Proceedings of the 6th Ph.D. Retreat of the HPI Research School on Service-oriented Systems Engineering

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

Technische Berichte Nr. 76

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 | 76

Proceedings of the 6th 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
Patrick Baudisch

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.de/ abrufbar.

Universitätsverlag Potsdam 2013
http://verlag.ub.uni-potsdam.de/

Am Neuen Palais 10, 14469 Potsdam
Tel.: +49 (0)331 977 2533 / Fax: 2292
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.

ISSN (print) 1613-5652
ISSN (online) 2191-1665

Online veröffentlicht auf dem Publikationsserver der Universität Potsdam
URL http://pub.ub.uni-potsdam.de/volltexte/2013/6677/
URN urn:nbn:de:kobv:517-opus-66777
http://nbn-resolving.de/urn:nbn:de:kobv:517-opus-66777

Zugleich gedruckt erschienen im Universitätsverlag Potsdam:
ISBN 978-3-86956-256-8

Contents

Synonym Discovery in RDF Data

Ziawasch Abedjan
Information Systems Group
Hasso-Plattner-Institut
ziawasch.abedjan@hpi.uni-potsdam.de

Linked Open Data brings new challenges and opportunities for the data mining community. Its underlying data model RDF is heterogeneous and contains machine readable semantic relations. The amount of available open data requires profiling and integration for desired applications. One of the promising underlying techniques is association rule mining. However there has been only limited application of association rules on semantic web data. We introduce the concept of mining configurations that allows us to mine RDF data on statement level. We described elaborated use cases such as ontology engineering, data imputation that are based on configurations in the context of RDF subjects. A novel application that is based on mining configurations is synonym discovery. Synonym discovery is useful for discovering globally valid synonyms for a thesauros as well supporting the completeness of SPARQL query results by including results that are connected to synonym predicates. We show that synonym discovery in RDF data can be done efficiently using the proposed techniques based on association rule mining.

1 Synonyms in LOD

The increasing amount of Linked Open Data (LOD) in the World Wide Web raises new opportunities and challenges for the data mining community [13]. LOD is often represented in the Resource Description Framework (RDF) data model: Data is represented by a triple structure consisting of a subject, a predicate, and an object (SPO). Each triple represents a statement/fact. Fig.1 illustrates some RDF facts describing Barack Obama. When processing RDF data, meta information such as ontological structures and exact range definitions of predicates are desirable and ideally provided by a knowledge base. However in the context of LOD, knowledge bases are usually incomplete or simply not available. When dealing with an LOD source without a satisfiable knowledge base, it is useful to automatically generate meta information, such as ontological dependencies, range definitions, and topical associations of resources. These metadata faciliate the understanding and integration of the data source. Data mining applications, such as association rule mining and frequency analysis are obvious approaches to create such metadata.

The URI representation of subjects, predicates, and objects and their connections within statements harbor many hidden relations that might lead to new insights about the data and its domain. Resources are connected with each other through multiple predicates, co-occurring in multiple relations. At this point, frequencies and co-occurrences of statement elements become an interesting object of investigation.

subject	http://dbpedia.org/resource/Barack_Obama
predicate	http://dbpedia.org/ontology/birthDate
object	1961-08-04
subject	http://dbpedia.org/resource/Barack_Obama
predicate	http://dbpedia.org/ontology/birthPlace
object	http://dbpedia.org/resource/Honolulu
subject	http://dbpedia.org/resource/Barack_Obama
predicate	http://dbpedia.org/ontology/orderInOffice
object	President of the United States.
subject	http://rdf.freebase.com/ns/en.barack_obama
predicate	http://rdf.freeb.../celebrities.friendship.friend
object	http://rdf.freebase.com/ns/en.oprah_winfrey

Table 1: RDF triples from DBpedia and Freebase

One sort of meta data that can be derived by applying association rule mining on RDF data is synonymic relations. Synonym discovery might first of all be interesting for the general purpose of enriching an existing synonym thesaurus using new synonyms that have been evolving through the time as multiple people use different terms for describing the same phenomenon. Current research on synonym discovery is mostly based on web mining techniques [5, 19]. We elaborate the task of synonym discovery in LOD sources. Linked Open Data is semi structured. Therefore synonym candidate terms are easy to extract and easier to compare with regard to their contextual occurrence. Note, synonym discovery in unstructured data such as web documents needs to consider natural language processing rules. In addition to the general purpose of creating a synonym database, the discovery of synonym predicates benefits the usage of LOD. As we mentioned before for many data sources meta-data is only poorly provided. Identifying synonymly used predicates can support the evaluation and improvement of the underlying ontology and schema definitions. Furthermore the quality of query results on RDF corpora can be improved. When a user is looking for artists of a movie and uses the predicate `artist`, the knowledge base could also provide him with artists that are connected to movies by the predicate `starring`. Usage of global synonym databases is not sufficient and might lead to misleading facts in this scenario because of the heterogeneity of LOD, as predicates are used in different knowledge bases for different purposes by different data publishers. So it is necessary to have a data-driven approach dissolving the existing synonym dependencies. We introduce an approach that is based on aggregating positive and negative association rules at statement level based on the concept of mining configurations. A configuration specifies the context of rule mining (the transaction identifiers) and the target of rule mining (the items and transactions) within a triple. As a proof-of-concept we applied our algorithm several LOD sources including the popular DBpedia data set [6].

The rest of this paper is organized as follows: In the next section we present related work with regard to synonym discovery and schema matching. Next we present necessary foundations with regard to RDF and association rules. In Section 4 we describe our algorithm. Section 5 contains our evaluation plan and some results and we conclude in Section 6

2 Related Work

In this paper, we apply existing data mining algorithms on the new domain of LOD and propose a resulting synonym discovery approach. Therefore, we present related work with regard to data mining in the semantic web as well as existing applications in the fields of synonym discovery. As most of our techniques for synonym discovery derive from schema matching approaches, we also give an overview of existing schema matching approaches.

2.1 Mining the Semantic Web

There is already much work on mining the semantic web in the fields of inductive logic programming and approaches that make use of the description logic of a knowledge base [14]. Those approaches concentrate on mining answer-sets of queries towards a knowledge base. Based on a general reference concept, additional logical relations are considered for refining the entries in a answer-set. This approach depends on a clean ontological knowledge base, which is usually not available. Furthermore, that approach ignores the interesting opportunities of mining of rules among predicates.

As RDF data spans a graph of resources connected by predicates as edges, another related field of research is mining frequent subgraphs or subtrees [15]. However, in LOD no two different nodes in an RDF graph have the same URI. Therefore, frequency analysis cannot be performed unless we assume duplicate entries in the data set. But if we consider the corresponding type of each URI pattern analysis can be performed because multiple URIs belong to the same type. Thus, any graph mining would be restricted to type mining and not data mining.

Among profiling tools, ProLOD is a tool for profiling LOD, which includes association rule mining on predicates for the purpose of schema analysis [7]. The method of mining association rules on predicates is also applied in our work, however we go further than just analyzing the schema und show concrete applications that are based on this method and show how it can be combined to rule mining scenarios that also involve the objects of RDF statements.

2.2 Synonym Discovery

Most existing work for discovering synonyms is based on different language processing and information retrieval techniques. A commmon approach is to look for co-occurence of synonym candidates in web documents [5, 19]. The idea behind this approach is that synonymous word co-occur in documents. So they calculate the ratio of real co-occurrence of two terms and the independent occurrence of each term. Note that for these approaches there are already known candidate pairs that have to be validated. In our scenario this assumption does not hold as we also have to retrieve the candidate pairs.

While Baronis work concentrates on globally valid synonyms the authors of [19] address context sensitive synonym discovery by looking at co-clicked query results.

Whenever the distance between two clusters of clicked query results is below a certain threshold, the query terms can be seen as synonyms.

The approaches so far are very different from our domain where we want to discover synonym schema elements in semi-structured data. An approach that has a similar characteristic is the synonym discovery approach based on eextraced webtables [9]. The authors introduce a metric that enables to discover synonyms among table attributes. However their approach is quite restrictive as they assume a context attribute given for making attributes comparable. Furthermore they ignore instance-based techniques as they only process extracted table schemata.

2.3 Schema Matching

Schema matching differs from synonym discovery within schemata in the sense, that two schema elements may be synonyms but still may not share a remarkable number of values. On the other hands two attributes may share a lot of values but their corresponding labels may not be synonyms from a global point of view. Still approaches for the discovery of attribute matches and synonyms follow similar intuitions [18]. According to the classification of Rahm and Bernstein, we would classify our approach as mixture of an instance-based and a schema level matching algorithms. On schema level we apply existing techniques to RDF data and evaluate their effectivity.

Existing instance-based approaches are different from our work as they compare the content of each attribute column-wise [10,11,17]. Chosing features for matching is cumbersome and algorithms that look for value overlaps lack efficiency. We propose an association rule based approach that discoveres overlaps between attribute values in an RDF corpus.

One could also perform schema matching on schema element level by using dictionaries but existing work already neglects the fact it works well in real data scenarios [16]. Furthermore as RDF data does not always conform to dictionaries and contains abbreviations and domain-specific labels as predicates makes it even more difficult to discover similarities on this level.

3 Association Rules and Triples

We briefly define the relevant foundations of association rule mining on RDF data.

3.1 Association Rule Mining

The concept of association rules has been widely studied in the context of market basket analysis [3], however the formal definition is not restricted by any domain: Given a set of items $I = \{i_1, i_2, \dots, i_m\}$, an association rule is an implication $X \rightarrow Y$ consisting of the *itemsets* $X, Y \subset I$ with $X \cap Y = \emptyset$. Given a set of transactions $T = \{t | t \subseteq I\}$, association rule mining aims at discovering rules holding two thresholds: minimum support and minimum confidence.

Support s of a rule $X \rightarrow Y$ denotes the fraction of transactions in T that include the union of the *antecedent* (left-hand side itemset X) and *consequent* (right-hand side itemset Y) of the rule, i.e., $s\%$ of the transactions in T contain $X \cup Y$. The confidence c of a rule denotes the statistical dependency of the *consequent* of a rule from the *antecedent*. The rule $X \rightarrow Y$ has confidence c if $c\%$ of the transactions T that contain X also contain Y. There are also other metrics for evaluating rules, such as lift and conviction [8], but for our approach support and confidence as the basic metrics are sufficient. Algorithms to generate association rules decompose the problem into two separate steps: (1) Discover all frequent itemsets, i.e., itemsets that hold minimal support. (2) For each frequent itemset a generate rules of the form $l \rightarrow a - l$ with $l \subset a$, and check the confidence of the rule.

While the second step of the algorithm is straightforward, the first step marks the bottleneck of any algorithm. The three best known approaches to this problem are Apriori [4], FP-Growth [12], and Eclat [20]. We use the FP-Grwoth algorithm for our paper.

3.2 Association Rules on RDF Statements

To apply association rule mining to RDF data, it is necessary to identify the respective item set I as well as the transaction base T and its transactions. Our mining approach is based on the subject-predicate-object (SPO) view of RDF data as briefly introduced in [2]. Table 2 illustrates some SPO facts extracted from DBpedia. For legibility, we omit the complete URI representations of the resources and just give the human-readable values. Any part of the SPO statement can be regarded as a *context*, which is used for grouping one of the two remaining parts of the statement as the *target* for mining. So, a transaction is a set of target elements associated with one context element that represents the transaction id (TID). We call each of those *context* and *target* combinations a *configuration*. Table 3 shows an overview of the six possible configurations and their preliminarily identified use-cases. Each can be further constrained to derive more refined configurations. For instance, the subjects may be constrained to be of type Person, as happens to be the case in our example.

Subject	Predicate	Object
B. Obama	birthPlace	Hawaii
B. Obama	party	Democratic Party
B. Obama	orderInOffice	President of the US
A. Merkel	birthPlace	Hamburg
A. Merkel	orderInOffice	Chancellor of GFR
A. Merkel	party	CDU
J. Lennon	birthPlace	Liverpool
J. Lennon	instrument	Guitar

Table 2: Facts in SPO structure from DBpedia

The application of Configuration 1 from Tab. 3 to our example data set would transform the facts into three transactions, one for each distinct subject as illustrated in Tab. 4. In this example, the itemset *{birthPlace, party, orderInOffice}* is a frequent item-

Conf.	**Context**	**Target**	**Use case**
1	Subject	Predicate	Schema discovery
2	Subject	Object	Basket analysis
3	Predicate	Subject	Clustering
4	Predicate	Object	Range discovery
5	Object	Subject	Topical clustering
6	Object	Predicate	Schema matching

Table 3: Six configurations of context and target

set (support 66.7%), implying rules, such as *birthPlace* $\rightarrow$ *orderInOffice, party* and *orderInOffice* $\rightarrow$ *birthPlace, party* with 66.7% and 100% confidence, respectively.

TID	transaction
B. Obama	*{birthPlace, party, orderInOffice}*
A. Merkel	*{birthPlace, party, orderInOffice}*
J. Lennon	*{birthPlace, instrument}*

Table 4: Context: Subject, Target: Predicate

TID	transaction
birthPlace	*{B. Obama, A. Merkel, J. Lennon}*
party	*{B. Obama, A. Merkel}*
orderInOffice	*{B. Obama, A. Merkel}*
instrument	*{J. Lennon}*

Table 5: Context: Predicate, Target: Subject

The reverse configuration (Conf. 3 in Tab. 3) in the context of predicates would create the transactions presented in Tab. 5. The frequent itemsets here contain subjects that represent similar real world objects, such as heads of government *{B. Obama, A. Merkel}*. The analysis of each configuration requires the investigation of its capabilities, technical challenges, and use cases. In this paper, we concentrate on Configurations 1 and 6, which have both predicates as their targets.

4 Synonym Discovery

We introduce three basic strategies that we combine for synonym discovery. Our approach makes direct usage of the configurations 1 and 6 from Tab. 3. These configurations benefit two major intuitions. In configuration 1 we do schema analysis in the context of subjects. Configuration 6 enables to mine similar predicates in the context of objects. Additionally we look into range structure of predicates by looking at value type distributions. All three applications are derived from existing schema matching scenarios with schema-level and instance-based strategies.

4.1 Schema Analysis

Configuration 1 enables us to do frequency analysis and rule discovery per entities. For instance positive rules between predicates can be used for re-validating existing ontologies [1]. In our use case we have a different intuition. Synonym predicates should not co-occur for entities. It is more likely for entities to include only one representative of a synonymous predicate group within their schema. That is why we look for negative correlations in Configuration 1. For this purpose we adapted an FP-Growth [12] implementation that retrieves all negative correlations for a set of synonym candidate pairs. The approach can also be used stand-aloe looking at all possible pairs that have a negative correlation in the data set. Negative correlation can be expressed by several score functions. One could look at the bidirectional correlation coefficient or consider some kind of aggregations of the negative rules' confidence values such as minimum, maximum or the f-measure of both.

Bare schema analysis leads also to results including pairs such as `birthDate` and `author` as both occur for different entities. So a negative correlation is not a sufficient condition. The context or the range of the predicates should also be taken into account. In [9] the authors introduce an approach for synonym discovery in different webtables that includes a context attribute. We also adapted this score function and compared the results to the scoring functions named before. In the following we describe our strategies that complement the schema analysis by considering also the range of predicates.

4.2 Range Content filtering

The first intuition is that as synonym predicates will have a similar meaning they also share a similar range of object values. Configuration 6 constitutes a mining scenario where each transaction is defined by a distinct object value. So each transaction consists of all predicates containing the distinct object value in there range. Frequent patterns in this configuration are sets of predicates that share a significant number of object values in their range. As each Configuration is an adaption of frequent itemset mining the threshold that decides whether two predicates are similar or not is support and depends on the number of all baskets or all existing distinct objects. Normally when trying to compute the value overlap between two predicates one would look at the ratio of overlaps depending on the total number of values of such a predicate. Furthermore our approach ignores value overlaps that occur because of multiple occurrence of one distinct value in the ranges. We will analyze the effect of these differences and show that our approach is much more efficient without any loss of quality. We further restrict the configuration to discover only frequent predicate pairs that will be regarded as synonym candidate pairs in the schema analysis or range structure filtering phase. Similar to the schema analysis strategy also the range content filtering based on value overlaps is not a sufficient condition for discovering synonyms. For example the predicates `birthPlace` and `deathPlace` will share a remarkable percentage of their ranges but are obviously no synonyms. However this candidate pair can be pruned looking at their exclusion rate per entity during schema analysis.

4.3 Range Structure Filtering

In some scenarios value range content filtering might not be the most appropriate technique as it requires two synonym predicates to share a portion of exactly equal values. However, real world data might contain synonym predicates with completely disjoint range sets where the range elements are only ontologically similar. This is often the case when looking at predicates describing numbers and dates. Therefore existing work not only looks at exact overlaps but also on general string or token characteristics such as string length and character distributions [10, 17]. However as the motivation of our work was to analyze the capabilities of mining on statement level we do not approach this problem on string level. Instead we look at type distributions in predicate ranges. So for every object in the range of a predicate we retrieve its type from the graph and create type vectors per predicate containing the number of the occurences of this type. As each entity in RDF might have several types due to existing type hierarchies, i.e., Barack Obama is a Politician as well as a Person, we examined two different vector constructions. The first construction just retrieves all types per entity in a predicate range and the second construction only considers the most specific type of an entity. Having type vectors for a predicate pair both can be compared using measures such as cosine similarity or weighted jaccard similarity. After preliminary experiments weighted jaccard similarity seems more promising because cosine similarity results into high scores as soon as one component value of one vector is very large while all other components have very small values. Missing type values, e.g., in case of dates and other numerical values, have been handled as unknown types, whereas no two unknown types are equal.

4.4 Combined Approach

We have introduced three different ways of generating or evaluating synonym candidate pairs. It is crucial to find a reasonable order for combining those three to make best use of the intuitions and achieve optimal quality and to be efficient at the same time. One strategy is to first retrieve all predicate pairs through range content filtering filter those pairs by range structure filtering and then analyzing their schema co-occurrences. This strategy has two advantages: as retrieving negative correlations and type vectors is inefficient, it is reasonable to perform both on given candidates instead of using them on the complete data set to retrieve candidates. Furthermore as we compare very different correlation measures on schema level all of them can be performed at the end of the discovery process after having done the instance-based approaches first.

5 Evaluation Plan

We will evaluate the synonym discovery algorithm on different levels. First of all, we will compare all measures on each level of our synonym discovery approach. Further we will compare our three strategies each as a stand alone approach to another and identify weaknesses and strengths in our use case. Finally, we will evaluate the algorithm with regard to our claims that are:

- There are synonym predicates in RDF data sets that can be useful for optimizing queries.
- Configuration 6 is as effective as a naive overlap approach but more efficient.

6 Summary and Future Work

We proposed a statement-level mining methodology for RDF data that is based on six basic configurations with the aid of different real examples. On this basis, further research directions include reasoning and formalizing constraints and refinements that allow unlimited configuration scenarios for different purposes. We use this methodology for creating a syonym discovery algorithm that identifies synonymous predicate pairs in RDF data sets. All described approaches have been implemented and the next step is to evaluate the claims. Further study aims at combining the first and second Configuration as a pre-processing step for advancing frequent graph mining in RDF data.

References

[1] Ziawasch Abedjan, Johannes Lorey, and Felix Naumann. Advancing the Discovery of Unique Column Combinations. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, Maui, Hawaii, October 2012.

[2] Ziawasch Abedjan and Felix Naumann. Context and target configurations for mining RDF data. In *Proceedings of the International Workshop on Search and Mining Entity-Relationship Data (SMER)*, Glasgow, 2011.

[3] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 207–216, Washington, D.C., USA, 1993. ACM.

[4] Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 487–499, Santiago de Chile, Chile, 1994.

[5] Marco Baroni and Sabrina Bisi. Using cooccurrence statistics and the web to discover synonyms in technical language. In *International Conference on Language Resources and Evaluation*, pages 1725–1728, 2004.

[6] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia - A crystallization point for the Web of Data. *Journal of Web Semantics (JWS)*, 7:154–165, September 2009.

[7] Christoph Böhm, Felix Naumann, Ziawasch Abedjan, Dandy Fenz, Toni Grütze, Daniel Hefenbrock, Matthias Pohl, and David Sonnabend. Profiling linked open data with ProLOD. In *Proceedings of the International Workshop on New Trends in Information Integration (NTII)*, pages 175–178, 2010.

[8] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, Proceedings of the ACM International Conference on Management of Data (SIGMOD), pages 255–264, New York, NY, USA, 1997. ACM.

[9] Michael J. Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. WebTables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment*, 1:538–549, August 2008.

[10] AnHai Doan, Pedro Domingos, and Alon Y. Halevy. Reconciling schemas of disparate data sources: a machine-learning approach. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 509–520, New York, NY, 2001.

[11] Georg Gottlob and Pierre Senellart. Schema mapping discovery from data instances. *Journal of the ACM*, 57(2):6:1–6:37, 2010.

