Modelling security has been a research topic in the recent years. Several approaches emerged [6, 10], but none of them is suitable to express simple security intentions that could easily be integrated in different modelling languages.

In [2] Basin and Lodderstedt introduce SecureUML that provides a security design language to describe role-based access control and authorisation constraints. In addition, they describe a general schema to integrate security modelling languages into arbitrary system design languages.

I have adapted this schema to provide a security modelling language that enables the definition of security intentions in SOA. This language is called SecureSOA and is the foundation of my model-driven approach. Security intentions modelled in a language based on SecureSOA are translated to my meta-model for Security in SOA that has been introduced in [7]. This translation is driven by security pattern as described in [8]. Finally, the security requirements defined in the meta-model are used to generate enforceable WS-SecurityPolicy.

The structure of this report is as follows. In Section 2 I will outline my model-driven approach for SOA. Section 3 introduces my concept to enhance design modelling languages with security intentions, while SecureSOA is described as a security design language to model these intentions in the next Section. A SecureSOA dialect based on FMC is introduced in Section 5. The next Section presents a use case that is modelled using the dialect. Section 7 presents related work, while Section 8 concludes the report.

2 Model-driven Security in SOA

My model-driven approach should enable SOA Architects to state security intentions at the modelling layer and to facilitate a generation of enforceable security configurations [7]. As illustrated in Figure 1, my approach consist of three layers.

To state security requirements, I foster an annotation of system design models e.g. FMC block diagrams or BPMN models with security intentions that are defined by my

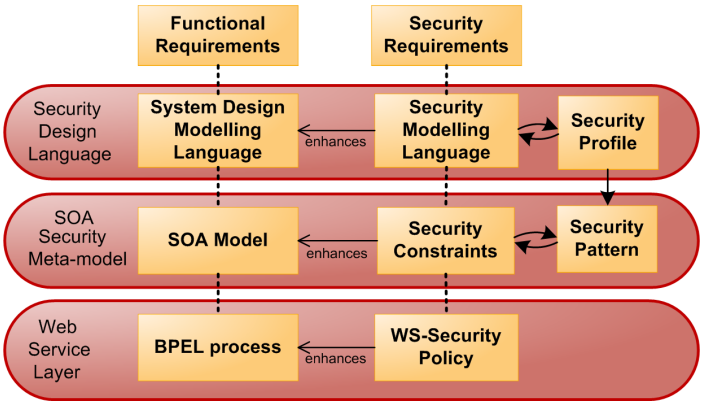


Figure 1: Model-driven Security in SOA

security modelling language SecureSOA. These languages are combined in a modelling dialect that combines modelling elements from both languages, as described in the next section. The modelled security intentions represent security goals that must be enforced and refer to a security profile. Profiles are used to abstract from technical details that should be hidden from the modeller. Instead of specifying, for instance, the algorithms, key strength and other technical details, the modelled instance of an intention refers to a profile that provide this information.

However, intentions and profiles are not sufficient to generate security policies, since additional technical information are required. Therefore, I defined a security constraint model that is used to capture these technical details. Information at the modelling layer are gathered and translated to this model.

To perform the transformation from security intentions to security constraints, further knowledge might be needed. Expertise knowledge might be required to determine an appropriate strategy to secure services and resource, 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. To describe these strategies and their preconditions in a standardised way, I foster the usage of security patterns as described in [8]. Security patterns have been introduced by Yoder and Barcalow in 1997 [13] and are based on the idea of design patterns as described by Christopher Alexander [1]. Based on this work, I defined a formalised system of security configuration patterns that describes patterns for each security intention and that is used to resolve appropriate set of security constraints.

The final step in my model-driven approach is the transformation of security configuration into enforceable security policy languages, depending on the capabilities of the target environment.

3 Modelling Security Intentions for SOA

To integrate security intentions in system design languages, an enhancement of these languages is required. In general, three approaches can be distinguished to implement such an enhancement:

1. Light-weight extensions – The easiest way to enhance a particular system design language is the usage of extension points provided by the languages itself. For instance, UML provides stereotypes (enhances the semantics of modelling elements) and tags (name/value pair) to extend UML modelling elements. Light-weight UML extensions are used by UMLsec to express security requirements.

   The advantage of using extension points is the simplified integration of security requirements in existing modelling tools. However, the visualisation of complicated security requirements might get confusing. Moreover, not all modelling languages define extension points to enhance modelling elements.

2. Heavy-weight extensions – Another approach to enhance modelling languages is based on the extension of its meta-model. For example, this approach is used by Rodríguez to define his security extensions for BPMN and UML [10]. The fact that the definition and integration of security requirements is done specifically for a particular system design modelling language based on its meta-model is one major disadvantage of this approach.

3. Defining a new language – To avoid the drawbacks mentioned above, a new modelling language can be defined. This modelling language integrates security elements and contain specific redefined elements of the system design modelling language.

   SecureUML uses this approach to model security requirements as an integral part of system models. Therefore, Basin and Lodderstedt described a generic approach to create a new security design languages by integrating security modeling languages into system design modelling languages as described in [2].

The advantage of the schema described by Basin and Lodderstedt is its flexibility. A security modelling language can be defined once with certain extension points and than be integrated into different design modelling languages. The resulting language is called a modelling dialect.

I have adopted this approach to model security intentions as shown in Figure 2.

The schema consists of the following parts:

1. A security modelling language is used to express security requirements for a specific purpose. While SecureUML provides a security modelling language to model authorization constraints, I have defined SecureSOA that enables a modelling of security intentions for Web Services.

2. The structure of a system is described using a system design modelling languages. While different types of modelling languages can be used, my approach is based on FMC Block diagrams that is used to visualise system architectures.

3. Both languages are integrated by merging their vocabulary using the extension points of the security modelling language. The resulting language is called a dialect.

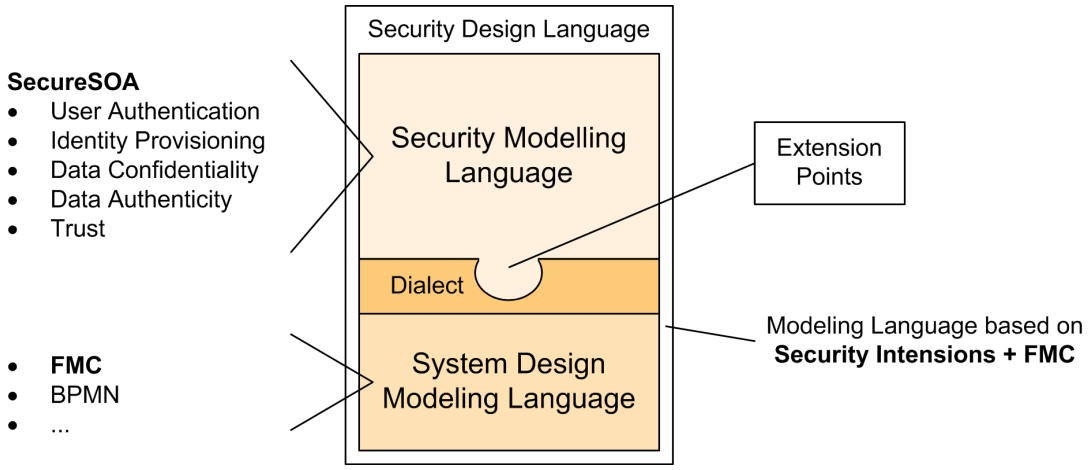


Figure 2: schema for constructing security design languages

In order to provide extension points, the security modelling language has to formalise entities that are subject of the security requirements and that can be identified in the system design models as well. For example, I formalise participants in an SOA such as services as objects that participate in an interaction by exchanging information. The security intentions defined in my security language state requirements that refer to a particular object or information. FMC visualises system architectures that are composed of agents communicating over a channel. Therefore, each agent is an object and can be integrated using subclassing.

4 Secure SOA - a security design language for SOA

SecureSOA enables a modelling of security intentions for Web Service-based systems and is defined by a MOF-based meta-model. The concrete syntax (notion) is defined using UML profiles.

The SecureSOA meta-model [7] consist of two parts. The meta-model for SOA introduces the basic entities in my model and their relationships to describe interactions in a Service-oriented Architecture. Based on this model, a model for security intentions is provided as well.

4.1 A metamodel for SOA

As introduced in [11], one of the basic entities in my model is an *object* that consists of a set of *attributes* and can participate in an *interaction*, see Figure 3. An interaction is always performed on a *medium* that is connected to the objects. For instance in the scope of Web Services, an object could be a Web Service client or a Web Service itself. In addition, each interaction also involve the exchange of *information*.

To enable a detailed description of Web Service messaging, I model transferred information as *data transfer objects* as introduced by Fowler in [3]. Figure 4 shows

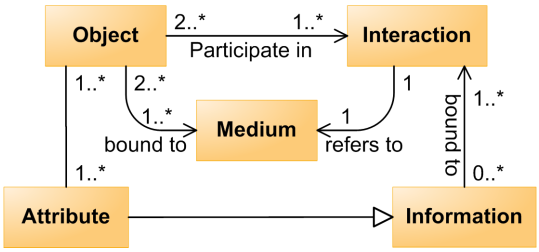


Figure 3: The Security Base Model

the adaptation of this concept to my model. A *data transfer object* represents serialised information and is an information itself. However, it can also contain information. This recursive structure facilitates the description of SOAP messages and its message parts. In addition, Figure 4 visualises the mapping to the SOAP message structure as defined in the SOAP messaging framework specification [4]. A SOAP envelope is a data transfer object that can contain different message parts that are data transfer objects itself.

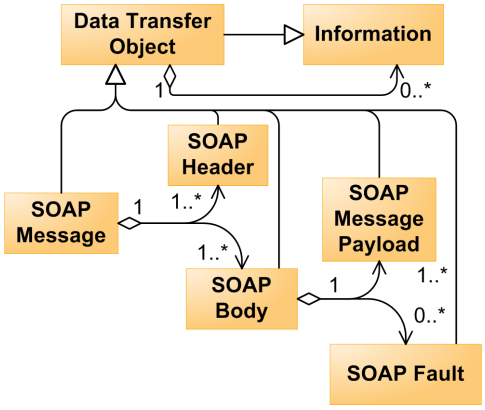


Figure 4: Messaging in SOA

Moreover, a data transfer object has a *target* and an *issuer*. This reflects that a data transfer object can be send over several objects acting as intermediaries. Therefore, issuer and target do not have to correspond necessarily to the objects that are involved in an interaction exchanging a data transfer object. In the scope of Web Service technology, WS-Addressing [5] would be used to represent the issuer and target in a SOAP-message by including a SOAP header that is also a data transfer object.

4.2 Modelling Security Intentions

Security intentions are defined specifically for one security aspect in terms of Web Service security and are related to one or more security goal. I have defined the following set of security intentions: User Authentication, Non-Repudiation, Identity Provisioning, Data Authenticity, Data Confidentiality, and Trust. Data Confidentiality, for instance, requires the security goal confidentiality for a particular piece of information, while identity

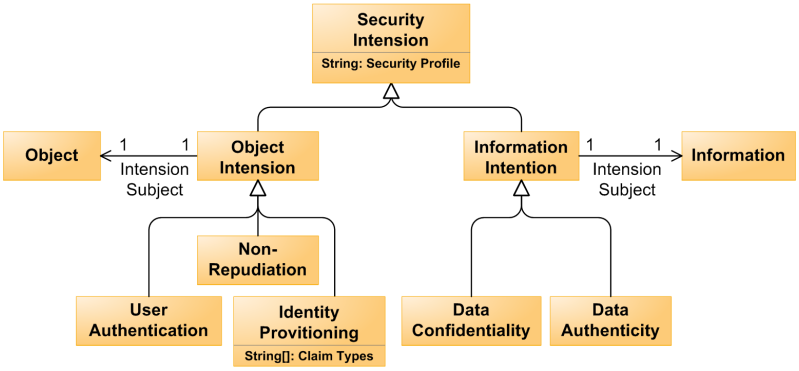


Figure 5: Modelling Security Intentions

provisioning states that the trustworthy identification and authentication of a user at a particular object is required. However, SecureSOA is not limited to this constraints and supports custom enhancements by adding additional security intentions.

As shown in Figure 5, each security intention is related to a security profile. The fundamental idea is to hide technical details at the modelling layer. The modeller should not be bothered with details such as security algorithms and mechanisms that are used to enforce this intention. This set of information is predefined in a security profile and is referenced by the security intention. Moreover, security intentions state requirements for a specific subject that is either an object or a data transfer object. Therefore, each security intention has an intention subject. Object intentions refer to an object, while information intentions refer to a data transfer object.

5 A SecureSOA dialect based on FMC

SecureSOA offers the possibility to express security intentions in various modelling languages. I have chosen FMC Compositional Structure Diagram (Block Diagram) to visualise software systems with security annotations, since FMC offers a suitable foundation to describe an SOA on a technical layer in terms of the involved participants and their communication channels.

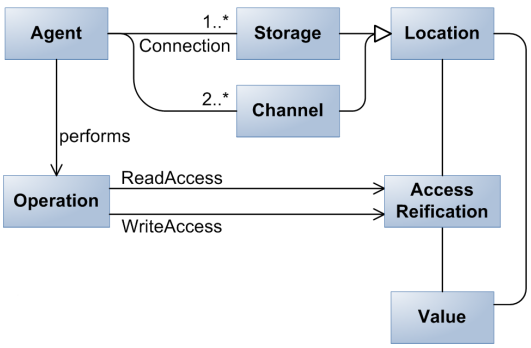


Figure 6: FMC Meta-Model

The FMC meta-model is depicted in figure 6. Agents interact by performing read or write operations on a channel.

To integrate SecureSOA in FMC, the entities in FMC have to be mapped to its corresponding entities in SecureSOA. As aforementioned, the easiest way to perform the integration is to subclass elements in SecureSOA. Object is subclassed by Agent, while information is subclassed by value. However, there is no corresponding element in FMC that can be mapped to service, client, sts and data transfer object, although their parents have been mapped to entities in FMC. To integrate these elements, it is necessary to add new elements to the meta-model of the dialect that subclass related elements in FMC and SecureSOA, as shown in Figure 7.

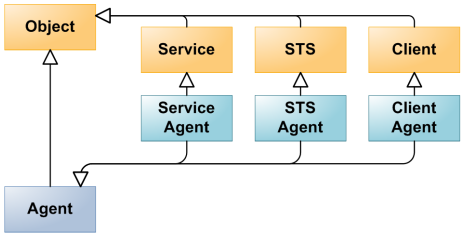


Figure 7: FMC Meta-Model

Finally, the element interaction has to be mapped to FMC. Subclassing will not work as integration technique, since interaction is not just a channel in FMC. It is composed of a channel in combination with an operation that is performed on this channel. Therefore, associations and an OCL-Constraint must be defined to perform the integration.

6 Example

The following section introduces a common claim-based service composition scenario as shown in Figure 8.

The order scenario contains an order process, in which a user is requesting goods using an online store web application. This application invokes a composed order service that uses two external services; a payment and a shipping service. The payment service represents an external service which handles the payment of the order process. In order to do so, the service needs payment information including a payment amount and credit card information like card type, card holder, card number, expiration date and a security code. The shipping service initiates the shipping of the goods using the recipients address. Based on SecureSOA, the notion of each FMC actor has been enhanced to indicate its type (client, service or STS) using stereotypes.

Users in this example have an account at his trusted bank and at the registration office, who act as identity providers managing the user's digital identity. The user can authenticate at the identity providers to request a security token that can be used to access a specific service.

In addition, this SecureSOA dialect is used to annotate security intentions to various actors in this use case. In particular, the payment service has established a trust relationship with the trusted bank, while the shipping service trusts information from

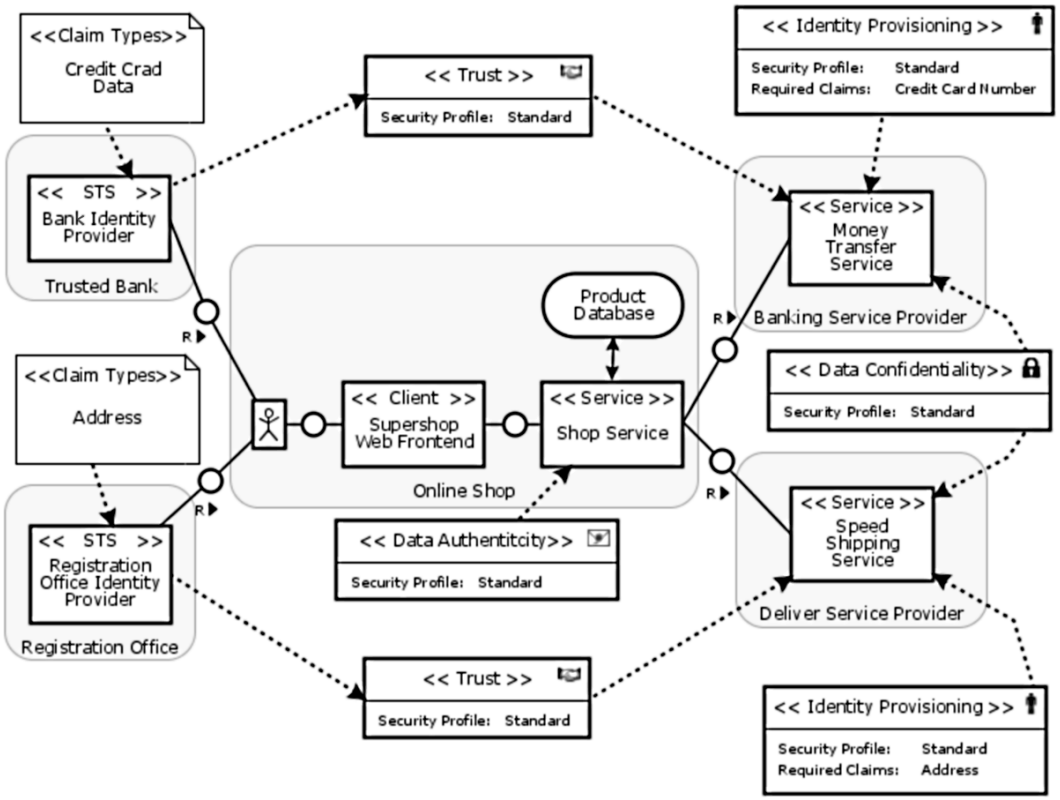


Figure 8: The Order Scenario with Security Intentions

the registration office. Therefore, the money transfer service and the speed shipping service are annotated with the security intention identity provisioning, while the identity providers are annotated with the identity information that they offer. For instance, the money requires credit card information while the STS trusted bank is capable to issue this information. To secure the exchanged information, the intentions data authenticity and data confidentiality are annotated as well in this example.

7 Related Work

The domain of model-driven security in the context of and SOA and business processes is an emerging research area. Previous work done by Rodriguez et al. [9], [10] discusses an approach to express security requirements in the context of business processes 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. A model-driven scenario based on their annotations is considered as future work.

Model-driven security and the automated generation of security enhanced software artefacts and security configurations have been a topic of interest in recent years. Similar to SecureUML, Jürjens presented the UMLSec extension for UML [6] in order to express security relevant information within UML-diagrams. This approach relies on

the usage of UML profiles, tags, and stereotypes to express requirements such as confidentiality, access control and non-repudiation. One focus of UMLSec lies on the verification of these security security requirements that facilitate a verification of security protocols modelled with UMLsec. However, the verification of security protocols and system models also implies that all security relevant information must be included in the UML-model and results in models that are difficult to understand due to its complexity. Although UMLsec can be adapted to verify communication related requirements in SOA, it does not provide a simple, high-level notion for security intention.

8 Conclusion and Future Work

System design languages provide a suitable abstract perspective to specific security goals on a more accessible level, as we have shown in [12]. In this report, I presented an approach to enhance arbitrary system design model with security annotations. My approach is based on an universal schema that has been introduced by Lodderstedt and Basin in [2]. To express security intentions related to Web Service Security, I have defined SecureSOA that has been integrated in FMC as a system design language.

Moreover, I presented an order service scenario that was the basis to illustrate the expression of security intentions in FMC concerning trust relationship, identity provisioning, and confidentiality. The specification of security requirements on an abstract level is the basis for my model-driven approach that addresses the difficulty to generate security configurations for Web Service systems. The foundation constitutes my generic security model that specifies security goals, policies, and constraints based on a set of basic entities as described in [7]. Security intentions and related requirements defined at the modelling layer can be mapped to this model. To resolve concrete security protocols and mechanisms, a security pattern-driven approach has been introduced in [8] that resolves appropriate security protocols with regard to specific preconditions. The gathered information can be mapped to policy specifications such as WS-SecurityPolicy.

Altogether, my proposed modelling enhancement constitutes a suitable foundation to describe and implement a model-driven transformation of abstract security intentions to enforceable security configurations in different application domains. As described in [8], security patterns are a promising approach to resolve additional information needed in the transformation process. In the next step, I will use my security model and the pattern system to provide an automated translation of security intentions modelled with SecureSOA to WS-SecurityPolicy.

References

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

[2] David Basin, Jürgen Doser, and Torsten Lodderstedt. Model driven security: from uml models to access control infrastructures. *ACM Transactions on Software Engineering and Methodology*, 15(1):39–91, January 2006.

[3] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[4] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, Anish Karmarkar, and Yves Lafon. Soap version 1.2 part 1: Messaging framework (second edition). Specification, April 2007.

[5] Martin Gudgin, Marc Hadley, and Tony Rogers. Web services addressing 1.0. Specification, May 2006.

[6] 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.

[7] Michael Menzel and Christoph Meinel. A security meta-model for service-oriented architectures. In *Proc. SCC*, 2009.

[8] Michael Menzel, Ivonne Thomas, and Christoph Meinel. Security requirements specification in service-oriented business process management. In *ARES*, 2009.

[9] 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.

[10] 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.

[11] 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.

[12] Christian Wolter, Michael Menzel, Andreas Schaad, Philip Miseldine, and Christoph Meinel. Model-driven business process security requirement specification. *Journal of Systems Architecture Special Issue on Secure Web Services*, 2008.

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

Automatic Extraction of Locking Protocols

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

1 Introduction

Monitoring and tracing applications raises many challenging problems. First, one has to find the right amount of tracing points in the system so that the information gathered is sufficient for a particular analysis. Secondly, once the trace points have been determined, one has to decide what information to store. If too few information is stored, it might not be possible to reconstruct any useful control flow of the system. On the other hand, if too much information is stored, it might confuse the analysis process. Finally, the tracing solution must not degrade the performance of the system under inspection too much. How much too much actually is, depends heavily on the intent the monitoring tool is used for. Developers may accept performance degradations at an order of magnitude in order to find the reason for an error, while system administrators running production systems will agree only to a small performance penalty.

One particular problem in the problem area described above, is how to ensure that any retrieved data is valid with respect to the system. That means, if a monitoring tool retrieves a data object, can you rely on that data or is there a possibility that the data is corrupt? This matter becomes even more important when multi-threaded systems are concerned, which is more likely nowadays with the advent of many- and multi-core architectures. In such systems, when data objects are shared between different actors, or *threads*, the system implements a consistency model on its data [10], *i.e.*, all actors agree on when they see updates on shared data objects. There are multiple ways to achieve that, for example distributed shared memory systems, or hardware and software transactional memory systems. Another approach to adhere to a consistency model is to design a locking protocol, which defines how participating actors determine who gets access to a shared object. The solution to the Producer-Consumer Problem is an example of such a locking protocol. In consequence, by applying the locking protocol, all accesses to a shared data object are serialized into a sequence. It is this sequence that prevents shared data objects from corruptions.

This issue has long not been addressed by both monitoring and tracing tools. To overcome that problem, we present the KStruct approach, which is a data structure inspection tool that allows accessing data objects at runtime while ensuring the consistency of the retrieved data. KStruct was carefully designed with locking protocols in mind. In order to define a locking protocol, we introduced the KStruct Access domain specific language [15], a subset of the C programming language. KStruct Access is based on the observation that many multi-threaded systems implement locking protocols at the object or data structure level. For that reason KStruct Access allows to annotate structure definitions and to declare *locking brackets i.e.*, what data structure is protected by which lock. For each data structure that should be monitored, there must be a definition in KStruct Access, which is created during the Annotation phase. Based on that, the KStruct compiler extracts that knowledge and compiles it into a driver that is capable of accessing the system under inspection. The driver incorporates the KStruct Runtime system in order to get access to the system's locking functions and data heap. The driver further provides several interfaces to reveal the monitored data. We denoted the phase of using the driver the Audit phase. As the KStruct runtime system lets you interactively browse through the

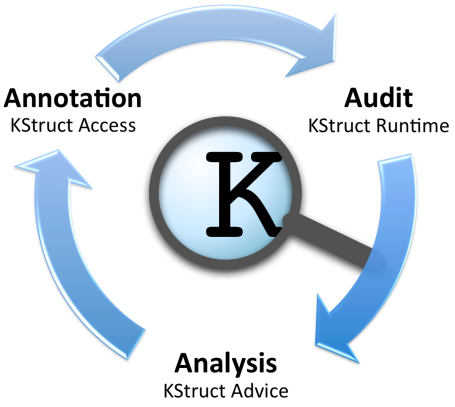


Figure 1: The KStruct AAA aproach.

object heap of the system, it may become necessary at some point to extend the annotated data structure which close the basic KStruct cycle.

So far, all the defined locking protocols have been defined manually for a small subset of data structures of the Windows Research Kernel. To provide access to greater amount of data structures, a more convenient way had to be found to derive the locking protocol for a particular data structure. Within this paper, we extend the previous KStruct cycle by an Analysis phase which is run by our tool KStruct Advice. KStruct Advice leverages static data-flow analysis to extract locking policies from a given code base. Based on that analysis, it is much easier to properly annotate data structures in order to be used by our compiler. We denote that approach as the *KStruct AAA* approach, shown in Figure 1. In the remainder of this paper, we will present the details on the data-flow analysis that is leveraged by KStruct Advice before we conclude the paper.

1.1 KStruct Advice

The KStruct Advice approach is based on the observation that existing multi-threaded applications are usually carefully designed with a particular consistency model [10] in mind, which resolves to a certain synchronization protocol. However, this protocol is hardly ever explicitly written down. If the code base is complex with several hundred people working on it, it may well happen that someone has to modify some code that was written by someone else in the first place. Without the synchronization protocol being expressed in some form, figure out the underlying protocol is both error prone and costly. It is not only for that that Greenhouse *et al.* demand making the original design intent explicit [7]. KStruct Access [16] has been created for that regard.

KStruct Advice has been designed to assist in analyzing an existing code base for its locking protocols. This knowledge may guide a programmer when writing code that needs to access shared data structures. It may help to understand what locks need be held in the first place before accessing a data structure by relating a particular access to similar accesses in the given code base. Also, KStruct Advice has the potential to provide further input for the KStruct [15] family of tools that can leverage the synchronization protocol to monitor the state of a system.

In order to extract the inherent synchronization protocol, we apply a context-sensitive, inter-procedural data-flow analysis, a well know technique used throughout compilers and static verification tools [1]. However, existing analysis frameworks are incapable of detecting the synchronization protocols. We thus will present a new set of data-flow equations to overcome this limitation. As a proof-of-concept implementation, we examine the Windows Research Kernel [12]. Therefore, in our analysis framework we only focus on procedural programming languages in general and the C programming language in particular.

2 Synchronization Protocols and the System Model

KStruct Advice is based upon the observation that non-trivial, parallel applications typically implement a fine grained synchronization protocol on shared data structures in order to scale on multi-core architectures. A synchronization protocol describes a serialization of concurrently executing threads with regard to either control flow or shared data access. A synchronization protocol therefore leverages either control locks or data locks, which we will further explain in the subsequent sections.

2.1 Control Locks

Control locks not necessarily protect data from race conditions, they solely provide an ordering of events in the control flow of the system. For example, consider the Producer-Consumer synchronization problem [18]. A single producer eventually produces an abstract item and stores it in a queue. Whenever there is an item in the queue, a consumer eventually may remove the item from the queue to somehow consume it. This problem is usually modeled with a semaphore that signals the consumer when there is something to consume, and signals the producer when there is any space in the queue. This is an example of a mere control lock. Neither queue operations nor any element is protected by it. It rather controls the activity of both participating parties. There are various implementations of control locks like monitors, barriers, or counting semaphores.

2.2 Data Locks

We further want to distinguish the class *data locks*. The purpose of this class is to protect shared data from race conditions. We define a data lock to be a synchronization primitive that ensures mutual exclusion among participating threads. With respect to data structures, data locks ensure that at any given point at most one thread may access the data structure. Similar to control locks, there is a variety of implementations of such locks. The Windows Research Kernel [12], for example, provides spin-locks, mutexes, push-locks, queued spin-locks, etc.

KStruct Advice is based upon the observation that non-trivial, parallel applications, *e.g.*, operating system kernels, need to perform fine grained locking on internal data structures to be scalable on multiple CPUs. Based on this observation we define our hypotheses that most of these locks are associated with the data they guard. In the Windows kernel, for example, data structures that can be accessed by more than one thread, usually contain a dedicated field that protects other fields from race conditions. This lock relationship may cascade on nested data structures. However, not all structures of a system are locked in such a fine-grained manner. The Windows kernel, for example, contains several global data structures that are still protected by global locks, too.

2.3 Lock States

In order to detect the synchronization protocol, our tool needs to know the fundamental synchronization primitives and its operations used in the system. For example, for a semaphore synchronization primitive [3], KStruct Advice must know what functions implement the *p()* and *v()* operation, respectively. To minimize efforts for a developer to use our tool, fundamental synchronization primitives are expressed as a list of functions that have acquire and release semantics, respectively. The particular type of the lock is then derived from the list of formal parameters. A function has *acquire semantics*, if it acquires a lock and the lock is held after returning from the function. Similarly, a function has *release semantics*, if it releases a lock and after returning from the function, the lock is free for others to use. These assumptions are safe as each parallel program must maintain a consistency model [10] in order to prevent itself from memory corruptions.

Basically, a lock is in one of two states: Acquired or Released. If a lock is in the state Acquired, it is assigned to a thread. If multiple threads try to acquire a lock, only that thread may continue to execute that the lock has been assigned to. Other threads must wait until the lock is assigned to them. If a lock is not acquired, it is in the Released state. Please note that semaphores can be assigned to multiple threads. In addition to that, recursive locks may be present, *i.e.*, a thread may acquire a lock that is already assigned to that thread. If we would not track multiple acquisitions of the same lock, a single release would mark the lock as released, although it is not in the state Released. We therefore extend the set of lock states by the following states. This set is henceforth referred to as V.

1. A number that represents the difference of pairwise encountered acquire and release operations. The domain of the counter is $\mathbb{N}_0^+$. If the number is 0, the lock is determinedly released, *i.e.*, it is in the state RELEASED. A counter value greater than zero implies that the lock has been acquired (at least once), and is in the state ACQUIRED.

2. The distinguished value UNKNOWN, which represents the unknown state. All locks of the system initially have this value, as we cannot reason about a lock before having seen it.

3. The distinguished value CONFLICT. A lock has that value if, during our analysis, we found conflicting states of the lock, *e.g.*, a release operation on an already released lock. This may happen, if the code base we use is itself erroneous. While an UNKNOWN state just states that we do not know anything about the lock, the CONFLICT state tells us that this lock is unsafe to reason about.

2.4 Call Graph

Our analysis requires a call graph for the system under investigation. The call graph is a directed graph where each node represents a function and each edge represents a function call. Suppose $G = (F, E)$ is a call graph, F is the set of functions and $E = F \times F$ the set of function calls, *i.e.*, there is an edge $(a, b) \in E$, iff function a calls function b. Further suppose $(a, b) \in E$. We henceforth continue to denote a as caller and b as callee. We denote the location of the actual function call of b in a as the call site in a for function b. While in the call graph there is only one edge from a to b, a may indeed have several call sites for function b.

In this call graph, there are nodes that have no predecessors, *i.e.*, $\nexists a \in F : \forall b \in F, a \neq b, (a, b) \in E$. Such nodes b have no callers and are therefore called *root functions*. Root functions typically denote entry points of a system. We need to determine root functions in

order to start our analysis pass. Using root functions as starting point for the analysis minimizes potential repetitions of analyses. Also, we only analyze functions where there is an actual control flow through the system. Dead code, *i.e.*, code that is not directly called within the given code is not considered for the analysis. Although this prevents us from analyzing indirect function calls, we consider our approach still appropriate for detecting locking protocols.

2.5 Context Sensitive Control Flow Graph

For each function in the system to analyze, we construct a context sensitive control flow graph (cCFG). A *context sensitive control flow graph* is a graph, where each node represents a basic block of the code. Furthermore we define two types of edges: *control flow* edges and *call edges*. A control flow edge represents control flow through the function itself, *i.e.*, from one basic block to another. Each basic block consists of a stream of instructions. By definition, a jump, or branch, instruction is the last instruction of a basic block. Additionally, a cCFG contains two special nodes that represent the function entry and exit point.

The reason, why we need a cCFG becomes clearer, when we consider the C code given in Listing 1. There are two possibilities to treat function calls. First, the simplest approach is to treat the content of the called function as sub-flow graph and create an edge from each call site in the system to the respective sub-flow graph. For the example given in Figure 1, this would mean (1) to add a new basic block containing the code of function `InitData`. (2) We need to split those basic blocks containing a call site to the function and, instead, add a control-flow edge from the instruction preceding the call instruction to the flow graph of the called function. And (3), although not necessary in our case, we need to map formal parameters at the call site to the formal parameters within the callee. This includes the potential return value of the function.

Listing 1: This listing demonstrates the necessity for context-sensitive.

```
struct bar {
  Lock lock;
  int data;
};

void InitData(int *data)
{
  *data = 0;
}

void Sample(BOOLEAN DoBranch)
{
  struct bar s1;
  int data2;

  if (DoBranch) {
    AcquireLock(s1.lock);
    InitData(&s1.data);
    ReleaseLock(s1.lock);
  }
  InitData(&data2);
}
```

Although this approach is straightforward, it treats all callers of a function indifferently, with-

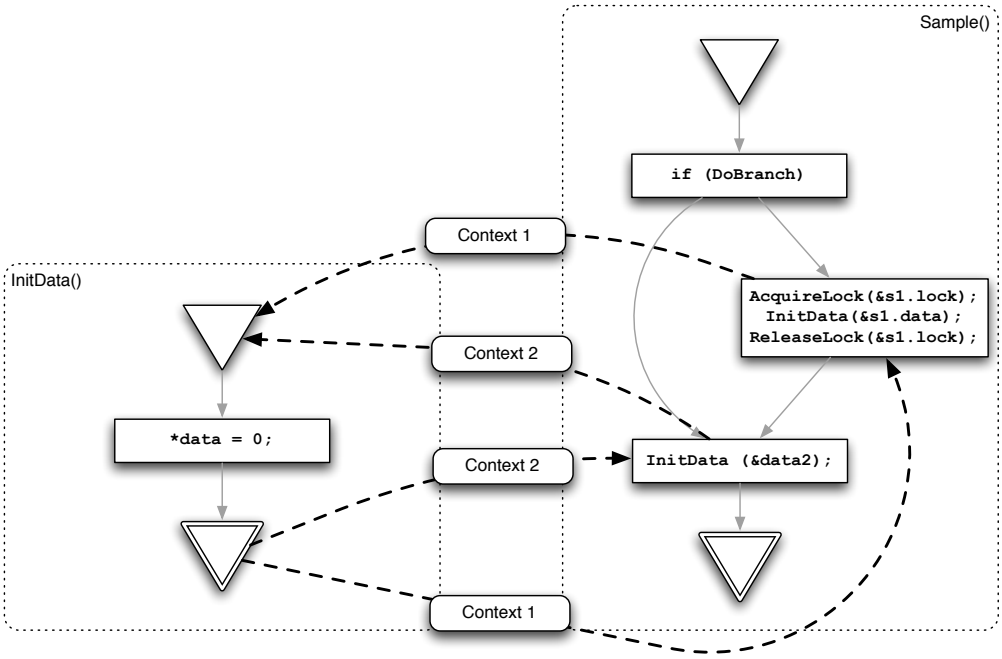


Figure 2: Example of a context-sensitive control flow graph.

out regarding the current context at the call site. For our goal, to detect what locks are held at a given point in program execution, it is however crucial to distinguish between call sites. We thus construct a super control-flow graph, as shown in Figure 2. To distinguish between different callers of a function, a simple approach would be to have a sub-flow graph for each call site. Instead, we create a flow graph only once for each function and clone and update only the context parameters. This cloning based approach [1] is necessary to minimize memory consumption during the analysis phase. As each callee may itself call functions, recursively creating a flow graph for each call site would result in an exponential number of flow graphs. In order to support context-sensitivity, we introduce *call edges* to the super control-flow graph, *i.e.*, we annotate each control-flow edge from a call site to the respective called function with the current context that must be considered when analyzing the function.

We define the context at a call site as the list of formal parameters that are passed to the callee. Furthermore, we need to extend the context by the list of locks that we are aware of at the call site. During analysis of the callee, the list of locks may be updated. We thus need to propagate that list back to the caller.

Our approach means that, potentially, the amount of times we analyze a function increases exponentially with the number of functions in the call graph. To minimize execution time, we leverage the partial transfer function approach by Wilson and Lam [21].

3 KStruct Advice Data-Flow Analysis Framework

In a first phase KStruct Advice leverages data-flow analysis to compute the state of any lock at any point within the statically reachable code. In this section, we briefly summarize key aspects of a data-flow analysis framework. Data-flow analysis is a well-known technique most often

used by compilers [1, 8], which is based on the concept of abstract interpretation, introduced by Cousot and Cousot [2]. A data-flow analysis framework describes a body of techniques to derive information about the flow of data along program execution paths.