[12] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 1–12, 2000.

[13] Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan Claypool Publishers, 2011.

[14] Joanna Józefowska, Agnieszka Lawrynowicz, and Tomasz Lukaszewski. The role of semantics in mining frequent patterns from knowledge bases in description logics with rules. *Theory Pract. Log. Program.*, 10:251–289, 2010.

[15] Michihiro Kuramochi and George Karypis. Frequent subgraph discovery. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 313–320, Washington, D.C., 2001.

[16] Wen-Syan Li and Chris Clifton. Semint: A tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data and Knowledge Engineering (DKE)*, 33(1):49 – 84, 2000.

[17] Felix Naumann, Ching-Tien Ho, Xuqing Tian, Laura M. Haas, and Nimrod Megiddo. Attribute classification using feature analysis. In *Proceedings of the International Conference on Data Engineering (ICDE)*, page 271, 2002.

[18] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, December 2001.

[19] Xing Wei, Fuchun Peng, Huihsin Tseng, Yumao Lu, and Benoit Dumoulin. Context sensitive synonym discovery for web search queries. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 1585–1588, New York, NY, USA, 2009.

[20] Mohammed J. Zaki. Scalable Algorithms for Association Mining. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 12:372–390, May 2000.

Methodologies for Goal-Oriented Process Performance Management

Evellin Cardoso

Business Process Technology Group
Hasso Plattner Institute
evellin.cardoso@hpi.uni-potsdam.de

Although modeling goals provide a more strategic view for business processes; little attention is devoted to explicitly modeling goals as well as using the concept of goal to increase the value of the process modeling techniques. The purpose of this report is to enhance the current BPM lifecycle with new activities in order to support goal-oriented process performance management. In order to so, we propose new methodological steps within the BPM lifecycle that must be done in order to support such goal-orientation and how the existent methodologies can support the execution of these new activities.

1 Introduction

The increasing competitiveness drives organizations to constantly evaluate their position in the market and promote changes in an attempt to improve the quality of the services and products they offer. In recent years, many of the efforts related to the management of organizations have been conducted in the scope of Business Process Management (BPM) activities. BPM is based on the fact that each product or service that a company provides to the market is the outcome of a process. As a consequence, changes in processes should generate radical improvements in critical performance measures (such as cost and quality).

However, predicting how an environment should respond to changes by simply adopting a process-centered view is unfeasible since there are a large number of issues to be considered, such as infrastructure, power and politics, managerial control, organizational culture, among others. Given this multitude of issues, understanding an organizational setting often requires a number of perspectives or domains.

Among these perspectives, the domain of "motivation" has been recognized as an important element of organizations as highlighted in Zachman framework's motivation column. Goal modeling is the artifact employed for capturing the motivational aspect and strategies behind the organizational practices. Moreover, by adopting goal modeling, the company can systematically express the choices behind multiple alternatives and explore new possible configurations for an organizational setting. This is essential for business improvement once changes in the company's goals have significant consequences within all domains of the organization. However, although modelling goals provide a more strategic view for business processes; little attention is devoted to explicitly modelling goals as well as using the concept of goal to increase the value of the process modelling techniques. Moreover, few approaches in *process performance management* and *business process management* address the problem of monitoring processes in order to quantify the progress towards strategic goals, rather taking an operational view for process performance.

The purpose of this report is to situate our goal-oriented approach within the current BPM lifecycle and how it enhances the current state of art. In order to so, we describe the current state of art in each of the phases of the BPM lifecycle in terms of languages and methodologies and how we enhance some phases of this lifecycle. The rest of this paper is organized as follows: section 2 describes the state of art in the current BPM lifecycle (in terms of activities, languages and methodologies that support a goal-oriented view) and how we intend to enhance this lifecycle with this thesis and section 3 concludes the paper with directions for future research.

2 General Overview of a Goal-Oriented Methodology for Process Performance Management

In order to obtain the benefits from adopting the alignment between goal-related concepts and the business processes in enterprise architectures, researchers must rely on two main components [1]. The first one is a metamodel for defining the modeling language for goals and business process models, i.e., a metamodel for defining the modeling constructs syntax, semantics and graphical notation used to create models. The second component corresponds to a systematic methodology for creating aligned models. During this process different ways of working are applied in order to elicit and develop the knowledge of business stakeholders or domain experts.

Regarding the language component, we have already provided a framework [8] with the correspondent concepts for modeling goals, indicators and business processes. In relation to the methodology component, this report intends to enhance the current BPM lifecycle with the correspondent activities to perform a goal-oriented analysis of business processes. The current BPM lifecycle required to manage operational business processes [28] [26] [25] consists of four major phases, namely: business process strategy, business process design, business process implementation and business process controlling. The addition of strategic concepts in the BPM lifecycle in fact do not require the creation of additional phases, but instead, only requires the introduction of new activities within the existent phases.

In the remainder, we have two sections, namely business process strategy and business process controlling, each one corresponding to the phases of the BPM lifecycle that we contribute. For each section, we present a current overview of the existent state of art (in terms of activities, languages and methodologies) with their open points for contribution and how we already contributed with our framework. In the following, we enumerate how our framework addresses some open points in terms of language and the methodologies that we have obtained so far.

2.1 Business Process Strategy Phase

The **Business Process Strategy phase** forms the foundation for aligning business processes with general corporate strategy [27], establishing the organizational prerequisites for the project that implement business processes management initiatives. Once the long-term strategies are set up, projects can be established to implement them [28]. In this phase, the strategies are captured as organizational requirements that support the designing and management of enterprise architectures [22]. Further, the numerical quantification in which extent goals have been achieved is also addressed in this phase by providing a set of metrics and Key Performance Indicators (KPI) that are used to measure organizational/processes performance in later phases [28]. Currently, these goals and strategies are established in an ad hoc manner, but no explicit knowledge about goals and strategies are currently captured in some artifact like a model. We divide the approaches for addressing the performance within business processes into two groups:

2.1.1 Approaches for Goal Modeling

These approaches are concerned about providing languages and methods for goal modeling. Approaches:

Enterprise Architecture Methods (Business Requirement Modelling). Enterprise modelling approaches structure an enterprise architecture in terms of various related architectural domains or viewpoints which focus on specific aspects of the enterprise like business processes, information systems that support organizational activities, organizational structures and so forth. Among the various architectural domains, the domain of "motivation" has been recognized as an important element of enterprise architectures. Within these approaches, goal modelling is the artefact employed for capturing the motivational aspect and strategies behind the organizational practices, helping in clarifying interests and intentions from different stakeholders. Among the approaches, architectural methods are the only methodologies that explicitly model goals and strategies in the format of artefacts (goal models). The explicit modelling of such artefacts enables the development of automated and model-based techniques to analyze business goals. However, in terms of languages for goal modeling, there is a little support in the current literature, as evidenced in [2]. Among these languages, the most expressive as argued in [22] is the so called Motivational ArchiMate Extension or ARMOR language. The language enables one to model business goals and requirements for enterprise architectures, but do support KPI modeling. Concerning methodologies, an open point in literature is how to identify, elicit and model business goals in the context of enterprise architectures, since any papers concerning this topic have been found.

Our contributions here are twofold:

- Language: We proposed a framework [8] that provides the concept of goals to capture the organizational goals as in the ArchiMate Motivational Extension. The contribution here is that we include some goal attributes that are important to characterize goals and exclude other concepts whose practical application is not clear according to [7]. Furthermore, the language is not precise with respect to the semantics for one to decide if goals are achieved or not. A precise relation with the concepts of Enterprise Architecture would be sufficient to assign this semantics. However, the only current relation of the goal language with the enterprise architecture is the realization (means-end relationship) between a Goal and the Services/Processes. We solve this problem creating a relation between a Goals and a KPI, that in its turn, the KPIs serve to measure in which extent the goals are achieved.

- Methodology: we propose a methodology for goal modeling here based in the attributes that must characterize goals and how to measure their achievement.

2.1.2 Approaches for KPI Modeling

These approaches are concerned about providing languages and methods for KPI modeling. It is also important to highlight that indicators must be derived from the organizational goals to monitor the organizational performance in this phase. Concerning the creation of process indicators, they may be set up in this phase (through a refinement of the organizational indicators) or it can be postponed until the business process design phase where process indicators are modeled together with the business process (and subsequently aligned with the organizational indicators). Approaches:

Business Activity Monitoring approaches (BAM) [11,15]. BAM approaches are concerned about determining process performance by monitoring indicators values in a real-time fashion. For this reason, there are several approaches that provide a language for KPI modeling [11] [6] [3] [4] (in order of expressiveness of modeling language), but no methodology standpoint to elicit and derive KPIs on the basis of existent ones.

Performance Measurement Systems [9,18,21,23]. Performance Measurement (PM) systems consist of a number of indicators (and their relationships) that are used to describe and evaluate the organizational performance. These approaches arise from several areas, mainly in management sciences, such as strategy management, operations management, human resources, organizational behavior, information systems, marketing, and management accounting and control [9] and for this reason, they do not provide languages for KPI modeling, but instead, methodologies for the construction of PM systems. Basically speaking, there are two types of methodologies for developing PM systems [27]: (1) develop the indicator system based on existing generic indicators and/or complete PM systems libraries and (2) develop the indicator system selecting the appropriate indicators for the company on the basis of its business objectives and success factors, resulting in a list that it is company-specific.

In the first type, several PM libraries for the development of PM systems within a company-wide scope are available, such as the best-known Balanced Scorecard [14], Performance Prism [20], SMART System/Performance Pyramid [5]. These libraries have a number of indicators and come with a methodology perspective associated so that who use the library also is able to derive indicators for a specific organization, adapting the questions suggested in the library to some specific organizational context. The challenge consists in selecting the appropriate indicators from an extensive list of potential indicators. Furthermore, other disadvantage that can be mentioned is the fact that these libraries aim at providing indicators for determining the performance of the entire company or organization units (the indicators are very high-level and related with several business process at the same time), in opposition to the evaluation of individual business processes [27] [16].

The second type of methodologies is based on the evidence that it seems very unlikely that a universal set of performance indicators can be applied successfully to all business processes. For this reason, some papers propose methodologies for derivation of process-specific indicators on the basis of goals. In contrast to the use of libraries, these methodologies (i) develop PM systems to evaluate the performance of business processes (in opposition to measure the performance of entire corporations or organizational units) and (ii) are related to the goals of each business process that are a good starting point for gathering the right indicators [16] [27] [10]. However, while there is a plethora of libraries for business indicators and methodologies for derivation of process-specific indicators, only very few approaches propose a modeling method for indicators [10].

Our contribution here is:

- Creation of modeling methodology for the development of process-indicator system based on the previously modeled goals. In relation to the existing methods, we may highlight that our method adds value because: (i) present a modeling method (just one related methodology does [10]) (ii) concentrate in the derivation of process-specific indicators instead of in organizational ones like literature in PM systems (iii) assume goals models before deriving indicators in opposition to use strategic goals in a ad-hoc manner as the proposals in current literature.

2.2 Business Process Controlling Phase

Once the implementation of measurements has been defined in the previous phase, the **Business Process Controlling phase** enables the obtainment of insights about qualitative and quantitative measures, revealing areas with potential for improvement. These insights can be motivated by either *performance* or *compliance* considerations [27]. From a *performance perspective*, the intent of process analytics is to shorten the reaction time of decision makers to events that may affect changes in the process performance, and to allow a more immediate evaluation of the impact of process management decisions on process metrics. From a *compliance perspective*, the intent of process analytics is to establish the adherence of process execution with governing rules and regulations, and to ensure that contractual obligations and quality of service agreements are met. From a compliance perspective, although there is a body of knowledge that addresses the enforcement of "compliance goals" during the design/execution time of business processes; our focus is concentrated on performance issues within business processes. We divide the approaches for addressing the performance within business processes into two groups:

2.2.1 Approaches for *guiding the (re)design*

These approaches are concerned about guiding the (re)design of processes so that they contain only activities that generate value for the organization (i.e., they address the structural properties of business processes). Approaches:

Business process reengineering. Constitutes an area which is closely related to optimizing business process quality. Reengineering approaches commonly comprise recommended best practices and other informal methods like "classic" reengineering view [13] which are mostly based on anecdotal evidence. This view is also reflected in the OMG Business Process Maturity Model [12] which suggest criteria to allocate business processes to maturity levels without giving clear evidence on how this structure is devised. While this informal character fits well with practical applicability, we still lack an overarching comprehensive model to ensure causal relations between measures recommended and intended results as well as completeness of coverage of quality aspects [19].

Architectural Analysis (Enterprise Architecture Methods) [17]. Functional analysis is performed to gain insight into the functional aspects of an architecture, illustrating the dynamic behavior of a system. Among others, it is used to understand how a system that conforms to an architecture works, to find the impact of a change on an architecture, or to validate the correctness of an architecture.

2.2.2 Approaches for *evaluating the operating process*

Once the business process has already structurally designed e evaluated accordingly, the operating quality of the executing business process must be evaluated. In that respect, there are three types of analysis that can be made on the basis of the execution of business processes [27]: to evaluate what happened in the past (past analysis), to detect what is happening at the moment (real-time analysis) and to predict what may happen in the future (predictive analysis):

Past analysis. Process Controlling [30] focuses on post analysis of completed processes that may or may not be based on a pre-existing formal representation of the business process. Process mining techniques [24] inductively discover the process model when no explicit process model exists,

Real-time analysis. Business Activity Monitoring (BAM) [15, 29] is concerned about real-time monitoring of currently active business process,

Predictive analysis. Process Intelligence [3, 11] uses business process data to forecast the future behavior of the organization through techniques such as scenario planning and simulation. Quantitative simulation in enterprise architecture methods [17] is used to make statistical statements about the quantitative measures of a system based on multiple simulation runs. It can be seen as performing measurements in a given model, enabling the examination of performance measures in a model in a specific situation. Further, Analytical techniques [17] are not statistical by nature and produce a unique and reproducible result. They are usually more suitable for quantitative analysis than quantitative simulation, usually providing architects with indications of performance measures and bottlenecks in a given process model. They are useful when a comparison of a large number of alternatives is needed in a so called "what-if" analysis.

Currently, no activities for process analysis considering explicitly modeled goals have been found. In order to perform a goal-oriented analysis, we identified three methodological steps:

- Mapping the goals to the other architectural domains (operationalization),
- Determine if the goals are being achieved in the current enterprise architecture,
- If not, one must:
  - Redesign the business processes considering structural issues like removing activities that are adding no value, decisions that are no longer required,
  - Redesign the business processes addressing performance violations like trends of negative performance in the target values of indicators.

The current surveyed methods can be used to perform these aforementioned methodological steps in the following way:

- Approaches for guiding the (re)design can be used to redesign business process considering structural issues
- Approaches for evaluating the operating process can be used to redesign the business processes addressing performance violations.

At this moment, the literature review is pointing out that the best methodologies to be adapted in order to perform a goal-oriented analysis are the techniques for enterprise architecture analysis depicted in [17]. Furthermore, we are not sure yet about the scope of the thesis, i.e., which of the aforementioned steps we aim at solving with this thesis.

3 Conclusions

This paper reported the state of art in BPM in terms of activities, languages and methodologies to support a goal-oriented BPM. Furthermore, it positions our approach for a goal-oriented methodology, by depicts the current achievements in terms of language and methodologies that we have obtained so far.

As a general conclusion, we may say that there is a huge gap between linking business strategies to the management of operational performance from a methodological point of view, as also noticed by [19]. This can be accounted by the fact that goals are stated from a management perspective, while the existent methods in BPM for evaluation of process quality are not yet fully effective from a management perspective [19].

A roadmap of this thesis can be summarized as follows:

- Contributions obtained up to this moment: proposal of a framework to tackle the language component of the problem of goal-oriented process analysis,

- Work in progress: methodology for goal elicitation and modeling and modeling methodology for process specific indicators,
- Expected future contributions: goal operationalization and goal-oriented enterprise architecture redesign method, however, the scope is not well-defined yet.

References

[1] Persson Anne Bubenko, Janis and Janis Stirna. D3 Appendix B: EKD User Guide. *Project Deliverable*, 2001.

[2] Evellin Cristine Souza Cardoso, João Paulo A. Almeida, and Renata S. S. Guizzardi. On the support for the goal domain in enterprise modelling approaches. In *EDOCW*, pages 335–344, 2010.

[3] M. Castellanos, F. Casati, U. Dayal, and M. Shan. A Comprehensive and Automated Approach to Intelligent Business Processes Execution Analysis. *Distrib. Parallel Databases*, 16:239–273, November 2004.

[4] M. Castellanos, F. Casati, M. Shan, and U. Dayal. iBOM: A Platform for Intelligent Business Operation Management. *Data Engineering, International Conference on*, 0:1084–1095, 2005.

[5] K. Cross and R. Lynch. The SMART way to define and sustain success. 1989.

[6] A. del Rio-Ortega, M. Resinas, and A. Ruiz-Cortes. Defining process performance indicators: An ontological approach. In Robert Meersman, Tharam Dillon, and Pilar Herrero, editors, *On the Move to Meaningful Internet Systems: OTM 2010*, volume 6426 of *Lecture Notes in Computer Science*, pages 555–572. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-16934-2_41.

[7] Wilco Engelsman and Roel Wieringa. Goal-oriented requirements engineering and enterprise architecture: Two case studies and some lessons learned. In Björn Regnell and Daniela Damian, editors, *Requirements Engineering: Foundation for Software Quality*, volume 7195 of *Lecture Notes in Computer Science*, pages 306–320, London, March 2012. Springer Verlag.

[8] Cardoso Evellin. A Conceptual Framework for Goal-Oriented Process Monitoring. *BPT Technical Report 12*, 2012.

[9] M. Franco-Santos, M. Kennerley, P. Micheli, V. Martinez, S. Mason, B. Marr, D. Gray, and A. Neely. Towards a definition of a business performance measurement system. *International Journal of Operations & Production Management*, 27(8):784–801, 2007.

[10] Ulrich Frank, David Heise, Heiko Kattenstroth, and Hanno Schauer. Designing and utilising business indicator systems within enterprise models-outline of a method. In *MobIS*, pages 89–105, 2008.

[11] D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, and M. Shan. Business process intelligence. *Comput. Ind.*, 53:321–343, April 2004.

[12] The Object Management Group. Business process maturity model (bpmm), 2008.

[13] James Hammer, Michael und Champy. *Reengineering the Corporation: A Manifesto for Business Revolution*. Harper Business, 2003.

[14] R. Kaplan and D. Norton. Using the Balanced Scorecard as a Strategic Management System. *Harvard Business Review*, (January-February):75–85, 1996.

[15] J. Kolar. *Business Activity Monitoring*. PhD thesis, Czech Republic, 2009.

[16] P. Kueng and A. J. W. Krahn. Building a process performance measurement system: some early experiences. *Journal of Scientific and Industrial Research*, 58:149–159, 1999.

[17] Marc Lankhorst. *Enterprise Architecture at Work: Modelling, Communication and Analysis*. Springer, 3 edition, 2012.

[18] M.J. Lebas. Performance measurement and performance management. 1995.

[19] Matthias Lohrmann and Manfred Reichert. Understanding business process quality. In *Advances in Business Process Management*. Springer, 2012.

[20] A. Neely, C. Adams, and P. Crowe. The performance prism in practice. 2001.

[21] A. Neely, M. Gregory, and K. Platts. Performance measurement system design: A literature review and research agenda. *International Journal of Operations & Production Management*, 15(4), 1995.

[22] Dick Quartel, Wilco Engelsman, Henk Jonkers, and Marten Sinderen van. A goal-oriented requirements modelling language for enterprise architecture. In *Proceedings of the IEEE International Enterprise Distributed Object Computing Conference, EDOC '09*, pages 3–13, Los Alamitos, CA, USA, 2009. IEEE Computer Society Press.

[23] S. and J. Ballantine. Performance measurement in service businesses revisited. *International Journal of Service Industry Management*, 7:6–31, 1996.

[24] Wil M. P van der Aalst. *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.

[25] W.M.P. van der Aalst, A. H. M. Ter Hofstede, and M. Weske. Business Process Management: A Survey. In *Proceedings of the 1st International Conference on Business Process Management, volume 2678 of LNCS*, pages 1–12. Springer-Verlag, 2003.

[26] Jan vom Brocke and Michael Rosemann. *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[27] Jan vom Brocke and Michael Rosemann. *Handbook on Business Process Management 2:Strategic Alignment, Governance, People and Culture*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[28] M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2 edition, 2012.

[29] B. Wetzstein, P. Leitner, F. Rosenberg, I. Brandic, S. Dustdar, and F. Leymann. Monitoring and Analyzing Influential Factors of Business Process Performance. In *2009 IEEE International Enterprise Distributed Object Computing Conference*, pages 141–150. IEEE, September 2009.

[30] Michael zur Muehlen. *Workflow-Based Process Controlling: Foundation, Design, and Application of Workflow-Driven Process Information Systems*. Logos Berlin, 2004.

Hybrid parallel computing with Java

Frank Feinbube
Frank.Feinbube@hpi.uni-potsdam.de

State-of-the-art computer systems are a combination of general purpose processors and special function accelerators. The most common type of accelerators are GPU compute devices, which are used for some years to compute a variety of data parallel tasks fast and energy efficient. Since the release of Intel's Sandy Bridge architecture and AMD's APU technology graphic compute units are integrated into each new processor. Despite their wide availability only a relatively small amount of projects makes extensive use of these hybrid systems. That applies in particular to projects that are realized in standard languages like Java. One main reason for this is the fact that it is laborious to learn to use the software libraries for hybrid architectures and that a deep understanding of the characteristics of the underlying hardware is often necessary, as well. The burden of that this creates is often too much for small and mid-sized enterprises and project teams.

In this paper we show a way to allow such projects to get access to the hidden performance of hybrid systems while keeping the effort to learn new language constructs as low as possible. Our solution supports the development of new software projects as well as the refactoring of existing software systems.

1 Introduction

The solution that we present in this paper serves as a bridge between developers using standard languages and state-of-the-art hybrid hardware that can be found in almost every computer system on the market. These two classes are described in detail in Section 1.1 and 1.2.

1.1 Target Group

Standard languages like Java and C# still have a strong popularity. [24] This is not surprising, since they enable developers to create portable programs for different domains and tasks in a time and cost efficient way. This is achieved mainly by abstraction. The virtual machine (JVM, .NET Framework, ...) allows developers to focus on their application logic without having to think about the underlying hardware and operating system. This is particularly interesting for small and medium-sized projects, because they rarely can afford experts that get into the details of the underlying systems. Developers in such projects are usually already working to their full capacity with the implementation of the business logic. Thus they want to be able to rely on a runtime library, which supports them with all tasks that they need to implement and if necessary use a few special libraries with which they have experience or that provide very fast and easy access - which means in most cases, a limited set of special features.

There is a huge code base of such Java-based projects. And ideally this code is reused in follow-up or side projects. If the code has to be changed in order to suit these projects, usually refactoring and subsequent expansion is applied. Thereby the code is converted to a more appropriate representation, which then allows the implementation of the new features. As for the rebuild of software development time and costs are important here as well.

1.2 Target Systems

Computer system architectures are currently at a crossroads. Due to the power wall, the ILP wall and the memory wall it is clear that the economic potential of traditional solutions to improve performance with new generations of processors are exhausted. [23] Nevertheless, in order to provide further resources for performance-hungry applications alternative architectures are tested, that violate well-established system features such as cache coherence or fixed memory mappings. [12] Also, accelerators are studied [7, 10] and integrated into end-user systems progressively. [3, 25, 26] Hybrid systems of accelerators and conventional general purpose processors are now available in almost all computer classes: from server systems [15] to COTS computers to netbooks [20]. The current trend in smartphones suggests that their powerful accelerators will be programmable soon, as well.

The most common type of accelerators is GPU compute devices. Many successful projects show what performance gains can be achieved with them. [4, 18] This performance potential is waiting in a myriad of systems to accelerate computationally intensive projects of all kinds.

Currently, it is necessary to have a deep understanding of the characteristics of the hardware of the accelerator and the interaction with the host system in order to take advantage of hybrid systems. But that is about to change soon ...

2 Bridging the Gap

How to bridge the gap between the target audience introduced in Section 1.1 and current hardware presented in Section 1.2 while making it as easy as possible to program new software and adapt existing software. In this chapter, the possible approaches to bring hybrid computing to developers using standard languages are presented.

2.1 Don't do anything

The first way to deal with this situation is to do nothing at all. In this case, developers rough the additional performance gain and the accelerator resources remain untapped. The advantage of this solution is clear in the minimum effort for the developer - namely none at all. The biggest disadvantage beside the untapped potential is mainly the fact that such software systems cannot expect any performance gains by future generations of processors, since these will not be faster, but more parallel instead. [23]

2.2 Let the Operating System / Virtual Machine handle it

The main task of operating systems and virtual machines is the abstraction from and standardization of access to lower system layers. Therefore, they are the ones who take care of the provision of uniform interfaces to existing hardware. Particularly for the management of execution resources and the distribution of tasks and data, the operating system and the virtual machines were responsible.

The new generations of hardware are supported already indeed and NUMA-based systems can be used also, and their run-time behavior can be improved by the use of thread pinning, for accelerators, however, this does not apply. Both GPU compute devices as well as the integrated GPU compute units of Intel's Sandy Bridge architecture and AMD's APUs are neither managed automatically by the virtual machines nor by the operating system. Initial work regarding the integration of these accelerators with the operating system exist already [21], but an actual implementation in mainstream operating systems will leave some time in coming. In contrast to this, there are already promising approaches to accelerate parts of the runtime libraries of common virtual machine. [16]

The advantage for the developers is clearly the lack of additional effort. The downside is the unpredictability of whether and how their software systems will benefit from future enhancements in the operating system and virtual machines. Short to medium term we cannot expect any performance gains this way.

2.3 Use Special Libraries

Accelerators have been designed for a specific purpose; the primary function of GPU compute devices for example, is the transformation and rendering of vertices, matrices and textures. Therefore usually there are already some libraries that include an efficient implementation of algorithms related to their specific purposes for the corresponding accelerators. They range from matrix multiplications and fast Fourier transforms to radix sort, and more. Many of these libraries are freely available and some of them even combine a variety of useful functions. [6, 11] The portfolio of libraries that provide special solutions grows daily.

From the perspective of a developer these libraries are especially interesting because the cost of developing and maintaining the functionality is being taken care of by others. On the other hand, however, one needs to become acquainted with these libraries and meet the conditions of open source licenses where necessary. Moreover, developers using special libraries become dependent on them; if such a special library is no longer maintained, it could negatively affect their software project. Also, as a user of such a library one usually has only limited influence on its further development direction.

2.4 Dig into Details

If developers want to have full control of the implementation of an algorithm essential for their project or want to implement their own, accelerated algorithms not yet provided by others, they need to familiarize themselves with the details of the accelerators hardware. This demands the use of C/C++ and low level APIs. [17, 19] In any case, it is necessary to know the manufacturer-specific and version-specific features of the hardware in the system and to take into account. [8] This is often realized by shipping a variety of specific implementations with the program - one implementation for each hardware generation / configuration of each manufacturer. The appropriate implementation is selected, compiled and executed at runtime.

The advantage of this approach is the full control of every detail of the implementation and the associated possibility of achieving the best possible performance. However, this approach also results in the greatest effort. In general, this approach is, therefore, impractical for small and medium projects.

2.5 Use Abstractions

There are already some APIs that promise abstractions and easier access indeed, [1,5] but in order to achieve acceptable performance, it is still necessary to be aware of the execution behavior and the memory hierarchy of the accelerators and evaluate specific best practices. [9] Still it is important to be aware of the specific characteristics of differing manufacturers, hardware generations, and the configuration of the accelerators that are available in the system where the program is running and to respond to this in code accordingly at runtime.

For developers, these APIs are very promising because they allow integrating the accelerator-specific algorithms seamlessly into their software and reducing the coding overhead that is required to use of low-level APIs. Unfortunately, it is still often necessary to formulate the code that is executed on the accelerator in a C/C++ derivative. [5, 22]

Our solution that overcomes these disadvantages and combines the advantages of abstractions with the advantages of special libraries is presented below.

3 HyFor.parallel - Hybrid Loops in Java

Although there are already technologies that make it possible to use accelerators, there is still no solution, which allows an efficient and convenient use of hybrid systems. Our solution fills this gap.

3.1 Requirements

From the approaches presented in Section 2, as well as, the experiences of other projects [13] we conclude that the following properties are relevant for acceptable, useful solutions:

1. The effort (lines of code, code complexity, time, and cost) for new development, software advancement (refactoring) and maintenance should be as low as possible. Then the solution is suitable for small and medium sized projects, as well.

2. The available resources should be utilized. The performance gain should be rewarding.
3. A good share of the large variety of compute devices types that are available on the market should be supported. The solution should also be suited for future hardware generations.
4. The details of the underlying hardware (execution behavior, memory hierarchy) should be abstracted. This division of responsibilities also makes it possible that different developers focus on their respective area of expertise.
5. Common functions (sorting, linear algebra, ...) should be part of the package and implemented as efficiently as possible. [13]
6. A solution should be comprehensive. In this way, it reduces the number of dependencies, which arise in projects that use many partial solutions.
7. Developers should have comprehensive control on the execution characteristics of the library. In particular it should be possible to conduct high-level performance tuning. [13]
8. The solution should be implemented as a high-level API, so that it seamlessly integrates with the developer's language of choice.
9. The execution layer should realize the known best practices as well as possible to ensure good execution performance. [9]

3.2 Concept

In order to reduce the burden of training and to ensure seamless integration in a familiar environment, we decided to use a parallel loop as an interface to our implementation. Loops are a fundamental programming construct; parallel loops in which the independent loop bodies are executed in parallel, are widespread. [2,14] Parallel port loops are thereby suitable for both task parallel problems and for the calculation of data-parallel algorithms. (Requirement 1, 4, 8)

The access to the accelerators is realized via OpenCL [17]. This standard was precisely designed to create a bridge between high-level languages, as well as, abstractions and the underlying hybrid systems. It is thus very detailed and close to hardware. Since we use OpenCL, we can not only support all current accelerators, but at the same time guarantee the future viability of our solution. (Requirement 3, 2)

The best practices in the area of performance optimization of hybrid systems show that many of them can be implemented automatically by the execution layer. To enable this static and dynamic code analysis can be used. Better results are achieved, however, if developers enrich their code with information to improve runtime decisions on possible optimizations. In Java this could be realized with Java annotations. In particular the distribution of data to the different kinds of memory in the memory hierarchy and to individual devices is very performance-critical. By using annotations to provide information to influence these mappings, developers can conduct performance tuning and influence the realization of best practices by the runtime. Here is a rule of thumb that the solution should already deliver good performance with little effort - namely the absence of annotations - and that a slightly higher effort should result in a large potential for further performance. The effort to performance gain ratio should therefore be reasonable. (Requirement 7, 9) To ensure that the abstraction remains and that developers are relieved from learning the details of the accelerator hardware, these annotations are only used to describe the data access pattern of the algorithm. The mapping on the memory of the accelerator is taken care of by our execution layer. Because developers know their algorithms best, they are able to describe the type of access, the access frequency, the relative sequence of accesses, etc., which are crucial for the mapping. (Requirement 1, 4, 8)

In addition, our solution delivers a portfolio of useful and commonly used functions that are applied in many data-parallel application domains: sorting, graph algorithms, linear algebra, ... (Requirement 5)

Our solution opens the way for the use of hybrid systems in Java projects. The special focus is on light-weight, seamless integration, the full control of your own code, and the resulting mapping onto the accelerator hardware. While developers focus on the description of their algorithms, our library takes care of the realization of possible optimizations. Because we support a fundamental construct countless algorithms can be implemented and parallelized. To further increase the comfort and improve the performance of the projects based on our solution even further, we provide a set of high-performance implementations for frequently used functions. By using our comprehensive solution the dependence on many small projects that provide partial solutions using hybrid systems is reduced. (Requirement 6)

3.3 Implementation

Listing 1 shows how our solution integrates with a Java program. The call to *HyFor.parallel* replaces the normal loop. The inclusive start value and the exclusive target value must be passed - as usual with loops; the loop body is described in the overridden *run* method on an anonymous class of type *ForLoopBody*.

If HyFor.parallel is called, the course of action shown in Figure 1 is executed. First, if this step has not already been taken before, our solution analyzes the system structure and existing accelerators. In addition, metadata is created for the algorithm. These are used in the second step to sort out those accelerators that are not capable of executing the algorithm. For example, because they do not support recursion, atomics or double precision floating point operations, but these are used in the algorithm. The third step is to estimate the performance of the execution of the given algorithm on the system components. In order to do this, the instructions used in the algorithm are considered and compared with the characteristics of the accelerator. In addition, the current workload and the performance numbers collected from the previous executions on the accelerators are taken into account. Now suitable accelerators for execution are selected in consideration of the task size. After that the actual code for the accelerator is generated. In this step both, the known features of the hardware and the properties of the algorithm that are apparent from the code or identified by annotations are con-

```
  int size = 512;

  final float[] values = new float[size];
  for (int i = 0; i < size; i++) {
    values[i] = i;
  }

  final float[] squares = new float[size];

  HyFor.parallel(0, size,
    new ForLoopBody<Integer>() {

    @Override
    public void run(Integer id) {
      squares[id] = values[id] * values[id];
    }

  });

  for (int i = 0; i < size; i++) {
    System.out.printf("\%6.0f \%8.0f\\n", values[i], squares[i]);
  }
}
```

Listing 1: Squares example using our approach. *values* contains the numbers from 0 to *size*; as a result of the calculation, *squares* contains the squares of these numbers.

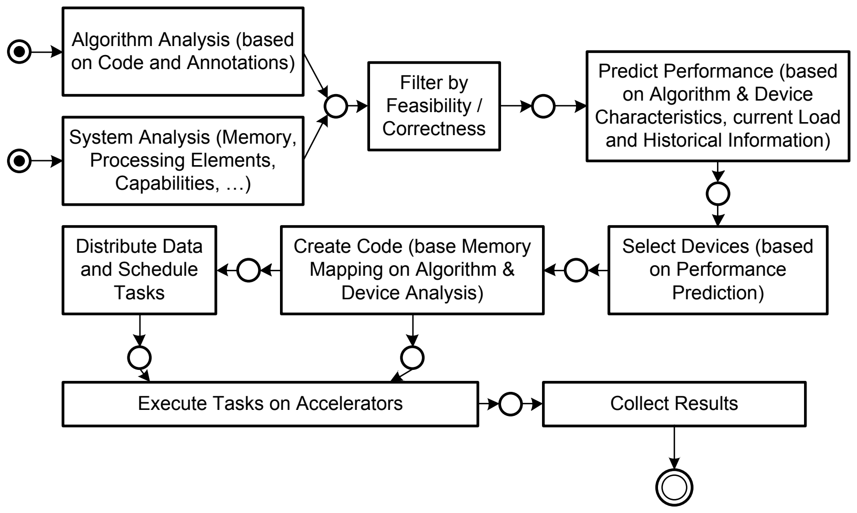

Algorithm Analysis (based on Code and Annotations)
System Analysis (Memory, Processing Elements, Capabilities, ...)
Filter by Feasibility / Correctness
Predict Performance (based on Algorithm & Device Characteristics, current Load and Historical Information)
Select Devices (based on Performance Prediction)
Create Code (base Memory Mapping on Algorithm & Device Analysis)
Distribute Data and Schedule Tasks
Execute Tasks on Accelerators
Collect Results

Figure 1: Hybrid Parallel Library - Course of Action

sidered. The code is loaded onto the accelerator, the data is distributed, the portions of the calculation are identified by indices and finally the calculation is triggered. When the results are ready, they are joined together and the function returns.

Our solution is implemented as a library that is structured as shown in Figure 2. The user program uses *HyFor.parallel* and annotations if applicable to describe the data access. *Algorithm Analysis* and *System Analysis* compile information about the

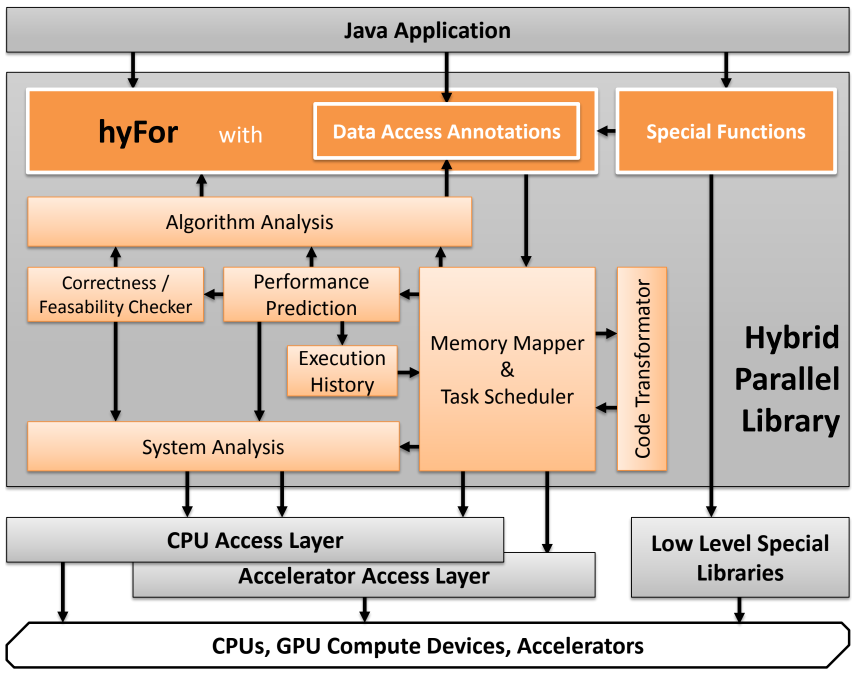


Figure 2: Hybrid Parallel Architecture. (Boxes represent software components, arrows represent usage relations, the eight-edged shape at the bottom represents hardware.)

mix of instructions of the algorithm and the characteristics of the hardware. The *Correctness Checker* and *Performance Predictor* select appropriate compute devices and *Memory Mapper* as well as *TaskScheduler* then assign data and subtasks to them. The code transformer delivers the source code that is compiled for each accelerator. It is realized as an adaptation and extension of the project Aparapi [1], which was originally developed by AMD and is now available as open source. In order to access the accelerators, OpenCL is used.

Our library ships with a portfolio of implementations of frequently used functions. These functions are realized either based on highly optimized native solutions or are implemented by the use of *HyFor.parallel* themselves.

4 Conclusion

An important factor in the creation of our solution was the strong focus on creating a light weight tool for the developer. The goal was that with a few lines of code and a simple construct, programmers should be enabled to use our library and take advantage of hybrid systems. In this paper we discussed a programming model as well as an implementation. Based on our research hybrid programming could be made available to programmers by learning only one new function: HyFor.parallel.

References

[1] Advanced Micro Devices, Inc. Aparapi .

[2] and Chuck Koelbel. High Performance Fortran Specification Version 2.0, January 1997.

[3] A. Branover, D. Foley, and M. Steinman. AMD Fusion APU: Llano . *Micro, IEEE*, 32:28–37 , 2012.

[4] Shuai Che, Michael Boyer, Jiayuan Meng, David Tarjan, Jeremy Sheaffer, Sang-Ha Lee, and Kevin Skadron. Rodinia: A benchmark suite for heterogeneous computing. *IEEE Workload Characterization Symposium*, 0:44–54, 2009.

[5] CLyther. CLyther. `http://srossross.github.com/Clyther/`.

[6] J. Diaz, C. Munoz-Caro, and A. Nino. A Survey of Parallel Programming Models and Tools in the Multi and Many-Core Era. 23:1369 –1386 , 2012.

[7] A. Duran and M. Klemm. The Intel®Many Integrated Core Architecture. In *High Performance Computing and Simulation*, pages 365 –366.

[8] Frank Feinbube, Bernhard Rabe, Martin Löwis, and Andreas Polze. NQueens on CUDA: Optimization Issues. In *2010 Ninth International Symposium on Parallel and Distributed Computing*, pages 63–70, Washington, DC, USA, 2010. IEEE Computer Society.

[9] Frank Feinbube, Peter Tröger, and Andreas Polze. Joint Forces: From Multithreaded Programming to GPU Computing. *IEEE Software*, 28:51–57, October 2010.

[10] H. Franke, J. Xenidis, C. Basso, B. Bass, S. Woodward, J. Brown, and C. Johnson. Introduction to the wire-speed processor and architecture. *IBM Journal of Research and Development*, 54, 2010.

[11] Jared Hoberock and Nathan Bell. Thrust: open-source template library for developing CUDA applications.

[12] Intel Labs. *SCC External Architecture Specification (EAS)*, April 2010.

[13] Ken Kennedy, Charles Koelbel, and Hans Zima. The rise and fall of High Performance Fortran: an historical object lesson. In *third ACM SIGPLAN conference on History of programming languages*, pages 7–1, New York, NY, USA, 2007. ACM.

[14] Daan Leijen and Judd Hall. Parallel Performance - Optimize Managed Code For Multi-Core Machines, 2007.

[15] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym. NVIDIA Tesla: A unified graphics and computing architecture. *Micro, IEEE*, 28:39–55, 2008.

[16] Microsoft Corporation. *Microsoft Accelerator v2 Programming Guide*, 2011.

[17] Aaftab Munshi. *The OpenCL Specification - Version 1.1*, June 2010.

[18] Nvidia. CUDA Show Case.

[19] NVIDIA. CUDA Developer Zone. `http://developer.nvidia.com/category/zone/cuda-zone`, 2011.