Every program point, *i.e.*, any instruction or statement, has two data-flow values: an entry value that denotes the value before the instruction and an exit value that denotes the value right after the execution of the statement. We denote the entry value of statement s as $\text{IN}[s]$ and its exit value as $\text{OUT}[s]$. For any program point s and a forward data-flow analysis, the following equations apply:

$$\text{OUT}[s] = f_s(\text{IN}[s]) \tag{1}$$

$$\text{IN}[s] = \bigwedge_{p \in \text{PREDECESSOR}(s)} \text{OUT}[p] \tag{2}$$

Where f_s denotes a transfer function specific to s that maps the entry value to an exit value, and $\wedge$ denotes the meet operator that aggregates the exit values of any predecessor p of s in the cCFG into an entry value of s. Within KStruct Advice we group instructions into *basic blocks*, where a basic block is a sequence of instructions not containing any jump or call instruction except for the last one. Furthermore, no instruction inside the basic block except for the first one may be the target of any jump instruction. Within a basic block, each statement s has exactly one predecessor p in control flow order, *i.e.*, the meet operator for any two consecutive statements s_1, s_2 is the identity function $\text{IN}[s_2] = \text{OUT}[s_1]$.

3.1 Lock Semi-Lattice

The foundation for our analysis is the lock semi-lattice presented here. It allows KStruct Advice to determine the state of any lock at any point in the system. We define our lattice as $L = (V, \wedge_L)$, where V is the set of lock states as defined in Section 2.3.

We define the meet operator $\wedge_L$, $V \times V \rightarrow V$, as follows. For all $x \in V$:

$$\text{UNKNOWN} \wedge_L x = x$$

and

$$\text{CONFLICT} \wedge_L x = \text{CONFLICT}.$$

For any two constants c_1 and c_2, $c_1, c_2 \in V$ the $\wedge_L$ operator is defined as follows:

$$c_1 \wedge_L c_2 = \begin{cases} \text{CONFLICT} & \text{if } c_1 \neq c_2 \\ c_1 & \text{otherwise} \end{cases}$$

It can easily be shown that our meet operator $\wedge_L$ is idempotent, commutative, and associative.

Based on our semi-lattice, we define a partial order on the set V of lock states. For all x and y in V:

$$x \leq_L y \Leftrightarrow x \wedge_L y = x.$$

Because $\wedge_L$ is idempotent, commutative, and associative, it can be easily shown that our lock order $\leq_L$ is reflexive, anti-symmetric, and transitive. We thus have defined a sound semi-lattice and poset, respectively, for the set of lock states. □

3.2 Product Lattice

In the previous section we have defined a lattice L that represents the state, or the value, of a single lock at a given point in the system's control flow. However, when analyzing a system, we do not want to run the computation again for each single lock. Suppose Λ is the set of all locks in the system. We compose a product lattice $\mathbb{L} = (V^{|\Lambda|}, \wedge_{\mathbb{L}})$, where each value represents a map from each lock variable in the system to one of the values described in the previous section. A map $m \in V^{|\Lambda|}$ is a tuple $(v_1, v_2, \ldots, v_{|\Lambda|}), v_1, \ldots, v_{|\Lambda|} \in V$. The i-th entry denotes the value of lock i in the system. To index a certain value in the map, we introduce $m(\lambda)$ to denote the value of lock λ.

We now define the meet operator $\wedge_{\mathbb{L}}$ for the product lattice. Suppose m, m', and m'' be three distinct maps. Then

$$m \wedge_{\mathbb{L}} m' = m'' \Leftrightarrow \forall \lambda \in \Lambda : m''(\lambda) = m(\lambda) \wedge_L m'(\lambda),$$

i.e., the meet operator $\wedge_{\mathbb{L}}$ operates component-wise on the map. Similarly, we define the partial order $\leq_{\mathbb{L}}$ on the product lattice:

$$m \leq_{\mathbb{L}} m' \Leftrightarrow \forall \lambda \in \Lambda : m(\lambda) \leq_L m'(\lambda).$$

That is, a map m is less than or equal to a map m' if and only if all elements of m are less or equal than all components of m'.

3.3 Transfer Functions

KStruct Advice analyzes a series of statements in the control flow of a system. Each statement has an entry and an exit state. A *transfer function* f_s transfers the entry state of statement s into the exit state of statement s. We describe the transfer functions f_s for our analysis by correlating m and m', where $m' = f_s(m)$.

1. If the statement s is an acquire operation on lock λ, then $\forall \gamma \in \Lambda, \gamma \neq \lambda : m'(\lambda) = m(\gamma)$, *i.e.*, all the other locks not affected by the statement remain their state.

$$m'(\lambda) = \begin{cases} m(\lambda) + 1 & \text{if } m(\lambda) \text{ is already acquired} \\ 1 & \text{if } m(\lambda) \text{ is UNKNOWN} \\ \text{CONFLICT} & \text{otherwise} \end{cases}$$

2. If the statement s is a release operation on lock λ, then $\forall \gamma \in \Lambda, \gamma \neq \lambda : m'(\gamma) = m(\gamma)$.

$$m'(\lambda) = \begin{cases} m(\lambda) - 1 & \text{if } \lambda \text{ has been acquired} \\ & \text{at least once.} \\ \text{CONFLICT} & \text{otherwise.} \end{cases}$$

3. If the statement is a call instruction, the transfer function is the composition of all transfer functions for the called function, the callee. Let Δ be a subset of Λ that contains all the locks λ affected by the callee. For all locks λ in Δ, the transfer function is

$$m'(\lambda) = \text{OUT}[F_c](\lambda)$$

where $\text{OUT}[F_c]$ is the output map of callee F in context c. $\text{OUT}[F_c]$ is computed by analyzing the callee itself. We leverage the partial transfer function approach to optimize analysis time [21]. Further details on how we deal with context-sensitivity are presented in the following section. For all other locks not affected by F the lock value remains the same, *i.e.*, $\forall \gamma \in \Lambda - \Delta, m'(\gamma) = m(\gamma)$.

4. For all other statements the transfer function is just the identity function $I(m) = m$, *i.e.*, $\forall \lambda \in \Lambda : m'(\lambda) = m(\lambda)$.

3.4 Execution

To solve the data-flow problem, KStruct Advice solves the data-flow equations for all basic blocks of the root functions and its dependents. Once the analysis reaches a fixed point, *i.e.*, none of the basic blocks changes its value, KStruct Advice gathers information about the state of any lock found during the analysis and correlates that information with field accesses to data structures. Based on this information, KStruct Advice provides guidance when annotating data structures.

4 Related Work

For many years, researches are focusing on automatic verification of parallel programs with respect to deadlocks or data races. Although knowing that a program might be free from races or deadlocks is less concrete than to know which particular lock must be held when accessing a field, KStruct Advice and those verification tools share some properties. First of all, we need to classify existing solutions into dynamic and static ones, which will be covered separately. Both classes share with KStruct Advice the basic necessity to detect (1) when an access occurs, (2) what lock is currently held, and (3) what locks had been held on previous accesses. Race detection tools forget that information, as soon as they found a violation. KStruct Advice, however, keeps this information to build an information data base for developers.

4.1 Dynamic Detection Tools

Dynamic concurrency verification tools try to detect access anomalies during runtime of the system under investigation. The typical approach of dynamic tools is the *lockset* approach [14]. That is, each shared field of a data structure is associated with a set of locks that guard the respective data structure. On initialization, the set contains all locks of the system. Whenever a thread accesses a shared object, its current lockset is intersected with the current lockset of the object and stored as the new lockset of the object. Thus, if the lockset becomes empty, an access anomaly has been found. The original approach by Savage *et al.* has been improved and modified over time [5, 23].

An alternative approach is based on Lamport's *Happened-Before* relation [9]. The basic idea here is to determine, if accesses to a shared object or a member of a shared object happens in a partial order. As approaches based on the Happened-Before relation usually require more time and space, a hybrid approach has been developed [4, 11, 20].

All these dynamic approaches usually impose a significant overhead, both with respect to execution time and memory consumption, to the system under investigation. And, although dynamic approaches only track feasible paths, they cannot guarantee that the system is free of races, as long as their code coverage is less than 100%, which is very likely in complex software systems.

4.2 Static Detection Tools

Static race detection tools try to detect data races during the compilation phase, just as KStruct Advice does. They have the benefit that they cover the whole source code and run off-line, *i.e.*,

they do not impose any runtime overhead.

Among the first concurrency analysis tools was that of Young and Taylor [22]. In their approach, besides static analysis, they leveraged symbolic execution to detect any violations to the consistency model. However, the state space explosion makes this approach unfeasible for complex software projects. The same “flaw” holds for Warlock [17], one of the first tools using solely static analysis. However, it requires thorough annotations throughout the code to retrieve any useful result, which renders it impractical on large software system like operating systems.

Recent approaches, like *RacerX* [6], or *Locksmith* [13], require less annotations. However, they do not consider recursive locks in their model. Also, RacerX, in some cases, is too conservative with respect to analyzing recursive function calls. That might be suitable for race detection, but not for identifying the synchronization model. Von Praun and Gross [19], finally, present also a static analysis framework consisting of optimized super control-flow graphs as in KStruct Advice. However, they focus only on object-oriented languages and the Java language in particular.

5 Conclusions

In this paper we presented KStruct Advice, a approach for automatically extracting locking policies of shared data structures, which is based on data-flow analysis. For a given system in the C programming language KStruct Advice derives (1) what data structures and fields, respectively, depend on what locks and (2) what particular lock protects which data structure or field, respectively.

We have implemented our algorithm in a prototype that has been used on the Windows Research Kernel to improve understanding of its inherent locking protocols. However, we continue evaluating the quality of our results as well performing case studies on the impact on developers. We hope that using KStruct Advice will minimize their efforts to understand and adopt the synchronization protocol of the system.

We have further shown how KStruct Advice integrates with the KStruct family of tools, which focus on consistently monitoring system state at runtime. While most other monitoring frameworks only focus on the control flow level, *i.e.*, what action happened before another, KStruct also focuses on the consistency aspect of any retrieved data. If there is a chance that the retrieved data itself can be corrupt due to race conditions imposed by the monitoring tool, reasoning about any errors based on that data is unsound and merely educated guessing. KStruct ensures that, given the locking protocol is correct, any retrieved data is consistent with the consistency model of the system. Therefore it relies on the underlying locking protocols that, prior to KStruct Advice, had to be specified manually. KStruct Advice is thus the final, missing part of the KStruct tool chain and my thesis.

References

[1] Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. *Compilers Principles, Techniques, and Tools*. Pearson, second edition, 2007.

[2] Patrick Cousot and Radhia Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL '77: Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 238–252, New York, NY, USA, 1977. ACM.

[3] Edsger W. Dijkstra. Cooperating sequential processes. Technical Report EWD-123, Technical University Eindhoven, Eindhoven, the Netherlands, 1965.

[4] Anne Dinning and Edith Schonberg. Detecting access anomalies in programs with critical sections. *SIGPLAN Not.*, 26(12):85–96, 1991.

[5] Tayfun Elmas, Shaz Qadeer, and Serdar Tasiran. Goldilocks: a race and transaction-aware java runtime. *SIGPLAN Not.*, 42(6):245–255, 2007.

[6] Dawson Engler and Ken Ashcraft. Racerx: effective, static detection of race conditions and deadlocks. *SIGOPS Oper. Syst. Rev.*, 37(5):237–252, 2003.

[7] Aaron Greenhouse, T. J. Halloran, and William L. Scherlis. Using eclipse to demonstrate positive static assurance of java program concurrency design intent. In *eclipse '03: Proceedings of the 2003 OOPSLA workshop on eclipse technology eXchange*, pages 99–103, New York, NY, USA, 2003. ACM.

[8] Gary A. Kildall. A unified approach to global program optimization. In *POPL '73: Proceedings of the 1st annual ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 194–206, New York, NY, USA, 1973. ACM.

[9] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978.

[10] David Mosberger. Memory consistency models. *SIGOPS Oper. Syst. Rev.*, 27(1):18–26, 1993.

[11] Robert O'Callahan and Jong-Deok Choi. Hybrid dynamic data race detection. In *PPoPP '03: Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 167–178, New York, NY, USA, 2003. ACM.

[12] 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.

[13] Polyvios Pratikakis, Jeffrey S. Foster, and Michael Hicks. Locksmith: context-sensitive correlation analysis for race detection. *SIGPLAN Not.*, 41(6):320–331, 2006.

[14] Stefan Savage, Michael Burrows, Greg Nelson, Patrick Sobalvarro, and Thomas Anderson. Eraser: a dynamic data race detector for multithreaded programs. *ACM Trans. Comput. Syst.*, 15(4):391–411, 1997.

[15] Alexander Schmidt. Kstruct: A language for kernel runtime inspection. Technical Report 23, HPI Research School, Reinsberg, Germany, October 2007.

[16] Alexander Schmidt, Martin von Löwis, and Andreas Polze. Kstruct – preserving consistency through c annotations. In *Proceedings of the 5th Workshop on Programming Languages and Operating Systems (PLOS '2009)*, Big Sky, MT, USA, October 2009.

[17] N. Sterling. Warlock: Astatic data race analysis tool. In *USENIX Winter Technical Conference*, pages 1–12, 1993.

[18] Andrew S. Tanenbaum. *Modern Operating Systems*. Prentice Hall, 3rd edition, 2007.

[19] Christoph von Praun and Thomas R. Gross. Static conflict analysis for multi-threaded object-oriented programs. *SIGPLAN Not.*, 38(5):115–128, 2003.

[20] Liqiang Wang and Scott D. Stoller. Runtime analysis of atomicity for multithreaded programs. *IEEE Trans. Softw. Eng.*, 32(2):93–110, 2006.

[21] Robert P. Wilson and Monica S. Lam. Efficient context-sensitive pointer analysis for c programs. *SIGPLAN Not.*, 30(6):1–12, 1995.

[22] M. Young and R. M. Taylor. Combining static concurrency analysis with symbolic execution. *IEEE Trans. Softw. Eng.*, 14(10):1499–1511, 1988.

[23] Yuan Yu, Tom Rodeheffer, and Wei Chen. Racetrack: efficient detection of data race conditions via adaptive tracking. In *SOSP '05: Proceedings of the twentieth ACM symposium on Operating systems principles*, pages 221–234, New York, NY, USA, 2005. ACM.

Service-Based, Interactive Portrayal of 3D Geovirtual Environments

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

This report summarizes my work in the HPI Research School on Service Oriented Systems Engineering. It motivates and points out the underlying research questions in the field of service-based 3D portrayal of large and complex 3D geovirtual environments, presents my approach to solving these questions, and demonstrates and highlights results by example. A focus is on my work during the past months.

1 Introduction

Continuously, massive geodata is acquired by authorities, municipalities, private companies, and public communities. This includes, e.g., digital elevation and terrain models, aerial images, other photogrammetric data, land use information, infrastructure data, or building footprints. This data can be used in various application areas, e.g., urban planning, thread response, disaster management, fleet management, tourism, or location marketing. Geodata in general represents a data class that includes heterogeneous information, which can differ, e.g., in data semantics, models, formats, or complexity. However, the spatial reference of geodata forms a foundation for combining this heterogeneous data and generating new knowledge about complex spatial structures, relationships, and processes.

To tap the full potential of geodata and adding value to it, applications and systems are required that provide methods and techniques for geodata management, provisioning, access, integration, analysis, and visualization that take into account the characteristics of this data, mainly its distributed storage, and massive data size. Interoperability represents one primary issue of such systems and techniques:

> *"Geographic interoperability" is the ability of information systems to 1) freely exchange all kinds of spatial information about the Earth and about the objects and phenomena on, above, and below the Earths surface; and 2) cooperatively, over networks, run software capable of manipulating such information.* [1]

In the geo-domain, service-based systems have evolved as *the* approach for making distributed geodata accessible, processing this data, and presenting it to end users. Various consortiums and standardization bodies are engaged in the interoperable specification of geodata formats and services. Examples include the International Organization for Standardization (ISO), the Open Geospatial Consortium (OGC), the

Open Source Geospatial Foundation (OSGeo), and others. Additionally, large companies and vendors participate in the standardization processes, e.g., by providing de facto standards.

Geovisualization, i.e., the generation of visual representations dedicated for human perception, plays a major role within the process of geodata provisioning and in today's spatial data infrastructures. For this purpose, the Open Geospatial Consortium (OGC) proposes several portrayal services for 2D and 3D geodata. So far, there is only one widely used "workhorse", the Web Map Service (WMS), for generating 2D maps. Standards for visualizing 3D geovirtual environments (3DGeoVEs) such as virtual 3D city models and landscape models have not been elaborated to a similar degree.

From this, the research question raised **Q1)** of how to provide complex 3DGeoVEs in a service-based manner. As a first finding, we identified image-based 3D portrayal services as an effective means for integrating and presenting complex, heterogeneous geodata from distributed sources. A second research question was **Q2)** how to make this image-based presentation as efficient and interactive as possible, which includes, e.g., navigation and analysis capabilities. A third question was **Q3)** how to orchestrate image-based portrayal services and how to provide their integration into more complex portrayal scenarios, aiming at higher-level geovisualization services.

My approach for solving these questions is presented in the remaining sections of the report. Section 2 introduces the approach of service-based, image-based 3D portrayal. Section 3 presents my work for introducing interaction capabilities into an image-based 3D portrayal service; Section 4 addresses the same issue by introducing a smart navigation technique. Section 5 demonstrates the potentials of orchestrating image-based 3D portrayal services, and Section 6 summarizes my research work.

2 Approach: Image-Based 3D Portrayal Services

For portraying 3D geodata, two approaches have been presented by the OGC, the Web 3D Service (W3DS) and the Web Perspective View Service (WPVS).[1] – While the W3DS provides scene graphs that have to be rendered by the service consumer, the WPVS supports the generation of images that show 2D perspective views of 3DGeoVEs encoded in standard image formats. The key advantages of this image-based approach include low hardware and software requirements on the service consumer side due to service-side model management, 3D rendering, and moderate bandwidth requirements; only images need to be transferred regardless of the model and rendering complexity. Furthermore, it can be used with simple, lightweight clients without specific 3D rendering hardware because only standard viewing functionality is required such as provided by web browsers.

As part of the OGC web services initiative, phase 4, we investigated the applicability of service-based access to distributed two-dimensional and three-dimensional geodata from the GIS, CAD, and BIM domain; terrain models, aerial-images, and building models have been accessed via WMS and Web Feature Services (WFS) and have been

[1] The W3DS has discussion status. The WPVS is an OGC-internal draft specification and represents the successor of the OGC Web Terrain Server discussion paper.

integrated by an interactive 3D viewer client, which also provides functionalities for building inspection and analysis. It turned out, that interoperability can be achieved for 3DGeoVEs based on a service-based architecture and based on 3D standards such as the Industry Foundation Classes (IFC) and CityGML [3]. As key contribution, the 3D viewer client integrates 2D and 3D GIS data as well as 3D CAD and BIM data at the visualization level in a lossless and seamless way by using 3DGeoVEs as a key concept and providing information retrieval and analysis functionalities.

Based on these findings, a high-level image-based 3D portrayal web service for the visualization and analysis of 3D building information models (BIM) had been developed [4]. This specialized WPVS integrates GIS, CAD, and BIM data at the visualization level; building information are mapped to components of building virtual 3D city models. High interoperability is ensured by providing the integration result as image to a service consumer; rendering styles can be configured by service operation parameters and different views for analysis purposes can be obtained.

3 An Interactive, Image-Based 3D Portrayal Service

The weaknesses of the current WPVS approach include the limited degree of interactivity of client applications. For example, it is not possible to navigate to a position identified in the image or to retrieve information about visible geographic objects.

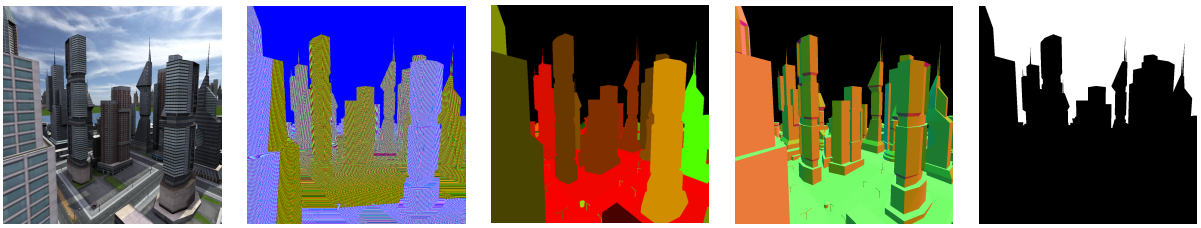
To overcome these limitations, we propose the *WPVS++*, an extension of the OGC WPVS that is capable of augmenting each generated color image by multiple thematic information layers encoded as images. These additional image layers provide for instance depth information and object identity information for each pixel of the color image. Additionally, we propose operations for retrieving thematic information about presented objects at a specific image position, simple measurement functionality, and enhanced navigation support. This allows WPVS++ clients to efficiently access information about visually encoded objects in images, to use that information for advanced and assistant 3D navigation, and thereby to increase the degree of interactivity.

In the following, the main functionalities of the proposed interactive WPVS++ are described.

3.1 Image Layers

Besides color images, the WPVS++ generates additional image layers storing information such as depth or object id values [8] for each pixel of the image. Figure 1 shows examples of such image layers. This additional data is provided as separate images, in which each pixel value does not necessarily encode a color and is not necessarily dedicated for human cognition:

Color layer: A color layer contains a color value (e.g., RGB) for each pixel. According to the current WPVS specification, the service consumer can request a transparent background (RGBA), which requires image formats that support transparency, e.g., PNG or GIF.



(a) (b) (c) (d) (e)

Figure 1: Examples of image layers that are provided by WPVS++: (a) color layer, (b) depth layer, (c) object id layer, (d) normal layer, and (e) mask layer.

Depth layer: A depth layer describes the distance to the camera for each pixel. Depth images can serve for multiple effects, e.g., for computing 3D points at each pixel of the image (image-based modeling) or for various rendering techniques such as depth-of-field effects. Furthermore, depth information represents a major means to compose multiple images generated from the same camera position.

Typically, graphics hardware provides logarithmic and linearized depth values $z \in [0, 1]$. Different from this, we suggest to provide depth values as distances to the virtual camera in meters. This representation abstracts from computer graphical details and the distance values can be used without additional computation in many applications.

Depth values are stored in IEEE 32 bit Float representation as images by segmenting their byte representations and assigning them to the color components of a pixel. Due to the direct storage as color components, resulting depth images possess very little pixel-to-pixel coherence, and should not be stored by lossy image formats. Thus, we suggest at least 32 bit (RGBA) images for this type of depth value storage. Consequently, a service consumer has to recompose the depth values from the color components of the requested depth layer.

Object id layer: An object id layer contains a unique id for each pixel that refers to a scene object. Using this information, we can select all pixels that show a specific feature, e.g., for highlighting, contouring or spatially analyzing the feature. For facilitating consistent interaction across multiple images, object ids should be unique over multiple requests. For that, they could be computed from an object id that is unique for the whole dataset.

Normal layer: A normal layer describes the direction of the surface normal at each pixel. This could be used for the subsequent integration of additional light sources and the adjustment of the color values around that pixel, e.g., for highlighting scene objects. Each vector component (x, y, z) of the normalized normal vector in camera space is encoded as color component (R, G, B) of a 24 bit color image.

Mask layer: A mask layer contains a value of 1 for each pixel that covers a scene object and 0 otherwise. Thus, a mask layer can be stored as 1 bit black and white image. It could support the easy altering of the scene background or provide information about

unused image space, which could be used, e.g., for blending advertisement, labels, etc.

Category layer: A category layer classifies the image contents on a per-pixel basis; the classification could be based on, e.g., feature types, object usages, or object names. The Category layer extends the so far described concept; it consists of an object id layer and a look-up table that lists for each object id the corresponding category.

Each image layer is encoded by standard image formats. This means to support the same principles for data encoding, data exchange, and client-side data loading and processing for all image data. Additionally, using images allows for applying state-of-the-art compression algorithms. For color data, which is dedicated for perception by humans, lossy compression algorithms can be used; for this data, JPEG mostly represents a suitable data encoding format. For data that requires exact values for each pixel, lossless image compression must be applied; we suggest PNG to encode this image data. PNG is also used for color images that shall contain transparency. Technically we encode JPEG images by 24 bit per pixel and PNG images by 24 or 32 bit per pixel. Higher data accuracies could be reached by using image formats with higher bit rates, e.g., 32 (48) bit images or even 48 (64) bit images. Drawbacks include increased image size and, thus, transfer load, and processing time at both the service-side and the consumer-side.

3.2 Image Sets

Various applications could require to fetch multiple, spatially coherent image layers from the WPVS++. As the WPVS++ is a stateless service (i.e., each service request is self-contained and its processing is independent from any preceding request), it potentially has to fetch, map, and render disjunctive sets of geodata for each service request. In particular, this is relevant when considering large 3DGeoVEs or remote data sources to be visualized.

From a performance point of view, it could be better to support multiple image layers within a single WPVS++ response. Provided that the requested views are spatially coherent (i.e., they are located close to each or have overlapping view frustums), this could reduce the number of switches of the rendering context. As an additional benefit, the network communication overhead caused by multiple requests is reduced.

These image sets could provide multiple image layers for the same camera specification (e.g., a set consisting of a color layer, depth layer, and object id layer) as well as images for different camera specifications (e.g., perspective views along a path through the 3D world).

3.3 Convenient Camera Specification

The current OGC WPVS provides a camera specification that is based on a point-of-interest (POI) and is not convenient for many applications. For example, specifying a setting that shows a view out of a window to a specific point in the 3D world demands for laborious computations for deriving distance, pitch, yaw, and roll angles.

As various applications could profit from a different specification of the view frustum, the WPVS++ replaces the camera specification by another one that is based on only three vectors: the 3D coordinate of the camera position (POC), the 3D coordinate of the point-of-interest (POI), and an optional camera up vector (UP) for specifying camera rolls. Furthermore, parameters for near plane and far plane are required; they describe the culling volume used during the rendering process and are necessary, e.g., for the generation and usage of depth image layers. For the specification of distorted images, we suggest replacing the angle-of-view parameter by a field-of-view angle in degrees in horizontal direction (FOVY) and to complement it by the optional vertical angle (FOVX). If FOVX is not specified, it is derived from the image dimensions.

3.4 Non-Perspective Projections

The current OGC WPVS is intended for central perspective projections only. Complementary, an image-based 3D portrayal service could offer additional projection types, such as orthographic projections, which are used, e.g., for architectural applications.

For extending the WPVS++ for orthographic projections, we suggest an alternative camera specification Orthographic that is controlled by the parameters Left, Right, Top, and Bottom, which describe the borders of the cuboid view frustum.

Beyond perspective and orthographic projections, a 3D portrayal service could support more advanced projection types such as oblique images, panoramic views, or multi-perspective views. While some of these projections could be generated from 2D images as provided by the current WPVS implementation, others require geometric information and, thus, are an integral part of the rendering process and need to be implemented by the 3D portrayal service itself.

3.5 Requesting 3D Coordinates and 2D Positions

Retrieving 3D geocoordinates for a specific 2D pixel position in a generated image is useful for many applications (e.g., specifying the position of an intended annotation). For this, the WPVS++ provides a GetPosition operation. As the WPVS++ is stateless, in addition to the 2D pixel position the image specification of the original GetView request has to be part of the GetPosition request. This operation is useful for navigation in the visualized 3DGeoVE: A user could select two image pixels that specify the desired point of camera and point of interest; the client requests corresponding 3D coordinates, and uses these for specifying the camera in a subsequent GetView request.

The other way round, the WPVS++ can compute the pixel position of a 3D coordinate if this 3D position is within the view frustum. Again, this could support the

interaction capabilities; for keeping orientation values high, a client could e.g., request the pixel position, where the virtual camera was located in the view before.

3.6 Requesting Feature Information

Similar to a WMS, a GetFeatureInfo operation provides extra information about the geoobjects at a specific pixel position. According to the underlying data source and its capabilities, various response formats are supported, e.g., attribute names and values in XML format or GML-structured data sets. Additional formats can be supported and specified as mime type within the service request. In the case of structured responses (e.g., in GML format), the WPVS++ operation GetLayerInfo can be used for retrieving schema information for specific datasets.

3.7 Navigation Support

Navigation represents a fundamental interaction type for 3DGeoVEs. From virtual globes and online map systems users know to navigate by using mouse and keyboard in a real-time interactive manner. However, especially due to the image-based approach, the WPVS++ inherently provides only non-real-time step-by-step navigation. Assistant navigation techniques could compensate that drawback and allow a user to interact with the image, e.g., by sketching a desired point of interest, which serves as input of a GetCamera operation. The portrayal service automatically interprets this navigation input, taking into account scene objects, their types, and their navigation affordances, and computes and responds a camera specification, which can be used for requesting the corresponding view.

3.8 Analysis Support

Measuring within a 3D view represents a main functionality used to retrieve information about the spatial extent of features, their spatial relationships, and the overall spatial layout of a 3D scene. Thus, the WPVS++ supports the measurement of distances, paths, and areas. *Path measurement* computes the sum of the Euclidean distances between the 3D positions derived from each pair of consecutive 2D pixel positions. If only two 2D pixel positions are provided as input, the WPVS++ performs distance measurement. *Area measurement* computes the area which is outlined by the input points. In 3D, area measurement is not straight forward, as the derived 3D points are not likely to be coplanar.

3.9 Interactive, Lightweight JavaScript Client API

Navigation capabilities mainly depend on the functionality of the clients that consume the WPVS++. Simple clients can provide a step-by-step navigation based on requesting and displaying single views. For this, the additional GetPosition operation facilitates

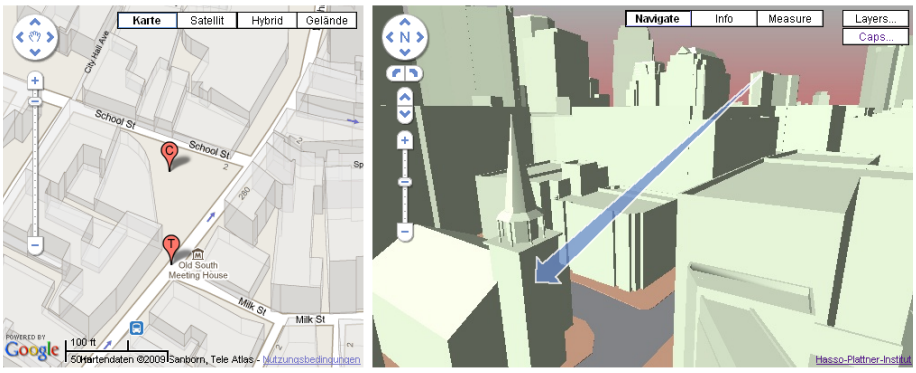


Figure 2: 3D view client (right) integrated with Google^{TM}maps (left). 2D and 3D visualization are synchronized, i.e. navigation on the 2D map immediately updates the 3D view and vice versa. The blue arrow indicates a navigation operation based on the WPVS++ GetPosition operation, i.e. setting the point of camera and point of interest. (Data: Boston Redevelopment Authority)

an easy and targeted camera manipulation. Based on depth information, more complex clients could apply, e.g., image-based rendering techniques for providing more convenient and close to real-time visualization and navigation functionality [6].

As example of lightweight clients, a JavaScript client has been implemented, which can be run in a web browser without additional plug-ins or libraries. It provides step-by-step navigation such as moving left/right, tilting up/down, or rotating around the POC or POI. Moreover, this client takes advantage of the GetPosition operation for retrieving 3D coordinates for changing the virtual camera's orientation and for implementing a "move there and orient here" navigation. The GetFeatureInfo operation is used for retrieving object information about features in the image. The GetMeasurement operation is facilitated for allowing users to measure distances in the portrayed 3DGeoVE.

The 3D view client provides an API for positioning the virtual camera and for corresponding callback functionalities. So it can be easily embedded into existing applications such as web-portals. As an example the 3D view client has been integrated with Google^{TM}maps (Figure 2), which allows for exploiting the functionalities of both 2D maps and 3D views.

4 Sketch-Based 3D Navigation Techniques

Navigation represents the fundamental interaction technique in 3DGeoVEs as it enables users to explore the 3D world and to interact with its objects. In web-based 3D portrayal services, efficient navigation techniques and strategies can help to increase the interactivity of these applications: Smart navigation techniques could hide the step-by-step image retrieval process and provide efficient navigation means, potentially compensating the absence of real-time navigation capabilities, e.g., by preventing "getting-lost" situations, confusing view configurations, or loss of visual context information.

For the application with 3D city models, we developed a sketch-based navigation technique [5]. This allows a user to sketch navigation commands by drawing gestures

and object-related sketches on a 3D view. These sketches are processed, classified, and interpreted by a navigation command system, which takes into account the sketch class, its shape, the sketching speed, potential preceding sketches, and the semantics and inherent navigation affordances of the scene objects affected by the drawings. According to the current context, the interpreted sketches are mapped to camera animations, which are automatically performed by the navigation system. This approach provides a higher-level navigation: A user is not forced to directly control the virtual camera, e.g., by keyboard, but can maneuver by task- and object-related sketches.

Within a service-based system architecture, the sketch-based 3D navigation technique could be provided by the 3D portrayal service itself, i.e., integrated within a WPVS++ and, e.g., provided by the proposed GetCamera operation. Alternatively, the sketch processing and interpretation could be performed by an additional service, e.g., an OGC Web Processing Service (WPS). In this case, the sketch interpretation service would request image layers that provide thematic information about the image such as object id images or category images.

5 Orchestration of Image-Based 3D Portrayal Services

A core concept of service-based systems represents the composition of distributed functionality in a standardized manner. This enables the construction and flexible adaption of complex and value-added systems and applications. With image-based 3D portrayal services, orchestration mainly refers to A) the combination of different 3D views, B) the embedding of additional information into a 3D view, C) the altering of a 3D view, or D) the computation of metrics (e.g., object visibility). For such functionalities, the extended WPVS++ provides various sources for computation. Considering depth-related image blending and combination, the depth layer represents a major resource for image-based processing chains [2].

We demonstrated the applicability of such a service composition for the example of 3D annotations, which represent essential elements to communicate textual and symbolic information for cartographic maps and within 3DGeoVEs [7]. A number of criteria, e.g., a clear correlation with the feature, the legibility of the annotation, and the occlusion of other annotations or important image parts have to be considered. In our approach and implementation, a composition client receives a color image and a depth image from an extended WPVS and forwards them to a new Web View Annotation Service (WVAS) along with a set of annotation descriptions and configurations. The WVAS computes and overlays embedded textual annotations. WPVS++ interaction extensions turned out to be essential for supporting such higher-level functionality. In the case of 3D annotations, picking a 3D positions from the image is fundamental for positioning the 3D annotations within the 3D view.

We showed, how two complementary 3D visualization techniques can be seamlessly combined by implementing these techniques by two independently designed, implemented, and deployed web services; chained together, they form a higher-level web service chain. The approach offers high degree of interoperability because the individual web services do not need to exchange contents of 3DGeoVEs.

6 Summary

This report provides a description of my research work during the past months and gives an overall summary of my research work in the field of service-based, image-based 3D portrayal. The contribution of my research work is in the field of interactive, image-based visualization of complex, heterogeneous, and distributed geodata by service-based 3DGeoVEs. By example I showed the feasibility of integrating such data at the visualization level, I extended the concept of image-based 3D portrayal service in a way that allows high interactivity, and I demonstrated the potentials of chaining image-based 3D portrayal services. Ongoing work includes the standardization of the interactive image-based 3D portrayal service as an OGC implementation specification; the functionalities and the prototype implementation have been presented to the OGC, a specification draft is currently prepared.

References

[1] Geographic Information – Services. ISO 19119, International Standard, 2005.

[2] Nadine Alameh. Chaining Geographic Information Web Services. *IEEE Internet Computing*, 7(5):22–29, 2003.

[3] Jürgen Döllner and Benjamin Hagedorn. Integrating Urban GIS, CAD, and BIM Data By Service-Based Virtual 3D City-Models. In Massimo Rumor, Volker Coors, Elfriede M. Fendel, and Sisi Zlatanova, editors, *Urban and Regional Data Management: UDMS 2007 Annual*, pages 157–170, Stuttgart, Germany, October 2007. Taylor & Francis Ltd.

[4] 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.

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

[6] Benjamin Hagedorn, Dieter Hildebrandt, and Jürgen Döllner. Towards Advanced and Interactive Web Perspective View Services. In *4th International 3D GeoInfo Workshop*. Springer, 0 2009. to appear.

[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] Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3-D shapes. *SIGGRAPH Comput. Graph.*, 24(4):197–206, 1990.

An overview on the current approaches for building and executing mashups

Emilian Pascalau
emilian.pascalau@hpi.uni-potsdam.de

1 Introduction

Future Internet is the new playground and the new environment that allows for generative content [29] to be developed.

The technological advances, the new emerging paradigms such as software as a service, business processes, business rules etc. have enriched and created a proper environment where innovation can be fostered in the new context of the Future Internet.

SAP co-CEO Henning Kagermann talks in [15] about the internet of services as being *largely based on a service-oriented architecture (SOA), which is a flexible, standardized architecture that facilitates the combination of various applications into interoperable services. The Internet of Services also uses semantic tools technologies that understand the meaning of information and facilitate the accessibility of content (video, audio, print). Thus, data from various sources and different formats can easily be combined and processed toward a wealth of innovative Web-based services.*

In this context several major paradigms get mixed: Software as a Service (SaaS) [2], Web 2.0 [22], and Enterprise 2.0 [17].