[20] Nvidia. NVIDIA ION Graphics Processors. `http://www.nvidia.com/object/picoatom_specifications.html`, October 2012.

[21] Christopher J. Rossbach, Jon Currey, Mark Silberstein, Baishakhi Ray, Emmett Witchel, Ted Wobber, and Peter Druschel. PTask: operating system abstractions to manage GPUs as compute devices. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles 2011*, pages 233–248. ACM, 2011.

[22] SourceForge. OpenCL .Net.

[23] Herb Sutter. The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software. *Dr. Dobb's Journal*, 30, March 2005.

[24] TIOBE Software. TIOBE Programming Community Index. `http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html`, October 2012.

[25] M. Yuffe, E. Knoll, M. Mehalel, J. Shor, and T. Kurts. A fully integrated multi-CPU, GPU and memory controller 32nm processor . In *Solid-State Circuits Conference Digest of Technical Papers*, pages 264 –266. IEEE International.

[26] Ying Zhang , Baton Rouge, and Lu Peng. Architecture comparisons between Nvidia and ATI GPUs: Computation parallelism and data communications . In *Workload Characterization*, pages 205–215. IEEE.

On the Complex Nature of MDE Evolution – A Meta Study

Regina Hebig
System Analysis and Modeling Group
Hasso-Plattner-Institut
regina.hebig@hpi.uni-potsdam.de

In context of MDE, evolution plays an important role to improve productivity and address newly arising requirements on the usually built software, like the need to provide functionality in form of services. Therefore, it is of paramount importance to understand how MDE evolves in organizations. In this report, we approach the question how MDE evolution manifests in practice. Using a meta study we systematically compared literature that proposes techniques for evolving or changing MDE settings with reports about actual applications of MDE in industry. We categorize the observed evolution steps and illustrate a mismatch between literature and practice. In that context, we characterize an important class of evolution steps that has substantial impact on the MDE settings and has been not recognized yet.

1 Introduction

Model-driven engineering (MDE) is used in many domains for software development. The overarching characteristic of MDE is the use of models and dedicated automated development steps that operate on models to systematically derive large fractions of the software from models rather than general purpose programming language code.

In context of MDE evolution plays an important role to improve productivity and reduce time-to-market, e.g., by improving automation of development. Also newly arising requirements on the usually built software can be addressed by evolution, e.g., by evolving applied domain specific languages (DSLs). Therefore, it is of paramount importance to understand how MDE evolves in organizations and how to cost-effectively realize the necessary evolution steps.

The literature on MDE evolution is mainly considering language evolution [2, 10, 26], phenomena related to metamodel evolution, and the corresponding adjustment of consuming or generating transformations [13,20,33,35]. An additional form of evolution focuses on adding languages to the MDE settings [32]. Literature that targets the problem at a broader scope, looking at aspects such as Capability Maturity Model (CMM) for MDE [28] or the life cycle for transformation development [11], employs a similar understanding of evolution. However, in a number of studies on capturing MDE settings (the details how the models and manual as well as automated activities relate to each other) in practice [12], we got the feedback that the people have been exposed to rather substantial changes during the evolution of MDE settings in their organizations. Thereby, they experienced great impact on their daily work.

In this report, we approach the question how MDE evolution manifests in practice. In a first step we performed a meta study comparing systematically literature that proposes techniques for evolving or changing MDE settings with reports about actual MDE settings in industry. We categorize the observed evolution steps and illustrate a mismatch between literature and practice. In that context, we characterize an important class of evolution steps that has substantial impact on the MDE settings and has not been recognized yet.

2 Meta Study on Evolution

We performed a meta study to answer the question, whether the scope of current research approaches is sufficient to capture evolution of MDE settings that occur in practice. Following the review process is described followed by a survey of research support for evolution and reports on evolution that occurred in practice. We classify change types that occur in practice and change types that are supported by research approaches. We discuss differences between the change types and characterize a lack in literature that prevents us from understanding the nature of MDE evolution.

2.1 Review process

To identify literature for this meta study, we systematically searched through the proceedings of the MODELS conference from 2007 to 2011 and ECMFA, respectively ECMDA-FA conferences from 2007 to 2012. the proceedings of the Workshop on Models and Evolution ME, as well as its predecessors MCCM (Workshop on Model Co-Evolution and Consistency Management) and MoDSE (Workshop on Model-Driven Software Evolution) from 2007 to 2011, the proceedings of the OOPSLA Workshops on Domain-Specific Modeling from 2007 to 2011, as well as the Software and Systems Modeling journal (SoSyM) from 2007 to 2012, including papers published online first until end of July 2012. In addition, we performed online key word search and followed references in reviewed papers. In particular we used ACM digital library for keyword search in the proceedings of the ICSE conference. First, we searched for approaches that aim to support changes in model operations (e.g., transformations and generation of code), languages, or combinations of modeling techniques. The sample in this report was chosen so that the existing diversity of types of supported changes is represented: [2,4,6,7,10,13,16,18–20,22–27,31,33,35].

Second, we searched for reports on the application of model-driven techniques or domain-specific modeling languages in practice. Thereby, we identified thirteen reports or case studies that describe MDE introduction or usage: [1,3,5,7,17,21,25,29,30,34] and three case studies in [14]. Cases where different aspects of the same application are described in multiple papers are counted as one report. We filtered the reports according to two criteria. First, it was necessary to ensure that the chosen reports capture a period of time that is long enough to expect evolution. Thus, reports that focus only on the an initial introduction of MDE or on settings that were used for a single project only, were not suitable. Second, the changes need to be described sufficiently for rating. Reports that only mention changes without further descriptions could not be included. Finally, we chose five reports for this meta study: [3, 7, 17, 25, 30]. These reports stem from different domains, such as telecommunication industry, financial organizations, and development of control systems.

2.2 Literature on support for evolution

In this section we give an overview of evolution steps or changes in MDE approaches that are expected and supported by literature approaches. A first group of approaches deals with the application of changes to the implementation of a transformation (referred to as change type *C1* in the following). For example, the MDPE workbench, which is introduced in [8, 9] and further discussed as one of the case studies in [25], contains a transformation chain that transforms input models of different (partly proprietary) languages to a representation that conforms to an intermediate meta-model (Tool-independent Performance Model (TIPM)). The TIPM models are then automatically transformed to input formats of different performance analysis tools. The analysis results are transformed back to be presented within the context of the input model. Due to this mechanism, the MDPE workbench enables the extension of the quality assurance with additional performance analysis tools. From the perspective of the user, such an extending evolution step only has effects on the results. Thus, the evolution is experienced as an evolution of the implementation of an automated analysis activity.

Evolutionary changes are not the only motivation for changing a transformation. For example, in [18], a generator is built so that it can easily be configured to implement architectural decisions met within a project. In [19], the incremental development of a transformation chain is discussed. Thereby, implementations of involved transformations are evolved systematically to reach the desired result.

A second group of approaches deals with language evolution and migration of models, such that they become valid for a new version of a meta-model (referred to as change type *C2* in the following). In [26], the Model Change Language (MCL) is introduced. MCL can be used to specify migration rules for models between two versions of a meta-model. In [10], the differences between two meta-models are analyzed and used to generate a transformation that can be used to migrate the corresponding models. In [2], the Modif meta-model for describing meta-model evolution is introduced. Based on that description the new meta-model version and a transformation for migrating corresponding models is generated. Further examples are [31], [24], or [4], where the usage of higher-order model transformations to support co-evolution of models to changes in the corresponding meta-model is proposed.

A third group of approaches deals with the adaption of transformations when meta-models of consumed models change. For some of these approaches the change of a language is the motivation for providing techniques for modular design and adaption strategies for transformations. For example, Yie et al. [35] deal with the question how a fully automated transformation chain can be adapted due to additional concepts in the input language. They propose a solution where the initial transformation chain is complemented by a further chain of transformations that handles the additional concepts and a correspondence model to maintain the relations between the different transformation results. Similarly, the flexibility of the MasterCraft code generator presented in [18] is motivated by the introduction of additional concepts.

Other approaches deal with the question how information about language changes can be used systematically to adapt consuming transformations (e.g., [13]). In [20], a semi-automated approach for the adaption of the model transformations is introduced. Vermolen et al. lift the semi-automated evolution of models and consuming transformations to a more technology-independent layer by allowing also migrations of programming languages or data structures [33].

Similar to the third group of approaches, the fourth group of approaches is based on exchanging modeling languages and transformations. However, here the motivation is not the evolution of a language, but the migration to a new platform. As already proposed in the OMG's MDA [27], the transformation between platform-independent model and platform-specific model might be exchanged to address the needs of a new platform during generation. In [7], a corresponding migration process is presented. The proposed approach has to be configured by exchanging the transformation between platform-independent model and platform-specific model. In addition, languages of code might change. Meyers et al. present a combined view of evolution approaches. They subdivide language- and model evolution into four primitive scenarios, describing how evolution of models, meta-models, or transformations enforces co-evolution among each other [22,23]. The considered scenarios are combinations of the changes supported by the approaches described above (supporting change types *C1* and *C2*).

Finally, we identified two examples for approaches that lead to an addition of input models to an automated transformation or generation activity, which is exchanged or evolved (referred to as change type *C3* in the following). This changes not only the number of models that play a role within the affected MDE approach (referred to as change type *C4* in the following), but might also increase the number of used (modeling) languages (referred to as change type *C5* in the following). Furthermore, adding new input models can imply a change in the number of manual modeling activities (referred to as change type *C6* in the following). This is not necessarily the case, since the new models might be reused or automatically created. However, the changes *C4*, *C5*, and *C6* introduced here are manifested in change *C3*. As it will be shown below, the numbers of used models, languages, and manual activities can also be affected for other reasons than the addition of an input model to an automated activity.

Motivated by the need to raise the level of abstraction, starting from an existing DSL, Johannes et al. present an approach where a composition system is used to automatically compose models of different DSLs that are defined hierarchically on top of an existing DSL in [16]. Thus, the definition of a new DSL within this approach will lead to an additional input language for the automated composition and generation based on the composition result. Estublier et al. present a similar approach for the composition of DSLs as an alternative to extending existing DSLs in [6]. They present a modifiable interpreter, where a composition model can be used to define how different domain-specific models are related. Additionally, this approach allows adaptation for the interpreter in case of changes in the DSL (manifestation of *C1*).

2.3 Evolution observed in practice

In the following we summarize the results of the five reports (selected in Section 2.1) on changes of MDE settings in practice. Since the description of the change was not the main focus of the papers, we can only include changes explicitly described in the papers, but cannot exclude that certain types of changes are part of the example.

As described in Section 2.2, Fleurey et al. present a process for the migration of systems to new platforms in [7]. To apply the process it is proposed to substitute the used transformations to fit the current use case. In addition, they describe how they actually did adapt the process to apply it for the migration of a banking application. Interestingly, the changes actually applied differ strongly from the proposed changes. In this special case, it was necessary that the resulting system conforms to the development standards of the customer. Thus, it was not sufficient to produce code, but to provide corresponding models that were synchronized with the code, such that round-trip engineering on the migrated system was possible. Therefore, they replaced the code generation with an automated UML extraction. They integrated the Rational Rose code generator used by the customer to generate code skeletons out of the models. Thus, the number of tools changed (referred to as change type *C7* in the following). Further, they added a generation to migrate the remaining code from the platform-independent model (extracted from the original code) into the code skeletons (*C1*, *C2*, *C4*,*C5*). Not only the chain of automated steps was affected. Conforming to the round trip engineering, some manual migration tasks have to be applied to the models (*C6*). The corresponding reapplication of the Rational Rose code generation adds an additional automated step to the MDE settings (referred to as change type *C8* in the following). Thus, instead of being only followed by manual migration, the automated migration is followed by manual migration activities on the Rational Rose model, a generation of code and further manual migration activities on the code. Thereby, the order of manual and automated tasks change, as manual migration is intermixed with automated code generation (referred to as change type *C9* in the following).

For the Telefónica case study presented in [25], it is reported that the developed DSML for the generation of configuration files was changed later on. The first change was the integration of the verification language EVL to incrementally check the correctness of the models during development. This feedback mechanism intermixes manual modeling activities with an added automated analysis for correctness (*C8*,*C9*).

As a second change, the generation of the configuration files was exchanged by an implementation that is based on a composition system conforming to the approach reported in [16], which was already discussed in Section 2.2. Motivation for this change was the wish to be flexible to reach higher levels of abstraction. Thus, the number of input models and used modeling languages changed from one to a flexible number (*C3*,*C4*,*C5*). With it, also the number of manual modeling activities changes for the developer, who has to create a number of different DSL models (*C6*).

In [17], a tool vendor reports how the language FBL together with its engineering environment changed. They started with providing the visual programming language FBL together with an editor and a code generator. Over time they introduced the tool function test to enable developers to debug FBL (*C7*). Starting from manual programming with a following automated generation, the introduction of automated verification or debugging operations changes the order of manual and automated tasks. As result manual programming is followed by automated debugging and further manual correction before the automated generation is applied (*C8*,*C9*). A second evolution step was the introduction of templates to allow programming on a higher level of abstraction. Developers have to choose and configure templates by specifying parameters. A chosen template with parameters is then automatically translated to FBL and from there code is generated. Thus, the used language (templates instead of FBL) and the generation implementation (transformation plus generation) changed (*C1*,*C2*).

In [30], the adoption of MDE in a financial organization is reported. The report ends with a note that further changes to the MDE settings are planned in the future. They want to reach better integration of different used tools (*C7*) and more automation of the construction phase of the lifecycle (*C8*). In [3], Baker et al. describe the usage of MDE within Motorola. They report about changing tools (*C7*) and a changing number of languages and used models (*C4*,*C5*) with the introduction of Message Sequence Charts (MSC) and SDL. Further, they report about changes in MSC (*C2*) that enabled the introduction of automated generation of test cases (*C8*).

2.4 Summary of change types

In the following we compare changes supported by literature approaches with changes occurring in practice. Therefore, we want to discuss the extent of the change types. Change *C1* concerns exchanging or evolution of an automated activity, which might be any model operation or code generation. Whether *C1* is applied or a new automated activity is added *C8*, is sometimes difficult to distinguish. Here, we consider all cases where an automated activity is added and used in each case together with an existing automated activity as a change of type *C1*. *C2* concerns exchanging or evolving a used language, while *C5* concerns the addition of languages in use. Thereby, both changes concern modeling languages as well as programming languages or other kinds of artifacts. Change *C3* actually describes a set of constellations how changes *C4*, *C5*, and *C6* can be applied. *C4* concerns the change of artifact roles (i.e., expected input and output of manual or automated activities). Thereby, an artifact can be a model or code. In case of additions of input models, we rated the modeling activity for creating the additional model as an additional manual activity (*C6*). However, whether modeling of two distinct models is experienced as two modeling activities or not is a subjective question. Similarly, it is not always distinguishable how an additional tool is experienced by developers. If a new tool is highly integrated into am already used tool or framework, developers might recognize it as new feature, while in other cases an additional tool can lead to extra effort (e.g., when artifacts have to be explicitly exported and imported to be used). Finally, the change of the order of activities (*C9*) describes a consequence of the application of change *C6* or *C8*.

2.5 Structural changes vs. non-structural changes

Evolution supported in literature results from a combination of three change types: change of a model's or artifact's language (*C2*), change of a model operation's implementation (*C1*), and addition of input artifacts consumed in a model operation (*C3*). However, combinations of these change types do not always suffice to describe evolution in practice. We first differentiate between *non-structural* and *structural* changes. *Structural changes* affect the number and order of tools, activities, languages, and models a developer has to deal with. In contrast, *non-structural change* only concern which (modeling) language is concretely used for an artifact/model and how an automated activity (i.e., transformation or generation) is implemented. Thus, only one kind of structural change is currently supported in the literature, which is the addition of input artifacts to an automated activity.

Changing an MDE setting leads to changes for developers and a company. This can affect the *degree of automation* of development, the *complexity* to work with the MDE setting, and the *changeability and maintainability* of the software that it built with the MDE setting. Further, the effort for *maintaining consistency* and *integration effort* when working with different tools is affected. Finally, tools and automated activities that are used in an MDE setting, need to be maintained and, therefore, affect the *costs of ownership* of a company. Since these aspects imply potentials and risks for the productivity of developers or the overall productivity of a company, respectively, we call them *productivity dimensions* in the following. Changes in an automated activity (*C1*) as well as changes in the number of automated activities (*C8*) can affect the *degree of automation*. Changing a used (modeling) language (*C2*) or the number of used languages (*C5*) can have benefits concerning the degree of abstraction, but yields the risk that the developers lack the know how to use that language (affecting the *complexity* of the MDE settings) [3]. Similarly, a growing number of models (*C4*), necessary manual activities (*C6*), or tools (*C7*) increases *complexity* for developers. In addition, a change in the number of models affects the need to *maintain the consistency* of different models [15]. Further, additional tools might lead to additional activities to move artifacts between them (increasing the *integration effort*). As tools and implementations of automated activities have to be maintained, changes in both, number of tools *C7* and number of automated activities *C8*, can affect the *costs of ownership*.

Finally, a main risk results from the addition of automated or manual activities if this leads to a change of the order how manual and automated activities occur (*C6*, *C8*, and *C9*). This is not only due to growing complexity for developers, but also the occurrence of constellations where automatically created artifacts are touched manually. Further these changes can lengthen the chain of activities that is required to apply a change. This implies risks for *changeability and maintainability*, when the automated step has to be reapplied. The main risks and potentials for non-structural changes concern the *degree of automation* and the *complexity* of the MDE settings. In contrast, structural changes imply some important additional risk and potentials, like changes in the effort required to maintain consistency of all required artifacts or changes in the *costs of ownership*. Further structural changes can have stronger effects on the different domains of productivity than non-structural changes. For example, increasing the number of used languages (*C5*) has in most cases a worse impact on the required know how than just applying changes to a used language (*C2*). Finally, there is a group of structural changes (*C6*, *C8*, and *C9*) that can affect the *changeability and maintainability* of the software that is built with the MDE settings. This group of changes can lead to MDE settings that contain constellations that are well known for their risks, but has also the potential to eliminate these risks. Therefore, we call these structural changes *substantial structural changes* in the following. Interestingly, the only structural change that is proposed in literature (*C3*: adding additional input artifacts to an automated activity) is restricted in its risks to an increasing *complexity* and increasing *effort in maintaining consistency* between the different models. With the introduction of additional input artifacts *C3* might lead to the additional introduction of manual activities preceding the changes activity for creating these input artifact (unless these artifacts are reused). If this change type is not combined with the introduction of automated activities *C8* to support the creation of the additional input artifacts automatically, *C3* does not lead to a risky change of the order of manual and automated activities.

To sum up, non-structural changes tend to affect only single productivity dimensions (*C1* affects *degree of automation*, *C2* affects the *complexity*). In contrast, a structural change affects multiple productivity dimensions (e.g., *C8* affects *degree of automation*, *costs of ownership*, and *changeability* and *maintainability*). In consequence, structural changes yield the difficulty that improvements in one productivity dimension can come along with a change for the worse for another dimension. *Evolution* of MDE settings results from a single or combined occurrence of changes. A change can comply with multiple change types. E.g., adding a new automated activity (*C8*) in between two manual activities changes the order of automated and manual activities (*C9*), too. However, different changes can also occur in combination, e.g., exchanging the implementation of an automated activity (*C1*) can render a manual activity unnecessary (*C6*). Evolution consists of evolution steps, which include one or more changes. Thereby, an *evolution step* contains all changes that modified a version of an MDE setting that was practically used to another version that is or was practically used. We call an evolution step *structural evolution step* if the set of changes contains at least one structural change. Similarly, we call an evolution step *substantial evolution step* if the set of changes contains at least one substantial structural change. We call the evolution of an MDE setting that contains a least one structural evolution step *structural evolution*. The combined occurrence of different changes in one evolution steps can enhance the difficulty to improve one productivity dimension without a change for the worse for another dimension.

Subsuming, changes that occur in practice (like *substantial structural changes*) can imply risks that are not caused by changes supported by literature approaches. Further, especially structural evolution, affects multiple productivity dimensions.

2.6 Threats to validity

We identified that structural changes occur. However, we cannot provide any empirical statement about the probability that an MDE settings is affected by structural changes. We do not expect structural changes to be seldom, since we found occurrences for structural changes not only in the meta study. Initially we were confronted with structural changes in a field study, where we did not focus on evolution. Further, it cannot be claimed that structural changes occur in every domain in software engineering. However, the examples discussed in this report are from different domains, such as telecommunication industry, financial organizations, and enterprise applications.

3 Conclusion

Using the meta study we identified the class of *structural evolution steps* that can be observed in industry. It turned out that this kind of evolution steps is rarely or, in case of substantial structural evolution steps, not at all covered by techniques proposed in literature for evolving or changing MDE settings.