Major players such as Salesforce and Google already experiment and provide services that meet the requirements of SaaS applications. They provide several APIs and interconnectivity technologies either between their services or between other services (such as Facebook for example). Despite open APIs, another perspective of services aggregation on the Web is offered by mashups.

The wisdom of the crowd as it is characterized by O'Reilly in [22] is more and more taken into account, because as stated in [27] *customers know best what range of functionality a software product should have and what the underlying processes they need to support look like.* As a consequence i.e. IDS Scheer provides ARIS MashZone, which enables departmental users to create mashups for visualizing and evaluating a wide range of data. Similarly SAP has SAP Research RoofTop Marketplace and IBM has the IBM Mashup Center.

The rest of the paper is organized as follows: next section presents several definitions that were given for mashups as well addressing several approaches used to build and execute mashups. In addition it addresses characteristics of mashups as well as pointing out research questions. The last subsection addresses the rule based approach for creating and executing mashups which has been introduced by us. Section 3 provides a comparison between SaaS approach and the Mashup approach. The paper closes with conclusions.

2 Current approaches for building and executing mashups

While initially mashups have been seen as a tool based approach, nowadays mashups have attracted a lot of interest also from academia, and a lot of effort is consumed in order to provide formalism for the new Future Internet with all its buzzwords: *mashups*, *cloud computing*, *cloud*, *internet of things*, *internet of services*, *Web 2.0*, *Web 3.0*. In addition in this new context, the "Future Internet", other paradigms such as business process management, business rules and so forth must be immersed, and their relation with such concepts must be addressed as well as dealing with the challenges that arise from *developing an understanding of further ways to interoperate, what the stakeholders might demand, and how to achieve it if necessary* [23].

Several definitions have been given to define the concept of *mashup*. Altinel et al. in [1] define mashups as *web application that combines content from two or more applications to create a new application. Situational applications are enterprise web applications built on-the-fly to solve a specific business problem. They are often developed without involvement of the IT department and operate outside of its control. They combine data from a variety of enterprise sources such as SAP or Office applications, back-end databases, and content management systems.*

In a similar manner Grammel and Storey [11] define mashups as *applications that reuse and combine data and services available on the web. They are developed in a rapid, ad-hoc manner to automate processes and remix information. This enables users to explore information in new ways and saves valuable time that may be lost in laborious routine tasks.*

Although all the definitions provided emphasize more or less the same characteristics the CEO of Google Eric Schmidt [1] defines Web 3.0 applications as *"applications that are pieced together" - with the characteristics that the apps are relatively small, the data is in the cloud, the apps can run on any device (PC or mobile), the apps are very fast and very customizable, and are distributed virally (social networks, email, etc).*

Even though there are already several definitions for the concept of mashups and even though they seem similar, there is no concrete and no general accepted understanding of what a mashup is. Moreover in a special context such as WWW we are dealing with a different type of business processes: web based business processes that have special characteristics because the context is different. New technologies such as Google Wave and the new W3C efforts of standardization for HTML 5 with event stream and so forth emphasize more that the Internet and web programming is rapidly changing towards new paradigms.

In consequence several research questions arise:

- formalization of the mashup concept
- methodologies of creating and executing mashups in the context, where browsers, the Internet and the demands of the customers are the key points.

[1] `http://www.youtube.com/watch?v=T0QJmmdw3b0&feature=player_embedded`

- modeling of web based business processes

Mashups are needed because they allow: (1) to foster innovation by unlocking and remixing information in ways not originally planned for; (2) to uncover new business insights by easily assembling information from multiple sources; (3) to increase agility by supporting dynamic assembly and configuration of applications; (4) to speed up the development; (5) to reduce development costs through lightweight integration, reuse and sharing.

In general mashups approaches are grouped into two major categories: server-side and client-side. Further more these could be refined into:

- data mashups: could be both on server-side as well on client side. The goal is to create new data that did not exist before by taking remote information (from Web services, feeds, or even just plain HTML) and combining it with data from another source.
- software mashups: both on server-side and client-side. On server side it is mainly about web services, while on the client-side the code is directly integrated into the browser
- presentation mashups: client-side and usually show existent information in another way (e.g. Ajax Based ”desktops”)

Next subsections will address both *server-side* approaches as well as *client-side* approaches.

Although *pure* mashups as also argued in [3] are client-side mashups, these are much more difficult from a technical perspective compared to the server side mashups [3]. However we have developed a client-side approach, rules-based that could be used as an alternative to server-side mashups. This will be also addressed later on in the paper.

2.1 Mashups as collection of Widgets

As presented in [9] *a `widget` is a small client-side application that is authored using Web standards, but whose content can also be embedded into Web documents*.

Although W3C has released a candidate recommendation [6] that deals with packing and configuration of widgets there is no standard for creation and execution of widgets. Companies such as Google, Yahoo, Amazon, Microsoft, IBM have developed their own widgets and widgets platforms.

Google Gadgets are stored as XML files (i.e. `http://www.google.com/ig/modules/calendar3.xml`). Google provides a repository of such gadgets that can be embedded in your iGoogle page or in any other web page. Such gadgets have to be published on the Google Gadget platform and only after that, they can be processed by the Google Gadget Engine.

Such gadgets are embedded with the help of `script` elements. The result of their execution is usually an `iframe` element that is added to the DOM [14] of the embedding page. The iGoogle page is a collection of such gadgets.

A Google Gadget comprises XML [5], HTML [13], JavaScript [7] and optionally CSS [4]. The XML content is used to describe the gadget structure, HTML and CSS are used to define the presentation and JavaScript is used to define the logic of the gadget. The formal grammar of Google gadgets is defined in [21].

Yahoo Widgets are developed by Yahoo and although these have several common characteristics with the Google Gadgets there are also some differences. These are also stored as XML files. Yahoo widgets have also a formal grammar. They must be also published on the Yahoo Widget Server (Konfabulator)[2]. In addition a widget engine is required to be installed on the client machine. Yahoo Widgets are embedded in web pages in similar manner to Google Gadgets by means of `script` elements.

However Yahoo widgets, as stated in [9] *are programmatic interfaces that implement `actions` triggered by specific `events`*. The Yahoo Widget specification, is a proprietary format, which provides XML elements to describe: presentation; actions; periodic events (timers); preferences. The metadata that could be defined in a Yahoo Widget is defined in a DTD[3] file.

Typically widgets are defined declaratively using an XML based language, but the generated executable code is platform dependent. Usually they are embedded using `iframe` elements.

2.2 Pipes based mashups

Pipes are mainly about web data aggregation [9]. Various sources of data (typically Atom or RSS feeds) are queried/aggregated to create a new data source (usually a new feed). Pipes were pioneered by Yahoo with their Yahoo Pipes[4]. Yahoo Pipes allows you to aggregate popular feeds and create data mashups using a visual editor. Pipes are created by combing together several building blocks, such as: `User Inputs`, `Sources`, `URL`, `Operators`, `Location` and so forth.

IBM Mashup Center has as its core a similar approach. But beside the core pipes-based technology which has been improved with XPath constructs provides also the means to create widgets. These widgets use as data sources, the feeds created with their pipes based tool.

Semantic Pipes have been introduced in [19], [18]. As stated in [9] *a Semantic Pipe is a stored sequence of SPARQL queries over one or more RDF sources. The result of each query is seen as a concrete and independent RDF resource*. Various blocks are defined to process data sources (but also HTML, XML and JSON) and blocks for processing data (mainly RDF data) i.e. operators (defined on top of SPARQL language) as well as user input blocks and conditional blocks. In addition DERI Pipes uses an XML language behind the visual notations.

As a consequence a `semantic mashup` is a data mashup using RDF(S) as data model and SPARQL services to implement the behavior [9]. However while in the basic case semantic pipes just aggregate semantic data, semantic mashups may have

[2] `http://widgets.yahoo.com/tools/`
[3] `http://widgets.yahoo.com/static/downloads/widget-meta-4.0.dtd`
[4] `http://pipes.yahoo.com/pipes/`

reasoning capabilities and semantic web reasoners such as KAON2[5], Pellet[6], Racer[7], Jena[8] and so forth might be used.

2.3 Intelligent mashups

The other approaches for creating and executing mashups that have been addressed here are characterized by the following general comments:

- using Gadgets users should be able to do JavaScript programming. The alternative is to use already existent gadgets i.e. add then on a web page.
- using existent gadgets limits the user interaction
- to implement gadgets dependencies users should really write another gadget
- catching user activity is at the programming level (hard for business people)
- using Yahoo Widgets a user needs to install proprietary software on his local machine;
- using Amazon Widgets users cannot change the widget behavior but just presentation

A different approach for doing mashups is to use a rule-based approach, [25]. Rules can be easily captured and modified, and they empower applications with grater flexibility. Moreover using rules to model and to execute mashups seems to be an appealing solution that could: (1) offer another solution for service aggregation (the main focus is on SOAP/REST based services) and (2) provide a simple way to understand, model and define behavior/interaction between services.

Use case such as: *usability* - IF the user reads Reuters news about swine flu, THEN deliver him similar video news from CNN and possibly update his swine flu Google Map; *advertising* - IF the user loads financial news, offers him a three months subscription to Financial Times; *accessibility* - Whenever the user clicks more than 3 times a menu item add this item to the fast access menu items; are better easier captured with rules rather than with plain JavaScript or some JS framework. In some conditions widgets dependencies can be also implemented with rules.

The approach uses the JSON Rules language introduced in [10]. The main goals are:

- move the reasoning process to the client-side;
- offer support for intelligent user interfaces;
- handle business workflows;

[5] `http://kaon2.semanticweb.org/`
[6] `http://pellet.owldl.com/`
[7] `http://www.racer-systems.com/`
[8] `http://jena.sourceforge.net/`

- allow rule-based reasoning with semantic data inside HTML pages (reasoning with RDFa);
- create intelligent mashups rule-based mashups;
- enable users to collaborate and share information on the WWW (rule sharing and interchange)

The requirements as emphasized in [24] are:

- rules run in the browser;
- rules are Event-Condition-Action (ECA) rules;
- rules defined on top of DOM structure;
- events taken into account are DOM events and user defined events;
- conditions address the content of the pages;
- actions are defined by users (any JavaScript function calls could be an action)

3 Mashups vs. SaaS

The following comparison has been initially introduced in [25] and deals with Mashups and SaaS.

[8] analyzes terms such as *software as a service*, *software on demand*, *adaptive enterprise* and *mashups* and concludes that they are overlapping to many extents.

[2] argued that the "*service-based model of software is one in which services are configured to meet a specific set of requirements at a point in time*". Components may be bound instantly, based on the needs and discarded afterwards.

There are some key points that characterize *software as a service* (see for example, [28]):

- network-based access and management of commercially available software
- activities managed from central locations rather than at each customer's site,
- enabling customers to access applications remotely via the Web
- application delivery typically closer to a *one-to-many model* (single instance, multi-tenant architecture) rather than to a *one-to-one model*, including architecture, pricing, partnering, and management characteristics
- centralized feature updating, which obviates the need for end-users to download patches and upgrades
- frequent integration into a larger network of communicating software - either as part of a mashup or as a plugin to a platform as a service.

Mashups are hybrid web applications, usually found out under the association of SOA plus REST principles. Mash-up content is usually accessed through APIs from third party providers, (sites, services), processed and then presented to the user in a different format and with new insights. In such way new value is provided. One simple way to explain mashups was introduced by ZDNet Executive Editor David Berlind in a video presentation, What is a mashup?[9], where, among other issues, he claims that mashups are the fastest growing ecosystem on the Web.

Berlind introduced the mashup model by comparing it with the well known software stack on the traditional computers. In traditional computer systems we have an operating system, and, on top of this, a number of application programming interfaces (APIs) to access different services (i.e. the network, the display, the file system) and UIs to get to different applications (i.e. the mouse, the keyboard) as in Figure 1. Developers use these APIs to access different necessary services to create their applications.

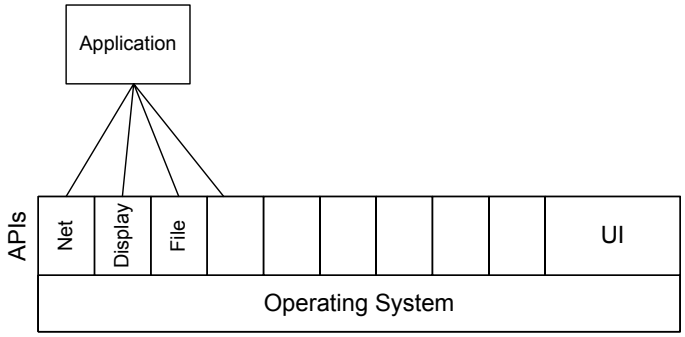


Figure 1: Computer Model (David Berlind, ZDNet), [25]

Somehow mashups follow the same model but with a different infrastructure. The Operating System is replaced by the Internet, and the old APIs are replaced with APIs offered by different service providers such as Yahoo, Google, Technorati, Amazon etc. as depicted in the Figure 2. In the same way developers use these APIs to get access the available services. These services reside in the Internet, or we may also say on top of the infrastructure offered by the Internet. In this way new applications are created from old ones.

The *software as a service* approach uses a very similar model (see Figure 3). Salesforce is a concrete example of this approach. However in this case the Operating System/Internet is represented by the Platform as a Service (i.e. Force.com).

Therefore, we see that while mashups are generally based on various service sources available on the Web, software as service is mainly based on a proprietary centric platform (which may handle different services). In SaaS, new applications are generated by using only the platform services (i.e. *platform as a service*).

Due to their *"open"* character, mashups besides the characteristics that define SaaS applications have some other characteristics that arise from the way they are implemented. Although some of these characteristics might overlap we present them here:

[9] `http://news.zdnet.com/2422-13569_22-152729.html`

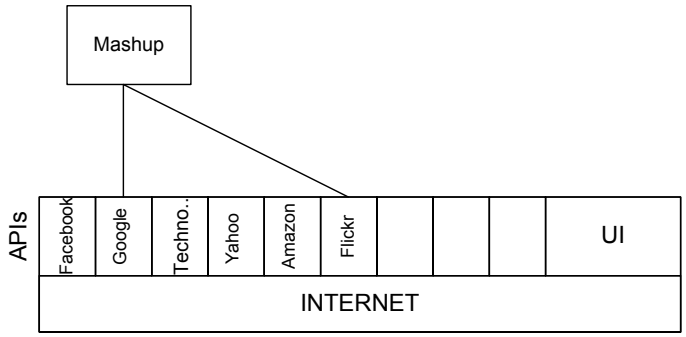


Figure 2: Mashup Model, (David Berlind, ZDNet), [25]

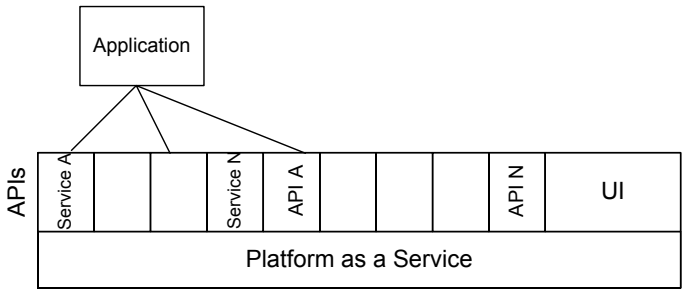


Figure 3: SaaS Model, [25]

- Mashups are usually created with a mashup editor such as: JackBe[10], IBM Mashup Center[11], Mozilla Ubiquity[12], Yahoo Pipes[13].
- Mashups use APIs from different platforms to aggregate and reuse the content.
- Usually mashups operate on XML based content such as Atom [20], [12], RSS 2.0 [26], and RDF [16], sometimes directly on the HTML level, strictly for presentation
- ”Melting Pot” style such that content is aggregated arbitrarily
- Create, read, update and delete (CRUD) operations are preferred to be based on REST principles

On the other hand the approach introduced in [25] removes the specific platform level and works directly on the content that any user has access to through a Web browser. This approach is in sync with the current trend where the browser is the server for the new type of web applications. A simple model of this approach with respect to mashup can be imagined as in Figure 4.

[10] `http://www.jackbe.com/`
[11] `http://www-01.ibm.com/software/lotus/products/mashups/`
[12] `http://labs.mozilla.com/projects/ubiquity/`
[13] `http://pipes.yahoo.com`

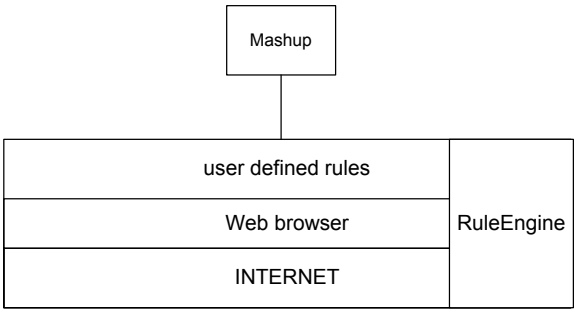


Figure 4: Our Model, [25]

4 Conclusions

We have presented an overview on the approaches for creating and executing mashups. The rule based approach has been introduced by us and strive to conform with most of the mashup characteristics. Moreover this approach is a client side approach, which makes it a *pure mashup* [3] approach. In addition we have touched also the problem of mashups in the general context of *Future Internet*.

References

[1] Mehmet Altinel, Paul Brown, Susan Cline, Rajesh Kartha, Eric Louie, Volker Markl, Louis Mau, Yip-Hing Ng, David Simmen, and Ashutosh Singh. Damia A Data Mashup Fabric for Intranet Applications. In *Proceedings of the 33th International Conference on Very Large Data Bases*. VLDB, 2007. `http://www.vldb.org/conf/2007/papers/demo/p1370-altinel.pdf`.

[2] K. Bennett, P. Layzell, D. Budgen, P. Brereton, L. Macaulay, and M. Munro. Service-Based Software: The Future for Flexible Software. In *Proceedings of the Seventh Asia-Pacific Software Engineering Conference (APSEC2000)*, pages 214–221. IEEE Computer Society, 2000. `http://www.bds.ie/Pdf/ServiceOriented1.pdf`.

[3] B. Bioernstad and C. Pautasso. Let it flow: Building Mashups with Data Processing Pipelines. In *Proc. of Mashups'07 International Workshop on Web APIs and Services Mashups at ICSOC'07*, 2007. `http://www.jopera.org/files/jopera_mashup07.pdf`.

[4] Bert Bos, Tantek Celik, Ian Hickson, and Hakon Wium Lie. Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification. W3C Candidate Recommendation, September 2009. `http://www.w3.org/TR/CSS2/`.

[5] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and Francois Yergeau. Extensible Markup Language (XML) 1.0 (Fifth Edition). W3C Recommendation, November 2008. `http://www.w3.org/TR/2008/REC-xml-20081126/`.

[6] Marcos Caceres. Widgets 1.0: Packaging and Configuration. W3C Candidate Recommendation, July 2009. http://www.w3.org/TR/widgets/.

[7] ECMA. ECMAScript Language Specification. http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf, December 1999.

[8] I. Foster and S. Tuecke. Describing the Elephant: The Different Faces of IT as Service. *Enterprise Distributed Computing*, 3(6):26–34, July/August 2005.

[9] Adrian Giurca and Emilian Pascalau. Building Intelligent Mashups. Tutorial presented at the Future Internet Symposium 2009 (FIS 2009), September 1-3, 2009, Berlin, Germany and at the 32nd Annual Conference on Artificial Intelligence (KI 2009), September 15-18, 2009, Paderborn, Germany, http://docs.google.com/View?id=dcff8ncf_181gxb3ss65.

[10] Adrian Giurca and Emilian Pascalau. JSON Rules. In *Proceedings of 4th Knowledge Engineering and Software Engineering, KESE 2008, collocated with KI 2008*, volume 425, pages 7–18. CEUR Workshop Proceedings, 2008.

[11] Lars Grammel and Margaret-Anne Storey. An End User perspective on Mashup Makers. Technical report, University of Victoria, September 2008. http://lars.grammel.googlepages.com/paper_mashup_makers.pdf.

[12] J. Gregorio and B. de hOra. Atom Publishing Format (RFC5023). http://tools.ietf.org/html/rfc5023, 2007.

[13] Ian Hickson and David Hyatt. HTML 5. A vocabulary and associated APIs for HTML and XHTML. W3C Working Draft, August 2009. http://www.w3.org/TR/html5/.

[14] A. Le Hors, P. Le Hegaret, L. Wood, G. Nicol, J. Robie, M. Champion, and S. Byrne. Document Object Model (DOM) Level 3 Core Specification. W3C Recommendation, April 2004. http://www.w3.org/TR/DOM-Level-3-Core/.

[15] H. Kagermann. Toward a European Strategy for the Future Internet A Call for Action. White paper, SAP AG, 2008. http://www.sap.com/about/company/research/fields/internet_services/index.epx.

[16] G. Klyne and J.J. Caroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation, February 2004. http://www.w3.org/TR/rdf-concepts/.

[17] A. P. McAfee. Enterprise 2.0: The Dawn of Emergent Collaboration. *MIT Sloan Management Review*, 47(3):21–28, 2006.

[18] Christian Morbidoni, Axel Polleres, Danh Le Phuoc, and Giovanni Tummarello. Semantic Web Pipes. Technical report, DERI, November 2007.

[19] Christian Morbidoni, Axel Polleres, and Giovanni Tummarello. Who the FOAF knows Alice? A needed step towards SemanticWeb Pipes. In Ruzica Piskac, Frank van Harmelen, and Ning Zhong, editors, *Proceedings of the First International Workshop Workshop "New forms of reasoning for the Semantic Web: scalable, tolerant and dynamic", co-located with ISWC 2007 and ASWC 2007*, volume 291. CEUR Workshop Proceedings, 2007. `http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-291/paper04.pdf`.

[20] M. Nottingham and R. Sayre. Atom Publishing Format (RFC4287). `http://tools.ietf.org/html/rfc4287`, 2005.

[21] OpenSocial. *OpenSocial Gadgets API Specification v0.9*. OpenSocial, 2009. `http://www.opensocial.org/Technical-Resources/opensocial-spec-v09/Gadgets-API-Specification.html`.

[22] T. O'Reilly. What is web 2.0. design patterns and business models for the next generation of software. *Oreillynet.com*, September 2005. `http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html`.

[23] John Palfrey and Urs Gasser. Case Study: Mashups Interoperability and eInnovation. *Breaking Down Digital Barriers*, pages 1–36, 2007.

[24] Emilian Pascalau and Adrain Giurca. A Lightweight Architecture of an ECA Rule Engine for Web Browsers. In *Proceedings of 5th Knowledge Engineering and Software Engineering, KESE 2009, collocated with KI 2009*, volume 486, pages 9–20. CEUR Workshop Proceedings, 2009.

[25] Emilian Pascalau and Adrian Giurca. A Rule-based Approach of creating and executing Mashups. In S. Sharma C. Godart, N. Gronau and G. Canals, editors, *Proceedings of the 9th IFIP Conference on e-Business, e-Services, and e-Society, I3E 2009*, pages 82–95. Springer, 2009.

[26] RSS. RSS 2.0 Specification, version 2.0.11. `http://www.rssboard.org/rss-specification`, March 2009.

[27] August-Wilhelm Scheer and Joerg Klueckmann. BPM 3.0. In *Proceedings of the 7th International Conference on Business Process Management, BPM 2009, Ulm, Germany*, pages 15–27. Springer, 2009.

[28] E. Traudt and A. Konary. Software as a Service Taxonomy and Research Guide. Technical report, IDC.com, 2005.

[29] Jonathan Zittrain. *The Future of the Internet And How to Stop It*. Yale University Press New Haven and London, 2008.

Requirements Traceability in Service-oriented Computing

Michael Perscheid
michael.perscheid@hpi.uni-potsdam.de

Software understanding, as a primary cost in software engineering, becomes more difficult since the complexity of service-based applications is steadily growing. Although the post-traceability of requirements is considered a critical component in program comprehension, current approaches often comprise only manual, tedious, and laborious processes with a small degree of automation.

In this paper, we present Gears, a programming language-independent approach for automated requirements traceability. Based on dynamic and static analysis in correspondence with the execution of acceptance tests, we connect requirements, architecture, and implementation artifacts closer together. A small example illustrates our idea for software understanding in service-oriented computing.

1 Introduction

Service-oriented computing allows for the loose coupling of service implementations for the composition of complex software systems. Such systems choose and combine several services to fulfill a certain requirement; they distribute the processing of a task; and in the case of a failure they can redirect to another service provider. With the concept of services, a system is divided into individual parts offering a new level of abstraction at the component level. Thus, service-oriented systems base on requirements, service components, and their implementations.

However, software understanding, as a primary cost in software engineering [7,25], becomes more difficult since the complexity of service-based systems is steadily growing. On the one hand, each application implements many requirements being distributed over the entire service landscape, on the other hand, the realization of individual services range from simple message passing to the orchestration of full-fledged applications [10]. Typical benefits of service-oriented computing such as the dynamic composition of services, the transparent interoperability, or the heterogeneity of service implementations hinder still more the comprehensibility of such systems [10]. Other approaches [26] for object-oriented or component-based comprehension cannot be blindly applied to service-oriented computing as they do not address these key issues. Thus, novel techniques must be developed to support the refinement from the early phases of requirements to their realization within services and their implementations [21].

Requirements post-traceability [11], as the capability to follow the life of a require-

ment down to architecture and implementation entities, provides significant information for understanding software systems. For instance, developers know where a certain requirement is implemented and in which requirements a certain entity is involved. Be it for reuse of existing components or to locate the cause of a failure. These days, the traceability of requirements is considered a critical component of any large-scale software system and its long term maintenance [5]. It improves not only program comprehension but rather enhances the overall quality of software and several software engineering activities [9, 12, 17, 22].

Current requirements traceability approaches, however, are often manual, tedious, and laborious [9, 12]. Especially, the small degree of automatization makes them impractical in the field of large-scale software development. Without appropriated concepts, software engineers can focus only on the most important traceability links. As a result, the traceability of requirements tends to get lost during software evolution. Another problem [23] is the large conceptual distance and the structural gap between different software views since no standard mapping between requirements, architecture, and implementation entities exists. Most traceability tools do not consider software from an integrated perspective where static and dynamic as well as all three software views are interrelated with each other. This leads to an incomplete picture and to difficulties in understanding software systems. All in all, requirements traceability remains a widely reported problem area in industry [11].

In this paper, we introduce Gears, a programming language-independent approach for automated requirements traceability. With Gears, we close the structural gap between requirements, architecture, and implementation artifacts by proposing a new view of software systems. Several existing meta-models are combined to define relationships between these heterogenous software artifacts and to create *one integrated model for requirements traceability*. We instantiate this model by adapted feature localization concepts in correspondence with the automatic execution of acceptance tests. Thus, we are able to provide trustworthy, up to date, and precisely captured trace information without the effort and complexity of manually gathered traces. Our approach is lightweight, adaptable to project-specific needs, and explicitly considers the component structure of software systems. We present the basic properties of Gears such as the conceptual meta-model, activities for application, and a first prototypic tool suite. In a small example, we demonstrate expected benefits of requirements traceability within software engineering.

The remainder of this paper is organized as follows: Section 2 describes Gears in more detail. Section 3 explains a small example of our approach. Section 4 presents related work. Section 5 concludes the paper and presents future work.

2 Gears: Automated Requirements Traceability

The goal of our approach is to maintain traceability information and to keep them up to date. We connect the three perspectives—requirements, architecture, and implementation—of a software system closer together and, so, we preserve the important information about the interrelationships between these views that will otherwise

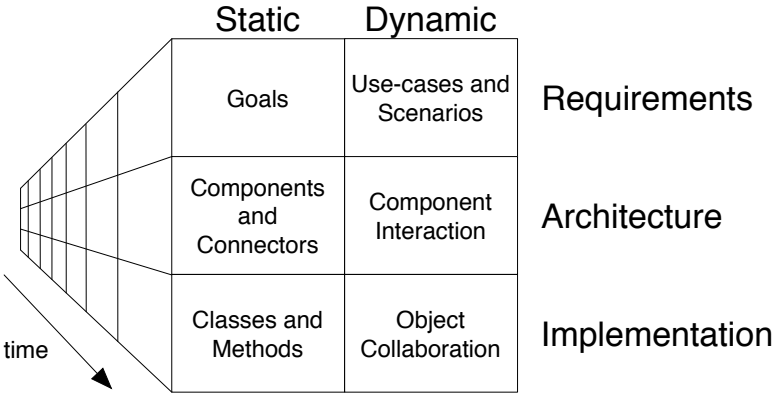


Figure 1: The conceptual meta-model.

be lost during software evolution. The principle of Gears is to leverage the similarity between the interaction and communication at different level of abstractions—users communicate with the system, services interact with each other, and objects send messages to other objects. In requirements, users interact with the system as a black box. This can be expressed by acceptance tests simulating the usage of the observed system. Such tests can be implemented within a testing framework making them automatically executable and repeatable. In architecture, the system is considered as a white box with components interacting with each other to fulfill a certain user communication step. In implementation, each component is more refined as the collaboration of objects that perform the request of a component. By connecting requirements with acceptance tests and analyzing them dynamically (feature localization [13, 27]), we can trace requirements down to architectural and implementation entities.

2.1 Meta-Model

The meta-model combines requirements, architecture, and implementation from both a static and a dynamic point of view. It structures trace information by defining concepts and relations that should be recorded during the automatic execution of acceptance tests.

Figure 1 describes the conceptual view of our meta-model. It consists of three dimensions—each part describes one particular aspect of the entire system from another perspective. By defining relationships between the six independent models from the first two dimensions, we create an integrated meta-model for requirements traceability. The time dimension is realized by multiple instances of the integrated model. The generic description of our view can be instantiated by several different models and, thus, it is useable in many different projects. For supporting project-specific adaptations, we define only minimal information capturing the common features of software systems. Figure 1 presents a concrete instance of our view, here, we assume typical concepts for each of the three development activities based on object-oriented and component-oriented paradigms.

Goals (model adapted from [22]) describe the functional requirements from a structural point of view—everything what the system has to provide. We map goals down to architecture and implementation entities. The behavioral specification of requirements is expressed by *use-cases* (model adapted from [22, 23]). Each use-case fulfills and concretes a certain *goal* by describing the interaction between actors and the system itself. Typically, a scenario step makes no assumptions about the system's internals. In architecture, this black box becomes clear and is built of components, connectors, and their interaction.

Primarily, software architectures (model adapted from [23]) consist of *components and connectors*. By using only both concepts, we assure a high-level abstraction where no assumptions about technical details are made. We distinguish between internal and external components whereas the former has access to the implementation level and can be composed of other components. Connectors link components together and determine a protocol with interfaces. As mentioned above, components are used in the realization of goals and the *interaction between components* (model adapted from [23]) is the refinement and internal behavior of use-case scenario steps. An architecture scenario consists of steps describing one interaction between components. These architecture scenario steps, in turn, trigger the collaboration of objects as the internal behavior of components.

Internal components are made up of source code entities such as *classes and methods* (model adapted from [2, 8]). This model is language-independent and supports several object-oriented programming languages [8]. By stopping at the granularity level of methods, we can hide the most language-dependent features that are not necessary for requirements traceability. Source code entities and goals are interrelated to each other analogous to components and goals. The internal run-time behavior of a component (model adapted from [2, 13]) is based on the *collaboration of objects*. Objects, as instances of classes, send messages from sender to receiver objects and, thus, execute methods. A trace scenario is the refinement of an architectural interaction after a component receives a request.

Due to the interconnections between all six views, our integrated meta-model constitutes a proper foundation for requirements traceability. Goals are concreted by use-cases and their scenarios. Scenarios are executed by actors or acceptance tests triggering internal behavior of the system. At the architecture level, components interact with each other to fulfill a use-case scenario step. At the implementation level, objects as part of components collaborate with each other to perform the next architectural interaction. Each object can be identified somewhere in the implementation structure and each source code entity is part of a certain component. Thus, we can derive traceability links between goals, components, and source code entities from the goal-dependent behavior of the system.

2.2 Activities for Application

Gears proposes activities describing how our approach can be integrated into existing software development processes. Three activities—definition, execution, and recovery—define tasks to instantiate the meta-model by particular tools or stakehold-

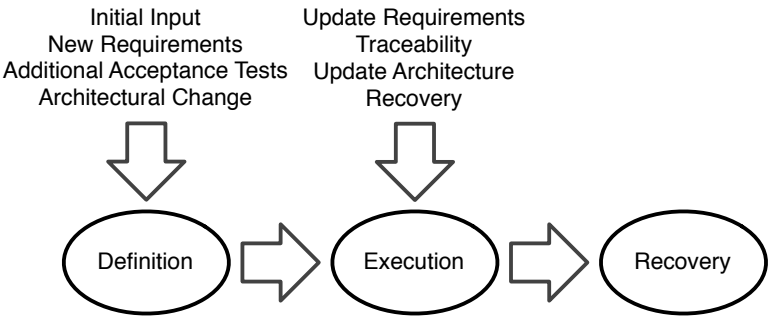


Figure 2: Activities for applying Gears within existing software projects.

ers. Figure 2 presents an overview of the activities and when they should be used.

The definition activity requires the most manual effort. Requirements engineers describe mostly manually what the system has to provide (goals) and how (use-cases). Simultaneously, software architects declare components and related source code entities in a coarse-grained view that will be refined later on. This task is supported by static analysis tools identifying structures in source code and connectors between components. Moreover, these tools search for test cases that can be linked by software testers with their corresponding requirements. The definition activity must be performed once the system gets new requirements, additional acceptance tests, or large architectural changes.

The execution activity is concerned with the automatic execution of acceptance tests and their observation by dynamic analysis tools. The meta-model is instantiated with information about the behavior of the system. Although we propose the automatic execution of tests, this step can also be done manually by software testers. Usually, dynamic analysis slows down the performance and produce a large amount of data. For that reason, we suggest to execute this task in a stand-alone test environment and at least in nightly builds.

The recovery activity deals with tools that use the collected information for recovery of requirements traceability and architecture. The former is done with adapted concepts from the feature location community [13, 27] and the latter connects the behavior of the system with its architectural description closing gaps such as forgotten connectors or refined components.

2.3 Tool Suite

We also suggest a tool suite for Gears (see Figure 3). The main component is a central repository that stores and provides the data of our meta-model. We prefer a relational database server as it is remotely accessible by different applications, can handle multiple users, and provides SQL as a unified API. A definition tool suite reflects the same-named activity for defining requirements, linking test, and describing the architecture within the central repository. The static analyzers support the software architect with the definition of the *big picture*. The dynamic analyzers observe the running system that

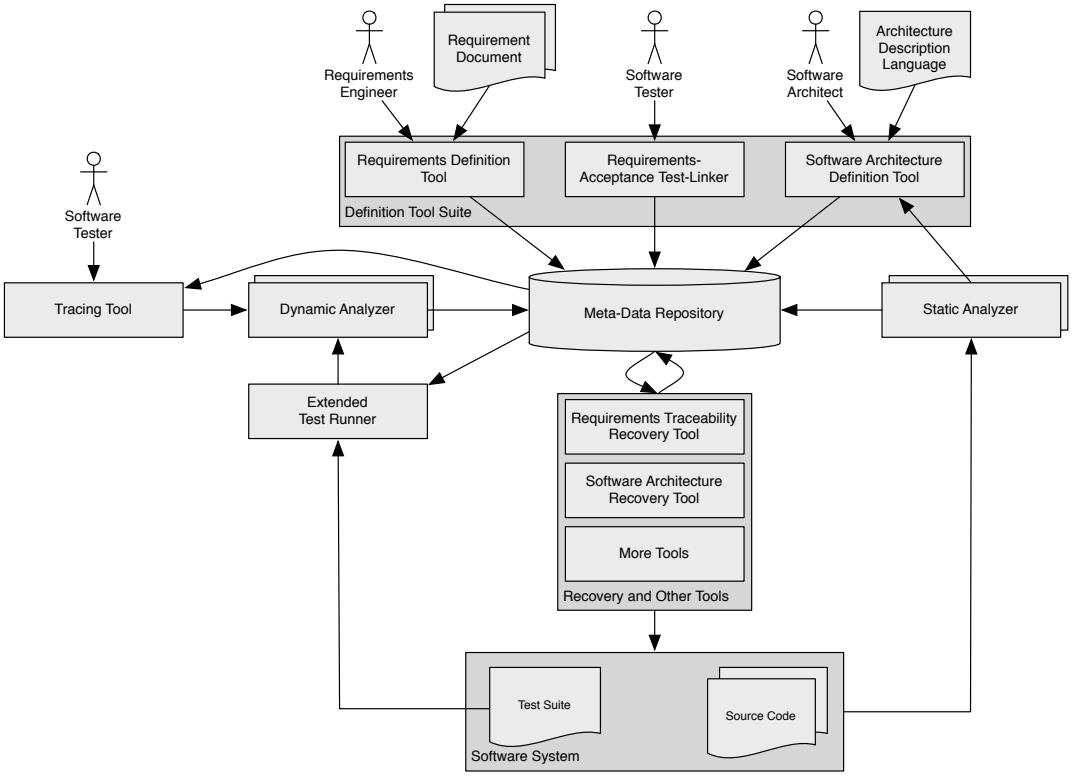


Figure 3: Proposed tool suite realizing Gears.

was triggered by an extended test runner or by a tracing tool for the manual execution of use-cases. We require various analyzers since they transform the language-dependent capabilities and properties of the system into our language-independent meta-model. For instance, in service-based systems, we need for monitoring the communication between services and for each available implementation a particular analyzer. Tools for the recovery activity can be seen under the repository as they read and manipulate the gathered information for particular purposes such as requirements traceability. These tools have also the possibility to influence the system itself, for instance for automatic reengineering.

3 Traceability Example

We demonstrate the applicability and the benefits of Gears within a small software system. The ”Travel Wizard” is a Web application where customers can query train connections, buy tickets, or combine different services to book a complete travel. In the following example (see Figure 4), we analyze the first requirement and its realization from all six views. At the top end, we define the requirements view. On the left-hand side, there is a goal for query train connections with two sub-goals for inserting data and choosing different modes. On the dynamic side, there is one use-case refining the

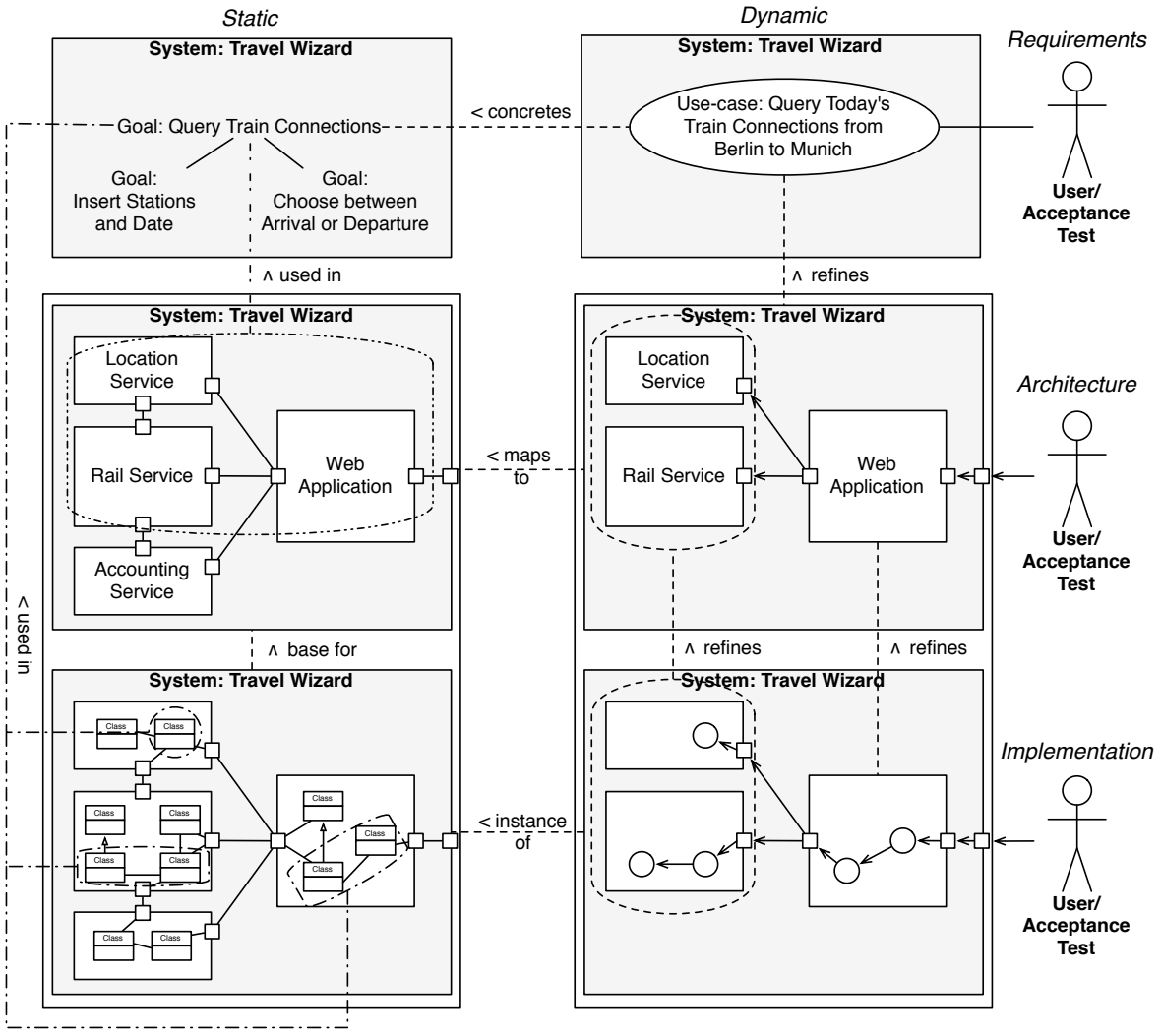


Figure 4: The dynamic analysis of the execution recovers the ”used in” relationship.

main goal. It describes what users (or acceptance test) have to do to query today's train connections from Berlin to Munich.

The architecture (center left) contains four components—three external services and the Web application itself. The location service returns information about towns such as their included rail stations. The rail service provides an API for external applications to query train connections or to reserve seats. The accounting service manages financial matters. Last but not least, the Web application provides a user interface for applying all the functionality of the Travel Wizard. We assume that the source code of all components is available (bottom left as classes).

The behavior of the use-case scenario can be seen on the right-hand side. The actor uses the Web application and its interface to query train connections from a town to another. During the input of arrival and departure stations, the Web application requests suggestions for rail stations from the location service. After all information

is given, the travel wizard forwards the request to the rail service. In the end, the result is presented to the user. The accounting service is not required in this use-case and the architecture scenario is traced between the three mentioned components. As refinement of the component interaction, the collaboration of objects describes trace scenarios at the implementation level.

Each use-case scenario concretes a certain goal and is followed by the interaction of components and objects. Due to the fact that each object was instantiated from a certain class and each component is unambiguous, we can identify for each executed component, object, and method a complement inside the static view. Thus, we can trace a goal from use-cases via internal behavior through to the structure of our system and reveal further information about the involved requirements (bottom left). For instance, both rail and location services and their corresponding classes are used in the goal "Query Train Connections". By using automatic acceptance tests and dynamic analysis concepts [13] this knowledge can be derived automatically.

Having these traceability links, we can improve several software engineering activities [22]. For instance, in *program comprehension*, developers get a new navigation concept as they can follow the life of a requirement down to architecture and implementation. They know which parts are involved in the implementation of a requirement as well as they know which requirements depend on a certain part of the system. In *project management*, customers know whether and where a certain requirement is implemented. By using various metrics, engineers can estimate the required development effort as they know the involved components and classes. Furthermore, similar requirements can be mapped to already implemented ones leading to a higher reusability of existing system parts. In *maintenance*, failures are often reported by users in natural language. These descriptions can easier be mapped to requirements as directly to source code. With related links to source code entities, developers can reduce the search space for the cause of a failure.

4 Related Work

A variety of manual requirements traceability approaches exist such as the most known concepts of requirements traceability matrices [24] and issue-based information systems (IBIS) [19] being implemented in several hyperlink tools [3, 6]. The components, connectors, overall system, and their properties (CBSP for short) [14] and the software architecture analysis method (SAAM) [18] are light-weight approaches for traceability between requirements and architecture capturing and maintaining arbitrary complex relationships. All approaches are flexible with respect to the regarded documents and the entire software development life cycle. However, they suffer from manual effort to define mappings and to keep them up to date.

Event-based traceability [4] establishes loosely coupled traceability links between requirements and other documents. After creating manual links, an event server automatically notifies all subscribed entities if a requirement changes. Based on a few manually created links, LeanArt [12] recovers traceability between use-case diagrams, types, and variables. LeanArt mimics the human-driven procedure of searching for

similarities between program entities and use-case elements with run-time monitoring, program analysis, and machine learning. The TraceAnalyzer [9] is similar to our approach since it uses run-time information of scenarios to recover traceability links. TraceAnalyzer considers only classes and the used propagation technique has the drawback to potentially derive incorrect relationships.

There are also some approaches based on information retrieval [1, 16, 20]. These techniques have the benefit to process fully automatic but they have even the drawback to create many false-positive mappings. In a position paper [15], the authors suggest the dynamic analysis of acceptance and unit tests for requirements traceability. This idea is similar to the research area of feature location [13, 27]. Both approaches do not consider the architecture of software systems.

5 Summary and Outlook

In this paper, we introduce an automatic approach for requirements traceability called Gears. Based on an integrated meta-model, we bridge the structural gap between heterogenous software artifacts and propose a minimalist, generic, and programming language-indepent view of software systems. By using static and dynamic analysis tools, we trace the execution of acceptance tests and instantiate our meta-model with the necessary information to recover relationships between requirements, architecture, and implementation entities.

The consideration of the architectural view enables Gears also in a service-oriented environment. Services can be represented by components that interact with each other at the architectural level and their internal behavior as the collaboration of objects. Thus, we can follow the life of a requirement step-by-step through the interaction of services as well as down to parts of their implementation in a transparent manner. With this knowledge, developers can understand software from the user's point of view as they know why a certain method is really required. A small example illustrates our idea.

We plan to extend Gears and our early prototype. Next steps include the creation of a proper implementation and infrastructure; the realization of tools for requirements traceability; and an evaluation of our approach.

References

[1] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo. Recovering Traceability Links between Code and Documentation. *IEEE Transactions on Software Engineering*, 28(10):970–983, 2002.

[2] E. Arisholm, Lionel C. B., and Audun F. Dynamic Coupling Measurement for Object-Oriented Software. *IEEE Transactions on Software Engineering*, 30(8):491–506, 2004.

[3] H.U. Asuncion, F. François, and R.N. Taylor. An End-to-End Industrial Software Traceability Tool. In *Proceedings of ACM SIGSOFT symposium on the foundations of software engineering*, pages 115–124. ACM New York, NY, USA, 2007.

[4] J. Cleland-Huang. *Robust Requirements Traceability for Handling Evolutionary and Speculative Change*. PhD thesis, University of Illinois, 2002.

[5] J. Cleland-Huang, C. K. Chang, and J. C. Wise. Automating Performance-related Impact Analysis through Event Based Traceability. *Requirements Engineering*, 8(3):171–182, 2003.

[6] J. Conklin and M.L. Begeman. gIBIS: A Hypertext Tool for Team Design Deliberation. In *Proceedings of the ACM conference on Hypertext*, pages 247–251. ACM New York, NY, USA, 1987.

[7] J. W. Davison, D. M. Mancl, and W. F. Opdyke. Understanding and Addressing the Essential Costs of Evolving Systems. *Bell Labs Technical Journal*, 5(2), 2000.

[8] S. Demeyer, S. Tichelaar, and S. Ducasse. FAMIX 2.1-the FAMOOS Information Exchange Model. *Research Report, University of Bern*, page 11, 2001.

[9] A. Egyed and P. Grünbacher. Automating Requirements Traceability: Beyond the Record & Replay Paradigm. In *Proceedings of the 17th IEEE International Conference on Automated Software Engineering*, pages 163–171, 2002.

[10] N. Gold, C. Knight, A. Mohan, and M. Munro. Understanding Service-Oriented Software. *IEEE Software*, 21(2):71–77, 2004.

[11] O. C. Z. Gotel and A. C. W. Finkelstein. An Analysis of the Requirements Traceability Problem. In *Proceedings of the First International Conference on Requirements Engineering*, pages 94–101, 1994.

[12] M. Grechanik, K. S. McKinley, and D. E. Perry. Recovering And Using Use-Case-Diagram-To-Source-Code Traceability Links. In *Proceedings of the Symposium on the Foundations of Software Engineering*, pages 95–104, 2007.

[13] O. Greevy. *Enriching Reverse Engineering with Feature Analysis*. PhD thesis, University of Berne, 2007.

[14] P. Grünbacher, A. Egyed, and N. Medvidovic. Reconciling Software Requirements and Architectures: The CBSP Approach. *IEEE International Conference on Requirements Engineering*, 0:0202, 2001.

[15] J.H. Hayes, A. Dekhtyar, and D.S. Janzen. Towards Traceable Test-driven Development. In *ICSE Workshop on Traceability in Emerging Forms of Software Engineering*, pages 26–30, 2009.

[16] J.H. Hayes et.al. REquirements TRacing On target (RETRO): Improving Software Maintenance through Traceability Recovery. *Innovations in Systems and Software Engineering*, 3(3):193–202, 2007.

[17] E. Hull, K. Jackson, and J. Dick. *Requirements Engineering*. Springer, 2005.

[18] R. Kazman, G. Abowd, L. Bass, and P. Clements. Scenario-based Analysis of Software Architecture. *IEEE Software*, 13(6):47–55, 1996.

[19] W. Kunz and H. W. J. Rittel. Issues as elements of information systems. Technical report, Universität Stuttgart Institut fur Grundlagen der Planung, 1970.

[20] A. Marcus and J. I. Maletic. Recovering Documentation-To-Source-Code Traceability Links Using Latent Semantic Indexing. In *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*, pages 125–135, 2003.

[21] M.P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann, and B.J. Kramer. Service-oriented Computing: A Research Roadmap. *International Journal of Cooperative Information Systems*, 17(2):223–255, 2008.

[22] K. Pohl. *Requirements Engineering*. Dpunkt, 2007.

[23] K. Pohl, M. Brandenburg, and A. Gülich. Integrating Requirement and Architecture Information: A Scenario and Meta-Model Approach. In *Proceedings of the 7th International Workshop on Requirements Engineering*, pages 68–84, 2001.

[24] B. Ramesh, C. Stubbs, T. Powers, and M. Edwards. Requirements traceability: Theory and practice. *Annals of Software Engineering*, 3:397–415, 1997.

[25] T. Standish. Essay on Software Reuse. *IEEE Transactions on Software Engineering*, 10(5):494–497, 1984.

[26] B. Steinert, M. Perscheid, M. Beck, J. Lincke, and R. Hirschfeld. Debugging into Examples - Leveraging Tests for Program Comprehension. In *Proceedings of the 21st International Conference on Testing of Communicating Systems*, to appear, 2009.

[27] N. Wilde and M. C. Scully. Software Reconnaissance: Mapping Program Features to Code. *Journal of Software Maintenance: Research and Practice*, 7(1):49–62, 1995.

Models at Runtime for Monitoring and Adapting Software Systems

Thomas Vogel
thomas.vogel@hpi.uni-potsdam.de

The use of model-driven engineering techniques for managing or self-managing software systems can yield to significant benefits as models can be used to provide a rich semantic base for monitoring, analyzing, planning and adapting software systems. Moreover, the administration of today's complex systems could be eased. This paper outlines the application of models at runtime for monitoring and adapting software systems. This includes a discussion of the causal relationship between those models and a running system, appropriate abstractions of runtime models, the maintenance of runtime models, and of emerged research challenges. The ideas are presented in the context of *IT Service Management* and of *autonomic*/*self-adaptive* software. Both have in common that runtime models reflecting a running system can be used for monitoring and adapting the system.

1 Introduction

The complexity of today's software systems impedes the management of these systems by humans. To increase the level of automation for management and to move towards the vision of *self-adaptive software* [6] and *Autonomic Computing* [19], the usage of *Model-Driven Engineering* (MDE) techniques can in principle help. Though the MDE discipline is focused on the design, implementation and deployment stages of software development, MDE techniques could be basically applied at runtime for managing software systems, as argued by France and Rumpe [10].

Models at runtime can yield to significant benefits as they can provide a rich semantic base for monitoring, analyzing, planning and adapting a managed software system [1]. MDE techniques, like transformation, synchronization, merge, comparison, and analysis on models, can increase the level of automation and finally reduce the complexity of managing systems.

This paper outlines approaches that use models conforming to meta models and selected model-driven techniques for monitoring and adapting software systems. Runtime phenomena that occur in a managed system are observed through models reflecting the structure and behavior of a running system at different levels of abstractions. Such models can be used for further activities, like analyzing a system and deciding about required adaptations. Finally, executing adaptations of a managed system are also based on such models.

The paper is structured as follows: the approaches are presented in the context of *IT Service Management* (Section 2) and of *autonomic*/*self-adaptive* software (Section 3), while outlining research challenges for future work. Section 4 concludes the paper.

2 IT Service Management

The de-facto standard to manage today's complex and heterogeneous IT systems is the *IT Infrastructure Library* (ITIL), currently in version 3, that provides a catalogue of best practices for *IT Service Management* (ITSM) containing common definitions by using a common terminology. The integral part of ITSM is the *Configuration Management System* (CMS), which is primarily a tool set and storage for *Configuration Items* (CI). CIs are manageable elements of a managed system and they can be services, components, incidents, hardware, software, buildings, persons, etc. The goal of a CMS is to increase the quality and efficiency of system management.

Several commercial ITIL implementations exist, e.g., IBM Service Management [2], which can be seen as state-of-the-art in ITSM. These approaches are quite powerful and comprehensive. Nevertheless, they rely on a set of XML-based standards and do not leverage the full strength of MDE. On the other hand, research approaches do not focus on ITSM as proposed by ITIL, but they embrace aspects of runtime models in different domains for managing running software systems [3, 5, 15, 18, 23].

This gap can be filled by exploiting model-driven techniques for a CMS, as we proposed in [13]. Models based on meta models improve the manageability by providing a suitable abstraction of a managed system, which enables direct user interaction as well as the application of MDE techniques, like model transformations. Furthermore, vital elements of a model-driven CMS are runtime models, which capture the structure and behavior of a running managed system. Due to the broad scope of ITIL, we focused on parts of it by addressing *Service Asset & Configuration Management* (reflecting CIs and configurations of CIs in a CMS), *Change Management* (deciding about and planning changes of CIs), and *Release & Deployment Management* (performing changes to a running system) through a model-driven CMS. However, the proposed CMS is open to extensions for other management activities as defined by ITIL.

2.1 Model-Driven Configuration Management System (CMS)

After presenting a generic CMS that conforms to ITIL, potentials for applying MDE are proposed.

Additionally to CIs, a generic CMS contains logical dependencies between CIs that have to be captured as well as information that is required by management processes, such as *Change Management* or *Release & Deployment Management*. Managing CIs in the CMS is the task of *Service Asset & Configuration Management*. Figure 1 shows a CMS in the context of ITSM. A CMS consists of a federated *Configuration Management Database* (CMDB) and a set of *Managed Data Repositories* (MDRs). An MDR focuses on a specific domain of a managed system, e.g., database servers or certain applications, and thus contains detailed information about CIs in that domain. MDRs gather information about CIs from different sources within a managed system through *Management Interfaces*. The federated CMDB is responsible for providing all relevant CIs of a managed system and their logical dependencies at a higher level of abstraction than the MDRs do.

Therefore, each MDR *federates* its CIs to the CMDB where a coherent and consistent set of CIs is *reconciled*. The CIs of the CMDB do not need to capture all details about managed elements but at least basic information and references to corresponding CIs in the MDRs, where detailed information about them are captured. Furthermore, a CMDB provides interfaces, which are used by management tools to *query* and *update* CIs and management information.

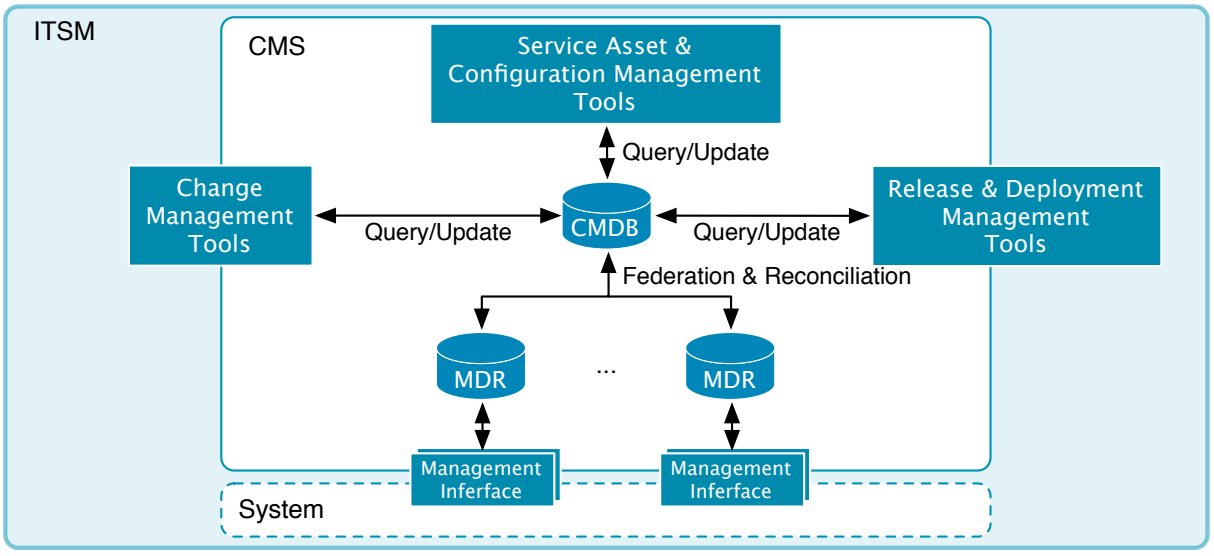


Figure 1: A Generic CMS in the Context of *IT Service Management* (ITSM)

Based on the generic CMS described above, models based on meta models and MDE techniques such as transformation, synchronization, merge, comparison, and analysis on models can be applied as follows.

Management Tools Management tools for processes, like *Change Management*, can use models as an underlying formalism and graphical model editors to provide a view on a managed system at an appropriate level of abstraction that facilitates the understanding and management of the system. For example, a *Change Management* tool is used to model changes that should be propagated to a running software system. Based on data from a CMDB, if multiple representations of models are beneficial, transformation, synchronization or merging are suitable applications of model-driven techniques. Model analysis, e.g. in the form of checking constraints, can be applied for reasoning, which supports decision making.

CMDB A runtime model is applicable within a CMDB capturing the structure of a managed system. Thus, CIs being part of a managed system have to be captured in the runtime model. In addition, the runtime model has at least to capture interconnections between CIs and changeable configuration properties whose adaptation influences the structure or behavior of items in a managed system. Such a runtime model is called a *configuration model*. For each management process, appropriate management models are beneficial, which capture all required management information and the relationships to CIs being part of the configuration model. For example, a *Change Management* model can capture information about planned changes that contain relationships to CIs on which these changes have to be performed.

MDR A runtime model is also applicable to MDRs. Each MDR captures a domain and therefore a specific subset of a managed system. Moreover, this runtime model is usually vendor specific. Hence, it contains vendor specific information which is not captured in the configuration model of the CMDB. A MDR runtime model is called a *vendor specific configuration model*.

Query & Update Connecting a management tool to a CMDB can be conducted by just copying the queried model into the management tool or by applying a transformation/merge that provides a more comprehensive model tailored to the corresponding management process. A transformation/merge combines several models of the CMDB into a single model that can be further used in a model editor of a management tool. Furthermore, changes that are made to the models in a management tool have to be transferred back to the models of the CMDB, which corresponds to a model update.

Federation & Reconciliation Transformation is applicable to federate the vendor specific configuration model of each MDR into a *partial configuration model* within a federated CMDB. A partial configuration model is a subset of the configuration model used in the CMDB. The configuration model is derived by reconciling partial configuration models. This reconciliation can be conducted by applying model merge.

2.2 Implementation

Currently, we have implemented several aspects of a model-driven CMS as proposed in the previous section. The implementation covers an MDR for *Enterprise Java Beans 3.0* (EJB) [7] applications that is based on the management interface provided by the infrastructure *mKernel* [4]. The implemented CMDB is based on Eclipse CDO[1], simple management tools for *Service Asset & Configuration Management*, *Change Management* and *Release & Deployment Management* are implemented within Eclipse and EMF[2], and all models being used are defined by meta models. Furthermore, model transformations based on Story Diagrams [9] and Eclipse ATL[3] have been applied to automatically derive a runtime model in the form of a configuration model and to derive and execute changes to a running software system.

2.3 Future Work: Vision of Autonomic IT Service Management

Applying MDE techniques, the level of automation in a CMS can be increased to make progress towards the vision of autonomic computing in ITSM. Therefore, as future work further steps towards engineering a full round trip in ITSM needs to be done, while easing system management for humans. Moreover, additional MDRs for other domains than EJB-based applications should be supported to investigate the usage of MDE techniques for reconciliation and to validate appropriate configuration models representing heterogeneous systems.

[1] Connected Data Objects; http://www.eclipse.org/modeling/emft/?project=cdo
[2] Eclipse Modeling Framework; http://www.eclipse.org/emf
[3] ATLAS Transformation Language, http://www.eclipse.org/m2m/atl/

3 Autonomic and Self-Adaptive Software Systems

To keep its value for the user, software has to be frequently adapted to changes in the environment [21]. Software adaptation can be conducted by modifying program variables (*parameter adaptation*) or by exchanging structural system components (*compositional adaptation*) possibly changing the software architecture [22]. However, adaptations are impeded by the complexity of today's software systems.

The vision of *Autonomic Computing* approaches this problem by applying the concept of control loops, which originates from the domain of control engineering, and adapting it to suit enterprise computing by self-management capabilities such as *self-configuration*, *self-healing*, *self-optimization* and *self-protection* [19]. Most of the work on autonomic computing employs parameter adaptation and therefore can employ well understood control engineering techniques [16]. For some systems this is sufficient, but sometimes compositional adaptations have to be employed to achieve the needed self-management goals [20].

Therefore, each capability requires its own corresponding abstract view on a managed software system. This view reflects the runtime state of a system regarding its architecture and parameters in the context of the concern being addressed by the capability, e.g., performance in the case of self-optimization. These views should be provided by models.

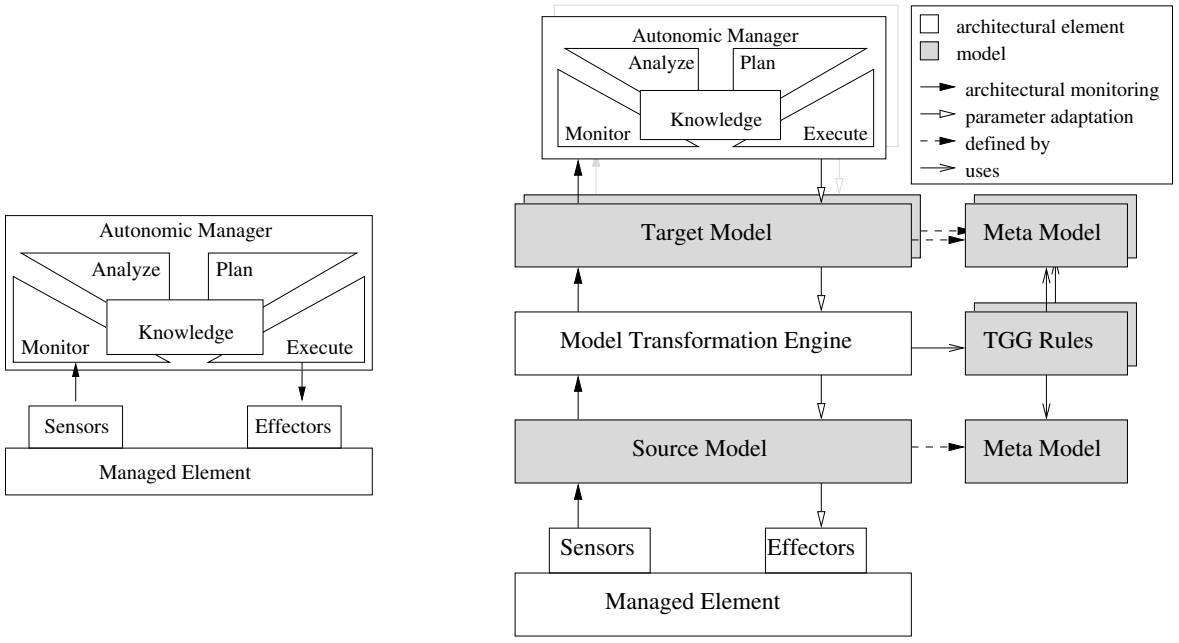
In this context, we propose a model-driven approach to ease the development of architectural monitoring and adaptation for autonomic systems [26, 27]. By employing MDE techniques and an optimized model transformation technique [12, 14] the runtime system and several models aiming at different self-management capabilities are synchronized online and incrementally.

While there are many examples for the benefits of adapting architectures at runtime (cf. [11, 24, 28]), the usage of MDE techniques and models at runtime in this context has recently found more attention [1]. Model-driven approaches considering runtime models, in contrast to our one, do not work incrementally to maintain those models [15] or they provide only one view on a managed system [8,23]. All these approaches [8,15, 23] do not apply advanced MDE techniques like model transformation between models being specified by different meta models. In this context, only first ideas exist, like the approach in [25], which performs only offline synchronizations, i.e., models have to be read from files, and therefore leads to a performance loss. Moreover, it seems that their approach involves non-incremental model transformations.

The next sections outline our model-driven approach and the implementation, and finally discusses future work.

3.1 Approach

Our approach combines MDE and the vision of autonomic computing. MDE techniques are employed to provide models describing different views on a managed software systems for different control loops being concerned with certain self-management capabilities. The generic architecture of our approach is depicted in Figure 2(b). It extends the control loop as shown in Figure 2(a) and proposed in [19] by introducing models as the interface between *Autonomic Managers* and the *Managed Element*.



(a) Control Loop [19] (b) Model-Driven Control Loop

Figure 2: Original (a) and Model-Driven Control Loop (b)

Instead of an *Application Programming Interface* (API) at a low level of abstraction, *Sensors* and *Effectors* provide a model-based view on a managed system in the form of a *Source Model* that can be used for monitoring or adapting the system in a model-based manner at runtime. The source model is maintained at runtime and updated if changes occur in a managed system. Nevertheless, a source model represents all capabilities of sensors and effectors. Consequently, a source model might be quite complex, which makes it laborious to use it as a basis for autonomic managers implementing the control loop activities monitoring, analysis, planning and execution. Thus, we propose to derive several *Target Models* at runtime using model transformation techniques. Each target model raises the level of abstraction with respect to the source model and it provides a specific view on the software system required for an autonomic manager focusing on a certain self-management capability. For example, a manager being concerned with *self-optimization* uses only those specific models that describe the resource utilization and performance attributes of a system, but does not have to consider views that are covered by other managers focusing on *self-healing* or *self-protection*. Consequently, several autonomic managers work on possibly different target models as depicted at the top of Figure 2(b).

The different target models are maintained by a generic *Model Transformation Engine* being based on *Triple Graph Grammars* (TGG) [12,14]. Source and target models are causally connected, i.e., changes in a source model are reflected in target models (monitoring) and vice versa (adaptation). This is possible due to the bidirectional transformation and synchronization capabilities of TGGs present in the engine, which work incrementally and therefore facilitate an efficient application at runtime. How two models have to be synchronized is specified declaratively by *TGG Rules* at the level of *Meta Models* for the source and target models. Hence, the rules are independent of concrete models. Thus, source and target models have to conform to user defined

meta models (cf. Figure 2(b)), and for each target meta model, TGG rules have to be defined that specify the synchronization between the source model and the corresponding target model. A TGG combines three conventional graph grammars: one grammar describes a source model, the second one describes a target model and a third grammar describes a correspondence model. A correspondence model explicitly stores the correspondence relationships between corresponding source and target model elements.

To detect model modifications efficiently, the transformation engine relies on a notification mechanism that reports when a source model element has been changed. To synchronize the changes of a source model to a target model, the engine first checks if the model elements are still consistent by navigating efficiently between both models using the correspondence model. If this is not the case, the engine reestablishes consistency by synchronizing attribute values and adjusting links. If this fails, the inconsistent target model elements are deleted and replaced by new ones that are consistent to the source model.

Synchronization between source and target models can be triggered on demand and the engine is able to synchronize two models that differ in more than one change. This enables the concurrent operations of managers and the decoupling of target models from a source model to ensure consistent analysis or to transfer several target model changes to the source model atomically.

3.2 Implementation

The implementation of the approach currently covers the capabilities to monitor the architecture and parameters and to adapt parameters of applications that have been realized with *Enterprise Java Beans 3.0* (EJB) [7] technology. It considers performance monitoring, architectural constraints checking and failure monitoring, and adapting values for *Simple Environment Entries* in these contexts. Simple environment entries are configuration properties of enterprise beans.

The implementation is based on sensors and effectors the autonomic computing infrastructure *mKernel* [4] provides for managing EJB-based systems. Based on the capabilities of these sensors and effectors, a meta model for the EJB domain has been developed that defines the source model. For target models, three meta models and the required TGG rules covering the architecture, performance and failure states, which aim at autonomic managers being concerned with self-configuration, self-optimization, and self-healing, respectively, have been developed. Our model transformation engine implementation and all meta models are based on the *Eclipse Modeling Framework* (EMF). However, the whole infrastructure can run decoupled from the *Eclipse* workbench.

3.3 Future Work

As the current solution fully automates the monitoring of a managed system, it is planned to cover the adaptation of architectures, i.e., compositional adaptations, and to investigate the degree of automation for adaptations. Compositional adaptations are

more complex and might consist of a set of changes. Due to such complex changes, there are several open research challenges that have to be tackled. First of all, the order of complex architectural changes performed on a target model might differ from the order of performing corresponding changes to the source model and therefore to the managed system. Not suitable orders might, e.g., affect the consistency of a system. Moreover, as source and target models reflect a managed system at different levels of abstraction, this abstraction step might be too large to enable easily a bidirectional synchronization. As a consequence, synchronizing target model changes to the source model requires additional information such as default values that depend on the concrete application. The same problem is discussed for model synchronization in round-trip engineering in [17] that emphasizes the difficulties of bidirectional model synchronization. Finally, autonomic managers working on different target models have to be coordinated to avoid conflicts when several of them want to adapt the system concurrently. Though, the common source model can be used for coordination, there is still the problem that concurrent changes of disjoint parts of a model might interfere at the semantic level.

4 Conclusion

This paper presented two approaches that use MDE techniques and models at runtime for managing software systems in the contexts of IT service management and autonomic/self-adaptive software systems. In both contexts, we have shown that models at runtime are an appropriate manner to support the monitoring and adaptation of a running system. Moreover, implementations and open research challenges for both approaches have been discussed. As the current results for both approaches are promising, these challenges will be addressed to move towards the vision of autonomic computing. Future work will mainly focus on enabling model-based architectural adaptations and to increase the level of automation, while investigating appropriate case studies from both contexts.

References

[1] *Workshop Models@run.time at the IEEE/ACM International Conference on Model Driven Engineering Languages and Systems (MoDELS)*, 2006-2009.

[2] IBM Service Management. *IBM Syst. J.*, 46(3), 2007.

[3] Anatoly Akkerman, Alexander Totok, and Vijay Karamcheti. Infrastructure for Automatic Dynamic Deployment of J2EE Applications in Distributed Environments. In *Proc. of the 3rd Intl. Working Conference on Cmponent Deployment*, pages 17–32. Springer, 2005.

[4] Jens Bruhn, Christian Niklaus, Thomas Vogel, and Guido Wirtz. Comprehensive support for management of Enterprise Applications. In *Proc. of the 6th Intl. Conference on Computer Systems and Applications*, pages 755–762. IEEE, 2008.

[5] Mauro Caporuscio, Antinisca Di Marco, and Paola Inverardi. Model-based System Reconfiguration for Dynamic Performance Management. *Journal of Systems and Software*, 80(4):455 – 473, 2007.

[6] Betty Cheng, Rogerio de Lemos, Holger Giese, Paola Inverardi, Jeff Magee, and et al. Software Engineering for Self-Adaptive Systems: A Research Road Map. Number 08031 in Dagstuhl Seminar Proceedings, 2008.

[7] Linda DeMichiel and Michael Keith. *JSR 220: Enterprise JavaBeans, Version 3.0: EJB Core Contracts and Requirements*. 2006.

[8] Jeremy Dubus and Philippe Merle. Applying OMG D&C Specification and ECA Rules for Autonomous Distributed Component-based Systems. In *Proc. of 1st Intl. Workshop on Models@run.time at MoDELS 2006*, 2006.

[9] Thorsten Fischer, Joerg Niere, Lars Torunski, and Albert Zündorf. Story Diagrams: A New Graph Rewrite Language Based on the Unified Modeling Language and Java. In *Proc. of the 6th Intl. Workshop on Theory and Application of Graph Transformations*, pages 296–309. Springer, 2000.

[10] Robert France and Bernhard Rumpe. Model-driven Development of Complex Software: A Research Roadmap. In *Proc. of the Workshop on Future of Software Engineering*, pages 37–54. IEEE, 2007.

[11] David Garlan, Shang-Wen Cheng, An-Cheng Huang, Bradley Schmerl, and Peter Steenkiste. Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure. *Computer*, 37(10):46–54, 2004.

[12] Holger Giese and Stephan Hildebrandt. Incremental Model Synchronization for Multiple Updates. In *Proc. of the 3rd Intl. Workshop on Graph and Model Transformations*. ACM, 2008.

[13] Holger Giese, Andreas Seibel, and Thomas Vogel. A Model-Driven Configuration Management System for Advanced IT Service Management. In *Proc. of the 4th Intl. Workshop on Models@run.time at MoDELS 2009*, volume 509 of *CEUR Workshop Proceedings*, pages 61–70, 10 2009.

[14] Holger Giese and Robert Wagner. From Model Transformation to Incremental Bidirectional Model Synchronization. *Software and Systems Modeling*, 8(1), 2009.

[15] Christian Hein, Tom Ritter, and Michael Wagner. System Monitoring using Constraint Checking as part of Model Based System Management. In *Proc. of the 2nd Intl. Workshop on Models@run.time at MoDELS 2007*, 2007.

[16] J.L. Hellerstein, Yixin Diao, S. Parekh, and D.M Tilbury. Control Engineering for Computing Systems - Industry Experience and Research Challenges. *Control Systems Magazine, IEEE*, 25(6):56–68, 2005.

[17] Thomas Hettel, Michael J. Lawley, and Kerry Raymond. Model Synchronisation: Definitions for Round-Trip Engineering. In *Proc. of the Intl. Conference on Model Transformation*, pages 31–45. Springer, 2008.