References

[1] Thomas Aschauer, Gerd Dauenhauer, and Wolfgang Pree. A modeling language's evolution driven by tight interaction between academia and industry. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2*, ICSE '10, pages 49–58, 2010. ACM.

[2] Jean-Philippe Babau and Mickael Kerboeuf. Domain Specific Language Modeling Facilities. In *Proceedings of the 5th MoDELS workshop on Models and Evolution*, pages 1–6, October 2011.

[3] Paul Baker, Shiou Loh, and Frank Weil. Model-Driven Engineering in a Large Industrial Context – Motorola Case Study. In Lionel Briand and Clay Williams, editors, *Model Driven Engineering Languages and Systems*, volume 3713 of *Lecture Notes in Computer Science*, pages 476–491. Springer Berlin / Heidelberg, 2005.

[4] Antonio Cicchetti, Davide Di Ruscio, Romina Eramo, and Alfonso Pierantonio. Automating Co-evolution in Model-Driven Engineering. In *Proceedings of the 2008 12th International IEEE Enterprise Distributed Object Computing Conference*, pages 222–231, 2008. IEEE Computer Society.

[5] Gan Deng, Tao Lu, Emre Turkay, Aniruddha Gokhale, Douglas C. Schmidt, and Andrey Nechypurenko. Model Driven Development of Inventory Tracking System. In *Proceedings of the ACM OOPSLA 2003 Workshop on Domain-Specific Modeling Languages*, October 2003.

[6] Jacky Estublier, German Vega, and Anca Ionita. Composing Domain-Specific Languages for Wide-Scope Software Engineering Applications. In Lionel Briand and Clay Williams, editors, *Model Driven Engineering Languages and Systems*, volume 3713 of *Lecture Notes in Computer Science*, pages 69–83. Springer Berlin / Heidelberg, 2005.

[7] Franck Fleurey, Erwan Breton, Benoît Baudry, Alain Nicolas, and Jean-Marc Jézéquel. Model-Driven Engineering for Software Migration in a Large Industrial Context. In Gregor Engels, Bill Opdyke, Douglas Schmidt, and Frank Weil, editors, *Model Driven Engineering Languages and Systems*, volume 4735 of *Lecture Notes in Computer Science*, pages 482–497. Springer Berlin / Heidelberg, 2007.

[8] Mathias Fritzsche, Hugo Brunelière, Bert Vanhooff, Yolande Berbers, Frédéric Jouault, and Wasif Gilani. Applying Megamodelling to Model Driven Performance Engineering. In *ECBS '09: Proceedings of the 2009 16th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems*, pages 244–253, 2009. IEEE Computer Society.

[9] Mathias Fritzsche and Jendrik Johannes. Putting Performance Engineering into Model-Driven Engineering: Model-Driven Performance Engineering. In Holger Giese, editor, *Models in Software Engineering*, volume 5002 of *Lecture Notes in Computer Science*, pages 164–175. Springer Berlin / Heidelberg, 2008.

[10] Kelly Garcés, Frédéric Jouault, Pierre Cointe, and Jean Bézivin. Managing Model Adaptation by Precise Detection of Metamodel Changes. In *ECMDA-FA '09: Proceedings of the 5th European Conference on Model Driven Architecture - Foundations and Applications*, pages 34–49, Berlin, Heidelberg, 2009. Springer-Verlag.

[11] Esther Guerra, Juan de Lara, Dimitrios Kolovos, Richard Paige, and Osmar dos Santos. Engineering model transformations with transML. *Software and Systems Modeling*, pages 1–23. 10.1007/s10270-011-0211-2.

[12] Regina Hebig and Holger Giese. MDE Settings in SAP. A Descriptive Field Study. Technical Report 58, Hasso-Plattner Institut at University of Potsdam, 2012.

[13] Markus Herrmannsdoerfer, Daniel Ratiu, and Guido Wachsmuth. Language Evolution in Practice: The History of GMF. In Mark Van den Brand, Dragan Gasevic, and Jeff Gray, editors, *Software Language Engineering*, volume 5969 of *Lecture Notes in Computer Science*, pages 3–22. Springer Berlin / Heidelberg, 2010.

[14] John Hutchinson, Mark Rouncefield, and Jon Whittle. Model-driven engineering practices in industry. In *Proceeding of the 33rd international conference on Software engineering*, ICSE '11, pages 633–642, 2011. ACM.

[15] John Hutchinson, Jon Whittle, Mark Rouncefield, and Steinar Kristoffersen. Empirical assessment of MDE in industry. In *Proceeding of the 33rd international conference on Software engineering*, ICSE '11, pages 471–480, 2011. ACM.

[16] Jendrik Johannes and Miguel Fernandez. Adding Abstraction and Reuse to a Network Modelling Tool Using the Reuseware Composition Framework. In Thomas Kühne, Bran Selic, Marie-Pierre Gervais, and FranÇois Terrier, editors, *Modelling Foundations and Applications*, volume 6138 of *Lecture Notes in Computer Science*, pages 132–143. Springer Berlin / Heidelberg, 2010.

[17] M. Karaila. Evolution of a Domain Specific Language and its engineering environment – Lehman's laws revisited. In *Proceedings of the 9th OOPSLA Workshop on Domain-Specific Modeling*, 2009.

[18] Vinay Kulkarni, Souvik Barat, and Uday Ramteerthkar. Early Experience with Agile Methodology in a Model-Driven Approach. In Jon Whittle, Tony Clark, and Thomas Kühne, editors, *Model Driven Engineering Languages and Systems*, volume 6981 of *Lecture Notes in Computer Science*, pages 578–590. Springer Berlin / Heidelberg, 2011.

[19] Jochen Küster, Thomas Gschwind, and Olaf Zimmermann. Incremental Development of Model Transformation Chains Using Automated Testing. In *Model Driven Engineering Languages and Systems: 12th International Conference, MODELS 2009*, volume 5795/2009 of *Lecture Notes in Computer Science (LNCS)*, pages 733–747. Springer Berlin / Heidelberg, 2009.

[20] Tihamer Levendovszky, Daniel Balasubramanian, Anantha Narayanan, and Gabor Karsai. A Novel Approach to Semi-automated Evolution of DSML Model Transformation. In Mark Van den Brand, Dragan Gasevic, and Jeff Gray, editors, *Software Language Engineering*, volume 5969 of *Lecture Notes in Computer Science*, pages 23–41. Springer Berlin / Heidelberg, 2010.

[21] Nikolai Mansurov and Djenana Campara. Managed Architecture of Existing Code as a Practical Transition Towards MDA. In Nuno Jardim Nunes, Bran Selic, Alberto Rodrigues da Silva, and Ambrosio Toval Alvarez, editors, *UML Modeling Languages and Applications*, volume 3297 of *Lecture Notes in Computer Science*, pages 219–233. Springer Berlin / Heidelberg, 2005.

[22] Bart Meyers, Raphael Mannadiar, and Hans Vangheluwe. Evolution of Modelling Languages. In *8th BElgian-NEtherlands software eVOLution seminar (BENEVOL)*, 2009.

[23] Bart Meyers and Hans Vangheluwe. A framework for evolution of modelling languages. *Science of Computer Programming, Special Issue on Software Evolution, Adaptability and Variability*, 76(12):1223 – 1246, 2011.

[24] Bart Meyers, Manuel Wimmer, Antonio Cicchetti, and Jonathan Sprinkle. A generic in-place transformation-based approach to structured model co-evolution. In *Proceedings of the 4th International Workshop on Multi-Paradigm Modeling (MPM'10) @ MoDELS'10*. Electronic Communications of the EASST, 2010.

[25] Parastoo Mohagheghi, Wasif Gilani, Alin Stefanescu, Miguel Fernandez, Bjørn Nordmoen, and Mathias Fritzsche. Where does model-driven engineering help? Experiences from three industrial cases. *Software and Systems Modeling*, pages 1–21. 10.1007/s10270-011-0219-7.

[26] Anantha Narayanan, Tihamer Levendovszky, Daniel Balasubramanian, and Gabor Karsai. Automatic Domain Model Migration to Manage Metamodel Evolution. In Andy Schürr and Bran Selic, editors, *Model Driven Engineering Languages and Systems*, volume 5795 of *Lecture Notes in Computer Science*, pages 706–711. Springer Berlin / Heidelberg, 2009.

[27] Object Management Group. *MDA Guide Version 1.0.1*, June 2003. Document – omg/03-06-01.

[28] Erkuden Rios, Teodora Bozheva, Aitor Bediaga, and Nathalie Guilloreau. MDD Maturity Model: A Roadmap for Introducing Model-Driven Development. In *Model Driven Architecture – Foundations and Applications*, volume 4066/2006 of *Lecture Notes in Computer Science*, pages 78–89. SpringerLink, 2006.

[29] Andrey Sadovykh, Lionel Vigier, Eduardo Gomez, Andreas Hoffmann, Juergen Grossmann, and Oleg Estekhin. On Study Results: Round Trip Engineering of Space Systems. In Richard Paige, Alan Hartman, and Arend Rensink, editors, *Model Driven Architecture - Foundations and Applications*, volume 5562 of *Lecture Notes in Computer Science*, pages 265–276. Springer Berlin / Heidelberg, 2009.

[30] Dov Shirtz, Michael Kazakov, and Yael Shaham-Gafni. Adopting model driven development in a large financial organization. In *Proceedings of the 3rd European conference on Model driven architecture-foundations and applications*, ECMDA-FA'07, pages 172–183, Berlin, Heidelberg, 2007. Springer-Verlag.

[31] Mark Van den Brand, Zvezdan Protic', and Tom Verhoeff. A Generic Solution for Syntax-Driven Model Co-evolution. In Judith Bishop and Antonio Vallecillo, editors, *Objects, Models, Components, Patterns*, volume 6705 of *Lecture Notes in Computer Science*, pages 36–51. Springer Berlin / Heidelberg, 2011.

[32] Arie van Deursen, Eelco Visser, and Jos Warmer. Model-Driven Software Evolution: A Research Agenda. In D. Tamzalit, editor, *CSMR Workshop on Model-Driven Software Evolution (MoDSE 2007)*, pages 41–49, March 2007.

[33] Sander Vermolen and Eelco Visser. Heterogeneous Coupled Evolution of Software Languages. In Krzysztof Czarnecki, Ileana Ober, Jean-Michel Bruel, Axel Uhl, and Markus Völter, editors, *Model Driven Engineering Languages and Systems*, volume 5301 of *Lecture Notes in Computer Science*, pages 630–644. Springer Berlin / Heidelberg, 2008.

[34] Regis Vogel. Practical case study of MDD infusion in a SME: Final Results. In Dalila Tamzalit, Dirk Deridder, and Bernhard Schätz, editors, *Models and Evolution Joint MODELS'09 Workshop on Model-Driven Software Evolution (MoDSE) and Model Co-Evolution and Consistency Management (MCCM),*, pages 68–78, 2009.

[35] Andrés Yie, Rubby Casallas, Dennis Wagelaar, and Dirk Deridder. An Approach for Evolving Transformation Chains. In Andy Schürr and Bran Selic, editors, *Model Driven Engineering Languages and Systems*, volume 5795 of *Lecture Notes in Computer Science*, pages 551–555. Springer Berlin / Heidelberg, 2009.

Steuerung der Datenübertragung in öffentlichen zellularen Netzen im Kontext telemedizinischer Anwendungen

Uwe Hentschel

FG Betriebssysteme & Middleware
Hasso-Plattner-Institut
uwe.hentschel@hpi.uni-potsdam.de

Die Bearbeitung und Weiterleitung von medizinischen Daten folgt dem Grundsatz der Datensicherheit. Das bedeutet vor allem, dass einmal erfasste Daten nicht mehr aus dem System verschwinden dürfen, auch wenn einzelne Teile oder das gesamte System ausfallen sollten. Speziell für die Übertragung heißt das, dass alle Daten beim Sender zumindest solange dauerhaft gespeichert werden, bis der Empfänger seinerseits die erfolgreiche Speicherung quittiert. Das heißt, eine reine Empfangsquittung des Empfängers ist nicht ausreichend, um die Daten auf der Senderseite zu löschen.

Verallgemeinert man die aus dem Fontane Projekt bekannte medizinische Nutzung der Endgeräte auf der Klientenseite, kann man von einer oder mehreren medizinischen Anwendungen auf jedem Endgerät ausgehen. Bei der Nutzung gemeinsamer Netzwerkressourcen müssen also unterschiedliche Endgeräte im Netzwerk als auch unterschiedliche Anwendungen auf jedem der Endgeräte synchronisiert werden.

1 Synchronisation auf Anwendungsebene

Die Anwendungen auf den mobilen Endgeräten sollen Daten entsprechend ihrer Dringlichkeit an den zentralen Server übermitteln und sich dabei auf unterschiedliche Kommunikationsbedingungen einstellen, indem sie die Menge der pro Zeiteinheit gesendeten Daten variieren. Dafür muss sich das Verhalten der Anwendungen entsprechend anpassen lassen – wir benötigen ein adaptives System innerhalb der Endgeräte. Dieser Ansatz geht davon aus, dass jedes Endgerät ein autonomes Element entsprechend des MAPE-K Modells darstellt [12, 13]. In unserem Fall ist das verwaltete Element die, für die Kommunikation zwischen den Client-Geräten und dem Server, zur Verfügung stehende Datenübertragungsrate; verwaltet wird es auf Anwendungsebene. Dieser Ansatz lässt sich mit Standardkomponenten für die Hardware, das Betriebssystem und den Protokollstapel realisieren.

1.1 Anwendung des MAPE-K Modells

Obwohl die Art und Weise wie sich die Kommunikationsstrategie einer Anwendung ändert, stark vom Zweck ihres Einsatzes abhängig ist, erscheint die Implementierung aller Komponenten zur Verwaltung der Kommunikation in den einzelnen Anwendungen als ineffektiv. Dies gründet sich vor allem darauf, dass einige dieser Komponenten, wie die Überwachung der Kommunikation und die Analyse dieser Ergebnisse, vom Umfeld der Kommunikation abhängig sind und sich somit bei allen Anwendungen in der selben Umgebung gleichen. Aus diesem Grund sind diese beiden Komponenten Teil einer von allen Anwendungen genutzten Middleware, die im Fontane Projekt SaPiMa genannt wird [14]. Die beiden restlichen Komponenten, die Planung und die Ausführung der notwendigen Anpassungen sind dagegen konzeptioneller Bestandteil jeder Anwendung (vgl. Abbildung 1).

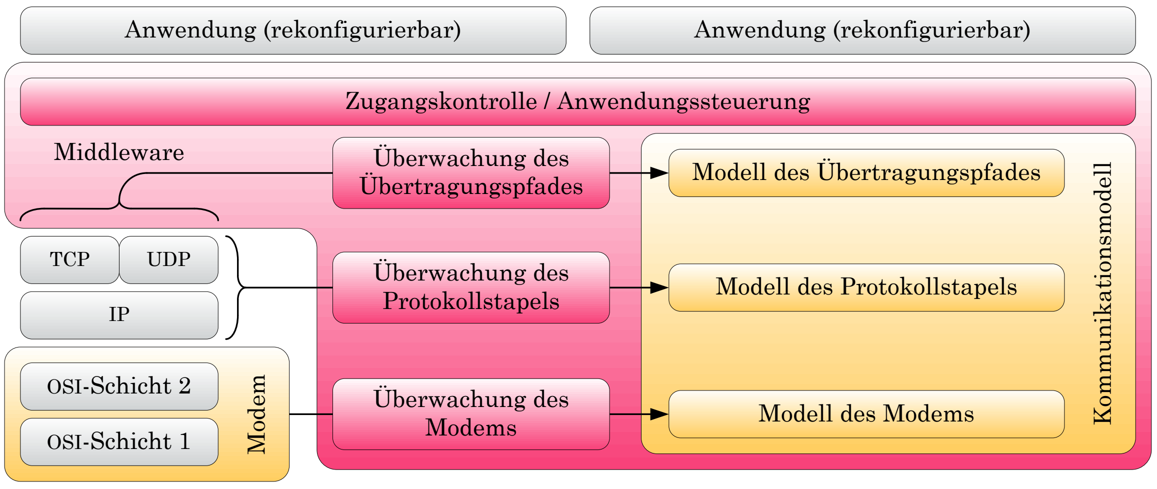

Anwendung (rekonfigurierbar)
Anwendung (rekonfigurierbar)
Zugangskontrolle / Anwendungssteuerung
Middleware
Überwachung des Übertragungspfades
Modell des Übertragungspfades
TCP
UDP
IP
Überwachung des Protokollstapels
Modell des Protokollstapels
OSI-Schicht 2
OSI-Schicht 1
Modem
Überwachung des Modems
Modell des Modems
Kommunikationsmodell

Abbildung 1: Modell der Steuerung auf Anwendungsebene (Client-Seite)

Die Schnittstelle zwischen der Middleware und den Anwendungen wird durch eine Zugangskontrolle (engl. *Call Admission Control*) gebildet. Bevor eine Anwendung Daten über das Netzwerk austauschen kann, muss sie der Zugangskontrolle ihren Kommunikationswunsch unter Angabe ihrer Anforderungen an die Übertragungsstrecke mitteilen. Anforderungen sind die Priorität der zu sendenden Daten und die unterstützten Übertragungsraten. Die Kontrollinstanz überprüft dann, ob die Kommunikationsumgebung einen gewünschten Datenaustausch ermöglichen kann oder nicht und akzeptiert die neue Datenverbindung oder weist sie ab (vgl. Abschnitt 1.2).

Eine weitere Aufgabe der Zugangskontrolle ist die Überwachung der Kommunikationsendpunkte. Sie kontrolliert, dass die vereinbarten Parameter der Kommunikation eingehalten werden, um eine ungewollte gegenseitige Beeinflussung der Datenverbindungen zu vermeiden. Dies wird erreicht, indem die maximal durch die jeweilige Anwendung benutzbare Übertragungsrate beschränkt wird.

Die Überwachung der Kommunikation teilt sich im Umfeld zellularer Funknetze in die drei Hauptbestandteile:

- Überwachung des Übertragungspfades
- Überwachung des Protokollstapels im Endgerät
- Überwachung des Funk-Modems (nur auf der Client-Seite)

Die Überwachung des Übertragungspfades basiert auf Ende-zu-Ende Messungen der verfügbaren Datenübertragungsrate. Die Überwachung des Protokollstapels erfolgt mit Hilfe der Funktionalität, die die pcap-Bibliothek[1] und das Socket-API bieten. Für die Überwachung des Funkmodems werden AT-Kommandos benutzt.

Zur Analyse der Ergebnisse der Überwachung wird ein modell-basierter Ansatz gewählt. Das Modell besteht aus drei Teilmodellen, die den jeweils überwachten Teilkomponenten der Kommunikation – dem Übertragungspfad, dem Protokollstapel und dem Funk-Modem – entsprechen. Jedes dieser Modelle leitet aus den Daten der Überwachung eigene Parameter ab, die dann zur Beurteilung des Umfeldes der Kommunikation benutzt werden. Die Middleware löst bei Änderungen des Kommunikationsumfeldes die Anpassung der jeweiligen Anwendungen aus.

1.2 Aufteilung der verfügbaren Übertragungsrate

Die Zugangskontrolle teilt die verfügbare Übertragungsrate entsprechend der Priorität der Daten auf die Anwendungen auf, die bereit sind Daten zu übertragen. Dafür erhält sie mit der Anmeldung des Übertragungswunsches von jeder Anwendung die Datenpriorität und eine Liste der unterstützten Übertragungsraten. Für die Aufteilung der verfügbaren Übertragungsrate gibt es zwei grundsätzliche Verfahren, deren Ziel es ist, dass entweder nur die Anwendungen mit den jeweils höchsten Datenprioritäten ihre Daten übertragen dürfen, oder die Zahl der Anwendungen, die ihre Daten übertragen dürfen maximiert werden soll.

Variante 1: Wenn nur die Anwendungen mit den jeweils höchsten Datenprioritäten ihre Daten übertragen sollen, dann muss die Zugangskontrolle die folgenden Schritte ausführen:

1. Weise der Anwendung mit der höchsten Datenpriorität die größtmögliche Übertragungsrate aus der Liste zu. Gibt es mehrere Anwendungen mit dieser Datenpriorität, dann:

   (a) Weise jeder von Ihnen die minimale Übertragungsrate aus der jeweiligen Liste zu.

   (b) Erhöhe die zugewiesene Übertragungsrate jeder Anwendung bis entweder die verfügbare Übertragungsrate vollständig ausgeschöpft oder die maximale Rate für jede Anwendung erreicht ist. Ist es nicht möglich, jeder Anwendung ihre maximale Übertragungsrate zuzuweisen, soll jeder dieser Anwendungen eine annähernd gleich große Datenrate ermöglicht werden.

2. Wiederhole den Schritt 1 mit der Anwendung mit der nächstkleineren Datenpriorität, bis entweder die verfügbare Übertragungsrate komplett aufgeteilt ist oder allen Anwendungen die größte unterstützte Rate zugewiesen wurde.