[18] P. Hnetynka. A Model-driven Environment for Component Deployment. In *Proc. of the 3rd ACIS Intl. Conference on Software Engineering Research, Management and Applications*, pages 6–13, 2005.

[19] Jeffrey O. Kephart and David M. Chess. The Vision of Autonomic Computing. *IEEE Computer*, 36(1):41–50, January 2003.

[20] Jeff Kramer and Jeff Magee. Self-Managed Systems: an Architectural Challenge. In *Proc. of the Workshop on Future of Software Engineering*, pages 259–268. IEEE, 2007.

[21] Meir M. Lehman. Software's Future: Managing Evolution. *IEEE Software*, 15(01):40–44, 1998.

[22] Philip K. McKinley, Seyed Masoud Sadjadi, Eric P. Kasten, and Betty H. C. Cheng. Composing Adaptive Software. *IEEE Computer*, 37(7), July 2004.

[23] Brice Morin, Olivier Barais, and Jean-Marc Jézéquel. K@RT: An Aspect-Oriented and Model-Oriented Framework for Dynamic Software Product Lines. In *Proc. of the 3rd Intl. Workshop on Models@run.time at MoDELS 2008*, pages 127–136, 2008.

[24] 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.

[25] Hui Song, Yingfei Xiong, Zhenjiang Hu, Gang Huang, and Hong Mei. A Model-Driven Framework for Constructing Runtime Architecture Infrastructures. Technical Report GRACE-TR-2008-05, GRACE Center, National Institute of Informatics, Japan, December 2008.

[26] Thomas Vogel, Stefan Neumann, Stephan Hildebrandt, Holger Giese, and Basil Becker. Incremental Model Synchronization for Efficient Run-time Monitoring. In *Proc. of the 4th Intl. Workshop on Models@run.time at MoDELS 2009*, volume 509 of *CEUR Workshop Proceedings*, pages 1–10, 10 2009.

[27] Thomas Vogel, Stefan Neumann, Stephan Hildebrandt, Holger Giese, and Basil Becker. Model-Driven Architectural Monitoring and Adaptation for Autonomic Systems. In *Proc. of the 6th Intl. Conference on Autonomic Computing and Communications*, pages 67–68. ACM, 6 2009.

[28] Jie Yang, Gang Huang, Wenhui Zhu, Xiaofeng Cui, and Hong Mei. Quality Attribute Tradeoff through Adaptive Architectures at Runtime. *Journal of Systems and Software*, 82(2):319 – 332, 2009.

Services for Real-Time Computing

Uwe Hentschel, M.Sc.
uwe.hentschel@hpi.uni-potsdam.de

Service oriented computing became more and more important in the last years. One Example is the increase in popularity of web services for business applications. The principles of service orientation are also interesting for real-time applications. One point that is difficult in this case is the network based implementation strategy of many services. In order to analyze the influence of network components a round-trip-delay measurement was made in an IP based network environment.

Furthermore, service oriented computing leads to distributed systems. The Fontane project, a telemedicine project, forms such a distributed system. The communication parts are of special interest. The vital signs of a patient are to be measured and stored under real-time conditions, the measured data and results of preliminary analysis are to be transmitted to the telemedicine center and all the attended doctors want to access the electronic health record. This paper discusses first thoughts to this project.

Introduction

In general services are something that provides a definite functionality and can be used by a client or application. In order to be useable by others services have to provide a public interface. Figure 1 shows different characteristics of services from the point of view of a particular user. Services in the outer circle, the white one, provides their functionality indirectly to the user. Because of that the user does not know anything about the service, the service can only be controlled by a third party.

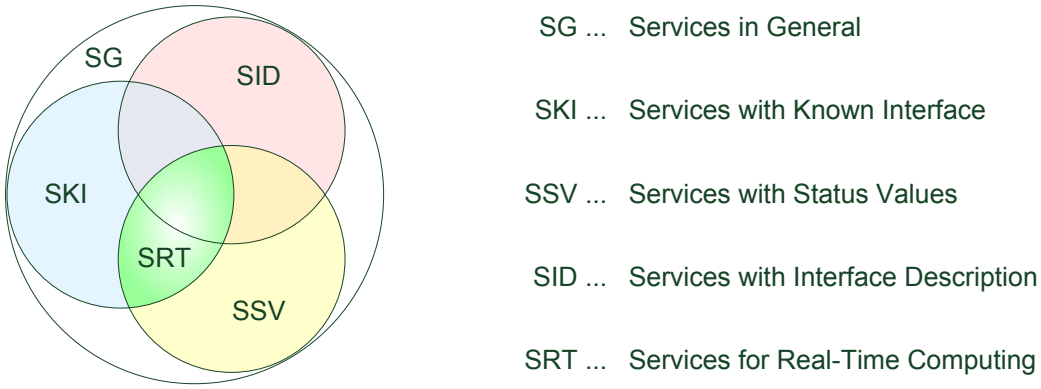


Figure 1: Service Categories

The next category represents services whose interface specification are known by the user – the blue circle. This one can be directly controlled by the user but they can

not dynamically managed. In this case the creator of the application has to implement the functionality to use the service in a static way. The services in the red circle provide a machine readable interface description. This makes it possible to control and combine the services dynamically at run-time.

The last category are services that provide status information. The easiest way to do so is to return a result value to the user but thinkable is each kind of status information. Status information are helpful for the user in order to detect problems and errors respectively or to synchronize different activities at run-time.

In the field of real-time computing services are interesting which have an user known interface in order to be direct controllable, return status information to the user which are useable for fault tolerance and synchronization of activities and have optionally a machine readable interface description for dynamic management. This group of services is placed inside the green area.

Measurement of Round-Trip-Delay

Within this experiment [2] an overview is given of the time required for communication in an IP-based network using Microsoft Windows as operating systems. Two computers are used whose communication speed was determined at different configurations using a simple client-server application. The basic specifications are listed below.

Computer 1 – Desktop:

Processor:	Intel Pentium 4 540
Processor clock:	3.20 GHz
Number of cores:	1
Hyper-Threading:	yes
CPU architecture :	32 bit
Main memory:	2 GByte DDR2 – 2 Channels
Memory clock:	266 MHz
Network:	Broadcom NetXtreme 57xx Gigabit Controller
IP versions:	4 and 6
Operating system:	Microsoft Windows XP Professional (SP 3)

Computer 2 – Laptop:

Processor:	Intel Mobile Core 2 Duo T9550
Processor clock:	2.66 GHz
Number of cores:	2
Hyper-Threading:	no
CPU architecture :	64 bit
Main memory:	4 GByte DDR3 – 2 Channels
Memory clock:	533 MHz
Network:	Intel 82567LM Gigabit Network Connection
IP versions:	4 and 6
Operating system:	Microsoft Windows 7 Ultimate Edition (Beta)

The client generates a data packet with a length of 256 bytes and sends it to the server. The server sends all received data back to the sender. On both sides, client and server, the time needed for sending and receiving is measured (see figure 2). The first measurement determined by the server is ignored since this also includes the time for starting and initializing the client.

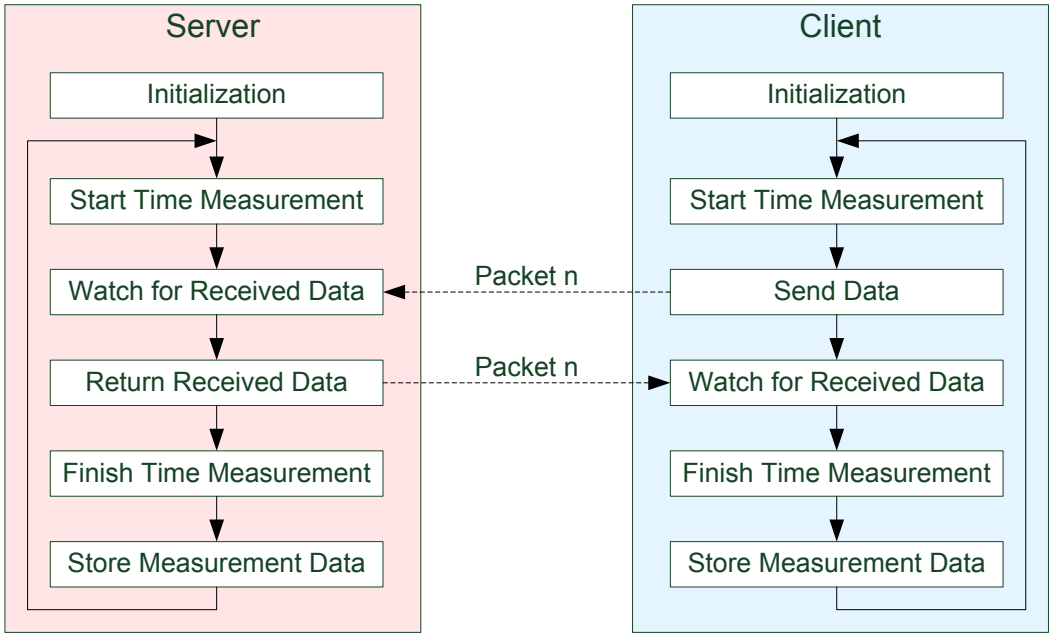


Figure 2: Principle of Time Measurement

The data transmission is connectionless with UDP packets. This will first limit the communication to the transmitted data – no additional acknowledgment data – and second the server processes exactly the same amount of data as the client because the data can only be read in packets. For communication on the network layer (OSI layer 3) IP is used in version 4.

Both tasks, server and client, have been set to priority 15 – the highest non-real-time priority for Microsoft Windows. The particular processes have retained their normal priority of 9 (base priority).

Accuracy of Time Measurement

On both computers a counter with high resolution (High Performance Counter) is used for time measurement. The Windows API provides the following two functions for using this counter:

- `QueryPerformanceFrequency ()` and
- `QueryPerformanceCounter ()`.

These functions can determine the resolution of the counter in steps per second and the current counter value.

To measure the time for a particular action, the counter value before and after this action is determined and the difference of the values is divided by the resolution. In this method the execution time of the function call to determine the current counter value influences each measurement. Therefore, it was previously analyzed how long it takes to determine the counter values and how high the resolution of the counter is on both computers. The values displayed in table 1 are made of a series of measurements each with a million individual measurements.

	Computer 1 – Desktop	**Computer 2 – Laptop**
Mean Value	330.3 ns	1084.7 ns
9th Decile	340.8 ns	1154.9 ns
Standard Deviation	219.7 ns	321.9 ns
Resolution	0.3 ns	385.0 ns

Table 1: Time of determining a counter value

The 9th decile indicates the value that applies to that at least 90% of the measured values are less or equal and at most 10% thereof are greater.

Time Measurement and Analysis

Because of the complexity of the experiment, the following explanation is based on the example running the client software on computer 1.

To make a distinction between the influences of operating system and network, in the first step both parts of the software running locally on one computer. The client uses the ordinary IP address of the computer to send data to the server and not the synonym '127.0.0.1' for the local computer. To determine the effects of other components in the network (routers, switches, ...), in the second step a comparative measurement was made on a local network formed by the two computers. And in the last step the measurements were carried out in the HPI network. Inside this network two switches and one router are arranged between the computers (see figure 3).

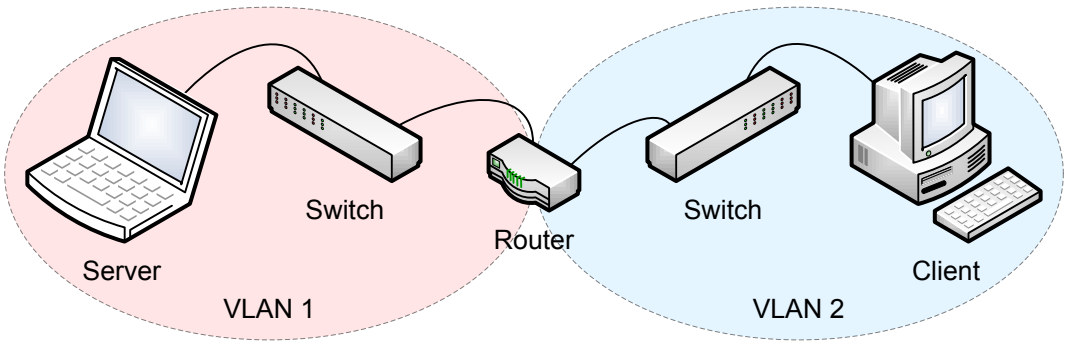


Figure 3: Network layout

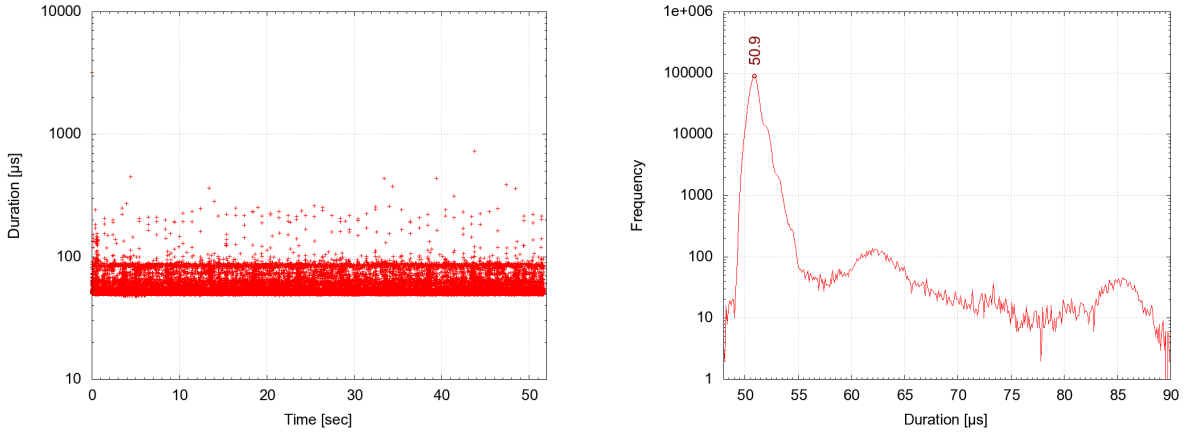
The values displayed in table 2 are made of a series of measurements per column each with a million individual measurements.

	Local Communication	Local Network	HPI Network
1st Decile	50.3 µs	359.7 µs	349.0 µs
Mean Value	51.2 µs	377.7 µs	378.4 µs
9th Decile	51.9 µs	390.9 µs	401.6 µs
Standard Deviation	4.4 µs	66.0 µs	70.5 µs

Table 2: Round-Trip-Delay

The 1st decile indicates the value that applies to that 90% of the measured values are greater or equal and at most 10% thereof are less.

The figure 4 shows the measured data and the histogram for the first step, the local communication on computer 1. One can see a clear maximum around 50.9 µs (89765 values) and some erratic values at higher times. Below 50.3 µs and above 51.9 µs are at most 10% of the measured values in each case. This meets the expected distribution of the measurement values.



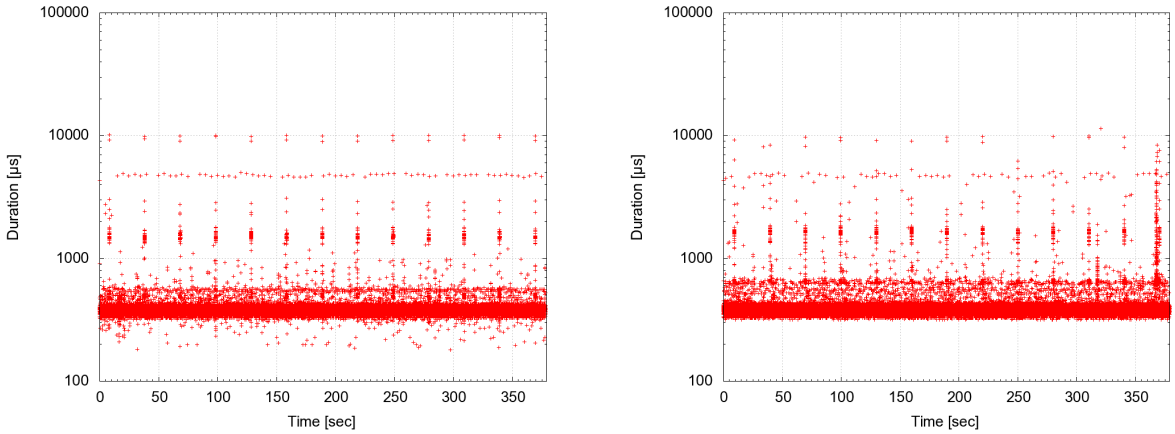
(a) Measured Data

(b) Histogram

Figure 4: Local communication

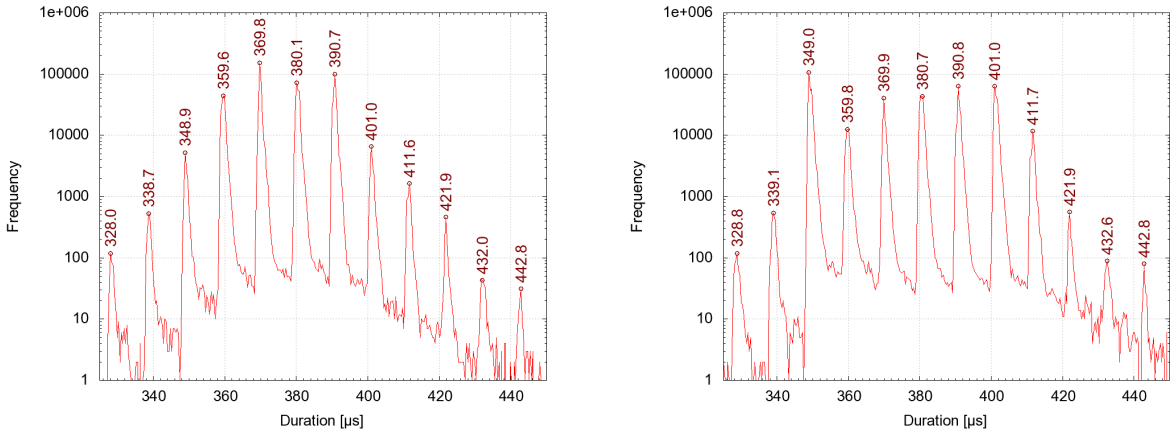
Figure 5 shows the timing values measured in the networks. At the first glance both diagrams look alike. The only difference seems to be that in the local network appear values below 300 µs and in HPI network not. Some small groups of values point to periodic higher durations.

But on closer examination and analysis the effects of the HPI network become visible (see figure 6). One point of interest are the detected peaks itself. This formation of groups about every 10 µs to 11 µs cannot be explained yet. But the reason for it can definitely be found in the area of network hardware and/or network control because this effect does not appear at the local communication.



(a) Local Network (b) HPI Network

Figure 5: Network Communication

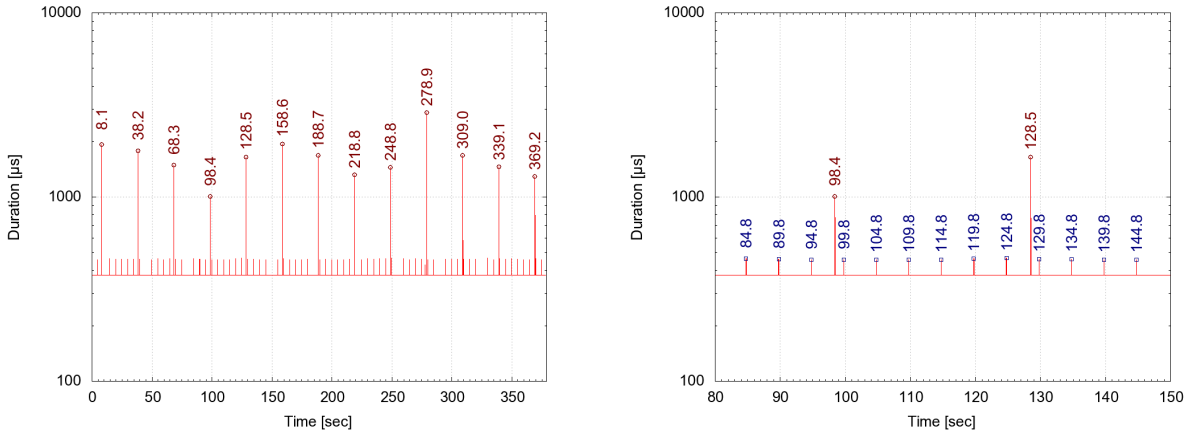


(a) Local Network (b) HPI Network

Figure 6: Histogram

The influence of the additional network components in the HPI network can be seen at the absolute high of the peaks. In the case of local network the highest peak is at 369.8 µs (153483 values). The envelope can be described as bell-shaped. In the case of HPI network the highest peak is at 349.0 µs (106865 values) and the envelope is rather rectangular. The peak at 369.9 µs (only 40837 values) is clearly smaller. Furthermore, in the case of HPI network the area between the 1st and the 9th decile is approximately 68% larger than in the case of local network. Both facts point to an additional delay of the data packets caused by the router and switches. Thereby the influence of this components is not large enough to change the mean value noticeable.

And also another effect, periodical higher timings, can be detected in both cases. The figure 7 shows this for the local network. As one can see, there are peaks every 30.1 sec – the higher one – and every 5.0 sec – the lower one. Certainly the operating system executes other tasks in these intervals which have basically lower priorities than the software used for the experiment.



(a) Entire Measurement (b) Zoom

Figure 7: Periodical peaks while gauging

Telemedicine – The Fontane Project

Telemedicine is a sector of medicine where the contact between the involved persons is held by a telecommunication infrastructure. The contact can be for example a phone call, a transmission of monitored vital signs or a video conference to conduct a real-time consultation. This is helpful if the involved persons are not able to meet each over or if it is not necessary.

The Fontane project [3] aims to improve the medical attendance of patients with cardiovascular diseases in the region Nordbrandenburg. The project is named for the apothecary and poet Theodor Fontane who was born in Neuruppin, Nordbrandenburg, and died in Berlin of 'unavoidable cardiac arrest'. The medical reasons for the region Nordbrandenburg are the low number of ambulant working cardiologists and the high cardiovascular death rate.

Medical Assessment at Home – Self-Monitoring

If a patient are vulnerable to a special kind of disease, it is helpful for him if he can monitor its vital signs at home and transfer the information to the attending doctor who checks the data. In this case the patient can live in familiar surroundings without to fear to ignore first signs of the disease.

The basic hardware structure of such telemedicine projects provides medical devices for self-monitoring at each patient and one telemedicine center. The devices on the patient side capture medical data and transfer these to the telemedicine center. Here the information are discussed by health professionals as well findings are created and stored inside the electronic health record. Additionally the family doctor or other authorized persons can visit the patient.

Today the medical devices on the patient side are designed as stand-alone devices. This means that they capture a special amount of characteristic information and transmit the data to the vendor of the device using specific software and data formats. The

attending doctor can maybe program the devices and will get the information from the vendors database. This makes it difficult to monitor more complex disease patterns or to use devices of different vendors. Another point is that the simple data capturing and transmission with the following manual analysis is only applicable for projects with low number of patients.

In the future telemedical products shall be modular and scalable. The devices on the patient side should be able to measure different vital signs and to analyze the data locally. Significant variations from the normal conditions of the patient result in the transmission of captured data and results of local analysis to the telemedicine center. This decreases the workload of the telemedicine center in standard cases. On the other hand the devices on the patient side will become more complex. One solution is to design different measurement devices which only capture the vital signs and one central device that can communicate with each of them, stores and analyzes the patient data and manages the transfers with the telemedicine center.

Requirements for Telemedicine Products

The ideal case would be if medical devices operates day and night without errors. But they are produced and controlled by humans and not every external influence is considered by the developer. That makes it necessary that each operating condition of such devices is well documented and tested. The key points are the safety of the devices and the prevention of the user if such a device fails or if it is diverted from its intended use.

General requirements for basic safety of medical electrical equipment are discussed in the standard DIN EN 60601-1 [1]. This document contains:

- Classification of medical electrical devices (ME devices) and medical electrical systems (ME systems)
- Terms and conditions for testing ME devices
- Identification, labeling and documentation of ME devices
- Protection against:
  - Electrical hazards
  - Mechanical hazards
  - Hazards through undesirable and exceeding radiation
  - Exceeding temperatures / fire
  - Hazardous output values
- Programmable electrical medical systems (PEMS)
- Design of ME devices
- Electromagnetic compatibility (EMC) of ME devices and ME systems

In terms of PEMS this standard refers to two additional standards:

1. DIN EN 62304 (Medical device software – Software life-cycle processes) and
2. DIN EN ISO 14971 (Medical devices – Application of risk management to medical devices).

Another point of view is the interoperability of different medical devices. The central device on the patient side, for instance, shall communicate with measurement devices of different vendors or the electronic health record at the telemedicine center should be readable and writable by different health professionals like the staff of the telemedicine center, the family doctor or medical specialists. Therefore, the devices has to fulfill special interface requirements. Relevant specifications and standards are:

- ISO/IEEE 11073 (Health informatics – Point-of-care medical device communication)
  - Part 10101 (Nomenclature)
  - Part 10201 (Domain information model)
  - Part 20101 (Application profile – Base standard)
  - Part 30200 (Transport profile – Cable connected)
  - Part 30300 (Transport profile – Infrared wireless)
  - Part 90101 (Analytical instruments – Point-of-care test)
  - Part 91064 (Standard communication protocol – Computer-assisted electrocardiography)
  - Part 92001 (Medical waveform format – Encoding rules)
- ISO 13606 (Health informatics – Electronic health record communication)
  - Part 1 (Reference model)
  - Part 2 (Archetype interchange)
  - Part 3 (Reference archetypes and term lists)
- ISO 12052 (Health informatics – Digital imaging and communication in medicine (DICOM) including workflow and data management)
- ISO/TR 22857 (Health informatics – Guidelines on data protection to facilitate trans-border flows of personal health information)
- ISO/HL7 21731 (Health informatics – HL7 version 3 – Reference information model – Release 1)
- ISO/HL7 27931 (Data Exchange Standards – Health Level Seven Version 2.5 – An application protocol for electronic data exchange in healthcare environments)
- Bluetooth Health Device Profile
- USB Personal Health Care Device Class

For the communication between the patient side devices and the telemedicine center normally a public network is used. The following standards refer to the intersection of medical devices and networks.

- ISO 16056 (Health informatics – Interoperability of telehealth systems and networks)
  - Part 1 (Introduction and definitions)
  - Part 2 (Real-time systems)
- ISO/TR 21730 (Health informatics – Use of mobile wireless communication and computing technology in healthcare facilities – Recommendations for electromagnetic compatibility (management of unintentional electromagnetic interference) with medical devices)

As well it is important to guarantee the security and availability of the transmitted information and the authorized access to all data stored in the system. On the patient side the patient itself, his family doctor and the emergency doctor, for example, should have access rights to the data. On the side of the telemedicine center all attending doctors should access to the electronic health record. The following standards are of interest to this points.

- ISO 17090 (Health informatics – Public key infrastructure)
  - Part 1 (Overview of digital certificate services)
  - Part 2 (Certificate profile)
  - Part 3 (Policy management of certification authority)
- ISO/TR 21089 (Health informatics – Trusted end-to-end information flows)
- ISO/TS 21091 (Health informatics – Directory services for security, communications and identification of professionals and patients)
- ISO/TS 22600 (Health informatics – Privilege management and access control)
  - Part 1 (Overview and policy management)
  - Part 2 (Formal models)

The technical committee 215 (TC 215 – Health informatics) of the International Organization for Standardization (ISO) works on further 45 draft standards.

The Fontane System Structure and Middleware

Within the Fontane project a control center is provided on the patient side which is able to preliminary analyze and store the vital signs measured by specialized devices like electrocardiograph or pulse oximeter. The measurement devices are wired or wireless connected to the control center. The tasks of communication and control are subject to real-time conditions. The control center sends the information, measured data and results of the preliminary analysis, to the telemedicine center in adjustable time intervals. The time can be defined by the attending doctor or in cases of emergency by the control center itself. After data transmission the cardiologist at the telemedicine center creates a finding and stores this in the electronic health record. All other medical professionals can read and write the electronic health record depending on their role in the health care process. The emergency doctor should be able to read the electronic health record as well the personal data stored by the control center on patient side. Additional the control center should have a positioning system which can transmit its data in case of emergency to the ambulance vehicle using the telemedicine center.

The system is characterized by loosely coupled devices. Therefore, directory services are one part of the middleware. On the side of the telemedicine center a list of registered patients including their devices is required. On the patient side the control center has to identify registered and activated measurement devices for transmitting data.

Other tasks of the middleware are the connection management and data transfers between all parties. The usage of wireless technologies makes trusted end-to-end information flows necessary. For dynamic bandwidth control and data prioritization

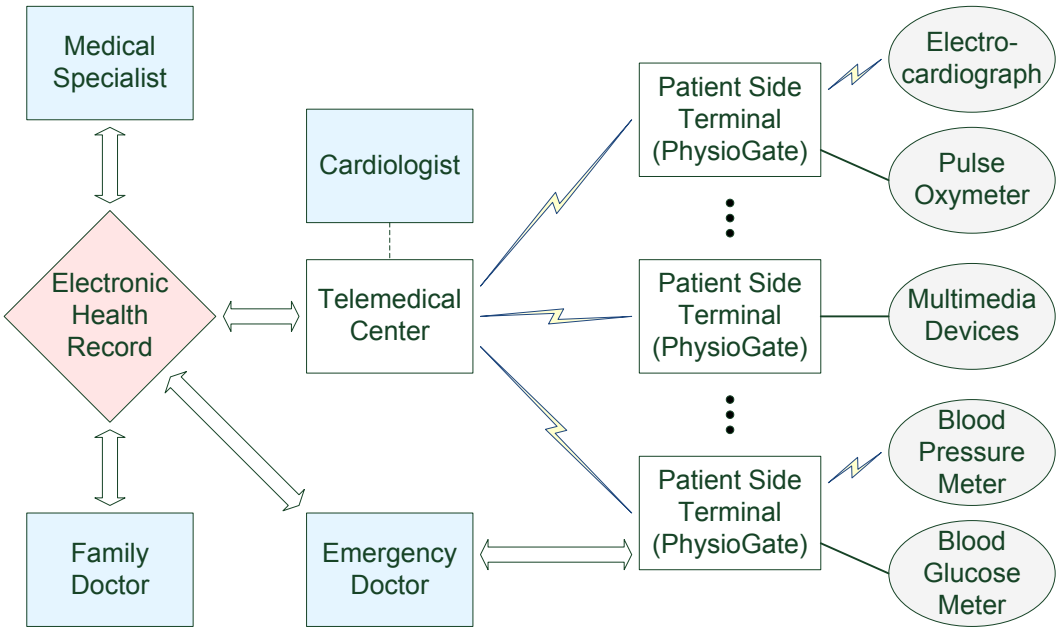


Figure 8: Fontane project – System structure

special attributes like priority, criticality, privacy and purge date have to be defined for each data set. The access to the electronic health record by the involved persons each with different tasks and/or rights needs role based policies.

Furthermore, the system itself is based on parameters which have to be dynamically adjusted, like the frequency of creating findings for a certain patient. This parameters can be set by using a system of rules defined and updated by the attended doctors. This rules are also used to define the different system behavior in normal and emergency cases.

Conclusion and Future Work

Service oriented computing offers a wide range of work and development. One part of this area can be applications and or services for real-time systems. Here comes in comparison with other working fields especially timing requirements into play. The round-trip-delay measurement has shown in terms of networks that the predominant number of data transmissions is concentrated around a mean value. But there are also some erratic values which can be much higher than the mean value. This experiment makes it clear that the use of services for real-time computing is a complex task if the services are connected via network.

Another important point is the preliminary work on the Fontane project. Fundamental questions have here to be answered in the next weeks together with other partners in this project. Our next part will be the participation on the development of the control center on patient side – the PhysioGate. Here the middleware shall be able to communicate with the connected sensors, store their data and transmit this data to the telemedicine center. All this actions run under real-time conditions.

References

[1] DIN EN 60601-1 (VDE 0750-1) Medizinische elektrische Geräte – Teil 1: Allgemeine Festlegungen für die Sicherheit einschließlich der wesentlichen Leistungsmerkmale (IEC 60601-1:2005); Deutsche Fassung EN 60601-1:2006, Juli 2007.

[2] Uwe Hentschel. Messung der Übertragungszeiten in IP-basierten Netzwerken unter Windows. September 2009.

[3] Friedrich Köhler. Fontane Gesundheitsregion Nordbrandenburg Antragsskizze für die Realisierungsphase im BMBF-Wettbewerb Gesundheitsregionen der Zukunft – Fortschritt durch Forschung und Innovation BMBF-GRDZ-46-105 “FONTANE-Projekt”, April 2009.

On Programming Models for Multi-Core Computers

Frank Feinbube
Frank.Feinbube@hpi.uni-potsdam.de

This document gives an overview of my current view on the topic of my promotion and the current state of the research I have done since May 2009 at the HPI Research School on Service-Oriented Systems Engineering within the operating systems and middleware group supervised by Prof. Dr. Andreas Polze.

Since services are hosted by application servers, shifts in the underlying systems have a great influence on their efficiency and functionality. Therefore it is necessary to get a deep understanding of trends in operating systems and middleware as well as hardware environments. This paper discusses some of these shifts. It will take a short look on energy efficiency and a deeper one on programming models for multi-core computers. Thereby first the application of graphic cards for parallel computing will be evaluated using the example of CUDA. Then a brief summary on the limits of threads for the Windows platform is shown.

1 Introduction

Services, especially web services, are hosted by application servers running in enterprise server environments. To create and maintain services that are reliable and deliver a good performance, one has to understand the underlying layers. Not only application servers have to be considered, but also trends in operating systems and hardware. Knowing the whole system provides the foundation to optimize every level for the execution of services. In addition this is the only way to generate useful service meta-data information such as the reliability, scalability and carbon footprint of a service.

One of the most popular trends is the architectural shift towards multiple processing units. It leads to a situation where software can only benefit from Moore's law, if it can be parallelized in a way that exploits that new architecture. In other words: if we can not adapt, new software will be slower on emerging processor architectures. The reason for this shift is the fact that increasing computational power and density of transistors not only results in increased power requirements, but also in additional heat production. For every 10 °C increase in temperature the failure rate of a system doubles (Arrenhius' equation) [8]. This is why cooling and power supply have become the most expensive parts in high-performance computer centers. Two ways lead out of this misery. On the one hand we have to consider energy savings to decrease the costs of maintaining computer systems. This topic is discussed in section 2. On the other hand we have to write software to be highly parallelizable to benefit from new architectures. This

way computation can be distributed and thus heat issues can be managed with greater efficiency. Unfortunately writing parallel programs is not as easy as using a new API or learning a new programming language. Actually we have to learn a new way of thinking about the software problems we face and the way we solve them. Section 3 is devoted to the subject of programming models for multi-core environments.

Another development, which rises straight out of its cradle, is cloud computing. The greatest change is the shift of responsibility for the availability and administrative work to the hands of an external provider. This allows users to concentrate on their core business. The cloud will be highly scalable and users will only have to pay for the compute power they use. Just as we now pay for electricity from the socket. Currently, many details are still in the dark. Thus, it is unclear what kinds of software systems will benefit from these platforms and how these will be written. But one thing is for sure: the underlying program must be highly scalable. In the context of this paper I will not consider cloud computing in detail.

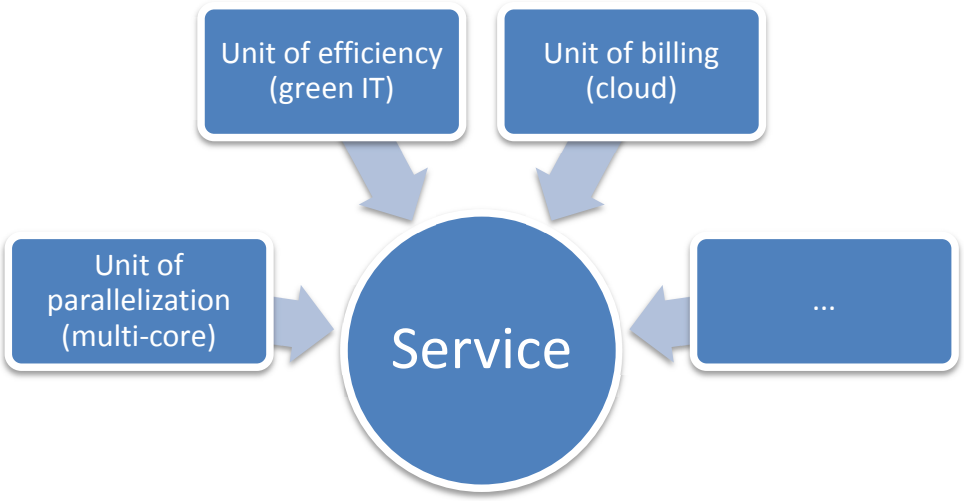


Figure 1: Services are the unit of scale in different domains.

As illustrated in Figure 1 all these shifts support the idea of seeing software as a service. But they also suggest that we should revise our service notion. This way we can check if it can handle the new developments in a good fashion and see spots where we can adapt it to derive the greatest benefits from the new situation.

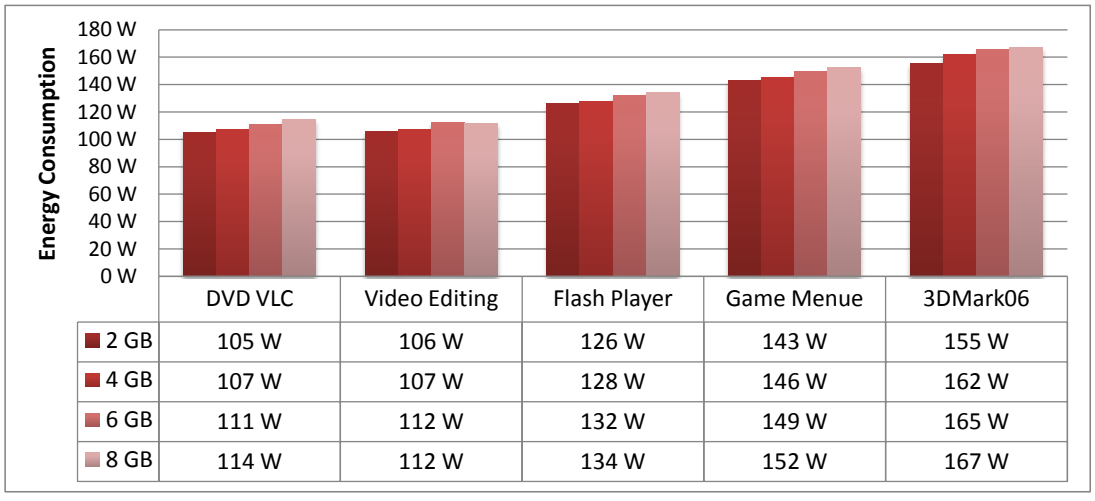
2 Energy Efficiency

Many Application servers are hosted in huge computer centers. In order to fulfill their high performance requirements these centers make use of a lot of computers. In the past the hardware of these computers was the main expense in building and maintaining a high performance computing center. But this is no longer true. Nowadays the costs for power supply and cooling of the hardware are about 50 to 70% [4] of the overall costs of the computer center.

In such a situation it is import to find new ways to reduce the energy consumption and the heat generation of the applied hardware components. This reduction should be accelerated by means of the operating system which should orchestrate the hardware components in an intelligent way. [8] and [22] show that such an approach can save a big amount of energy for a very small performance loss. In [18] a solution for CPUs is presented that not only reduces the energy footprint of the computer, but also increases the resulting performance.

Looking at the energy problem from the viewpoint of services we see a lack of convenient metrics. Having meta data information on the energy consumption and the CO_2 footprint of a service would make it possible to compose them in a more intelligent fashion. This way we could not only predict the energy needed to use a service, but also compose it in a way that minimizes the CO_2 footprint. Services are small, composeable and annotated with meta data by design. Thus they provide a solid basis for considerations on energy efficiency.

As described in [3] in the future power-aware features should be available to all major parts of the system. While there are big improvements in CPU and hard disk architectures, I found little research on the field of main memory. Thus I evaluated the energy consumption of main memory bricks for different usage scenarios. Therefore a test system was configured with different amounts of main memory. In all configurations the efficiency in handling the workload as well the energy consumption was measured.



	DVD VLC	Video Editing	Flash Player	Game Menue	3DMark06
2 GB	105 W	106 W	126 W	143 W	155 W
4 GB	107 W	107 W	128 W	146 W	162 W
6 GB	111 W	112 W	132 W	149 W	165 W
8 GB	114 W	112 W	134 W	152 W	167 W

Figure 2: Energy consumption of selected applications in relation to main memory brick count. A higher memory size leads to a bigger energy footprint, even if the additional memory is not used.

As depicted in Figure 2 the experiment showed that each memory brick adds about 2% to the memory footprint of the system. To make it worse this is the case even if the memory is not used at all. If this holds for most desktop computers it would be desirable to create hardware and operating system support to disable unnecessary memory bricks dynamically. This way energy savings could be brought one step further without any reduction in quality or service penalties. This technique would lead to approximately 6% power savings in the test system.

In order to get a broader view on the problem it is important to test some other desktop computer configurations as well as server computers. In addition a comparison of different memory brick sizes and vendors would be interesting.

3 Programming Models for Parallel Computing

Finding a good way to write programs for multi-core environments has again become a very popular topic. While many new languages arise, most of them reuse concepts that were already known, for example in *communicating sequential processes* [7] and its implementation *Occam*. With *Intel's Threading Building Blocks* [9] and *Microsoft's Parallel Extensions to .NET Framework* [13] APIs are provided that aim to ease the use of multi-threading in *C++* and the *.NET Framework*. Amongst others they bring constructs that allow programmers to declare that loops are to be executed in parallel. Another parallel programming model uses agents that communicate via channels. *Microsoft's Axum* is based on such a model.

Lots of influence arises from fields where parallel computing is long since applied. One example is the area of distributed computing, where parts of the program are running on different machines. Here we can find fast accessible memory on the local machine, but a fairly slow communication channel to the server or peers. Similar problems show up with the *Non-Uniform Memory Access* (NUMA) in modern processor architectures.

Another example for fields that already have highly parallelized programs is the area of graphic processing units (GPU). In order to execute their particular tasks, graphic cards benefit from calculating many equivalent functions in parallel. Programming for graphic cards has long since been a very specialized task. The applied programming models were only a reflection of the hardware of these devices. But in order to exploit the cards for new visual effects there has been a shift to general purpose graphic cards (GPGPUs) for some years. Though this shift has not finished yet, graphic cards are becoming more general and CPUs are becoming more parallel. So perhaps they will converge at some point in the future. In subsection 3.1 I present my experiences with the *NVIDIA CUDA* programming model as one representative for GPGPUs.

Besides programming models, another problem that we have to face when we think about parallel computing are the restrictions of hardware and the limits of the operating systems and middlewares we use. I took a first step in studying the limits of thread creations which is presented in subsection 3.2.

3.1 General purpose graphic processing using CUDA

For some years graphic cards were not only used to render pictures to screens, but also for mathematical processing. Therefore shader languages or vendor specific languages like *Brook+*, *Cal* or *Cg* were applied. Today with languages like *NVIDIA CUDA* [15] and the *AMD Stream Computing SDK* [1] it is even possible to write programs using only a few extension to the *C* programming language. The next step will be the application of the emerging *OpenCL* [2, 10] API. It will allow to program algorithms

that abstract from CPU and GPU as the underlying processing device. Current OpenCL implementations are not yet fully usable. *AMD* provides an implementation that is only capable to use the CPU while *NVIDIA* only supports the use of their CUDA-enabled graphic cards. This is why this section focuses on CUDA which is a well established "scalable parallel programming model and a software environment for parallel computing." [19]

It allows writing programs that run on general purpose graphic processors (GPGPU) by NVIDIA. Due to the hardware architecture of these devices, many complex computational problems can be solved much faster than on current CPUs. These problems include physical computations and video processing. Nowadays development for CUDA is done using some C extensions. The code is precompiled with a compiler provided by NVIDIA and finally compiled to binaries accessing CUDA-enabled graphic drivers. For CUDA works with all modern graphic cards from NVIDIA, even *NVIDIA ION* [14], it is available in hundreds of thousands of computers. This makes it particularly interesting for research on parallel computing.

3.1.1 Programming Model

The design goals of the CUDA programming model are to enable programmers to develop parallel algorithms that scale to hundreds of cores using thousands of threads. [19] Thereby the developers should not need to think about the mechanics of a parallel programming language and should be enabled to employ the CPU as well as the GPU.

The application parts that are executed on the graphic card are called kernels. They are executed by lots of lightweight CUDA threads which can communicate using slow device memory or fast shared memory. While the kernel is running the CPU is free to handle other workload. Each kernel has some equivalent tasks to fulfill. Listing 1 shows such a typical kernel execution scheme. At first each thread can calculate its unique thread identifier using some constructs provided by the CUDA environment. This identifier can be used to make control decisions and to compute the memory addresses of the input parameters. Finally the calculated result is written back to the global memory.

```
// derive absolute thread id from block id and relative thread id
int threadId = blockIdx.x * blockDim.x + threadIdx.x;
int parameter = dataGpu[threadId];}   // read from array (using id as index)
result = parameter * parameter;}      // calculate the result
dataGpu[threadId] = result;}          // write to array (using id as index)
```

Listing 1: CUDA kernel execution scheme

There is a strict separation of kernel routines and normal program routines. Kernel code cannot access host memory. Kernel methods must not be recursive and must not use static variables.

3.1.2 Evaluation with an example

In order to get a deep understanding of the programming model, I chose an active research problem that is complex enough to demonstrate the abilities and limits of

CUDA. The *n queens puzzle* fulfilled these requirements. Its goal is to find all ways to place a given number *n* of queens on a chessboard which has *n* times *n* fields. A configuration is only valid if no queen attacks another one. This holds if no queen is placed in a row, column or diagonal that is used by another queen.

All solutions for this puzzle are known up to a number of 26 queens. Preußer et al. from the University of Dresden [5] calculated all 22,317,699,616,364,044 valid configurations using specialized FPGA boards. Another project is *NQueens@Home* by Figueroa from the Universidad de Concepción de Chile [6] who uses an approach that is similar to *Folding@home* [17] where a middleware is provided that distributes work packages over the Internet. While calculations are done by the personal computers of registered users in both cases, NQueens@Home does not exploit their graphic cards. This is critical, because the statistics of the Folding@home project [16] show, that the application of graphic cards brings a huge performance benefit compared to common CPUs.

Both projects are based on an optimized single-threaded algorithm written by J. Somers [23]. After investigating the problem and programming an own solution, I decided to take the algorithm of J. Somers as the basis for my CUDA version as well. This decision founded on the fact that this algorithm does not use recursions, consumes little memory and is widely accepted.

The first and most important step was to parallelize the algorithm. Therefore I modified it in a way that allowed the precalculation of board settings for a given number of rows. Its results are then used as the input for an algorithm that calculates all solutions starting from the given setting. Applied to CUDA, the first algorithm runs on the host and the second one as a kernel on the graphic card. After the basic program was running, I step by step modified it to use fast shared memory instead of slow mapped device memory for the arrays. As shared memory is rare on graphic cards, using more of it reduced the number of threads I could deploy.

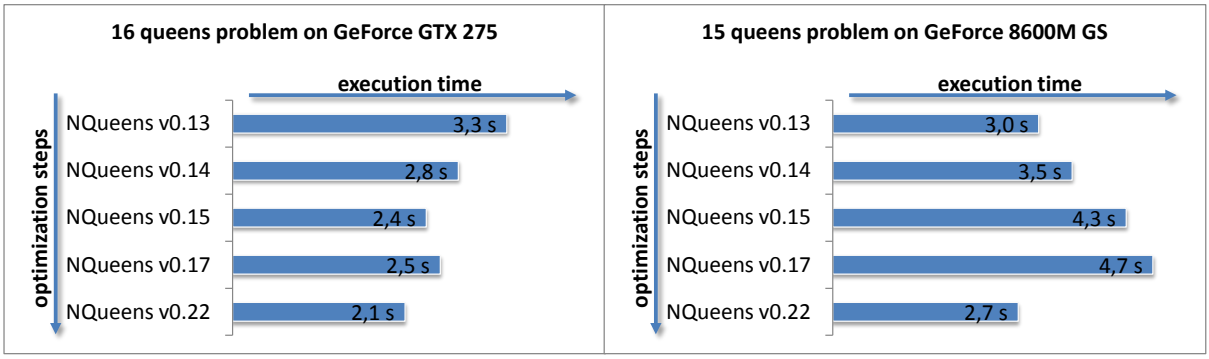


Figure 3: Runtime comparison of different versions of the NQueens program. The higher the version number the more shared memory is used instead of device memory. While the optimizations lead to better performance on the latest graphic cards (Geforce GTX 275), it results in worse performance on former CUDA-enabled versions (Geforce 8600M GS).

In Figure 3 the performance impact of these optimization steps is shown. Using a graphic card with the latest CUDA-enabled architecture from NVIDIA (*Geforce GTX 275*), the shift from device to shared memory lead to a performance increase. The benefit from faster memory accesses exceeded the penalty of the reduced thread count. On the other hand the same optimizations lead to a decreased performance on the former architecture (*Geforce 8600M GS*). Surprisingly after the last optimization step the performance is significantly improved and outperforms the original program version. In the last version shared memory is used for all arrays and the memory footprint per thread is minimized as well. The reason for the different results of the optimizations on the different architectures is a performance problem that arises from incoherent memory accesses on later CUDA-enabled architectures.

This evaluation shows that general purpose graphic cards are still far away from CPUs, because programmers can not even make assumptions on the effects that optimizations of a program will have. A program that is optimized for one architecture will not necessarily run fast on another CUDA-enabled architecture.

3.1.3 Conclusion

During my experience with the CUDA programming model I learned that there are a lot of restrictions a programmer must cope with. At first there is the unavailability of recursion and the differentiation of CUDA subroutines from normal routines. This is not only very unfamiliar to C programmers, but leads to replicated code as well. The second one is the focus on minimizing the memory usage of the algorithms. As a programmer one has to think "more carefully about memory access patterns and data structures sizes". [11] Another problem with memory that I faced was the unpredictability of the resulting register count. I was not able to derive a pattern for the number of occupied registers from code changes. Unfortunately one needs to know the number of occupied registers at coding time to optimize the code accordingly.

The last and most surprising restricting is that CUDA threads are only allowed to run for a very short period of time. Long running CUDA kernels result in a driver restart will thus be canceled. If a program needs to run several seconds or more, the display mode of the operating system must either be deactivated or additional graphics hardware must be used to take over the rendering work.

There is still a lot of work to do until general purpose programming for graphic cards will be as comfortable as programming for CPUs.

3.2 Pushing the Limits: Processes and Threads

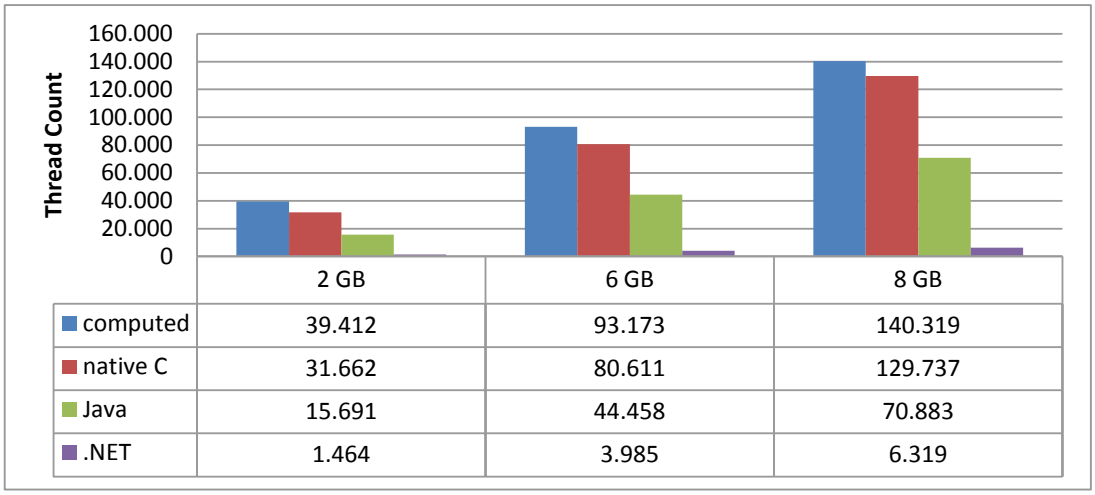
The usage of multiple processes and threads as workers is a widespread programming model. Inspired by an article of Mark Russinovich [20] and in cooperation with a vendor of complex enterprise applications, we tried to figure out what are the limits in the usage of processes and threads. The first approach which is described in this section was the creation of a model to predict the number of threads that can be created on a 64-bit Windows operating system. While lots of information is present for 32-bit [20], there is little information available for a 64-bit Windows operating system.

After studying the Windows internals to figure out how much memory a thread consumes [21], we came up with a fairly simple model:

$$max\_threads = \frac{(available\_memory + available\_pagefile\_memory) - program\_heap}{kernel\_thread\_size + thread\_commited\_memory}$$

$$kernel\_thread\_size = size(kernel\_stack) + size(TEB) + size(ETHREAD)$$

The maximal thread count equals the amount of memory that is available for thread stacks divided by the size of a thread. A thread's size is the sum of its user mode as well as its kernel mode representation. Due to some additional kernel constructs the size of kernel representation of a thread is negligible bigger than in our model. In order to check the accuracy of this model, we compared it against some benchmark results.



	2 GB	6 GB	8 GB
computed	39.412	93.173	140.319
native C	31.662	80.611	129.737
Java	15.691	44.458	70.883
.NET	1.464	3.985	6.319

Figure 4: Maximal thread count that can be achieved for different memory sizes on a 64-bit Windows 7 operating system using C, .NET and Java compared to a computed value based on our model.

Figure 4 shows the results of our evaluation. We found that our model fits very well for native C. The divergence is due to *Address Space Layout Randomization* (ASLR) which is included in Windows operation systems since *Windows Vista*. Using the Java environment half of this thread count can be created. That penalty is due to the overhead for the thread management of the Java Virtual Machine. With the .NET framework the possible thread count is even smaller. This is due to the fact that .NET relies on committed memory which potentially wastes resources for they are unused. Further inspection of the *Shared Source Common Language Infrastructure* (SSCLI) [12] would be necessary to check what improvements are possible here.

While studying the Windows internals and investigating the limiting factors for thread creations within the operating system, a lot of interesting questions arose that we will try to answer in the future.

4 Conclusion

This paper gives an overview of current architectural shifts and their impact on service computing. In section 2 energy efficiency considerations are presented. The need for operating system and middleware support to reduce the power consumption of a system by deactivating unnecessary system parts is illustrated by the example of memory bricks. Furthermore in section 3 the emerging importance of programming models for multi-core computers are discussed. Taking CUDA as one representative, general purpose GPU programming models were studied. The restrictions of these models were shown by evaluating a solution to the n queens problem. Some of the limits and problems of parallel computing were examined.

As shown in this paper we face a lot of interesting architectural shifts today. In order to let service computing benefit the most from these shifts, we need to get a better understanding of them and to revise our service notion appropriately.

References

[1] Advanced Micro Devices. ATI Stream Software Development Kit (SDK). http://developer.amd.com/gpu/ATIStreamSDK/Pages/default.aspx, 2009.

[2] Apple. Apple - Mac OS X - New technologies in Snow Leopard. http://www.apple.com/macosx/technology/, 2009.

[3] Wu chun Feng, Xizhou Feng, and Rong Ce. Green Supercomputing Comes of Age. *IT Professional*, 10(1):17–23, January 2008.

[4] Dell. DELL Leitfaden für Stromversorgung & Kühlung. http://www.dell.com/downloads/global/solutions/PowerAndCooling_Brochure_EMEA-de.pdf, 2008.

[5] Thomas B. Preußer, Bernd Nägel, and Rainer G. Spallek. Queens@tud. http://queens.inf.tu-dresden.de/?l=en&n=0, 2009.

[6] Israel Figueroa. Nqueens@home. http://nqueens.ing.udec.cl/.

[7] C. A. R. Hoare. *Communicating Sequential Processes (CSP)*. Prentice Hall, 2004.

[8] Chung-hsing Hsu and Wu-chun Feng. A power-aware run-time system for high-performance computing. In *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, page 1, Washington, DC, USA, 2005. IEEE Computer Society.

[9] Intel. Intel® Threading Building Blocks. http://www.threadingbuildingblocks.org/, 2009.

[10] Khronos. Opencl - the open standard for parallel programming of heterogeneous systems. http://www.khronos.org/opencl/, 2009.

[11] William Mark. Future graphics architectures. *Queue*, 6(2):54–64, 2008.

[12] Microsoft. Shared source common language infrastructure. http://www.microsoft.com/downloads/details.aspx?FamilyId=8C09FD61-3F26-4555-AE17-3121B4F51D4D&displaylang=en, March 2006.

[13] Microsoft. Parallel computing developer center. http://msdn.microsoft.com/en-us/concurrency/default.aspx, 2009.

[14] NVIDIA. Nvidia cuda-enabled products. http://www.nvidia.com/object/cuda_learn_products.html, 2009.

[15] NVIDIA. *NVIDIA CUDA™- Programming Guide - Version 2.3.1*. NVIDIA Corporation, 2.3.1 edition, August 2009.

[16] Vijay Pande. Folding@home - client statistics by os. http://fah-web.stanford.edu/cgi-bin/main.py?qtype=osstats.

[17] Vijay Pande. Folding@home - distributed computing. http://folding.stanford.edu/.

[18] J. Richling, J. H. Schönherr, G. Mühl, and M. Werner. Towards energy-aware multi-core scheduling. *PIK - Praxis der Informationsverarbeitung und Kommunikation*, 32:88–95, 2009.

[19] Greg Ruetsch and Brent Oster. Getting started with cuda, 2008.

[20] Mark Russinovich. Pushing the limits of windows: Processes and threads. http://blogs.technet.com/markrussinovich/archive/2009/07/07/3261309.aspx, July 2009.

[21] Michael Schöbel. NtCreateThread: memory allocations in kernel mode. http://www.dcl.hpi.uni-potsdam.de/research/WRK/?p=86, 10 2009.

[22] David C. Snowdon, Etienne Le Sueur, Stefan M. Petters, and Gernot Heiser. Koala: a platform for os-level power management. In *EuroSys '09: Proceedings of the 4th ACM European conference on Computer systems*, pages 289–302, New York, NY, USA, 2009. ACM.

[23] Jeff Somers. The n queens problem - a study in optimization. http://jsomers.com/nqueen_demo/nqueens.html.

Towards a Service Landscape for a Real-Time Project Manager Dashboard

Thomas Kowark
thomas.kowark@hpi.uni-potsdam.de

Continuously improving possibilities for electronic communication have facilitated a shift from co-located towards geographically distributed development teams. While those virtual development teams combine the cumulative knowledge of their members without the need for costly travels or relocations, they also have to cope with the challenges that the mainly indirect communication implies. Problems within the team structure or the communication behavior are much harder to recognize and can jeopardize project success. Hence, project managers should be able to have an overview about the communication patterns of their team during all, and especially early, stages of the development process.

This report presents the extensions that have been applied to the *d.store* platform in order to utilize it for capturing and analyzing the digital communication artifacts created by the members of software development teams. Furthermore, a case study is outlined that re-enacts a large scale development process during the course of a university lecture. Thus, it provides extensive data about the digital communication footprints of software developers that can be compared to analog observations to reason about indicators for communication problems present within the digital traces.

1 Introduction

The digital footprint of developers is steadily increasing due to various reasons. First of all, in geographically distributed teams (e.g., virtual development teams), electronic communication is often the fastest and most cost-effective way of communicating with other team members. But even co-located teams are producing more and more digital artifacts of their work because of modern collaborative application lifecycle management (ALM) tools such as Trac, Codebeamer, or RedMine [9]. These tools aggregate the formerly widespread landscape of CASE-tools into single applications. Thus, only one system needs to be set-up in order to provide functionality like bug tracking, wikis, or time management. However, some communication aspects, like instant-messaging or email, are in most cases still covered by external tools, even though some ALM solutions support those functionalities.

***d.Store* Platform** In order to provide a unified representation of this heterogenous data, the *d.store* platform has been developed [11]. It uses the Resource Description

Framework (RDF) [2] and the Web Ontology Language Extension (OWL) [3] to define semantic descriptions of the underlying models along with all relations between the model elements (e.g., the different relations a person can have to an email). The semantic description of domain ontologies is quite beneficial, since it (1) allows to easily extend the platform with support for new types of resources and (2) simplifies the generation of meaningful queries on the data since the data models are extensively described.

So far, the platform has been used for a post-hoc analysis of the digital communication of virtual design teams. Those studies have revealed patterns within the team communication behavior that can be used to assess expected team performance in the early stages of the examined projects and capture a variety of possible communication problems [10]. Accordingly, project managers might be able to intervene as soon as such patterns become apparent and thereby help to improve the overall project outcome. Additionally, side-projects have been started to further extend the platform with means to visualize the gathered data or extend some of the artifacts with information about their importance for the overall team communication.

The distinctive character of the evaluated projects has raised the question, whether the observations and findings also hold true in other set-ups and team structures. We have therefore extended the existing platform to capture digital artifacts that are very common in software development processes and additionally set-up a mid-sized software development project that serves as a case study. Those two measures help us to investigate the digital communication behavior of software developers and compare it to the patterns found for engineering design teams.

The remainder of this research report is structured as follows: Section 2 presents related work in the field of project team communication analysis. Section 3 gives an overview about our recent research activities and the extensions that have been developed for the *d.store* platform. Section 4 presents the outline of an upcoming case study, as well as other projects regarding the further improvement of the *d.store* platform and concludes the report.

2 Related Work

The analysis of development team communication in the field of software engineering has been the topic of preceding research and development efforts.

Various collaborative application lifecycle management tools like the aforementioned Trac, RedMine, or CodeBeamer try to provide a single point of access for developers and, additionally, allow basic queries on the data present within the systems. The amount of available datasources differs between those products, as does the flexibility of the definition of queries on this data.

An approach that aggregates data from different data sources has been developed by Ohira et. al [6]. The Empirical Project Monitor relies on numerous feeder applications that parse datasources like source code management systems, bug trackers, or email archives. It provides a number of preset visualizations for this

data. Additionally, an underlying communication model tries to detect flaws within the collaboration behavior based on empirical studies.

Reiner [8] presented a proposal for a knowledge modeling framework that supports the collaboration of design teams. Communication information has been deduced from explicit interactions between members of a design team by a software tool that he developed to provide a prototypical implementation for the proposed framework.

While all those approaches are able to create the same views on the available communication artifacts as our approach, the difference can be found within the representation of the internal models. The semantically annotated models of the *d.store* can be easily extended with new ontologies by adding the respective resource description to the application. Additionally, external services that parse the information available in various sources can feed the *d.store* networks with new data. This also allows to simply re-use existing resource definitions for the information from different implementations of the same concepts (e.g., different wiki types). The aforementioned implementations would require extensive adoptions of internal models to achieve the same behavior and defining arbitrary queries on this data would require knowledge about those internal data structures.

3 Recent Work

During the last six months, focus has been put on extensions to the *d.store* platform that, in addition to the existing support for emails, wiki pages, and WebDAV folders, allow for an investigation of team dynamics especially within software development teams. As shown by the highlighted boxes in Figure 1, the platform now is able to handle information retrieved from Subversion[1] repositories (SVN), as well as wiki pages and bug tracking information available within CodeBeamer - a solution for collaborative application lifecycle management.

Both additions to the platform included the implementation of a so-called feeder service that connects to the respective data source (i.e., the SVN repository and the CodeBeamer Web Service interfaces) and generates a Javascript Simple Object Notation (JSON) representation of the data. These JSON objects have to follow the resource definition specified within Resource Definition Framework (RDF) documents that have been created for both conceptual models. The resulting ontologies allow to incorporate the information into the team communication networks [11] created by the *d.store*.

Using wiki, SVN, and email information from previous projects of the development exercise of a software engineering lecture, it was tried to identify patterns within those graphs that indicate strong or weak individual or team performances. Since only little information regarding the roles of the team members and their distinct characteristics has been collected during this lecture, only weak indicators but no statistically significant evidence could be derived that certain communication and

[1] `http://subversion.org/`

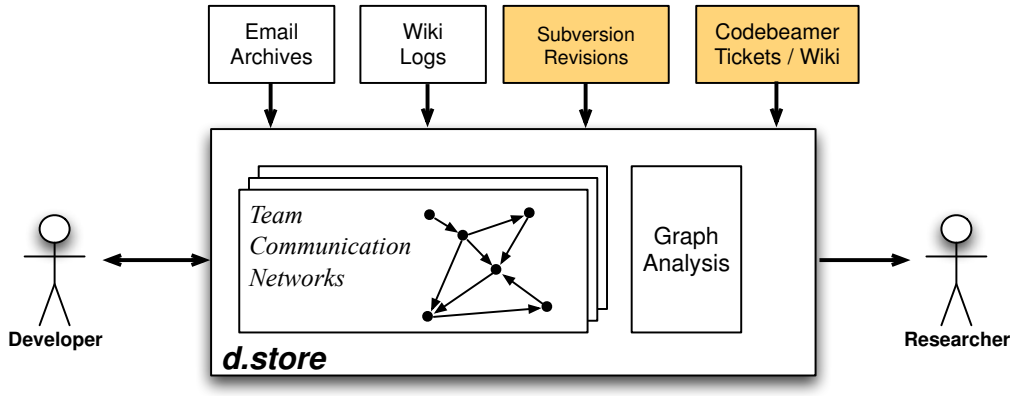


Figure 1: Extensions to the *d.store* service-landscape.

source code management behavior is beneficial for the project outcome. An example for such an indicator is the observation that team members with extreme check-in behavior seem to be more easily remembered by the project tutors. Generally, students that were responsible for many commits were considered to be "heavy performers". Those with very few were sought to have "struggled" keeping up with their team members.

While those factors might indicate that a certain person might be a good, or at least diligent, programmer, other factors are also important for project success. Recent studies indicate that the social network between the developers can be used to predict project failures [5]. Hence, good programming skills of single persons are no guarantee for an overall good team performance. To take this into account, a case study with a closer resemblance of real life development processes and a more structured observation process became necessary to provide the foundation for meaningful data collection, analysis, and accordingly, hypotheses creation [4]. The outline of the examined project and the methods used for analyzing the communication behavior of the project members is presented in the following section.

4 A Software Engineering Case Study

The implementation of the *d.store* extensions for SVN and CodeBeamer information is the foundation for a case study that is performed during the upcoming winter term. The purpose of this study is to determine whether indicators exist within the collection of digital artifacts created by software developers, that are evidential for individual team roles, performance of individual team members, or the performance of the entire development team.

4.1 Project Outline

The focus of the study is the observation of the development teams of a software engineering lecture during the project part of the course. The project goal is to develop a basic enterprise resource planning (ERP) system for small to mid-sized companies in a joint effort by all approximately 80 participants. The students are guided by both research assistants and senior students as tutors, but the main responsibility for organizing the group work remains within their own field of duty.

The entire project is further subdivided into 13 smaller teams, each of which is responsible for a certain set of requirements. The initial requirements given to each team are not prioritized. Hence, the teams have to perform user research to define the requirements that the target product has to fulfill. Furthermore, certain requirements not only have an impact on single teams, but also effect multiple ones. Thus, the teams not only need to coordinate the work within their owns teams, but also have to ensure the interoperability of components developed by different teams.

The SCRUM process is chosen to be the basic framework guiding the collaboration. A team of six “Product Owners” will be responsible for defining and prioritizing the requirements, as well as presenting them to the individual development teams. Furthermore, the product owners are responsible for evaluating the progress of the teams at the end of each development cycle - a so-called “Sprint”. Sprints last 3 weeks and are synchronized between all sub-teams. Within the sprints, the teams are required to have weekly meetings with the tutors to present recent developments, problems, and next steps. Delegates of the teams, so-called “SCRUM-Masters”, conduct an additional weekly meeting to further coordinate the work of the sub-teams by, for example, defining interfaces, discussing architectural decisions, or communication guidelines.

The teams are provided with a Subversion repository for source code management, global and team-internal mailing lists, and a CodeBeamer installation for feature tracking and wiki functionality.

4.2 Observation Process

In order to avoid distortion of the gathered data, the tutors are not granted access to the unified data representation of the team communication networks and thus do not have the possibility to perform queries on that data. They, however, have access to all artifacts created by the students and would theoretically be able to deduce all information that is stored within the *d.store*. However, as the amount of artifacts grows over time, this process is supposed to become more and more complicated and time-consuming. On the other hand, tutors get to know the members of the teams they supervise and, accordingly, might search through the available data more target-oriented.

As already mentioned, the tutors also have to log their impressions of the weekly team meetings. These logs have to include observations about the student behavior, e.g.:

- Which are the team leaders?
- Which team member is responsible for which aspect of the project?
- How did the team perform during the last development cycle?
- Are there any obvious problems within the team itself?
- How is the communication between interdependent teams handled?

All information that is gathered during those meetings can later be used for a detailed analysis of the information stored within the team communication networks of the *d.store* to answer the following research questions:

- Do the team communication networks contain indicators for the observations made by the tutors?
- Is it possible to determine patterns that indicate problems within the teams before they are noticed by the tutors?

In addition to these observations, the students are requested to perform the Belbin Team Role Inventory test [1] at the beginning of the project. This test assesses how the individual team members presumably behave within the team environment and what their alleged main characteristics are. Furthermore, the students are asked to complete a survey at the end of the lecture where they evaluate the teamwork and the support by the tutors. With this data, it is possible to analyze the *d.store* networks with a focus on the following questions:

- Is it possible to deduce team communication network patterns that indicate certain roles of team members?
- Which combinations of team roles might be beneficial for the project outcome and how does this affect the team communication network?

4.3 Future Work

To further broaden the database for the evaluation of the platform as a project communication analysis tool and not only focus on software development teams, the investigation of the communication artifacts created during the engineering design lecture that Uflacker et al. used for their initial research work will be continued, as well. This lecture is a cooperation with the Stanford University and provides insights into the collaboration of heterogenous, globally distributed engineering design teams.

While the main objective during the upcoming months is the aforementioned data analysis, further evolution of the *d.store* platform itself will also take place. The first topic in this area is the performance of the application. The expected graph representing the development teams of the case study project is likely to reveal possible

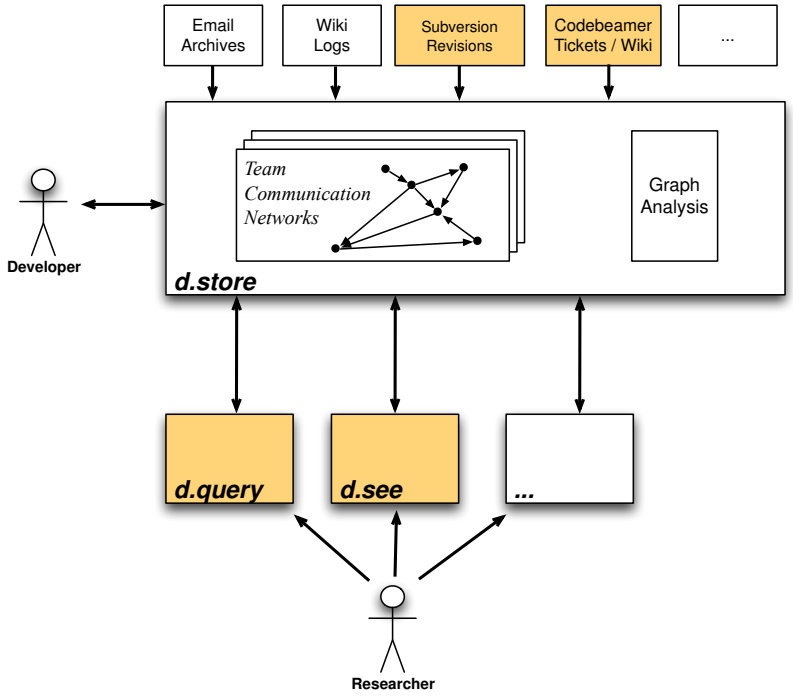


Figure 2: The *d.store* service-landscape.

performance bottlenecks in the storage of the networks in a relational database. We will therefore evaluate different approaches for storing the team communication networks in persistent storage spaces.

Furthermore, evaluating new services that utilize the data captured within the *d.store* and, thus, help evolving the landscape for a manager's dashboard (see Figure 2) will be a major topic for further research. This, for example, implies that new ways to visualize the gathered data have to be found since the new ontologies have different characteristics than the existing ones. While an approach to this problem has been developed with the *d.see* application, its applicability to the new models has yet to be determined.

Another aspects that has to be considered is the generation of queries. While the existing SPARQL [7] interface provides a powerful mechanism to query for arbitrary values, the long term target is the usage of the platform by project managers. To achieve this goal, means have to be identified to use the semantic annotations of the models to create *d.query*, a simple interface for the query creation that does not rely on the authors to learn a distinctive query language but use natural language.

References

[1] Meredith Belbin. *Management Teams*. John Wiley & Sons, 1981.

[2] D. Brickley and R. Guha. RDF vocabulary description language 1.0: RDF schema. W3C Recommendation, February 2004.

[3] M. Dean and G. Schreiber. OWL web ontology language reference. W3C Recommendation, February 2004.

[4] Kathleen M. Eisenhardt. Building theories from case study research. *The Academy of Management Review*, 14(4):532–550, 1989.

[5] Andrew Meneely, Laurie Williams, Will Snipes, and Jason Osborne. Predicting failures with developer networks and social network analysis. In *SIGSOFT '08/FSE-16: Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pages 13–23, New York, NY, USA, 2008. ACM.

[6] Masao Ohira, Reishi Yokomori, Makoto Sakai, Ken-ichi Matsumoto, Katsuro Inoue, Michael Barker, and Koji Torii. Empirical project monitor: A system for managing software development projects in real time. In *International Symposium on Empirical Software Engineering*, Redondo Beach, USA, 2004.

[7] Eric Prud'hommeaux and Andy Seaborne. SPARQL Query Language for RDF. W3C Recommendation, January 2008.

[8] Kurt A. Reiner. *A framework for knowledge capture and a study of development metrics in collaborative engineering design*. PhD thesis, Stanford, CA, USA, 2006. Adviser-Leifer, Larry J.

[9] M. Rita Thissen, Jean M. Page, Madhavi C. Bharathi, and Toyia L. Austin. Communication tools for distributed software development teams. In *SIGMIS-CPR '07: Proceedings of the 2007 ACM SIGMIS CPR conference on Computer personnel research*, pages 28–35, New York, NY, USA, 2007. ACM.

[10] Matthias Uflacker, Philipp Skogstad, Alexander Zeier, and Larry Leifer. Analysis of virtual design collaboration with team communication networks. In *Proceedings of the 17th International Conference on Engineering Design (ICED'09), Vol. 8*, volume 8, pages 275–286, 2009.