Variante 2: Wenn die verfügbare Übertragungsrate möglichst auf alle Anwendungen aufgeteilt werden soll, dann müssen die folgenden Schritte ausgeführt werden:

[1] Weitere Informationen sind für Windows Systeme unter `http://www.winpcap.org/` bzw. für alle anderen Betriebssysteme unter `http://www.tcpdump.org/` zu finden.

1. Weise der Anwendung mit der höchsten Datenpriorität die niedrigste Datenrate aus der Liste zu. Gibt es mehrere Anwendungen mit dieser Datenpriorität, dann verfahre mit allen anderen auf die gleiche Weise bis die verfügbare Übertragungsrate ausgeschöpft ist oder allen Anwendungen kleinste unterstützte Rate zugewiesen wurde.

2. Wiederhole den Schritt 1 mit der Anwendung mit der nächstkleineren Datenpriorität, bis entweder die verfügbare Übertragungsrate vollständig aufgeteilt ist oder allen Anwendungen die kleinste unterstützte Rate zugewiesen wurde.

3. Erhöhe die zugewiesenen Übertragungsrate der Anwendung mit der höchsten Datenpriorität auf den größtmöglichen Wert aus der zugehörigen Liste. Gibt es mehrere Anwendungen mit dieser Datenpriorität, dann erhöhe die zugewiesene Übertragungsrate bis entweder die verfügbare Übertragungsrate vollständig ausgeschöpft oder die maximale Rate für jede Anwendung erreicht ist. Ist es nicht möglich, jeder Anwendung ihre maximale Übertragungsrate zuzuweisen, soll jeder dieser Anwendungen eine annähernd gleich große Datenrate ermöglicht werden.

4. Wiederhole den Schritt 3 mit der Anwendung mit der nächstkleineren Datenpriorität, bis entweder die verfügbare Übertragungsrate komplett aufgeteilt ist oder allen Anwendungen die größte unterstützte Rate zugewiesen wurde.

1.3 Überwachung des Modems

Die Daten werden vom Funkmodem mit Hilfe von AT-Kommandos abgefragt, die mittels eines virtuellen seriellen Ports über eine USB-Verbindung gesendet werden. Die USB-Schnittstelle wird sowohl für das interne als auch das externe Modem verwendet. Das Format der Daten und deren Umfang hängt stark vom verwendeten Modem ab. Es gibt zwar eine Vielzahl standardisierter AT-Kommandos, aber ihre Implementierung ist größtenteils optional [1,3]. Gleiches gilt für die von den Abfragekommandos zurückgelieferten Parameter. Zudem kann der Hersteller eines Funkmodems auch zusätzliche selbst definierte AT-Kommandos zur Verfügung stellen.

Je nach Modem stehen demzufolge unterschiedliche Informationen zur Verfügung. Beim GenPro 35e von Erco & Gener, einem externen Funkmodem, kann man mit Hilfe der Kommandos *AT!GSTATUS?*, *AT!GSMINFO?*, *AT+COPS?*, *AT!GETRAT?*, *AT+CSQ*, *AT*CNTI=0*, *AT+USET?x*, *AT+RSCP?*, *AT+ECIO?*, *AT!HSDCAT?* und *AT!HSUCAT?* z. B. die folgenden Daten ermitteln:

- Zugriffstechnologie (*Global System for Mobile Communications* (GSM), *Universal Mobile Telecommunications System* (UMTS), ...)

- Übertragungsverfahren (*Enhanced Data Rates for* GSM *Evolution* (EDGE), *High Speed Uplink Packet Access* (HSUPA), ...)

- Ortsinformationen (*Location Area Code* (LAC), *Base Station Color Code* (BCC), ...)

- Statusinformationen von Protokollautomaten (GPRS *Mobility Management* (GMM), *Radio Resource Control* (RRC), ...)
- Empfangspegel für die aktuell genutzte Zelle und ihre Nachbarn
- Auswahlkriterien beim Wechsel einer Zelle (*Path loss criterion* C1, *Reselection criterion* C2, ...)

Das Modem HP un2400 stellt im Vergleich dazu deutlich weniger Informationen zur Verfügung. Mit den Kommandos *AT+COPS?*, *AT+CGREG?*, *AT+CGATT?*, *AT+CSQ* und *AT*CNTI=0* lassen sich folgende Daten ermitteln:

- Zugriffstechnologie
- Anschluss an die paketvermittelnden Domäne
- Empfangspegel der aktuellen Zelle (*Received Signal Strength Indicator* (RSSI))

In *Smartphones* mit *Windows Phone* als Betriebssystem gibt es eine andere Möglichkeit Informationen aus den untersten Schichten des Protokollstapels zu erhalten – das *Application Programming Interface* (API) des *Microsoft .NET Framework*. Die zur Verfügung gestellten Informationen entsprechen im Umfang etwa denen des Modems von HP, sind inhaltlich aber weniger präzise. Folgende Daten können ermittelt werden:

- Zugriffstechnologie (GSM nur indirekt über das Übertragungsverfahren)
- Verfügbare und aktuell genutztes Übertragungsverfahren (*General Packet Radio Service* (GPRS), EDGE, *High Speed Downlink Packet Access* (HSDPA))
- Anschluss an die paketvermittelnden Domäne
- Empfangspegel der aktuellen Zelle als prozentualer Anteil der vollen Signalstärke

1.4 Modell des Modems

Das Modemmodell leitet aus den Daten, die vom Funkmodem gewonnen werden können, Informationen zu Ressourcen des Zugangsnetzes ab. Da in Abhängikeit des verwendeten Modems die zur Verfügung gestellte Datenbasis sehr stark variieren kann, sind auch die, aus dem Modell gewonnen Informationen, stark vom eingesetzten Funkmodem abhängig. Im Rahmen dieser Arbeit wurde die Mehrzahl der Analysen mit dem externen Funkmodem durchgeführt.

In Abhängigkeit der genutzten Zugriffstechnologie werden unterschiedliche technische Verfahren auf der Luftschnittstelle eingesetzt – z. B. *Frequency Division Duplex* (FDD) und *Time Division Multiple Access* (TDMA) bei GSM aber FDD und *Wideband* CDMA (WCDMA) bei UMTS. Sie wird also auf die grundlegenden Übertragungsparameter, wie Übertragungsrate und Übertragungsdauer, Einfluss ausüben.

Das aktuelle Übertragungsverfahren, z. B. GPRS oder *Enhanced* GPRS (EGPRS) im Fall von GSM, und das verwendete Modulations- und Kodierungsschema (MKS) bestimmen die maximal mögliche Datenübertragungsrate der logischen Kanäle auf der

Luftschnittstelle [5]. Dieser Punkt kann im Rahmen der Arbeit jedoch nicht umgesetzt werden, da bei keinem der eingesetzten Funkmodems das verwendete MKS abgefragt werden kann.

Die Ortsinformationen geben den Aufenthaltsort des mobilen Endgerätes im Funknetz an. Sie können lokale und globale Gültigkeit haben. Globale Bezeichner, wie die *Cell Global Identification* (CGI), beschreiben den Ort eindeutig innerhalb des gesamten Funknetzes; lokale Bezeichner dagegen, wie der *Base Station Identity Code* (BSIC), sind nur im Umkreis des jeweiligen Strukturelementes, hier eine Basisstation, gültig und können innerhalb des Funknetzes mehrfach verwendet werden. Mit Hilfe der Ortsinformationen können Zellwechsel erkannt und die Endgeräte in Abhängigkeit ihres Aufenthaltsortes gruppiert werden.

Parameter, die den Status von Protokollautomaten beschreiben, können zur Bestimmung von Zustandes der Kommunikationsverbindung genutzt werden. Damit lässt sich z. B. erkennen, ob dem Endgerät ein Kanal zur Übertragung von Daten zugewiesen ist oder nicht, oder ob gerade ein Zellwechsel stattfindet.

Die Auswahl einer Zelle erfolgt oft anhand des Pegels des Empfangssignals. Ein hoher Empfangspegel macht ein hohes Signal-Rausch-Verhältnis und damit eine gute Signalqualität wahrscheinlich. Deshalb bilden die Empfangssignalpegel die Grundlage zur Bestimmung der Auswahlkriterien für die zu benutzende Zelle. Im Fall von GSM muss vor einem Zellwechsel das Kriterium C1 (*Path loss criterion*) erfüllt sein [4]. Es wird wie folgt berechnet [6]:

$$\begin{aligned} \mathrm{C1} &= \mathrm{A} - \max(\mathrm{B}, 0) \\ \mathrm{A} &= \mathrm{RLA\_C} - \mathrm{RXLEV\_ACCESS\_MIN} \\ \mathrm{B} &= \mathrm{MS\_TXPWR\_MAX\_CCH} - \mathrm{P} \end{aligned}$$

wobei alle Parameter in dBm angegeben werden. Der Wert A entspricht dabei dem, mit dem minimal notwendigen Empfangssignalpegel (RXLEV_ACCESS_MIN) bewerteten, gleitenden Mittelwert des gemessenen Pegels des empfangenen Signals (RLA_C). Er ist ein Maß für die Qualität des Empfangssignals. Der Wert B gibt an in welchem Verhältnis die maximal mögliche Sendeleistung des mobilen Endgerätes (P) und die maximal notwendigen Leistung liegt, die benötigt wird, um mit dem Netzwerk Informationen auszutauschen (MS_TXPWR_MAX_CCH). Er ist ein Maß für die Leistungsreserve des Senders. Das Kriterium C1 ist erfüllt, wenn der berechnete Wert größer als Null ist. Mit Kenntnis dieser Parameter von allen Zellen, die gerade vom Endgerät empfangbar sind, lassen sich so Zellwechsel mit gewisser Wahrscheinlichkeit vorhersagen.

1.5 Überwachung des Übertragungspfades

Bei den aktiven Ende-zu-Ende Messungen muss beachtet werden, dass die Messungen parallel zur Übertragung der Nutzerdaten durchgeführt werden sollen. Es ist also nicht möglich, eine eigenständige Datenquelle zu benutzen, die die Kapazität des Übertragungspfades voll auslastet, oder davon auszugehen, dass während des gesamten Messzyklus keine weiteren Daten gesendet werden. Deshalb wurde ein Verfahren eingesetzt, das auf der Übertragung von Paketpaaren beruht [10]. Die beiden Pakete müssen die gleiche Größe haben und direkt nacheinander gesendet werden.

Diese Bedingungen lassen sich auf zwei Wegen erreichen: zum einen kann man in bestimmten Abständen ein Paket aus dem Datenstrom der Anwendungen verdoppeln, indem man es zweimal nacheinander überträgt, und zum anderen kann man den Datenstrom neu formatieren und somit gleichgroße Pakete mit unterschiedlichem Inhalt erzeugen. Der zweite Ansatz vermeidet es die zu übertragende Datenmenge zu Messzwecken zu vergrößern; seine Umsetzung ist aber aufwändiger. Im Rahmen der vorliegenden Arbeit wurde deshalb der erste Ansatz umgesetzt. Sollen Ende-zu-Ende Messungen durchgeführt werden, wenn keine Anwendungsdaten zur Verfügung stehen, dann können die benötigten Datenpakete in vorgegebenen zeitlichem Abständen von der Messumgebung generiert werden.

Um den Empfänger des Datenstromes in die Lage zu versetzen, die Messdatenpakete zu erkennen und korrekt zu behandeln, müssen diese entsprechend gekennzeichnet werden. Dabei sind die folgenden drei Informationen zu berücksichtigen:

- Ist das Datenpaket Teil einer Ende-zu-Ende Messung oder nicht
- Handelt es sich bei einem Messdatenpaket um das Erste oder das Zweite Paket des Paares
- Stammt das Paket aus dem Strom der Anwendungsdaten oder wurde es speziell für die Messung erzeugt

Da alle drei Informationen binären Charakter haben, wurden zur Kennzeichnung der Daten drei Bits im Kopf jedes Paketes benutzt.

1.6 Kommunikationsmodell

Das Kommunikationsmodell liefert einen Schätzwert für die aktuell verfügbare Datenübertragungsrate. Dazu bedient es sich Informationen, die aus drei Bereichen des Protokollstapels gewonnen werden. Aufgrund der unterschiedlichen Ausrichtung der Kommunikation in den drei Bereichen, basiert das Kommunikationsmodell auf drei Teilmodellen – jeweils eines für jeden Informationsbereich.

Der Schätzwert der verfügbaren Datenübertragungsrate wird mit den Ergebnissen der Ende-zu-Ende Messungen und dem Wissen über den aktuellen Verbindungszustand bestimmt. Letzteres setzt sich aus Informationen über die physische Verbindung (OSI-Schicht 1) und über die Verbindung auf Ebene der Transportschicht (OSI-Schicht 4) zusammen. Der Zustand einer Verbindung in der Transportschicht wird aus dem Modell des Protokollstapels gewonnen und nur bei der Nutzung eines verbindungsorientierten Protokolls, wie *Transmission Control Protocol* (TCP), analysiert. Der Zustand der physischen Verbindung wird vom Modell des Modems zur Verfügung gestellt und stützt sich darauf, ob aktuell ein Zellwechsel im Funknetz stattfindet oder demnächst einer erwartet wird (vgl. Algorithmus 1.1).

Welchen Einfluss haben Zellwechsel auf den Zustand der physischen Verbindung? Ein Wechsel der Zelle kann als *Hard* oder *Soft Handover* durchgeführt werden. Im Fall eines *Hard Handover* existiert während des Wechsels keine physische Datenverbindung mit dem Netzwerk, da zuerst die alte, bestehende Verbindung beendet und erst danach eine neue aufgebaut wird. Im Fall eines *Soft Handover* dagegen wird erst

```
Eingabe: ein bei der Ende-zu-Ende Messung ermittelter Wert [bit/s],
    Informationen über die Verbindung in der OSI-Schicht 4,
    Informationen zu Zellwechseln im Funknetz
Ausgabe: Schätzwert der verfügbaren Datenübertragungsrate [bit/s]

1: $R \leftarrow 0$
2: if (aktuell wird kein Zellwechsel durchgeführt) and
   (demnächst wird kein Zellwechsel erwartet) and
   ((ein verbindungsloses Protokoll wird benutzt) or
   (die Übertragung auf OSI-Schicht 4 erfolgt momentan störungsfrei)) then
3:     $R \leftarrow$ Wert der Ende-zu-Ende Messung
4: end if
5: return $R$
```

Algorithmus 1.1: Algorithmus zur Schätzung der verfügbaren Übertragungsrate im Kommunikationsmodell

eine zweite, neue Verbindung zum Netz aufzubauen bevor die alte beendet wird. Da also jeder Zellwechsel mit einem kurzzeitigen Verlust der physischen Datenverbindung einhergehen kann, wird der Schätzwert der Übertragungsrate während dieser Zeit generell mit Null angenommen.

Wieso wird die Qualität der Verbindung in der OSI-Schicht 4 berücksichtigt? Bei der Nutzung von TCP deutet das wiederholte Senden von Daten auf Probleme innerhalb des Übertragungspfades hin. Mögliche Ursachen sind überlastete Knoten oder fehlende Verbindungen im Netzwerk. Bedingt durch ihre Funktion verwaltet jede TCP-Implementierung intern die noch zu sendenden und die noch nicht bestätigten Daten. Führt eine schlechte Verbindungsqualität aus Sicht von TCP zum Abbruch der Verbindung erhält die Anwendung über Standardschnittstellen, wie das Socket-API, keine Rückmeldung, welche der Daten noch nicht übertragen wurden. Aus diesem Grund ist es sinnvoll, der Transportschicht keine neuen Anwendungsdaten zu übergeben, wenn die intern noch gespeicherten Daten wiederholt übertragen werden.

2 Synchronisation der Endgeräte im Netzwerk

Die Endgeräte auf der Klientenseite sollen Daten entsprechend ihrer Dringlichkeit an den zentralen Server übermitteln und sich dabei in die gemeinsam genutzten Netzwerkressourcen teilen. Dafür muss sich das Kommunikationsverhalten der Endgeräte entsprechend regeln lassen – wir benötigen ein System zum Austausch von Informationen zwischen den Endgeräten. Da die Endgeräte der Klientenseite nur Kommunikationsverbindungen mit dem Server haben, wird dieser in den Informationsaustausch einbezogen [11].

Ziel der Synchronisierung ist es, die Übertragung von Nicht-Echtzeit-Daten einzuschränken, wenn Daten höherer Dringlichkeit im gleichen Bereich des Netzwerkes übertragen werden. Die Übertragung von Echtzeitdaten wird dabei nicht reglementiert. Damit wird die Datenpriorität aller Endgeräte im betrachteten Netzwerkbereich berücksichtigt ohne die Übertragung von kritischen Daten zu beschränken.

Gemeinsam genutzte Ressourcen treten in zellularen Netzen in jedem Fall in Abhängigkeit der Zellenstruktur auf, da innerhalb einer Zelle alle mobilen Endgeräte mit dem Punkt im Zugangsnetz kommunizieren, der die Zelle versorgt (vgl. Abbildung 2). Welche Ressourcen in welcher Weise gemeinsam genutzt werden, ist stark von der Netzstruktur und der Zugriffstechnologie abhängig. GSM verwendet z. B. unterschiedliche Trägerfrequenzen für den Zugriff in benachbarten Zellen und die Zellen sind nicht ineinander geschachtelt; UMTS dagegen verwendet unterschiedliche Spreizcodes und kann verschachtelte Zellstrukturen benutzten. Die Synchronisation der mobilen Endgeräte basiert deshalb auf Informationen über die Struktur des Netzwerkes.

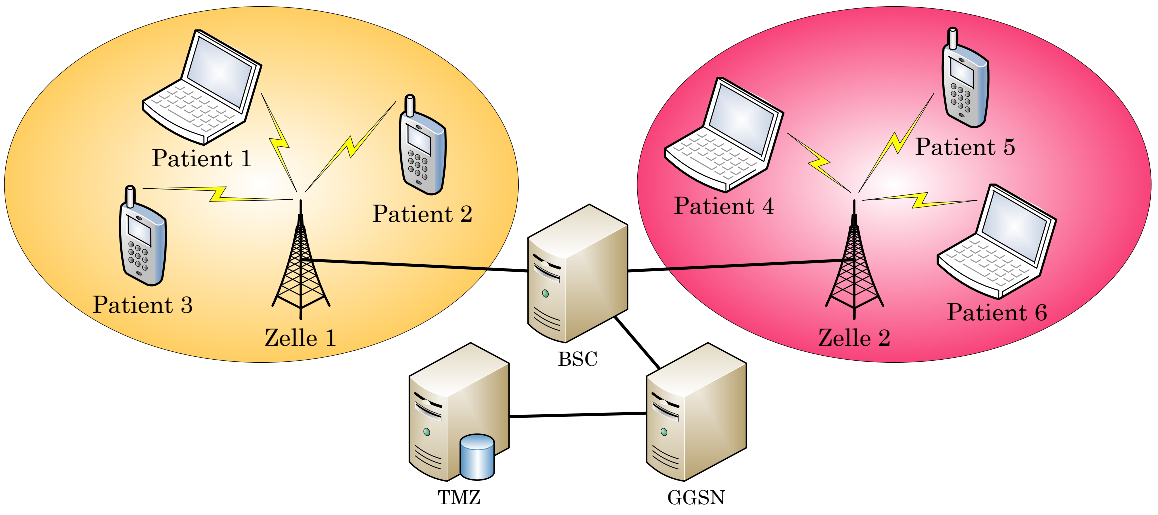


Abbildung 2: Prinzip der Synchronisation der unterschiedlichen Endgeräte im zellularen Netzwerk

Das grundlegende Prinzip der Synchronisation ist recht einfach: wenn die Datenkommunikation jedes einzelnen Endgerätes steuerbar, der Aufenthaltsort jedes Endgerätes im Netzwerk bekannt und ein zentraler Punkt zur Koordination vorhanden ist, dann lässt sich auch die Kommunikation der Geräte untereinander steuern. Wie lassen sich diese Punkte im einzelnen erfüllen? Die Middleware in den Endgeräten steuert die Datenkommunikation innerhalb eines Gerätes (vgl. Abschnitt 1). Der Aufenthaltsort im Netzwerk kann mit Hilfe der CGI bestimmt werden [7]. Dazu muss jedes Endgerät den für ihn gültigen Wert ermitteln und an den zentralen Koordinationspunkt übertragen. Der Server im Telemedizinzentrum (TMZ) kann zu Koordinationszentrale benutzt werden. Er bestimmt anhand der Datenpriorität und der Ortsinformationen die erforderlichen Steuerinformationen.

Als Basisstruktur zur Steuerung der Datenkommunikation im Netzwerk dienen *Location Areas* (LAs). Hier stellt sich die Frage: warum werden LAs und nicht die benutzten Zellen, innerhalb derer Ressourcen gemeinsam benutzt werden, für die Synchronisation benutzt? Der Grund dafür ist die Optimierung des Synchronisationsprozesses bezüglich das Menge und der Häufigkeit der auszutauschenden Daten – Zellwechsel treten deutlich häufiger auf als Wechsel des LA.