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

Towards Visualization of Complex, Service-Based Software Systems

Jonas Trümper
jonas.truemper@hpi.uni-potsdam.de

Traditional development tools such as IDEs and debuggers only provide partial support for developers to cope with a large system's complexity. Especially parallel execution poses a huge challenge for developers as it raises the system's runtime complexity by orders of magnitude. For example, synchronization has to be handled and each execution thread has its own local stack and state.

This work at the HPI Research School aims at developing concepts and tools for software visualization that help to cope with the complexity of such large software systems in various ways. Current research includes, but is not limited to, software visualization for debugging performance issues caused by flawed synchronization of shared memory access.

1 Motivation: Improve Productivity of the Software Development Process

Large software systems, in particular service-oriented software systems, typically consist of millions lines of code, are maintained over a long period of time and are developed by a large, diverse team. This poses an enormous challenge to developers in several dimensions. For example, (1) the knowledge about the whole system is typically distributed. That is, a single developer is no more able to memorize the complete system structure with all its details. More precisely, each developer working on the system typically has detailed knowledge about one or a few parts of the system and is roughly aware of the big picture. (2) The dependencies between system components may not be explicitly documented or visible. Dynamic binding due to polymorphism in object-oriented software systems complicates this even further as the exact dependencies are only visible at runtime. (3) Documentations and actual system implementations often exhibit significant differences in practice. Hence, the only reliable information sources are represented by the actual implementation artifacts, e.g., source codes and binaries.

As a consequence, creating a mental map of and understanding the internal dependencies of such a large software system by "source code reading" is a complex and time-consuming task for an individual. Locating bugs or identifying performance bottlenecks, however, requires even more: the specific interaction between the system's actors has to be understood.

Concurrent execution introduces new classes of complexity. Among others, the source code becomes more complex as synchronization must be handled, too. In

addition to that, the complexity of the system's runtime behavior typically rises with each additional thread[1] running in parallel. And third, a new class of bugs and performance problems is introduced due to the possibility of parallel access to shared memory. When it comes to multithreaded or even distributed applications the complexity of the conventional debugging workflow (i.e., debugger, breakpoints and step-by-step execution) typically rises by orders of magnitude.

As software visualization is "the art and science of generating visual representations of various aspects of software and its development process" [1], it aims to "help to comprehend software systems and to improve the productivity of the software development process" [1]. A lot of research exists in the area of software visualization for singlethreaded applications; however the visualization tools are typically incapable of exploration or analysis of parallel executions. However, extending those singlethread visualization tools is neither always possible nor straightforward – just as singlethreaded applications can not simply be switched to be multithreaded.

Software analysis tools, especially for parallel executions, are supposed to help finding answers to questions like:

- "Why is the system's execution in that part of the implementation so slow?"
- "What is going on in this parallel execution?"
- 'Which code segments actually run in parallel?"
- "Where is forced sequential execution due to dependencies?"

Which visualization techniques actually do provide real benefit in answering those type of questions is still unclear. In addition to that, each question/developer task may require completely different visualization strategies. So there is probably no one-size-fits-all visualization that is applicable to all kinds of issues introduced by parallel execution, and visualization needs to be tailored for each concrete task.

The remainder of this paper will focus on multithreaded software systems and performance bugs caused by flawed synchronization between threads. The concepts are intended to be applicable to service-oriented systems as well.

The paper is structured as follows: Chapter 2 outlines the peculiarities of debugging concurrent executions. Chapter 3 discusses existing work, Chapter 4 briefly presents a concept for software visualization of multithreaded applications using the example of performance bugs. Chapter 5 outlines planned next steps for the research.

2 Locating Performance Bugs in Concurrent Execution

Performance bugs in concurrent executions with multiple threads that access shared memory are typically caused by either an actually slow implementation or flawed synchronization with respect to the shared memory access. Whereas the former cause

[1] Within this paper, the term *thread* is used to identify a separate execution with local storage and stack. That is, the term *thread* is used for a separate execution within a process on a local machine or a service on a separate machine.

can be identified comparatively easy by means of, e.g., Call Stack Sampling (see Section 3.1), the latter cause is hard to identify: most of the calls to the flawed code run as fast as expected, but very few random calls are very slow.

Manually identifying the cause for very few outliers in a set of regular executions by reading the source code is hardly possible and typically requires guesswork. Likewise, examining such unintended behavior with a conventional debugger in concurrent executions is a hard task, because those concurrent paths have to be tracked mentally and the separate execution states have to be considered. Furthermore, it's typically not possible to predict which of the executions of a specific piece of code will constitute itself as an outlier and so all executions of the respective code would have to be stepped through manually. The lack of proper tool support again and again causes developers to manually add debug output into the source code in order to manually track a system's runtime behavior and execution costs. Tedious subsequent manual analysis of execution costs has to be done. Unfortunately, the very important and valuable original execution context is lost in this kind of post-mortem analysis.

3 Related Work

3.1 Data Acquisition and Performance Analysis

With regard to data acquisition concerning runtime behavior of software systems, lightweight techniques were introduced that aim to provide support for performance-related tasks. *Call Stack Profiling* (e.g., [2]) is a technique that records an instrumented system's call stack at specific time intervals. This, in turn, permits to derive averaged execution costs per function/method. *Call Graph Profilers* like *gprof* [2] introduced by Graham et al. additionally provide callee/call site context and summed up execution costs per callee/call site combination.

With regard to the above mentioned class of performance bugs (a few outliers in a huge set of regulars), Call Stack Sampling and Call Graph Profiling provide only average execution costs, so developers may notice a generally slowed execution of this particular piece of code; they won't be able to tell the slow outliers from the rest, though. Consequently, other data acquisition techniques like that of instrumenting profilers (e.g., PIN [7]), which are able to record Function Boundary Traces (FBT), are better suited for the identification of outliers. As a consequence of recording the full call history, *time warping* is possible during post-mortem analysis: the analyzed time slot within the recorded trace can be selected freely and so time can be virtually turned back. Developers are able to reconstruct the cause chain and context that lead to an event or function call *without* having to trigger another system run.

3.2 Visualization Tools

As a matter of fact, the size of FBTs typically exceeds the size of sampled and/or aggregated traces by orders of magnitude (e.g., several gigabytes of data) and so FBTs are considered to be not human-readable - although they may be stored in plain

text. Thus, FBTs need to be visualized in a way that allows developers to explore their content efficiently.

ThreadScope, proposed by Wheeler and Thain [8], is a tool that is able to visualize the execution of large, multithreaded applications in connected graphs. The graphs enable developers to visually identify bottlenecks in executions and possible deadlocks. They also propose techniques to reduce graph size for specific developer tasks, e.g., filtering read-only access to memory. This simplifies identification of erroneous synchronization that causes values to be overwritten unintentionally. However, their tool does not feature essential navigation and visualization techniques to simplify the analysis. The graphs become very huge for typical logging durations and as such exploring them in all is a tedious task again.

Zhao and Stasko present a tool to visualize multithreaded applications [9]. The tool supports visualization of lock usage, execution history and a high-level view on thread activity. However, it lacks a vital feature: users can analyze the activity of single threads, but there is no synchronized multi-thread view which would permit to actually examine parallel activities and their context.

ParaVision, a tool by Nutt et al. [5], provides multiple views on the runtime behavior of parallelized systems. Unfortunately, the views do not scale for systems with a high number of threads running in parallel and large executions traces.

4 Visualizing Function Boundary Traces of Multithreaded Systems to Locate Performance Bugs

Mentally tracking multiple concurrent executions within a debugger is a complex, expensive and sometimes even close-to-impossible task. In this work, the approach is to provide substantial support by means of visualization so that developers can track and analyze those parallel executions with ease.

The concept bases on a single assumption: developers that want to locate performance bugs typically have read the source code before and know where synchronization objects (namely locks) are accessed. With this premise, the general concept is as follows: each separate (chosen) thread of execution is visualized in a separate graph, which permits to analyze each threads activity separately and in parallel. The execution sequences are visualized using a stack depth based approach so that the resulting graph is as small as possible. (see Figure 1). Time is mapped along the x-axis of the visualization and stack depth along the y-axis. Calling relations between functions are implicit, that is, if function *foo* calls *bar*, then *bar* is drawn below *foo*. Since the whole graph exceeds a typical screen size, panning and zooming is implemented so that developers can adjust the shown time frame according to their needs.

Overview maps enable quick navigation in the graph and provide additional orientation within the visualized execution trace.

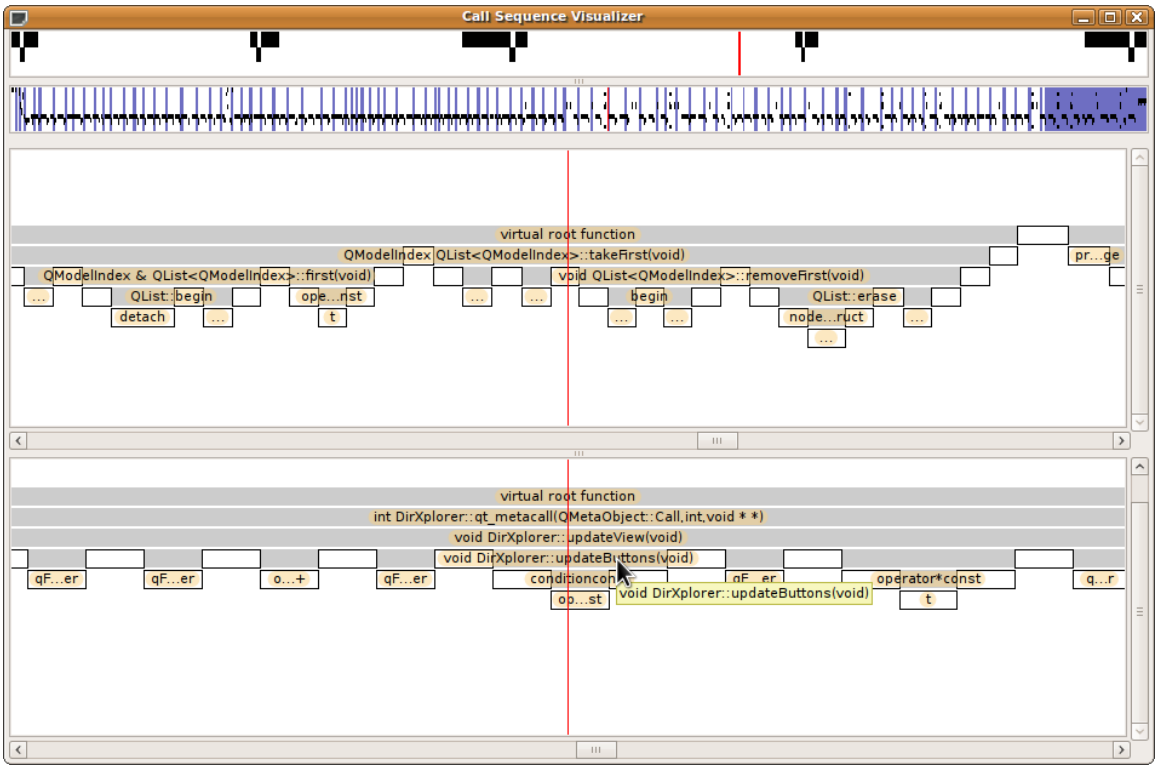


Figure 1: The prototype tool showing a sequence cutout of two threads. On top, overviews depict each threads activity and allow to directly navigate to sections of interest.

5 Next Steps

5.1 Synchronized Navigation Across Thread Boundaries

Navigation within the visualized execution trace(s) has to be as intuitive as possible. So synchronization between the separate thread views is a must - modifying the shown time range in one of the views should also update the other views accordingly.

However, synchronization, in this case, is not trivial. Non-linear time mapping is applied to execution times during import, because execution durations typically vary a lot. Without time mapping, long lasting function/method executions would span multiple screens whereas very short executions maybe only span a single pixel. As the current implementation applies time mapping separately for each thread, there is no global time that could be used for synchronization purposes. It would probably be desirable to calculate such time mapping on-the-fly for the currently shown time slot across the shown threads in order to ensure a consistent global time.

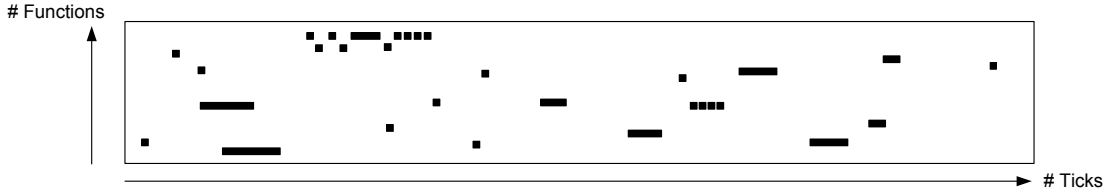


Figure 2: An execution trace interpreted as b/w 2-dimensional raster image.

5.2 Evaluating Techniques to Identify Relevant Sections in Execution Traces

As a matter of fact, Function Boundary Traces usually describe thousands to millions of sequential or parallel function executions. Depending on the purpose with which the trace was recorded, the relevant sections therein may be only a small percentage of the recorded data. So navigating to the relevant sections of the trace - where the synchronization of the analyzed applications is handled - can be a tedious task. Hence, the idea is to provide developers with pointers to the relevant time slots within such a trace.

Kuhn and Greevy [4] propose to interpret execution traces as a signal in time and introduce a visualization that enables recognition of stack depth rising and dropping. Their assumption is that complex functionality typically causes large stacks to be created at runtime. Consequently, their visualization is tailored for analyzing the complexity of explicitly traced features.

Inspired by Kuhn and Greevy, we aim to evaluate similar interpretation of execution traces as a signal in time to identify outliers within the whole recorded execution. More precise, the goal is to apply known image processing techniques to execution traces. Image registration is commonly used to, e.g., determine land mass movements in images generated by airborne microwave scanners. Land mass movements are determined by calculating the correlation between two images and thus deriving the relative shift between the two images. We aim to re-use this idea in order to identify outliers caused by flawed synchronization.

An execution trace can be thought of as a 2-dimensional matrix (or even a b/w 2-dimensional raster image) having the dimensions (#funcs, #ticks). Whenever a function is active, the value of that cell/pixel is 1, otherwise 0 (see Figure 2). Using the Discrete/Fast Fourier Transform (DFT/FFT) [6], it is possible to efficiently calculate frequency representations of the recorded threads (see Figure 3). An almost regular execution containing a single outlier will cause the frequency representation of thread 2 to contain two peaks: the highest peak representing the regular executions and a lower peak representing the outlier. A virtually correct execution can be obtained by asking the user to select a part of the execution which is correct, i.e., has acceptable execution costs. The differences in the frequency representation of the flawed execution and the virtual execution with correct synchronization provides hints to the outliers.

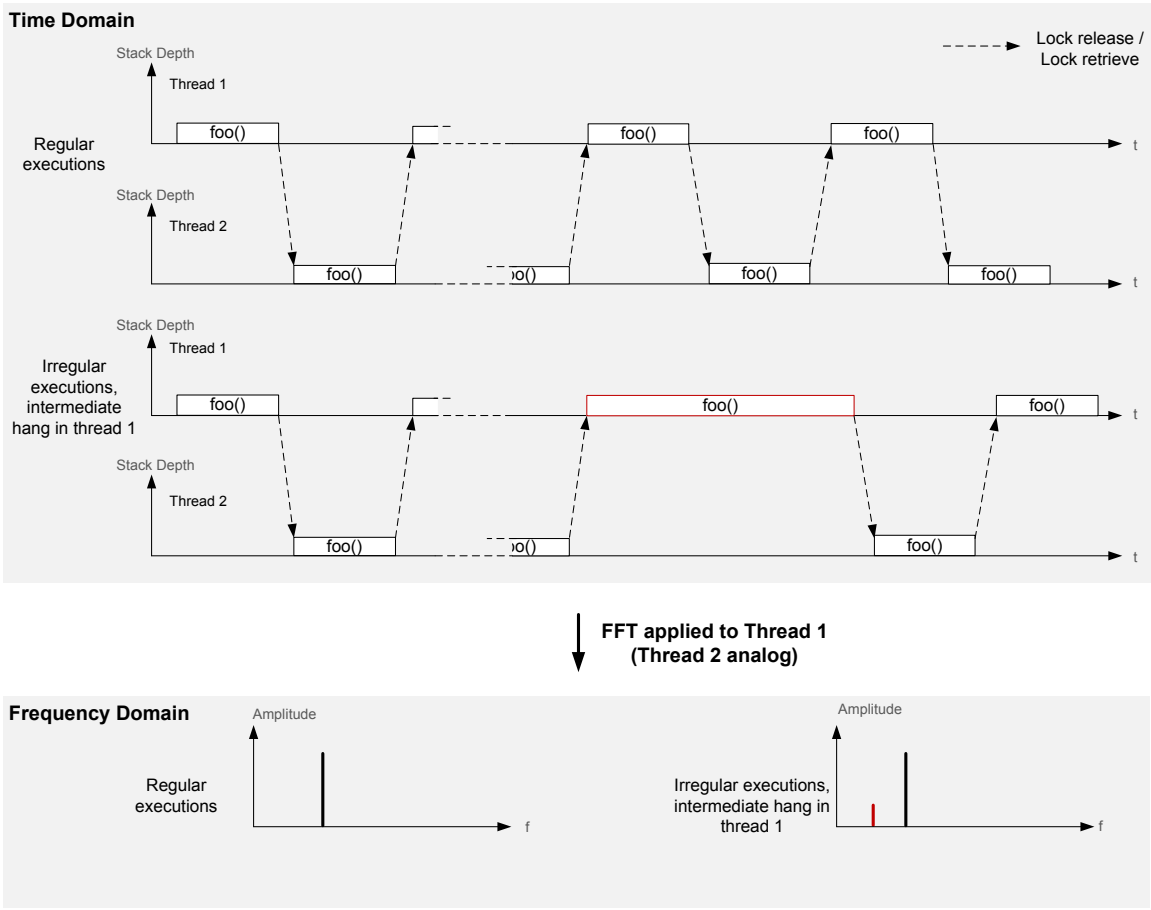


Figure 3: Example of time and frequency domain representations for an execution trace.

The inverse fourier transform of a frequency spectrum containing only the differences (outliers) will provide a virtual trace with repeating execution of the outlier. Superposing this virtual trace on top of the recorded thread trace then enables developers to identify the outliers within the actual thread traces.

Alternatively, data mining techniques could be applied; e.g., the *Subgroup Discovery Problem* [3] describes a similar problem: a huge data set containing mostly regular values and only several irregular values. The goal then is to identify the largest possible subset having different characteristics than all other elements (such as very few irregular executions).

5.3 Further Applications of Frequency Spectra Analysis

Interpreting execution traces as a signal in time can probably be useful in tackling other debugging problems, as well. For example, frequency domain comparison of two traces of the same feature – one before introducing a bug and one afterwards – will give some indication of the differences and thus likely help to find the bug in the proximity of the differences.

References

[1] Stefan Diehl. *Software Visualization. Visualizing the Structure, Behaviour, and Evolution of Software*. Springer, Berlin, 2007.

[2] Susan L. Graham, Peter B. Kessler, and Marshall K. Mckusick. Gprof: A call graph execution profiler. In *SIGPLAN '82: Proceedings of the 1982 SIGPLAN symposium on Compiler construction*, pages 120–126, New York, NY, USA, 1982. ACM.

[3] Willi Klösgen. *Handbook of data mining and knowledge discovery*. Oxford University Press, Inc., New York, NY, USA, 2002.

[4] Adrian Kuhn and Orla Greevy. Exploiting the analogy between traces and signal processing. In *ICSM '06: Proceedings of the 22nd IEEE International Conference on Software Maintenance*, pages 320–329, Washington, DC, USA, 2006. IEEE Computer Society.

[5] G.J. Nutt, A.J. Griff, J.E. Mankovich, and J.D. McWhirter. Extensible parallel program performance visualization. *Modeling, Analysis, and Simulation of Computer Systems, International Symposium on*, 0:205, 1995.

[6] Tao Pang. *An Introduction to Computational Physics*. Cambridge University Press, New York, NY, USA, 1997.

[7] Vijay Janapa Reddi, Alex Settle, Daniel A. Connors, and Robert S. Cohn. Pin: a binary instrumentation tool for computer architecture research and education. In *WCAE '04: Proceedings of the 2004 workshop on Computer architecture education*, page 22, New York, NY, USA, 2004. ACM.

[8] Kyle Wheeler and Douglas Thain. Visualizing massively multithreaded applications with threadscope. *Concurrency and Computation: Practice and Experience*, 2009.

[9] Qiang A. Zhao and John T. Stasko. Visualizing the execution of threads-based parallel programs. Technical report, Georgia Institute of Technology, 1995.

Web Service Generation and Data Quality Web Services

Tobias Vogel
tobias.vogel@hpi.uni-potsdam.de

1 Overview

My research on service-orientation during my first 5 month in the Research School was twofold. In the first part, which was more practical, I examined how existing applications can be provided for service-oriented architectures. More concrete, I integrated a service to wrap web applications into the PoSR framework which has been submitted to the IEEE Services Cup.

Second, I investigated on the provisioning of data quality Web Services for duplicate detection. I examined differences between traditional and service-oriented duplicate detection and identified seven separate problem classes and their corresponding properties.

2 Potsdam Service Repository (PoSR)

The joint PoSR project is a collaboration between the Business Process Technology group and the Information Systems group at HPI. It serves as a showcase to investigate on the benefits and usefulness of Web Service composition: existing Web Services are found by a crawler or generated on top of existing web applications. References for both, services and their meta data are stored in the repository where service requesters can search for appropriate services. Services can be graphically aggregated to composite Web Services while at the same time user interfaces are automatically created both for the single services and their compositions based on the discovered WSDL descriptions.

My contribution was the web application wrapper where I developed a methodology to semi-automatically generate Web Service (SOAP) interfaces for multi-stepped web applications.

Online services, e.g., shopping sites or travel information sites gain their user input via single or sequences of HTML forms. With this user provided information, web pages are created dynamically, called the *Deep Web*. This interaction model works well for interactions between humans and web applications. It is not feasible if computer programs have to autonomously use the functionality offered by web applications, for example to integrate web applications' functionality as a business process in existing landscapes. To allow this, it might be desired to obtain Web Service interfaces for this functionality, however, in general, they are not offered. Thus, they can be generated by

third parties on top of the existing, published web applications, which is possible with the developed Generator.

The generated Web Services “use” the web applications on behalf of the user by imitating the HTTP communication between the server and the user’s web browser, i.e. sending appropriate sequences of HTTP requests to the web server. Figure 1 shows how the wrapper integrates into the communication map between client and server and thus, provides a SOAP interface for the web application. The particular communication protocol, i.e. the sequence and characteristics of the respective HTTP calls, is implemented in the Web Service. The following sections show how this knowledge is derived from web applications.

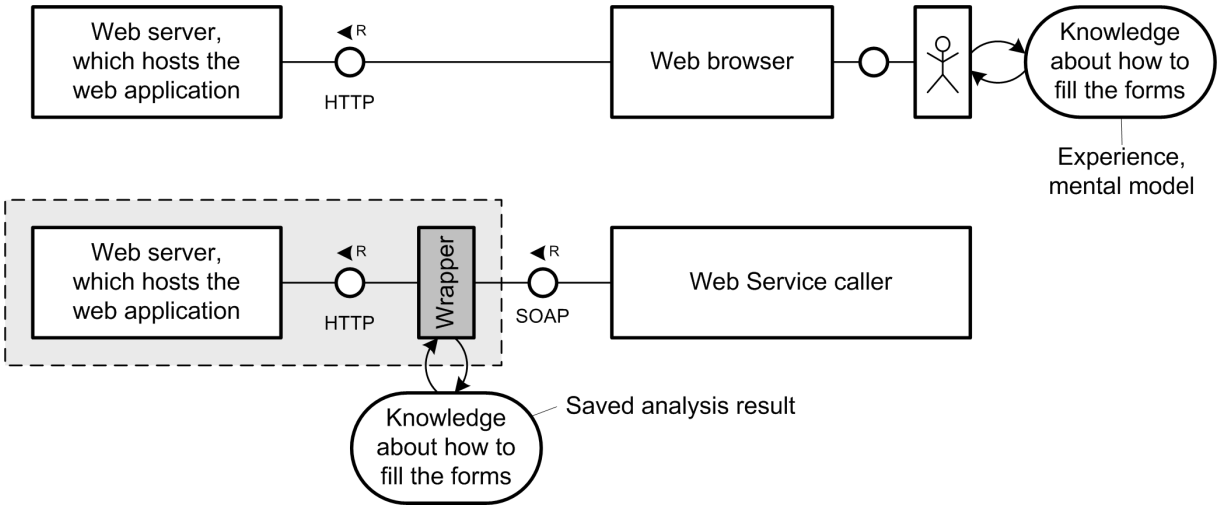


Figure 1: The upper part of the figure shows the intended interaction with the web application: a human user employing a web browser. Below, a wrapper s located between the web application and the client, accepting SOAP calls and submitting corresponding HTTP requests to the web server to imitate the upper behavior. Now, the web application is accessible as a Web Service.

2.1 Web Application Model

Web applications are computer programs that use HTML pages as user interface. User input is provided by HTML forms. Unlike traditional applications, web applications expose internal information, e.g., variable names or data types, in these forms. Furthermore, in multi-stepped web applications not all information are undisclosed in each form; they might appear in one or another form only, instead. This can be illustrated in a table (Figure 2) where columns are steps (or forms) and rows are the web application’s variables which appear occasionally. This is called the *web application’s model*, the set of all internal variables.

The Web Service caller needs a concise Web Service description with all the relevant, use-case dependent input parameters which are normally submitted little by

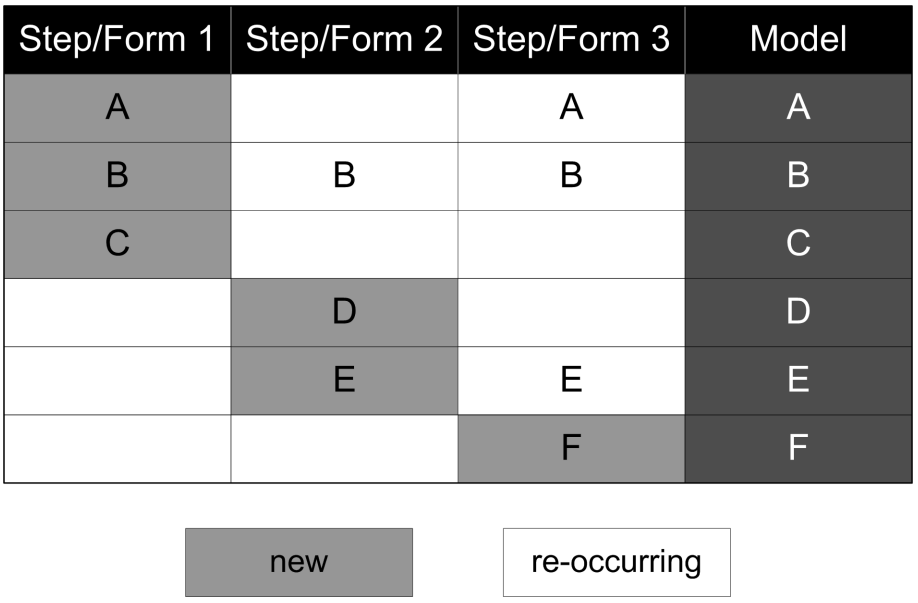


Figure 2: An web application's model containing 6 model elements from 3 forms.

little when visiting the multi-stepped web application with a browser. The generated Web Service takes these parameters and sends them to the web application. However, to function properly, also static, non-use-case dependent variables (e.g., hidden variables) have to be submitted and thus are to be contained in the Web Service's implementation. The information needed for this is retrieved from the model.

The main challenge is to aggregate form elements appearing in different forms to the same variable in the model. This can be achieved with the aforementioned exposed information, illustrated in Listing 1.

```
42 ...
43 <tr>
44   <td>
45     <label for="tel_field">
46       Tel:
47     </label>
48     <input type="text" name="telephone" value="01234567"
49       id="tel_field" class="address_field"
50       maxlength="40">
51   </td>
52   <td>
53     Insert your telephone number
54   </td>
55 </tr>
56 ...
```

Listing 1: Example snippet from an address entry form

Each element of a form has a name and a type (line 48) which are the core information of any form element. Furthermore, there might be an initial value set when the form is loaded. Additional meta information might be available. In the example, an `id` and a `class` attribute are given (line 49). While these may already provide some hints for a human user, better descriptions can be found as visible rendered text as in lines 46 and 53. These descriptions can be offered as labels (line 45 to 47) or as text in near proximity (line 53) of the form element [5, 8].

2.2 Form Element Matching

To wrap a web application, the relevant function has to be run through by a human user, while the HTTP traffic is monitored by the Web Service generator and complete forms as well as their form elements are analyzed. In each step, all new form elements are tried to be matched to existing model elements, which have been found in previous steps. The decision, whether or not to match these elements, is taken from a similarity measure. All of the information mentioned in Section 2.1 is used to estimate this similarity, however different weights are applied due to the different degree of uncertainty in the attributes.

The most reliable attribute is the form element's name, which is unlikely to change over different forms. Descriptive information (surrounding text, `class` attributes, etc.) might not be unique to this form element. This is why it adds a lower contribution to the overall similarity measure. The values of sent and received form elements provide the third contribution. In my experiments, I chose 1.0/0.75/0.5 for the previous weights (in the same order). I further applied a threshold of 0.6 to relaxate the matching, so that not every form element is forcefully matched to an existing model element, but new model elements can be created (Figure 3, compare to Figure 2 in step 3).

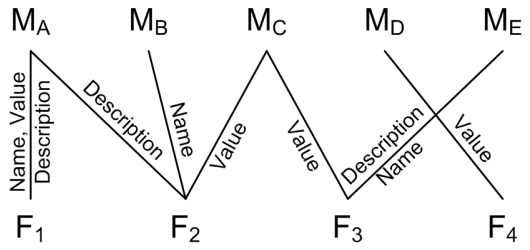
The Web Service is provided with the full model as well as the information, when to send which form with which model elements. The actual values are read from the Web-Service-invoking SOAP message. To allow the Web Service caller to provide meaningful data, the model element data is also used to create a WSDL file containing the descriptions, default values, types, etc. for all user-definable (i.e., non-hidden, non-disabled) model elements.

2.3 Outcome

The approach successfully matches corresponding form elements to model elements. With this, it is possible, to create single-call Web Services on the basis of multi-stepped web applications. However, the current prototypical implementation does not work on every existing web application, because, for example, the generated Web Services are incapable to execute JavaScripts, understand HTTPS or solve CAPTCHAs, etc. I documented the final outcome in a paper [7] which has been presented on the Workshop on Engineering Service-Oriented Applications (WESOA) in November 2009.

Similarity matrix for step 3

Already known model elements in step 3



Discovered form elements in step 3

Name = 1
Description = 0.75
Value = 0.5
Treshold = 0.6

	F_1	F_2	F_3	F_4	
M_A	2.25	0.75			
M_B		1.0			
M_C		0.5	0.5		
M_D				0.5	
M_E			1.75		

Figure 3: Matching algorithm during step 3. The left part shows five already known model elements (M_A to M_E) on the top and four form elements (F_1 to F_4) discovered in this step. The lines between them illustrate the similarity regarding to the three different similarity metrics. The table on the right shows the summed up similarities between any possible pair. Shaded cells constitute that this form element is regarded as being a representative of the corresponding model element and thus, is aggregated to it. Similarities below the threshold are ignored. M_C and M_D are not matched and therefore do not present any form elements in this step. F_4 could also not be matched and thus, constitutes an own, new model element.

3 Data Quality Web Services

Data quality plays an important role for entrepreneurial success. However, many companies do not care about the data quality in their ERP or CRM systems, as recent studies show[1]. Several measures can be driven to increase data quality, e.g., data normalization, duplicate detection, data fusion, etc.

In my research, I concentrate on the detection of duplicates. Traditional duplicate detection employs well-established algorithms and heuristics, which—in short—search through a database and estimate the similarity of pairs of tuples based on data type, value, and additional information to identify these pairs as possible duplicates. However, sometimes the amount of available information is restricted: the schema might not be up-to-date, the mapping is unclear, privacy issues prevent full access on all the data, etc.

Thus, the research question is, which information can be left out while still achieving appropriate results. In other words: which information is essential for a duplicate detection process and which information has to be inferred from the data.

Web Services share many characteristics with the restrictions mentioned before.

[1] http://www.pbinsight.com/about/newsroom/press-releases/detail/3390_data-quality-study-reveals-businesses-still-face-significant-challenges/ http://www.gartner.com/it/page.jsp?id=589207

Usually, they do not have access to the full amount of data and rely on the provided input information, generally available information, and/or other Web Services. They are invoked on-demand with exactly the information that is to be decided about, e.g., if only a small number of items has to be tested for similarity in an ad-hoc manner. Further, they provide a clearly specified functionality while remaining as general as possible to ensure a broad number of possible service requesters. These properties turn Web Services into the ideal foundation for evaluating duplicate detection algorithms with the limitations described above.

There are a number of providers and registries for data quality Web Services[2]. They provide services for data alignment (e.g., format telephone numbers properly), data completion (e.g., add postcodes to cities), or data verification (e.g., check ISBNs against book titles). However, there are no Web Services for duplicate detection.

3.1 Comparison Between Traditional and Service-Oriented Duplicate Detection

Traditional duplicate detection algorithms operate on whole databases and have relatively fast access to all tuples, which allows them to drive statistical analysis on this data, e.g., the distribution of values in specific columns or the detection of key constraints. These algorithms further base their similarity measure on the table's data types. This makes it possible to compare telephone numbers differently from names of persons. Also attribute names provide useful information, because similar family names are more significant than matching first names, for example. Furthermore, traditional algorithms are provided with a mapping between the schemas of two data items to compare. Finally, the attributes of the items are distinguishable.

In contrast, duplicate detection Web Services would only operate on the provided values as they do not access the Web Service requester's data storage. They might not have data type information, since they are only provided with attributes and their values which are deserialized as strings. Further, also the attribute names might be missing and thus, only lists of values would be provided. The mapping can be given (e.g., by the order of the attributes) but might also be unknown (or the order is insignificant). Finally, even field separators might be missing, e.g., when two pieces of textual information are compared.

To sum up, there are four different pieces of information which might or might not be available:

1. Field separators
2. Mapping
3. Attribute names
4. Data types

[2] `http://www.strikeiron.com/`, `http://www.xmethods.net/`, `http://www.webservicex.net/`

Calculative, this results in $2^4 = 16$ different combinations. However, further examination has shown that only seven combinations are possible or practically relevant, respectively.

3.2 Classes of Different Duplicate Detection Problems

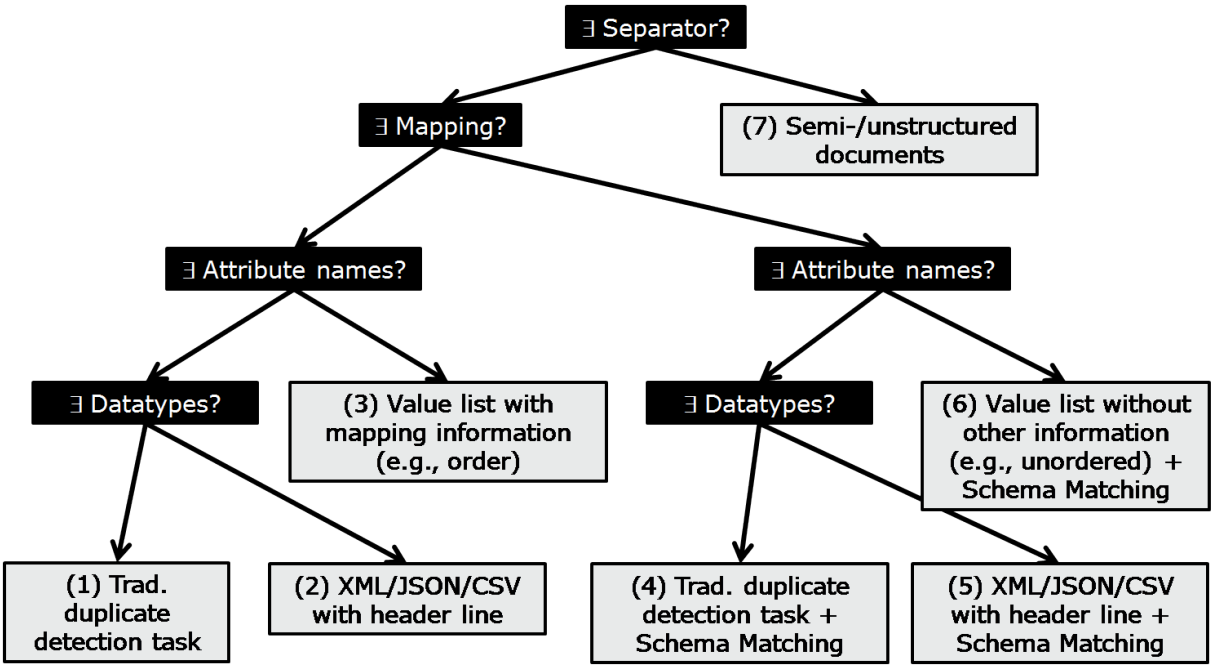


Figure 4: Seven different classes of duplicate detection are illustrated in a decision tree. Left means “available”, right means “not available”.

The four decisions are resulting in seven classes of duplicate detection problems, depicted in Figure 4. They are briefly characterized in the following.

1. Availability of separators, mapping, attribute names, and data types: this is most similar to the traditional duplicate detection task. However, only few tuples are available in this case, already.