Die Größe jedes LA wird vom Netzbetreiber festgelegt und soll zur Verringerung des Datenverkehrs für die Aktualisierung der Ortsinformationen im Netzwerk beitragen.

Vom *3rd Generation Partnership Project* (3GPP) wird ein LA wie folgt definiert:

> *The Location Area (LA) is defined as an area in which a mobile station may move freely without updating the VLR. A location area includes one or several GERAN/UTRAN cells.* [9]

> *A location area is an area in which, after having performed a location update once, MSs may roam without being required to perform subsequent location updates for reason of location change. A location area consists of one or more cells.* [8]

Ein zweiter Punkt, der hier beachtet werden muss, ist der Aufbau einer Verbindung zu einem mobilen Endgerät. Will das Netzwerk eine solche Verbindung aufbauen, muss die Zelle bekannt sein, von der das Endgerät versorgt wird. Da das Endgerät das Netzwerk nur auf Ebene von LAs über seinen Aufenthaltsort informiert, sendet das Netzwerk eine Broadcast-Nachricht im gesamten LA, um das Endgerät zu suchen. Dieser Vorgang wird als *Paging* bezeichnet [2]. Im Fall von GPRS wird die *Paging* Prozedur wie folgt beschrieben:

> *In A/Gb mode, paging is used by the network to identify the cell the MS has currently selected, or to prompt the mobile to re-attach if necessary as a result of network failure. ... In Iu mode, paging is used by the network to request the establishment of PS signalling connection or to prompt the mobile to re-attach if necessary as a result of network failure.* [2]

LAs sollen dazu beitragen, die Übertragung von Signalisierungsinformationen zwischen dem mobilen Endgerät und dem Netzwerk zu optimieren. Die Größe eines LA ist also ein Kompromiss: es muss groß genug sein, um die Anzahl der Aktualisierungsmeldungen zu reduzieren und es muss klein genug sein, um die Suche nach einem Gerät noch schnell und effizient zu gestalten. Es liegt also nahe anzunehmen, dass die Nutzung vonLA anstatt von Zellen auch zu einer besseren Synchronisation der Endgeräte im Netzwerk führt.

Die Synchronisation der Endgeräte wurde experimentell durch Einführung eines Steuerprotokolls und eines Kontrollalgorithmus umgesetzt und mit Hilfe von OMNet++ (siehe auch `http://www.omnetpp.org/`) simuliert.

References

[1] 3GPP. Technical Specification Group Core Network and Terminals; AT command set for User Equipment (UE) (3GPP TS 27.007 V10.4.0 (Release 10)), June 2011.

[2] 3GPP. Technical Specification Group Core Network and Terminals; Mobile radio interface Layer 3 specification; Core network protocols; Stage 3 (3GPP TS 24.008 V10.3.0 (Release 10)), June 2011.

[3] 3GPP. Technical Specification Group Core Network and Terminals; Use of Data Terminal Equipment - Data Circuit terminating Equipment (DTE - DCE) interface for Short Message Service (SMS) and Cell Broadcast Service (CBS) (3GPP TS 27.005 V10.0.0 (Release 10)), March 2011.

[4] 3GPP. Technical Specification Group GSM/EDGE Radio Access Network; Functions related to Mobile Station (MS) in idle mode and group receive mode (3GPP TS 43.022 V10.0.0 (Release 10)), March 2011.

[5] 3GPP. Technical Specification Group GSM/EDGE Radio Access Network; Physical layer on the radio path; General description (3GPP TS 45.001 V10.1.0 (Release 10)), November 2011.

[6] 3GPP. Technical Specification Group GSM/EDGE Radio Access Network; Radio subsystem link control (3GPP TS 45.008 V10.1.0 (Release 10)), May 2011.

[7] 3GPP. Technical Specification Group Services and System Aspects; Vocabulary for 3GPP Specifications (3GPP TR 21.905 V10.3.0 (Release 10)), March 2011.

[8] 3GPP. Technical Specification Group Core Network and Terminals; Location management procedures (3GPP TS 23.012 V11.0.0 (Release 11)), June 2012.

[9] 3GPP. Technical Specification Group Services and System Aspects; Network architecture (3GPP TS 23.002 V11.3.0 (Release 11)), June 2012.

[10] Constantinos Dovrolis, Parameswaran Ramanathan, and David Moore. Packet-Dispersion Techniques and a Capacity-Estimation Methodology. *IEEE/ACM Transactions on Networking*, 12:963–977, December 2004.

[11] Uwe Hentschel, Fahad Khalid, and Andreas Polze. An Approach to Control Transmission of Medical Data over Cellular Networks Using Location Information. In *Proceedings of the 2012 15th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*, ISORC '12, pages 87–94, Washington, DC, USA, 2012. IEEE Computer Society.

[12] IBM. An architectural blueprint for autonomic computing, 2003.

[13] Jeffrey O. Kephart and David M. Chess. The Vision of Autonomic Computing. *Computer*, 36:41–50, January 2003.

[14] Friedrich Köhler. Gesundheitsregion der Zukunft – Nordbrandenburg [FONTANE], 2009.

Heterogeneous Computing for Algorithms with Low Numerical Intensity

Fahad Khalid
Operating Systems and Middleware Group
Hasso-Plattner-Institut
fahad.khalid@hpi.uni-potsdam.de

The Scientific Computing community has witnessed a significant increase in applications that utilize GPUs as accelerators; complementing CPU based processing. However, the feasibility of porting an algorithm to the GPU is dependent on a complex combination of factors that include numerical intensity, data access patterns and data reuse. E.g. algorithms with very low numerical intensity are less likely to gain performance if implemented on a GPU.

In this report, a problem from Systems Biology is taken as a case study for utilizing heterogeneous computing to improve the performance of an algorithm with low numerical intensity. It is shown that the Map-Reduce structural pattern for parallel applications can be used to split the algorithm into two phases; where the Map phase can be optimized for a GPU implementation, while the Reduce phase can be efficiently implemented on the CPU. The result is a significant performance gain over a serial CPU-only implementation.

The primary objective of this research project is to determine performance improvement strategies that can be generalized to broader classes of low numerical intensity algorithms. If so, heterogeneous computing can be utilized to accelerate processing for a much larger set of problems.

1 Introduction

In recent years, research community in the High Performance Computing (HPC) and Scientific Computing sectors has witnessed an increasing application of Heterogeneous Computing [10]. The term Heterogeneous Computing (as used in this document) refers to the design of applications that can harness the power of both the CPU and the GPU for problems amenable to massive parallelism.

The rapid adaptation of the scientific community to Heterogeneous Computing is often attributed to the GPUs'capability to perform massive amounts of arithmetic operations, at very high speeds. Majority of the silicon chip area in a GPU is dedicated to a number of arithmetic processing units. This makes it possible for the GPU to execute a very large number of arithmetic instructions in parallel. However, since most of the chip area is committed to arithmetic processing, only small amounts of low latency memory resides on-chip. Therefore, a memory hierarchy (similar to the traditional CPU based systems) is employed, but with very small sizes for low latency memory as compared to the CPU.

A significant number of algorithms have been successfully ported to GPUs, attaining up to 100x speedup over serial execution on a CPU [7]. However, due to the aforementioned architectural limitations (w.r.t. low latency memory size), this success holds only for a certain set of algorithms that can take full advantage of the GPU architecture. This set of algorithms typically shows one important execution characteristic; i.e. high arithmetic/numerical intensity (compute-to-memory access ratio). The most common example for such an algorithm is matrix-matrix multiplication [1].

Nevertheless, even for algorithms with high numerical intensity, porting and optimizing the algorithm is a tedious task that requires investing a significant amount of time and effort. Moreover, several important algorithms have low numerical intensity e.g. Sparse Matrix-Vector Multiplication [2]. With the advent of Big Data revolution, the significance of such algorithms is further amplified; since this involves algorithms associated with processing of very large datasets. For such algorithms, CPU-only architectures (equipped with the much larger low latency memory) would appear more suitable.

In the context of Heterogeneous Computing, the above considerations raise the following important question:

- Is it possible to effectively utilize Heterogeneous Computing for algorithms with low numerical intensity? Or must such algorithms be executed on CPU-only systems?

In the sections to follow, the above posed question is approached by looking at a low numerical intensity algorithm from the domain of Systems Biology. The algorithm has already been successfully parallelized on both shared-memory [13] and distributed-memory [4] CPU based architectures. Here, a heterogeneous architecture based parallelization is presented.

2 Heterogeneous Computing for Enumeration of Elementary Flux Modes

The set of all Elementary Flux Modes (EFM) represents the complete set of minimal pathways in the metabolic network of an organism [11]. Under steady-state conditions, the problem of enumerating EFMs is mathematically equivalent to the enumeration of extreme rays of polyhedral cone [3]. The Nullspace algorithm [14] is known to be the most efficient algorithm for EFM enumeration. It consists of the following steps:

1. Solve a system of linear equations to get the Kernel matrix
2. Process the Kernel matrix to remove redundancies and permute it so that it is feasible for further processing
3. For each row in the Kernel matrix (EM Matrix):
    (a) Generate Candidate EM vectors

(b) Verify elementarity using Rank tests

(c) Update the EM matrix

The most compute intensive part of the algorithm is Candidate Generation. Following is the pseudocode for the serial candidate generation algorithm [4]:

```
❑ Input: Matrix A, Matrix B – where both are bit matrices
❑ Output: Candidate column pairs of the form:
    ❑ $\{(a,b) | a \in A \text{ and } b \in B\}$
❑ $For\ each\ column\ in\ MatA$
    ❑ $For\ each\ column\ in\ Mat$B
        ❑ $candidate = col(A) \vee col(B)$
        ❑ $nonZeros = popCount(candidate)$
        ❑ $If\ (nonZeros \leq \text{maxNonZerosAllowed})$
            ❑ $Keep\ the\ index\ pair\ corresponding\ to\ (a,b)$
Note: Index values are retrieved from another pair of arrays available as input
```

The core of the algorithm is based on an element-wise binary operation (bitwise OR) between the two input matrices. The $popCount()$ function is implemented as a lookup operation that computes the Hamming weight i.e. the number of set/on bits in the vector. Let the size of Matrix A be m, and size of Matrix B be n, then the total number of binary operations is $m \times n$. In the worst case, this leads to a result matrix that grows quadratically as a function of input size.

As can be inferred from the above description, the serial Nullspace algorithm has very low numerical intensity. In the sections to follow, the experience of parallelizing the Nullspace algorithm for execution on NVIDIA GPUs is presented.

2.1 Parallel Candidate Generation Model for GPU

Since the generation of each candidate vector is independent of the others, the parallelization strategy used is one where each thread computes a single candidate vector. This is a data parallel model that results in a massively parallel GPU application.

2.2 Memory Partitioning and Multiple Grid Invocations

The input size for the Nullspace algorithm grows very fast [6] and as a result just a few iterations lead to a very large size for the result data structure. The GPU global memory is quite limited; only 1GB for GTX 295. Therefore, a mechanism is required to divide the result data structure into smaller partitions that easily reside in the GPU global memory.

Even though partitioning is an obvious procedure, it has a rather important consequence in terms of performance. Ideally, a single kernel invocation should execute the entire candidate generation algorithm. However, the use of partitioning results into at least one kernel invocation per partition. This in turn leads to a lower bound on the number of kernel invocations, which is equal to the number of partitions.

$$lower\ bound\ on\ number\ of\ kernel\ invocations\ =\ number\ of\ partitions$$

For each kernel invocation, the result values have to be copied from the device to the host. This adds a performance penalty.

Moreover, there is an upper bound on the maximum number of blocks per grid. For GTX 295 as well as for Tesla 2050, this limit is 65535. The upper bound on the number of threads per grid is then derived as:

$$\begin{aligned} &upper\ bound\ on\ number\ of\ threads\ per\ grid\ = \\ &\qquad number\ of\ threads\ per\ block\ \times\ maximum\ number\ of\ blocks\ per\ grid \end{aligned}$$

The datasets computed by the Nullspace algorithm are very large, and as a result, multiple girds must be utilized. Therefore, for large input sizes, not only are multiple partitions required, multiple grid invocations per partition are necessary as well.

2.3 Index Algebra

In order for each thread to correctly index into the input matrices and the distributed result data structure, an Index Algebra is required. The unique thread ID for the corresponding element of the result matrix is given by:

$$\begin{aligned} indexMatC\ &= (blockIndex\ \times\ blockDim\ +\ threadIndex) \\ &+ (gridId\ \times\ gridDim\ \times\ blockDim) \end{aligned}$$

This thread ID is unique within a single partition. A thread ID unique over all partitions is then derived as:

$$uniqueId\ =\ indexMatC\ +\ (partitionId\ \times\ partitionDim)$$

The input matrices are indexed using Modular arithmetic. The required values are computed as follows:

- $period = sizeOf(MatB) \div columnLength$
- $indexMatA = (uniqueId \, / \, period) \times columnLength$
- $indexMatB = (uniqueId \,\%\, period) \times columnLength$

Integer division (for $indexMatA$) ensures that the value increments only once per period. For $indexMatB$, the value is incremented by one column per thread. However, at each period, the value is reset to 0. This is why the modulus operator is used.

2.4 Naïve Kernel

Following is the pseudocode for the kernel based on memory partitioning with the given index algebra:

```
❑ $indexMatC = (blockIndex \times blockDim + threadIndex)$
                    $+(gridId \times gridDim \times blockDim)$
❑ $uniqueId = indexMatC + (partitionId \times partitionDim)$
❑ $indexMatA = (uniqueId \, / \, period) \times columnLength$
❑ $indexMatB = (uniqueId \,\%\, period) \times columnLength$
❑ $candidate = MatA[indexMatA] \vee MatB[indexMatB]$
❑ $nonZeros = popCount(candidate)$
❑ $result[indexMatC] = (nonZeros \leq maxNonZeros)$
```

Given the massively parallel nature of the GPU, it would appear that the above kernel would perform magnitudes faster than the serial CPU-only algorithm. The results, however, show that the GPU performs even worse than the serial CPU-only code. Figure 1 shows that the GPU performance degrades with the increase in the input size. Overall, the serial CPU-only implementation outperforms the naïve GPU implementation.

An analysis of the GPU kernel shows that every column pair results in a write to the result matrix; which translates to a total of $m \times n$ global memory write operations.

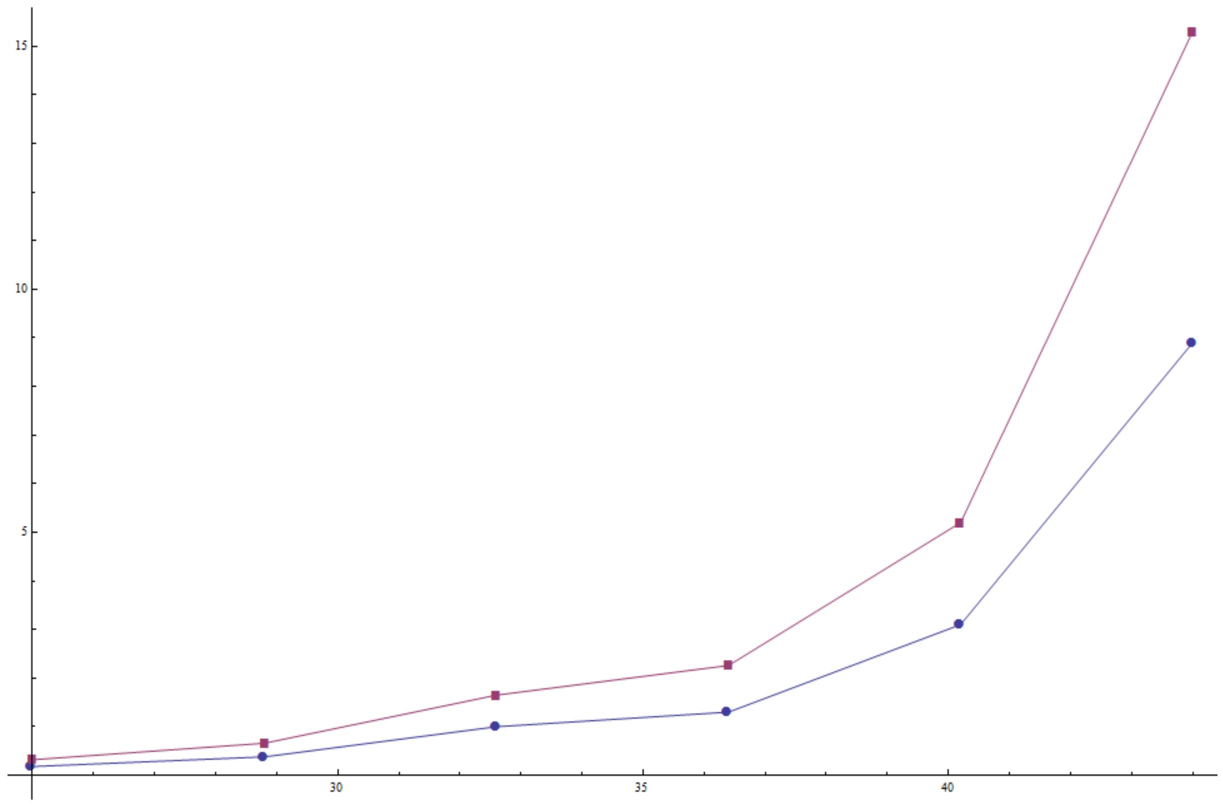


Figure 1: Performance comparison between the serial CPU-only implementation (blue) and the naïve GPU code (red). X-axis represents iterations of the Nullspace algorithm, which correspond to the growing input size. Y-axis represents the time taken in seconds.

Given that each result value is a 64-bit integer, this results in a very large result array. Therefore, most of the time is spent in transferring the result data structure between the device and the host.

2.5 Employing the Map-Reduce Structural Pattern for Parallel Applications

In the aforementioned pseudocode for the GPU kernel, the final step is a comparison that results in a Boolean value i.e. '0' or '1'. However, as per the serial version of the algorithm, this is just an intermediate step that eventually leads to fetching and storing the index pair corresponding to the input elements. The kernel has been designed in a way so that only the Boolean value is computed on the GPU, and the index pair computation is left as a post-processing step to be performed on the CPU. This division of work between the GPU and the CPU corresponds to the Map-Reduce structural pattern [5] with the following two phases:

- *Map*: The GPU kernel. Generates candidates and stores Boolean values as the result. Each result value represents the decision whether the pair of input vectors should be stored.

- *Reduce*: A post-processing step performed on the CPU. Parses the result data structure populated by the GPU. For all values where the decision is positive, the corresponding pair of indices is fetched and stored in a dynamic data structure. This phase is implemented as a shared-memory parallel procedure.

One of the benefits of implementing a heterogeneous Map-Reduce pattern is that the GPU kernel is relieved from executing further memory intensive index fetch operations. Also, the massively parallel nature of a GPU application results in concurrency issues (mutual exclusion) if a dynamic global data structure is used to store the resulting index values. Therefore, relegating such operations to the CPU, results in a relatively efficient kernel.

2.6 Introducing the Compression Factor

A significant advantage of employing the Map-Reduce pattern is the possibility to exploit the Boolean nature of the result computed per thread, in order to reduce the number of global memory write operations by a constant factor. Since each result value is a Boolean, instead of storing the result as an integer, it is stored as a single bit. Therefore, with $compressionFactor = 64$, 64 results can be stored in one element of the result array (assuming 64-bit unsigned integers are being used in the result data structure). This makes it possible to compute $compressionFactor$ number of candidates per thread.

As a result, size of the result data structure for the Map phase is reduced by a factor of $compressionFactor$. Following are two major advantages of the reduction in result size:

1. Previously, the time spent in data transfer between the device and the host overshadowed the time spent on computation. With the $compressionFactor$ scheme, the balance is shifted in the favor of computation time i.e. time spent in device-host-device data transfers is negligible in comparison to the time spent in kernel execution. This results in better utilization of the GPU resources.

2. Much more efficient user-managed caching [12] schemes can now be employed.

The performance results after implementing the Map-Reduce pattern with $compressionFactor$ are shown in Figure 2. The heterogeneous algorithm outperforms the serial CPU version by achieving a relative speedup of $\sim 3.5x$. Please note that this speedup does not include all the typical performance optimizations [9] applied to CUDA code. Efforts to further improve the performance by applying such optimizations are underway. These optimizations are well known within the CUDA programmers' community, and therefore, will not be discussed in detail here.