2. Availability of separators, mapping, and attribute names: data type information is missing. This can be the case when the input is provided in JSON format, for example. The mapping is derived from the attribute names or the order of the attributes. Data type information has to be inferred to be used for the similarity estimation.

3. Availability of separators and mapping: attribute names are missing. Thus, only lists of values can be provided while the mapping might be specified by the order of the items in the list, for example. To apply corresponding algorithms, the attributes have to be classified. The better this classification works, the more tailored algorithms can be used.

4.-6. These cases are the same as the corresponding classes with a mapping, however a schema matching process has to precede the detection of duplicates.

7. Here, no further information is available. This might be the case for semi- or unstructured documents such as HTML or RTF files.

Missing branches are irrelevant oder impossible cases, e.g., having no separators between the attributes, but having data types for them is an example for an impossible case.

There are two strategies to do duplicate detection within these different classes. First, the missing information can be tried to be inferred, so that the problem can be shifted in or towards easier classes which work with more information. Second, similarity measures have to be applied within the original problem classes.

3.3 Related Work

The different classes are already in focus of ongoing research, separately. For example, duplicate detection in XML structures [9] is examined by Weis et al.

Research in duplicate detection for unstructured or semi-structured texts is often applied in plagiarism detection systems as MOSS[3] (Schleimer et al. [6]) or YAP3 (Wise [11]).

Navarro [4], Winkler [10], and Elmagarmid et al. [1] give surveys on existing string comparison algorithms, based on edit distances as for example the Levenshtein Distance [3] and outline the specific usefulness for the corresponding values.

4 Next Steps

Further examination within the data quality area will answer the questions

- which pieces of information have to be inferred to handle class-specific problems,
- whether training data provided with a Web Service call supports the result quality, and
- whether tuples of different classes are comparable.

Further, data and a prototype have to be defined and implemented for being able to evaluate the results of different approaches in the diverse classes.

4.1 Publications

Concerning the first part of my research, I submitted a paper to the *Fifth International Workshop on Engineering Service-Oriented Applications*, taking place in Stockholm, November 2009.

Mohammed and I successfully submitted the joint work on PoSR to the IEEE Services Cup 2009 on the *International Conference on Web Services 2009* [2].

[3] `http://theory.stanford.edu/~aiken/moss/`

References

[1] Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 2007. Ahmed K. Elmagarmid and Panagiotis G. Ipeirotis and Vassilios S. Verykios.

[2] Mohammed AbuJarour, Mircea Craculeac, Falko Menge, Tobias Vogel, and Jan-Felix Schwarz. PoSR: A comprehensive System for Aggregating and Using Web Services. In *International Conference on Web Services*, 2009.

[3] Vladimir I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Technical report, 1966.

[4] Gonzalo Navarro. A Guided Tour to Approximate String Matching. *ACM Computing Surveys*, 1999.

[5] Sriram Raghavan and Hector Garcia Molina. Crawling the Hidden Web. In *International Conference on Very Large Databases*, 2001.

[6] Saul Schleimer, Daniel S. Wilkerson, and Alex Aiken. Winnowing: Local Algorithms for Document Fingerprinting. In *Special Interest group on Management of Data Conference*, 2003.

[7] Tobias Vogel, Frank Kaufer, and Felix Naumann. Encapsulating Multi-Stepped Web Forms as Web Services. In *Fifth International Workshop on Engineering Service-Oriented Applications*, 2009.

[8] Jiying Wang and Frederick H. Lochovsky. Data Extraction and Label Assignment for Web Databases. In *International Conference on World Wide Web*, 2003.

[9] Melanie Weis and Felix Naumann. DogmatiX - Track Down Duplicates in XML. In *Special Interest group on Management of Data Conference*, 2005.

[10] William E. Winkler. Overview of Record Linkage and Current Research Directions. Technical report, Bureau of the Census, 2006.

[11] Michael J. Wise. YAP3: Improved Detection of Similarities in Computer Program and Other Texts. In *Twenty-Seventh Special Interest Group on Computer Science Education Technical Symposium*, 1996.

Model-Based Extension of AUTOSAR for Architectural Online Reconfiguration

Basil Becker
basil.becker@hpi.uni-potsdam.de

In the last years innovations in the automotive domain have more and more been realized by software leading to a dramatically increased complexity of such systems. Additionally automotive systems have to be flexible and robust, e.g., to be able to deal with failures of sensors, actuators or other constituents of an automotive system. One possibility to achieve robustness and flexibility in automotive systems is the usage of reconfiguration capabilities. However, adding such capabilities introduces even higher degree of complexity. To avoid this drawback we propose to integrate reconfiguration capabilities into AUTOSAR, an existing framework supporting the management of such complex system at the architectural level. Elaborated and expensive tools and toolchains assist during the development of automotive systems. Hence we present how our reconfiguration solution has been seamlessly integrated into such a toolchain.

1 Introduction

Today most innovations in the automotive domain are realized by software. This results in a dramatically increasing complexity of the developed software systems[1]. The objective of the AUTOSAR framework is to deal with this complexity at the architectural level. Additionally these systems need to deal with diverse situations concerning the context in which the software is operating. Such systems and especially the software, which is realizing essential functionalities of the overall system, need to be flexible to react on changes of its context. Regardless if such a system need to react on failures or on other contextual situations[2], flexibility and robustness plays an important role in today's automotive applications.

Reconfiguration is one possibility to facilitate the flexibility and robustness of such systems. There exist different possibilities to realize reconfiguration within automotive software. One is to realize reconfiguration mechanisms at the functional level. Because the AUTOSAR framework primarily provides mechanisms to deal with the complexity at the architectural level also the reconfiguration aspects should be available at the same level. Because deriving architectural information from the functional level could be difficult or even impossible we propose to specify reconfiguration aspects at the

[1] The complexity concerning the size of the developed software, the functionality realized by the software system and so on.

[2] An example for such a situation, which is not related to a failure is in case the car is connected to diagnostic devices.

architectural level and to automatically derive the needed functionality based on the architectural information.

Further in a typical development scenario one has to deal with black-box components provided by third parties and elaborated information about the included functionality is not available, what also hampers the management of reconfiguration aspects at the functional level. Another possible solution is to introduce a new approach inherently facilitating reconfiguration aspects in the context of automotive systems. Today standard methods and tools already exist for supporting the development process of AUTOSAR. Because adapting existing tools or developing new once is very costly the propagation of such a new approach would be hardly suitable in practice. Summarizing we have identified the need for an development approach that is able to provide reconfiguration capabilities at the architectural level, can be seamlessly integrated into an exiting development solution and can also include third party components into the reconfigurable architecture. In this work we show how reconfiguration capabilities, which are currently not included in the existing AUTOSAR approach can be supported at the architectural level without degrading existing development solutions, tools or the standard itself. We further show how the needed functionality for realizing the reconfiguration logic can be automatically generated based on the architectural information describing the reconfiguration. The used application example for our evaluation is related to the field of fault tolerant systems and from our perspective such systems are one possible field to which reconfiguration like discussed in the remainder of this work can be applied.

The remainder of this paper is organized as follows. In Section 2 we briefly introduce the existing toolchain, which builds the technological foundation for our investigation concerning the developed extension for on-line reconfiguration within the AUTOSAR framework. Subsequently in Section 3 we show how such a system is usually modeled with the given tools and how the additional reconfiguration aspects could be formulated based on the input/output of the existing toolchain. In Section 4 we show how these created additional reconfiguration aspects are automatically merged back into the original architecture and how the merged result fits into the existing tools without discarding or degrading parts of the original toolchain. Finally we give short discussion concerning the current results of our work in Section 6.

2 Existing Development Approach

For the development of embedded systems – especially in the automotive domain – several tools exist that provide capabilities for model-based development of such systems. Tools used by companies typically are mature, provide reliable and optimized code generation mechanisms and are as expensive as complex. Hence, any technique that claims being usable in the domain of embedded / automotive systems must be integrated into the existing toolchain. We will use this section to exemplary describe a toolchain, which might be used in the context of the AUTOSAR domain specific language.

2.1 AUTOSAR

The AUtomotive Open System ARchitecture (AUTOSAR) is a framework for the development of complex electronic automotive systems. AUTOSAR provides a layered software architecture consisting of the Application layer, the Runtime Environment and the Basic Software layer. Figure 1[3] shows the different layer of the architecture. The Basic Software layer provides services concerning HW access, communication and Operating System (OS) functionality (cf. [1]). The Basic Software provides several interfaces in a standardized form to allow the interaction between the Basic Software layer and the application layer routed through the Runtime Environment. The Runtime Environment handles the communication between different constituents of the application layer and between the application layer and the Basic Software layer (e.g., for accessing Hardware via the Basic Software, cf. [2]). The Application layer consists of Software Components, which can be hierarchically structured and composed to so called Compositions. Software Components and Compositions can have ports and these ports can be connected via Connectors (see [3] for more details). The real communication is realized through the Runtime Environment in case of local communication between Software Components (Compositions) on the same node (Electronic Control Unit) or through the Runtime Environment in combination with the Basic Software in case of communication between different nodes.

The main focus of AUTOSAR is the modeling of architectural aspects and of structural aspects. The behavior modeling (e.g., needed control functionality for reading sensor values and setting actuators) is not the main focus of the AUTOSAR framework. For modeling such behavior existing approaches and tools can be integrated into the development process of AUTOSAR. One commonly used tool for the model based development of behavior is MATLAB/Simulink (like described in Section 2.2). For executing such functionality AUTOSAR provides the concept of Runnables, which are added as a part of the internal behavior of a Software Component. Developed functionality could be mapped to Runnables and these Runnables are mapped to OS tasks. Additionally events can be used to decide inside an OS task if specific runnables are executed at runtime (e.g., runnables could be triggered by events if new data has been received via a port of the surrounding Software Component). For more details about the OS provided by the AUTOSAR framework see [4].

Once the modeling and configuration is done, in the current release version of AUTOSAR[4] changes at run-time concerning the structure of the application layer (e.g., restructuring connectors) are not facilitated by the framework.

2.2 Existing Toolchain

The scheme in Figure 2(a) shows one possible toolchain for the development of AUTOSAR systems. Rectangles with rounded corners represent programs, rectangles with cogwheels stand for processes. The arrows indicate exchange of documents, the type of the document (i.e. models, C-code or parameters) is annotated to the arrows.

[3] Picture taken from http://www.autosar.org/gfx/media_pictures/AUTOSAR-components-and-inte.jpg.
[4] Release 3.1

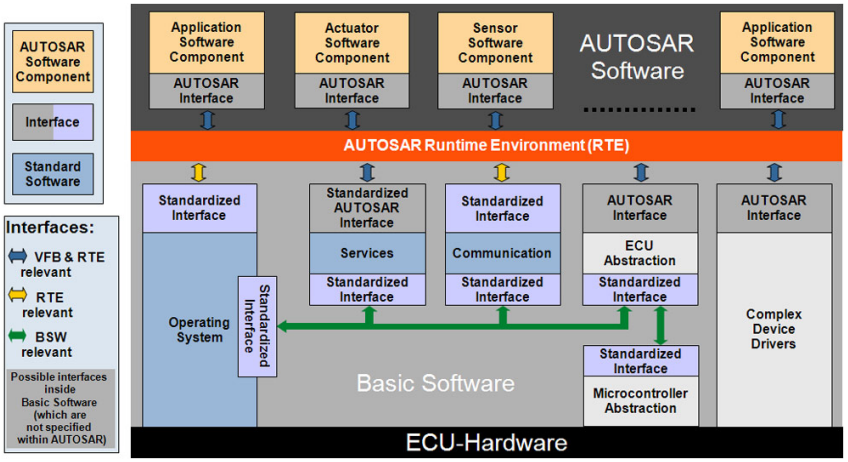


Figure 1: The AUTOSAR layered architecture

The system's architecture (i.e. components, ports and connectors) is modeled in SystemDesk[5]. Together with the architecture SystemDesk also supports the modeling of the system's deployment to several ECUs. The components behavior is specified using Matlab with the extension Simulink. For Matlab/Simulink (ML/SL) special AUTOSAR block sets exist, which allow the import of components specified in SystemDesk into Matlab and following the development of the component's functionality.

Further SystemDesk supports the generation of optimized C-Code, which conforms to the AUTOSAR standard concerning the Runtime Environment (cf. Subsection 2.1). Together with the C implementation of the software components modeled in SystemDesk the generated output also contains a configuration for the basic software layer. This layer is generated from specialized tools (e.g. Tresos by ElectroBit, abbreviated as BSW-C/G in Figure 2) and is specific to the system modeled in SystemDesk and the available hardware.

At the integration step a build environment compiles the generated C-Code and builds the software running on each ECU.

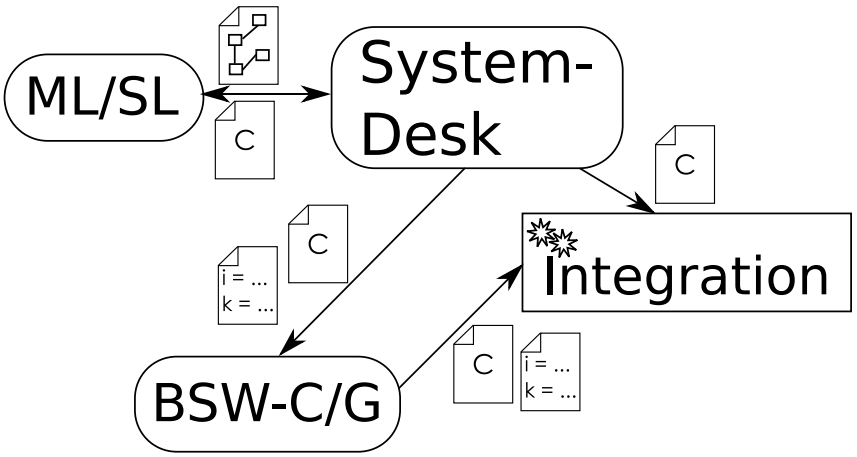
2.3 Evaluation Example

The used application example for showing the reconfiguration capabilities that are supplemented to the existing AUTOSAR framework in our approach is the reconfiguration of a set of adjacent aligned distance sensors. The discussed evaluation example allows reacting on sensor failures in the manner that the failure of individual sensor instances is compensated.[6]

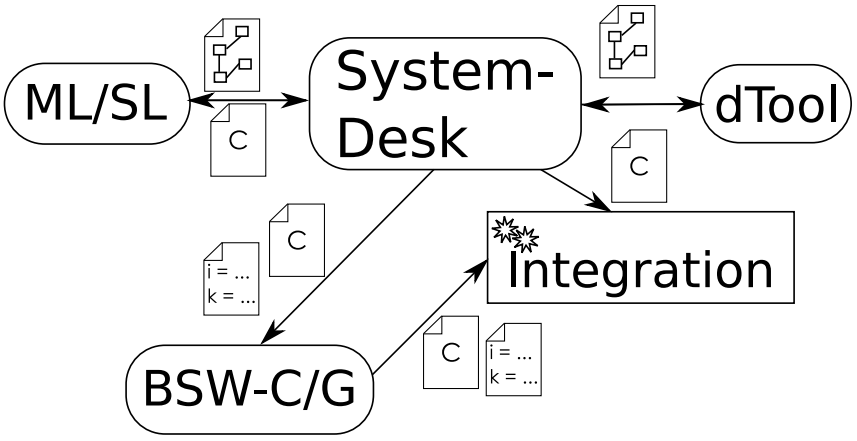
Such adjacent aligned sensors are commonly used in a modern car, e.g., in case of a parking distance control. Such a parking distance control uses sensors (e.g., ultrasonic sensors) embedded in the front or rear bumper for measuring the distance to nearby obstacles.

[5]http://www.dspace.de

[6]For our application example we assume that a sensor failure can be observed at the level of Software Components.



(a) Exemplary toolchain for development with AUTOSAR



(b) Tool chain for modeling reconfigurable AUTOSAR architectures

Figure 2: The current and the extended toolchain for the development with AUTOSAR

Additionally in Section 5 we discuss the evaluation results of experiments we have made on an evaluation platform using the techniques described in Section 3.

3 Modeling Reconfiguration

In order to make an AUTOSAR system architecture reconfigurable, some additional concepts are needed. The toolchain needs to be extended in a certain way that extensions do not make the existing toolchain invalid. From our perspective the best way is to integrate an optional tool that can be plugged into the existing toolchain.

3.1 Extended Toolchain

Our modeling approach is currently restricted to the modeling of AUTOSAR software architectures. The toolchain in Figure 2(b) shows our approach of extending the existing toolchain by another tool without degrading existing ones. By using this proposal

the developer is free to choose, whether he wants to use our given enhancement or not. He can either model an architecture, that does not provide any reconfiguration or he can use our tool in addition and empower himself to specify and realize reconfiguration aspects. The advantages are obvious: better control and overview due to the diagrammatic depiction.

3.1.1 SystemDesk

SystemDesk is a tool provided by *dSPACE*[7] supporting the modeling of AUTOSAR conform systems. Among other things it supports the modeling of the AUTOSAR HW and SW architectures. For modeling the SW architecture Software Components, compositions as well as ports, interfaces and connectors are provided as modeling artifacts. These artifacts can be used to describe the architectural aspects of a concrete SW architecture for a specific system like shown in Figure 3.[8] Besides modeling the architecture in SystemDesk, the tool also allows the linking of the Software Components to their behavior, written in C-Code or given in form of MATLAB/Simulink models.

Additionally the HW architecture including the used types of ECUs (Electronic Control Unit), the deployment of Software Components to these ECUs as well as additional information concerning the configuration (e.g., configuration concerning communication and the OS) can be specified. Based on this information SystemDesk automatically generates code, which can be compiled for the specified platform. Besides the code for the application layer SystemDesk also generates source code realizing the Runtime Environment functionality.

Figure 3 shows the relevant part of the SW architecture concerning our application example modeled in SystemDesk. Like depicted on the right side of Figure 3 the composition consists of four Software Components representing the distance sensors[9] connected to another composition *SensorLogic* evaluating the sensor values to a single value provided by the port *ShowDistanceOut*.[10]

The above mentioned elements (Software Components, ports and connectors) are used to describe the software when no reconfiguration is intended. Some additional elements shown in Figure 3 are described in more detail in the following section. These elements (*Interpolation*, *Reconfiguration* and the unused ports of the sensors) are used later to realize the reconfiguration functionality.

[7] www.dspace.de

[8] For the realization of control functionality other constituents can be imported into SystemDesk, e.g. in form of C-Code or Matlab/Simulink models, to realize the implementation of internal behavior of Software Components.

[9] The ports accessing the HW via the Runtime Environment and Basic Software are not shown here because they are not object of reconfiguration.

[10] To allow a better understanding *SensorLogic* calculates a single output value based on the different input values. Potentially also several output values can be computed.

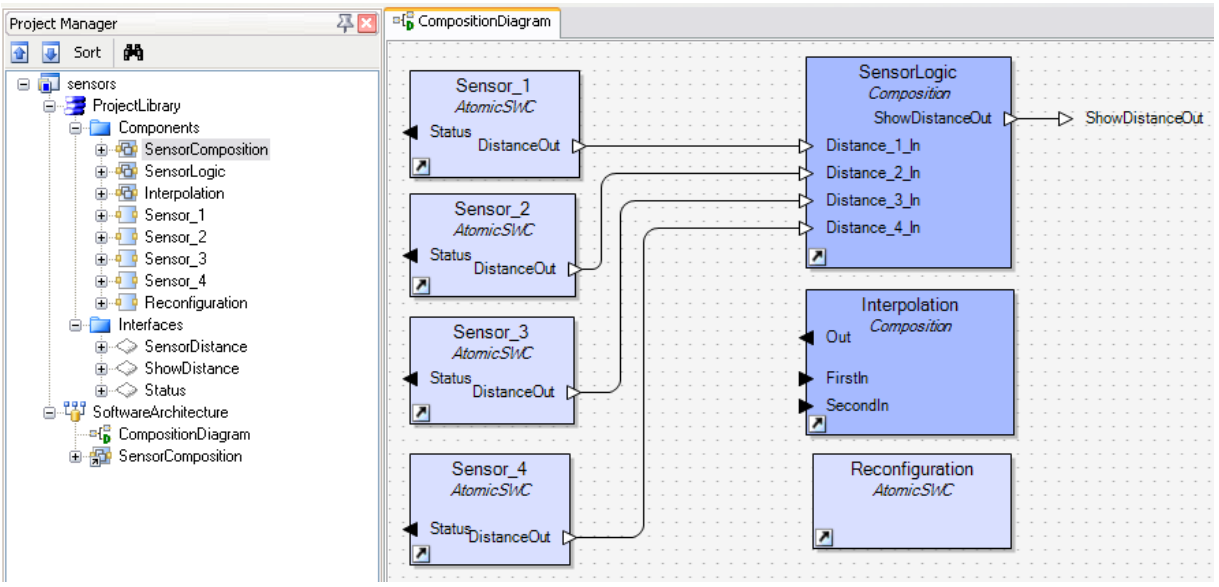


Figure 3: Configuration in SystemDesk

3.1.2 dTool

The usual modeling procedure is not altered until the modeling in SystemDesk[11] is initially done like described above. After the model from SystemDesk is exported in form of an XML file[12] and loaded into the *dTool* the constituents concerning the reconfiguration could be specified. Using the dTool we are now able to model two different aspects, relevant for the reconfiguration. On the one hand our tool allows creating new configurations, which differ from the initial one. Such differences are alternative connections (in form of connectors) between components and/or compositions. Which parts of the architecture are relevant concerning reconfiguration is indicated by the Software Component *Reconfiguration* included in the original SystemDesk model. Alternatively the dTool allows to manually choosing relevant parts of the imported architecture. On the other hand our dTool allows to model an automaton, which specifies how to switch between the modeled configurations.

Figure 4 depicts the configuration (modeled in the dTool) associated with the state that sensor two is broken. In the shown configuration the value of the port *DistanceOut* from the broken sensor *Sensor_2* is not available. Consequently the value sent to the port *Distance_2_In* of the composition *SensorLogic* is interpolated from the to sensor values of the first and the third sensor via the additional composition *Interpolation*.

Figure 3.1.2 shows the configuration associated with the state that sensor four is broken and the value sent to the port *Distance_3_In* of the composition *SensorLogic* is interpolated based on the sensor values of the second and the fourth sensor.

The composition *Interpolation* used here provides some functionality for interpolating two different sensor values. This functionality has been added specifically for our

[11] http://www.dspace.de/ww/en/ltd/home/products/sw/system_architecture _software/systemdesk.cfm

[12] The AUTOSAR framework specifies XML-Schemes for exchanging AUTOSAR models in a standardized form.

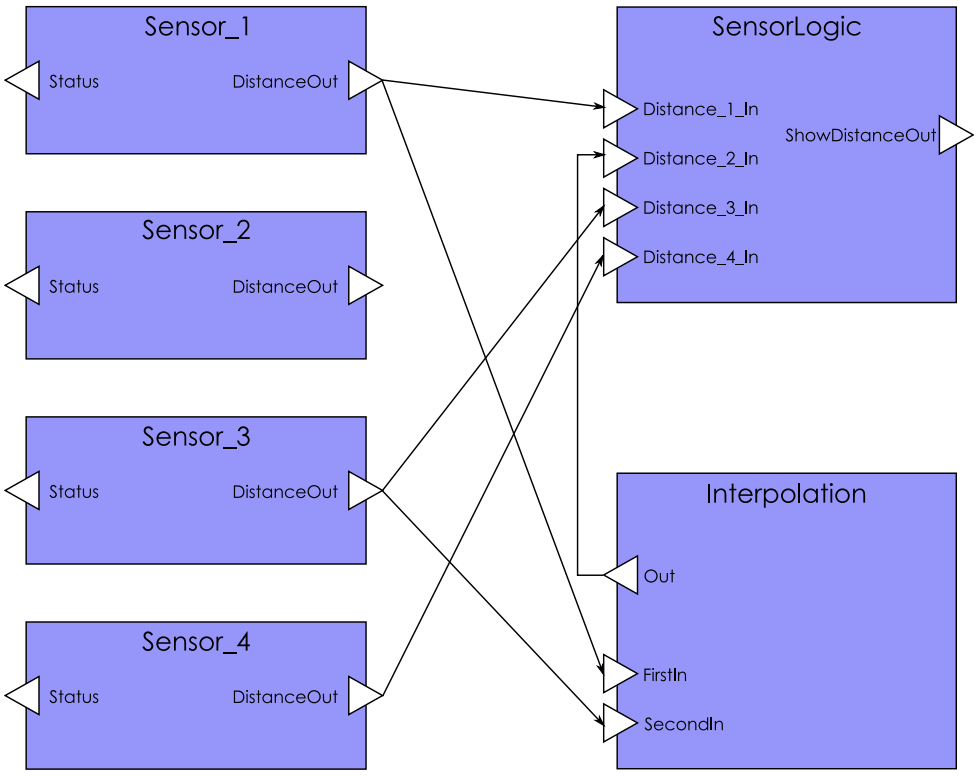


Figure 4: Configuration in case Sensor_2 is broken

application example.[13] This interpolation functionality is used to approximate the value of a broken sensor based on the values of two adjacent sensors. It is potentially possible to integrate this functionality into an existing Software Component, but for a better understanding, we decided to introduce a new Software Component for this purpose.

The second part, which could be modeled in the dTool relevant for the reconfiguration is the automaton shown in Figure 6 specifying how to switch between different configurations. The automaton consist of the initial state *initial*, where all four sensors work correctly, the state *sensor2broke* where the second sensor is broken, the state *sensor3broke* where the third sensor is broken and state *allfail* where the first or the fourth sensor or more than one sensor is broken. Transitions between these states specify which reconfiguration is applied at runtime. The transitions are further augmented with guards. These guards are expressions over the values provided by components within the reconfigurable composition, which provide information relevant for the reconfiguration (in our case these information are provided via the *Status*-ports of the four Sensor-Software Components). An example for such a guard is shown at the transition from state *initial* to state *sensor2broke* requiring that the status port of the Software Component *Sensor_2* provides the value 0 (indicating a broken sensor).

For the application example we assume that such status ports of the Software Components representing the sensors exist as we otherwise were not able to observe each sensors' status.[14]

[13] In our application example this functionality has been realized using Matlab/Simulink.

[14] Alternatively an observer could be realized in form of an additional Software Component evaluating

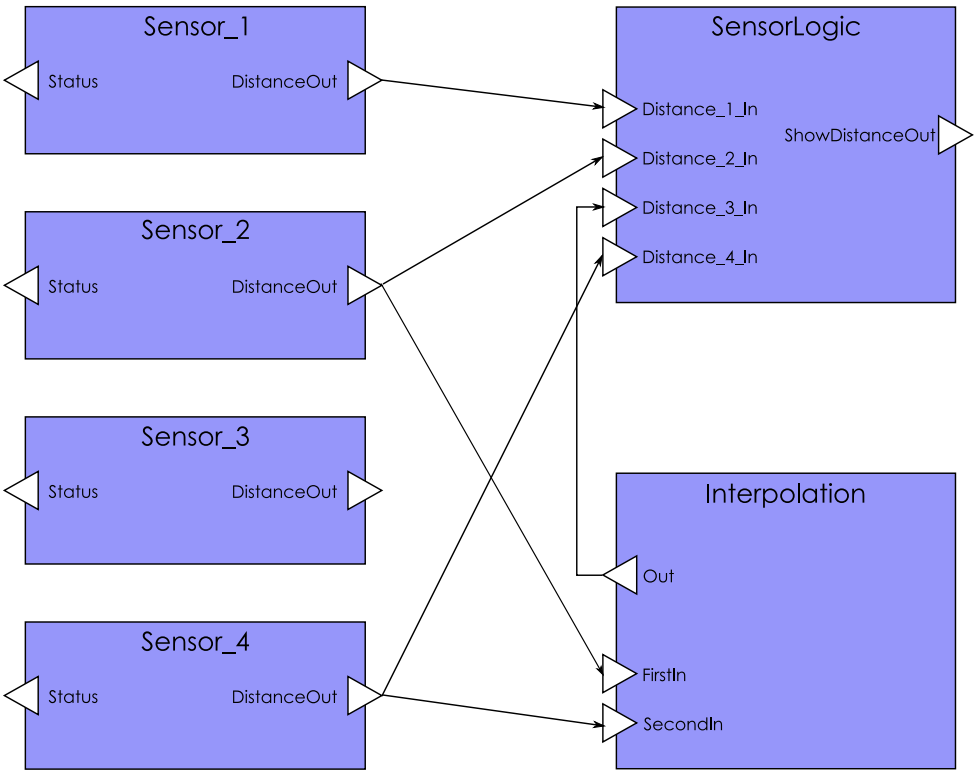


Figure 5: Configuration in case Sensor_3 is broken

4 Merge

In its current version the AUTOSAR standard does not support reconfiguration as a first class modeling element. Thus, SystemDesk also does not support modeling of diagrams that represent different variations of one composition. Hence the direct import of the reconfiguration, we have modeled in the *dTool*, is impossible. Nevertheless we want to make use of SystemDesk's elaborated and AUTOSAR standard conform code generation capabilities. We had to find a way to translate the reconfiguration behavior into a SystemDesk/AUTOSAR model. This is done by merging all configurations to one final model. In the final model, the reconfiguration logic will be encapsulated by two components, the RoutingComponent and the StateManager. A detailed description of the merge process can be found in [5].

5 Evaluation Results

The above described approach for the modeling and realization of reconfiguration aspects has been evaluated within a project arranged at Hasso-Plattner-Institute in collaboration with the dSPACE GmbH.

the sensor values over time and providing the status ports. If the measured values of consecutive points in time repeatedly have improper values (too big differences) a malfunction can be deduced.

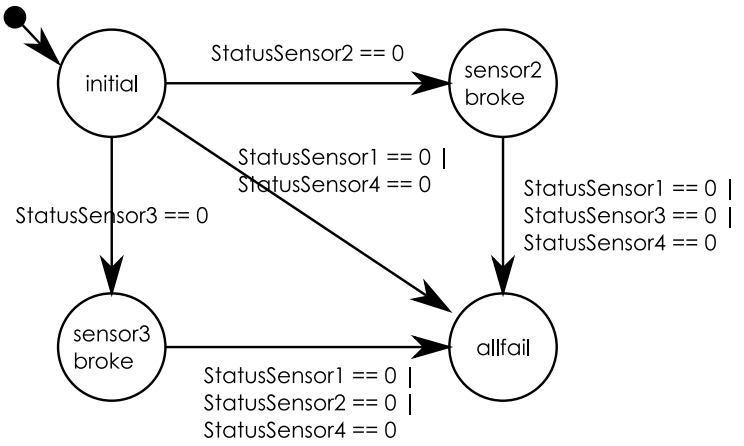


Figure 6: Reconfiguration automaton in the dTool

As an evaluation platform for the shown approach the Robotino robot[15] has been used, which provides an open platform for running C/C++ programs (among others) on a Real-Time Operating System (RTOS). The RTOS is provided in form of RTAI[16], which is a real-time extension for the Linux operating system. To be able to evaluate the developed concepts on this platform an execution environment has been realized based on the existing RTAI Linux, which allows to compile and execute the outcome of the above described extended toolchain including the resulting parts of the reconfiguration functionality.

The robot provides nine distance sensors uniformly distributed around its chassis. In the context of our evaluation experiments we modeled the reconfiguration of distance sensors accordingly to the above used evaluation example using nine instead of four sensors.[17]

The generated source code of the different tools has been compiled and executed on the platform to show the applicability of our approach. In addition we analyzed the overhead resulting from the reconfiguration functionality added by our approach in comparison to the original functionality without any reconfiguration capabilities. For this purpose we measured the execution time of the generated reconfiguration automaton included in the added StateManager in combination with the parts resulting from the routing functionality realized in the additional RoutingComponent.

In case of the nine sensors provided by the robot we measured execution times of the relevant parts concerning the reconfiguration functionality between 20 and 100 microseconds depending on the type of reconfiguration (react on the defect of one or several sensors at the same point in time). The tests have been realized on the equivalent execution platform on which the real functionality has been executed when running the application example on the robot.[18] While the robot provides a more powerful processor like it is the case for the most Electronic-Control-Units (ECUs) used within a modern car, even by using a platform or processor, which has only a tenth of the com-

[15] http://www.festo-didactic.com/int-en/news/learning-with-robots.htm
[16] For more details see https://www.rtai.org.
[17] For a better understanding we decided to only show four sensors in the previous sections.
[18] The robot is equipped with 300 MHz processor.

putation power we will not reach an overhead concerning the reconfiguration leading to an execution time much greater than one millisecond.

6 Conclusion

In this paper we have presented an approach to extend AUTOSAR architectures with reconfiguration capabilities. The approach fits into existing toolchains for the development of AUTOSAR systems and allows reusing tools, which where currently used. The overhead added to the resulting reconfigurable architecture has been shown to be minimal but the developer rewards an easier development of reconfiguration logic, which otherwise has to be done manually at the functional / implementation level. We have successfully shown that it is possible to use high-level architectural modeling techniques without generating massive run-time overhead.

Although our approach has only been evaluated in the context of AUTOSAR it should be applicable to almost any component based development approach.

For the future we plan to also support the reconfiguration of distributed compositions. From an architectural point of view a distributed composition does not differ from a local one, as AUTOSAR completely hides the communication details in the Runtime Environment-layer from perspective of the application layer. Anyway, a distributed scenario contains enough challenges such as timing delays, Basic Software configuration, deployment decisions concerning RoutingComponents, just to name a few. Further the high-level architectural modeling we have introduced in this paper also allows the verification of the modeled systems. First attempts in these directions have been very promising and we are looking forward to look into the details.

References

[1] AUTOSAR GbR. *List of Basic Software Modules*. Version 1.3.0.

[2] AUTOSAR GbR. *Specification of RTE*. Version 2.1.0.

[3] AUTOSAR GbR. *Specification of the Virtual Functional Bus*, 2008. Version 1.0.2.

[4] AUTOSAR GbR. *Specification of Operating System*, 2009. Version 3.1.1.

[5] Basil Becker, Holger Giese, Stefan Neumann, Martin Schenck, and Arian Treffer. Model-based extension of autosar for architectural online reconfiguration. In Stefan Van Baelen, Thomas Weigert, Ileana Ober, and Huascar Espinoza, editors, *Proceedings of the 2nd International Workshop on Model Based Architecting and Construction of Embedded Systems (ACES-MB 2009)*, volume 507 of *CEUR Workshop Proceedings*, pages 123–137. CEUR-WS.org, 10 2009.

Aktuelle Technische Berichte des Hasso-Plattner-Instituts

Band	ISBN	Titel	Autoren / Redaktion
30	978-3-86956-009-0	**Action Patterns in Business Process Models**	Sergey Smirnov, Matthias Weidlich, Jan Mendling, Mathias Weske
29	978-3-940793-91-1	**Correct Dynamic Service-Oriented Architectures: Modeling and Compositional Verification with Dynamic Collaborations**	Basil Becker, Holger Giese, Stefan Neumann
28	978-3-940793-84-3	**Efficient Model Synchronization of Large-Scale Models**	Holger Giese, Stephan Hildebrandt
27	978-3-940793-81-2	**Proceedings of the 3rd Ph.D. Retreat of the HPI Research School on Service-oriented Systems Engineering**	Hrsg. von den Professoren des HPI
26	978-3-940793-65-2	**The Triconnected Abstraction of Process Models**	Artem Polyvyanyy, Sergey Smirnov, Mathias Weske
25	978-3-940793-46-1	**Space and Time Scalability of Duplicate Detection in Graph Data**	Melanie Herschel, Felix Naumann
24	978-3-940793-45-4	**Erster Deutscher IPv6 Gipfel**	Christoph Meinel, Harald Sack, Justus Bross
23	978-3-940793-42-3	**Proceedings of the 2nd. Ph.D. retreat of the HPI Research School on Service-oriented Systems Engineering**	Hrsg. von den Professoren des HPI
22	978-3-940793-29-4	**Reducing the Complexity of Large EPCs**	Artem Polyvyanyy, Sergy Smirnov, Mathias Weske
21	978-3-940793-17-1	**"Proceedings of the 2nd International Workshop on e-learning and Virtual and Remote Laboratories"**	Bernhard Rabe, Andreas Rasche
20	978-3-940793-02-7	**STG Decomposition: Avoiding Irreducible CSC Conflicts by Internal Communication**	Dominic Wist, Ralf Wollowski
19	978-3-939469-95-7	**A quantitative evaluation of the enhanced Topic-based Vector Space Model**	Artem Polyvyanyy, Dominik Kuropka
18	978-3-939469-58-2	**Proceedings of the Fall 2006 Workshop of the HPI Research School on Service-Oriented Systems Engineering**	Benjamin Hagedorn, Michael Schöbel, Matthias Uflacker, Flavius Copaciu, Nikola Milanovic
17	3-939469-52-1 / 978-3-939469-52-0	**Visualizing Movement Dynamics in Virtual Urban Environments**	Marc Nienhaus, Bruce Gooch, Jürgen Döllner
16	3-939469-35-1 / 978-3-939469-35-3	**Fundamentals of Service-Oriented Engineering**	Andreas Polze, Stefan Hüttenrauch, Uwe Kylau, Martin Grund, Tobias Queck, Anna Ploskonos, Torben Schreiter, Martin Breest, Sören Haubrock, Paul Bouché
15	3-939469-34-3 / 978-3-939469-34-6	**Concepts and Technology of SAP Web Application Server and Service Oriented Architecture Products** (noch nicht erschienen)	Bernhard Gröne, Peter Tabeling, Konrad Hübner

ISBN 978-3-86956-036-6
ISSN 1613-5652