2.7 Brief Overview of Planned Optimizations

Following is a brief discussion of further optimizations applicable to the heterogeneous implementation:

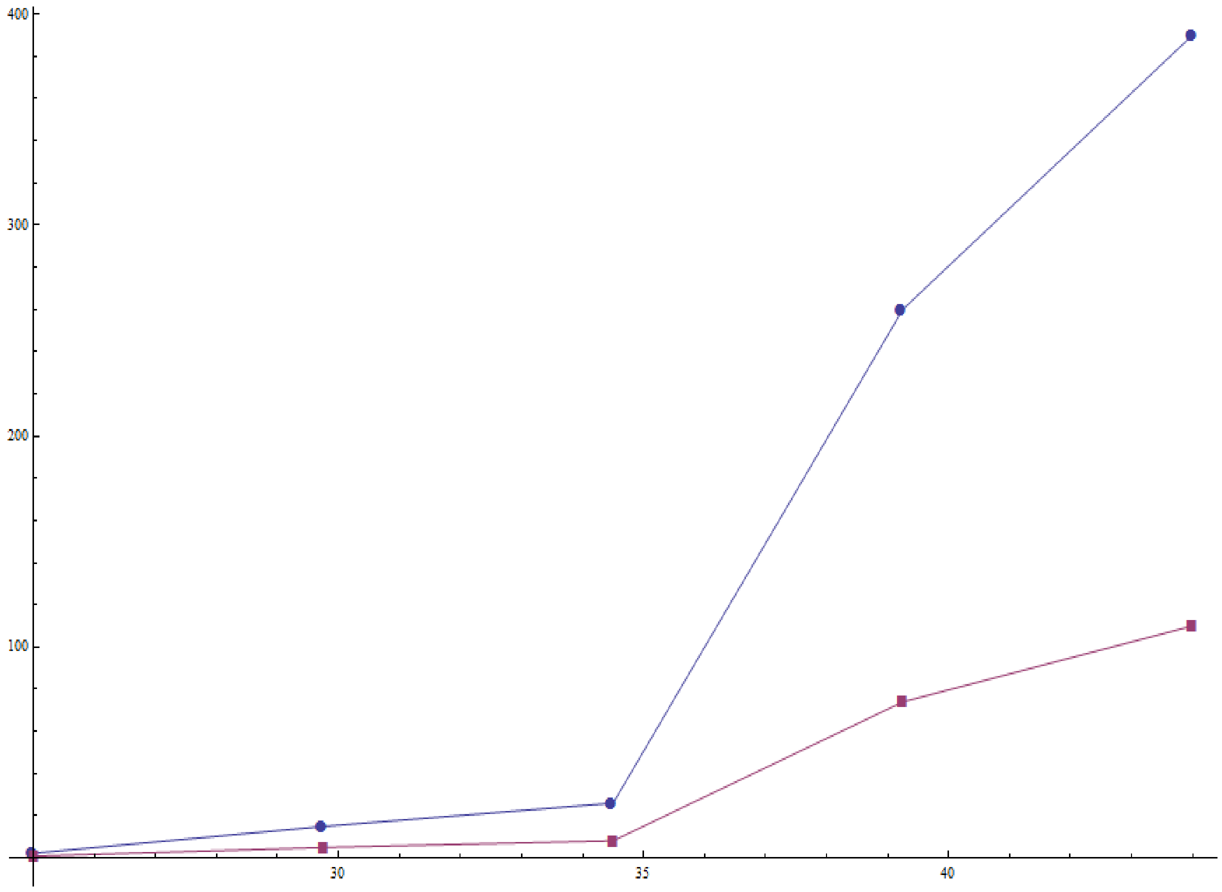


Figure 2: Performance comparison between serial CPU-only implementation (blue) and the heterogeneous Map-Reduce based implementation with compressionFactor (red). X-axis represents iterations of the Nullspace algorithm, which correspond to the growing input size. Y-axis represents the time taken in seconds.

- *Coalesced Global Memory Access and Result Caching*: Due to the use of $compressionFactor$, access to the global memory is no longer coalesced from threads in the same block. An improved index algebra, coupled with a cooperative caching mechanism is being implemented to ensure coalesced accessed to the global memory. This is expected to result in a significant performance improvement.

- *Tiled Input and Caching*: Tiling can be used to implement efficient caching for the input data. However, it is not clear at the moment if this will result in a significant performance gain.

- *Asynchronous Map-Reduce*: The current implementation executes the Map and Reduce phases in sequence. A multi-threaded asynchronous implementation may lead to better overall performance.

In addition to the above mentioned possible optimizations, several typical CUDA optimizations will be implemented.

3 Open Questions

At this point, not all possible optimizations have been implemented. Therefore, the full potential of heterogeneous computing is yet to be measured for the candidate generation algorithm.

The scope of this research project, however, is not limited to the implementation and optimization of the candidate generation algorithm. In terms of generalization of results, an attempt will be made to address the following questions:

- In the context of Heterogeneous Computing, can the use of the Map-Reduce structural pattern be generalized to a broad class of algorithms?
- For a given serial algorithm, how can one determine the optimal hardware architecture in order to maximize performance?
  - This is a critical decision in most situations, since cost is often a major constraint.
  - Also, this is a valid question even if the choice of hardware architectures excludes GPUs. Even for CPUs alone, it has been shown that the value for theoretical FLOPS is not always an accurate measure of CPU performance [8]. It depends on factors such as the numerical intensity of the algorithm.

4 Conclusions and Outlook

In order for software designers to take full advantage of heterogeneous computing, generic patterns in low numerical intensity algorithms need to be identified that can make it possible to gain significant performance gains over multi-threaded shared-memory CPU-only implementations. A positive result in this direction is presented by utilizing the Map-Reduce structural pattern to improve algorithm efficiency for an algorithm from Systems Biology.

A set of further optimizations will be implemented for this application, which will provide further insight into the full potential for utilizing Heterogeneous Computing for this particular problem. Moreover, research will be carried out in order to see if the results can be generalized for a broader class of algorithms.

References

[1] Aydın Buluç, John R. Gilbert, and Ceren Budak. Solving path problems on the gpu. *Parallel Computing*, 36(5-6):241–253, 2010.

[2] John D. Davis and Eric S. Chung. Spmv: A memory-bound application on the gpu stuck between a rock and a hard place. Technical report, Microsoft Research Silicon Valley, September 2012.

[3] Julien Gagneur and Steffen Klamt. Computation of elementary modes: a unifying framework and the new binary approach. *BMC Bioinformatics*, 5(1):175, 2004.

[4] Dimitrije Jevremović, Cong T. Trinh, Friedrich Srienc, Carlos P. Sosa, and Daniel Boley. Parallelization of nullspace algorithm for the computation of metabolic pathways. *Parallel Computing*, 37(6-7):261–278, 2011.

[5] Kurt Keutzer, Berna L. Massingill, Timothy G. Mattson, and Beverly A. Sanders. A design pattern language for engineering (parallel) software: merging the plpp and opl projects, 2010.

[6] Steffen Klamt and Jörg Stelling. Combinatorial complexity of pathway analysis in metabolic networks. *Molecular Biology Reports*, 29(1):233–236, 2002.

[7] Victor W. Lee, Changkyu Kim, Jatin Chhugani, Michael Deisher, Daehyun Kim, Anthony D. Nguyen, Nadathur Satish, Mikhail Smelyanskiy, Srinivas Chennupaty, Per Hammarlund, Ronak Singhal, and Pradeep Dubey. Debunking the 100x gpu vs. cpu myth: an evaluation of throughput computing on cpu and gpu. In *Proceedings of the 37th annual international symposium on Computer architecture*, pages 451–460. ACM, 2010.

[8] Josh Mora. Do theoretical flops matter for real application performance?, 2012. HPC Advisory Council Spain Workshop.

[9] NVIDIA. Cuda c best practices guide. Design guide, January 2012.

[10] John D. Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krüger, Aaron E. Lefohn, and Timothy J. Purcell. A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum*, 26(1):80–113, 2007.

[11] S. Schuster and C. Hilgetag. On elementary flux modes in biochemical reaction systems at steady state. *J. Biol. Syst*, 2:165–182, 1994.

[12] Mark Silberstein, Assaf Schuster, Dan Geiger, Anjul Patney, and John D. Owens. Efficient computation of sum-products on gpus through software-managed cache. In *Proceedings of the 22nd annual international conference on Supercomputing*, ICS '08, pages 309–318, New York, NY, USA, 2008. ACM.

[13] Marco Terzer and Jörg Stelling. *Accelerating the Computation of Elementary Modes Using Pattern Trees*, volume 4175 of *Lecture Notes in Computer Science*, pages 333–343. Springer Berlin / Heidelberg, 2006.

[14] C. Wagner. Nullspace approach to determine the elementary modes of chemical reaction systems. *The Journal of Physical Chemistry B*, 108(7):2425–2431, 2004.

3D Geovisualization Services for Efficient Distribution of 3D Geodata

Jan Klimke
Computer Graphics Systems
Hasso-Plattner-Institut
jan.klimke@hpi.uni-potsdam.de

Design, implementation, and operation of services for interactive 3D geovisualization are faced with a large number of challenges including (a) processing and integration of massive amounts of heterogeneous and distributed 2D and 3D geodata such as terrain models, buildings models, and thematic georeferenced data, (b) assembling, styling, and rendering 3D map contents according to application requirements and design principles, and (c) interactive provisioning of created 3D maps on mobile devices and thin clients as well as their integration as third-party components into domain-specific web and information systems. This report presents a concept and implementation of a service-oriented platform that addresses these major requirements of 3D web mapping systems. It is based on a separation of concerns for data management, 3D rendering, application logic, and user interaction. The main idea is to divide 3D rendering process into two stages. In the first stage, at the server side, an image-based representation of the 3D scene is created by means of multi-layered virtual 3D panoramas; in the second stage, at the client side, the 3D scene is interactively reconstructed based on these panoramas. Beside the interactive client introduced earlier, scalable variant for provisioning of virtual 3D city models is described, that utilizes the capabilities of our rendering service for data preparation.

1 Introduction

The availability of 3D geodata, its volume and its quality are constantly growing. Therefore, a high quality 3D visualization of massive and detailed data sets represents a computationally expensive task: It consumes large amounts of main memory, disk, network, CPU, and GPU resources in order to deliver an interactive user experience and a constantly good visual quality. For distribution of such 3D contents in a web environment, two principal concepts can be found in existing 3D web mapping approaches, client-side 3D rendering, i.e., assembly and streaming of 3D display elements such as 3D geometry and textures by a 3D server, whereby the corresponding 3D clients manage scene graphs and perform real-time 3D scene rendering (e.g., Google Earth, OGC W3DS); and server-side 3D rendering, i.e., assembly and rendering of 3D display elements by the 3D server, which delivers views of 3D scenes to lightweight clients (e.g., OGC WVS).

Taking into account the increasing complexity and size of geodata for 3D maps (e.g., complex 3D city models), the need for high-quality visualization (e.g., illustrative or photorealistic rendering), and rapidly growing use of mobile applications (e.g., smart phones and tablets), these principal concepts are faced by fundamental limitations: client-side 3D rendering fails to provide fast access to massive models due to bandwidth limitations and cannot guarantee high-quality rendering results as the graphics capabilities of nearly all mobile devices differ significantly, while a pure server-side 3D rendering is inherently limited with respect to interactivity and does not take advantage of todays mobile device graphics capabilities.

In this report i will present a framework based on a Web View Service as core component encapsulating the complexity of 3D model data and computer graphic techniques. Conventional techniques for visualization of 3D geodata, in particular virtual 3D city models, are built upon client-side rendering of 3D geometry and texture data transmitted from remote servers to client applications. These approaches are usually limited in terms of complexity of models and the visual quality they can provide, since high-end computer graphic technique demand for state of the art 3D hardware.

Utilizing this type of service as a basis for client applications allows for provisioning high quality visualization of 3D model data of nearly arbitrary sizes on devices and platforms, that would otherwise not be able to provide usable, high-quality 3D visualization. Through externalizing image-generation from client side on server side, a consistent presentation of 3D geodata is possible, regardless of client hardware or software.

Approaches for image-based client applications presented earlier, are allays bound by the performance of the underlying rendering server system in order to provide a fully interactive user experience. In applications, this full 3D interactivity is often not necessary and leads to a larger complexity for users regarding camera interaction. Therefore, in Section 5, we introduce a scalable application that was implemented for mobile devices as well as for web browsers. These applications addresses these two challenges, be limiting the possible camera interaction. This allows to pregenerate the necessary image data that has to be transmitted to clients.

The remainder of this report is organized as follows. Section 2 introduces related work in the are of image based remote visualization. Section 3 introduces the specific challenges that arise when building service-based visualization system in more detail. Section 4 describes briefly the overall architecture of the visualization system which forms the basis of my work. Finally, an outlook of planned future work concludes this report in Section 6.

2 Related Work

The interoperability of systems and applications dealing with geodata is an central issue in order to build systems out of interoperable software components for geodata access, processing, and visualization. Beside a common understanding on information models [3], definitions of service interfaces are necessary. The Open Geospatial Consortium (OGC) defines a set of standardized services, models, and formats for geodata encoding and processing. For example, a Web Feature Service (WFS) [16] can provide geodata, encoded in the Geography Markup Language (GML) [13] or City Geography Markup Language (CityGML) [6], and processed by a Web Processing Service (WPS) [15]. For geovisualization processes a general portrayal model is provided by the OGC that describes three principle approaches for distributing the tasks of the general visualization pipeline between portrayal services and consuming applications [1, 7]. While the OGC Web Map Service (WMS), providing map-like representations of 2D geodata, is widely adapted and used, 3D geovisualization services have not been elaborated to a similar degree. Several approaches for 3D portrayal have been presented [2]. Two types of 3D portrayal services, currently discussed in the OGC context, can be distinguished: Web 3D Service (W3DS) [14]: It handles geodata access and mapping to renderable computer graphics primitives (e.g., textured 3D geometry represented by scene graphs) and their delivery to client applications. Web View Service (WVS) [8]: It encapsulates the image generation process (i.e., it implements the geovisualization pipeline) of 3D models, delivering rendered image representations ("portrayals") to client applications.

By focusing on developing international standards for 3D Portrayal, an interoperable, service-based system can be built, that allows replacing component implementations selectively with other implementations. Further, system components, especially a 3D rendering service or a W3DS instance can be reused in other systems. Several approaches exist for remote rendering of 3D virtual environments [4, 12].

Mostly, they rely on constant transmission of single images [17] or video streams [10, 11] to client applications. In contrast to those applications that are completely dependent on the current data throughput of the network connection, our approach uses a latency hiding technique. This technique allows for a continuous user interaction operating on the locally available data also in situations with very low data transmission rates between 3D rendering server and clients. Another approach is to manage and render a low resolution 3D model on client side, to allow users to explore the model interactively; when interaction stops, remote rendering is used to create and display an image of the high-resolution 3D model [9]. For delivery of, e.g., large-scale textured 3D city models, this approach is not suitable, since a transmission and rendering of low resolution model representations on client-side would still exceed network and client-side rendering capabilities. In approach an image-based 3D approximation of a model is created by the client, using image data transmitted from a 3D rendering service.

3 Challenges

Design, implementation, and operation of interactive 3D geovisualization services are faced with a number of challenges including:

Data processing and integration Massive amounts of heterogeneous and distributed 2D and 3D geodata such as terrain models, buildings models, and thematic georeferenced data form the basis for 3D maps. This data has to be processed for visualization and integrated into the overall visualization system in order to be combined in a 3D map.

3D map content assembly, rendering and styling To communicate spatial and georeferenced information, 3D map content must be selected, styled and rendered according to application requirements and design principles.

Interactive provisioning Created 3D maps should be available on mobile devices and thin clients in an interactive manner. Further, third-party vendors should be able to integrate 3D maps as components into their domain-specific web and information systems.

Interoperability A system for 3D web mapping should rely on established standards in terms of data formats and service interfaces.

Upscaling When it comes to publishing of high-quality 3D visualization of 3D geodata, there should be a way to provide at least parts of the functionality to large numbers of concurrent users.

In particular, challenge (c) influences the way a service-oriented system for 3D mapping can be built. Large amounts of data have to be transmitted, processed, and stored for generating a useful 3D map, common client applications have to scale with the size of the underlying 3D model, because they deal with the massive amounts and the complexity of model data on client side. In order to provision 3D maps on a large variety of devices with heterogeneous hardware and software capabilities and undefined as network connection speed, the effort of processing, transfer, and rendering of 3D map content should be decoupled from a client application, while still providing a user with an interactive user experience. This diversity and performance considerations is especially an issue, when designing applications for mobile devices or browser-based 3D mapping solutions. Clients for these platforms should provide a equally high visual quality, regardless of the capabilities of the individual device or platform.

4 A Service-based 3D Visualization System

An architectural overview over the system is provided in Figure 1. There are many heterogeneous sources and formats for 3D geodata, many of them are not optimized for visualization purposes, although containing information about the detailed visual appearance of features. In order to be rendered efficiently by the 3D rendering service, the different formats are translated into an intermediate format that fulfills two requirements: (a) allow for highly efficient rendering of massive data amounts on the one hand side and (b) maintains the semantical structure of the original geodata in order to allow for professional applications requiring access to the underlying data sources and their structure. The Preprocessing service provides this type of functionality. In course of the preprocessing, a database is built up, containing meta informations, such as the coordinate bounding box, parent-child relation of objects, their type (e.g., building, road, city furniture, etc.), and a mapping from the original id of an imported feature and the assigned object id that is later used for rendering.

The resulting data format allows for a consistent implementation of 3D rendering techniques, since these can be built against a widely consistent kind of data (e.g., numerical scale of coordinates, geometry structure, and texture format). This eases the development of graphical stylizations and real-time rendering techniques for the 3D rendering service implementation significantly. The 3D rendering service, as core component of the service-based visualization system, creates rendered images of 3D geodata. It integrates data locally available preprocessed data as well as data available from external services, such as map data from standardized Web Map Services or 3D model data from Web 3D Services. The integration of external data services for provides more flexibility for visualization applications using the 3D rendering service. Especially, constantly changing, dynamic data, such as vehicle positions or water levels, can be integrated without a prior translation into an intermediate data format.

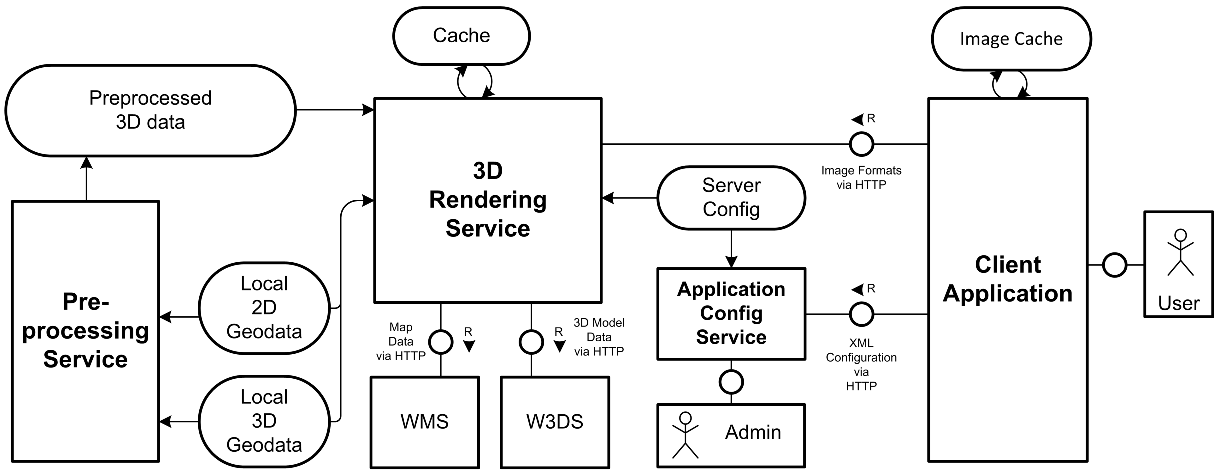


Figure 1: Architecture of a service-based system for 3D geovisualization. The 3D Rendering Service encapsulates the visualization pipeline for geodata and hides away the complexity of dealing with large, complex structured, and possibly distributed data sets.

To provide client applications that are both, universally applicable in different working processes and interactively manageable, an application configuration service is used to setup 3D rendering servers as well as corresponding client applications. Application specific configuration of multi purpose client applications is done remotely. This allows us to provide an individual application configuration, including, e.g., appropriate model data, map data, styling effects or predefined, or generated camera positions, for specific users, roles, or use cases.

4.1 Stylization of 3D Contents

Stylization of 3D maps is one of the main issues of the 3D map generation process. It supports the efficient communication of geoinformation. In our 3D rendering service, we support flexible stylization of 3D map contents. For this it provides a number of options to style the 3D maps. Specific textures can be applied to each model element, e.g., a terrain model can be textured using different 2D maps. Further, different 3D rendering effects can be applied per model element, e.g., a specialized rendering for planned constructions. Additionally, the 3D rendering service offers so called image styles that affect the overall appearance of the 3D scene.

This stylization is implemented as an image-based post-processing executed after the 3D rendering of the scene geometry and textures has been performed. Data from

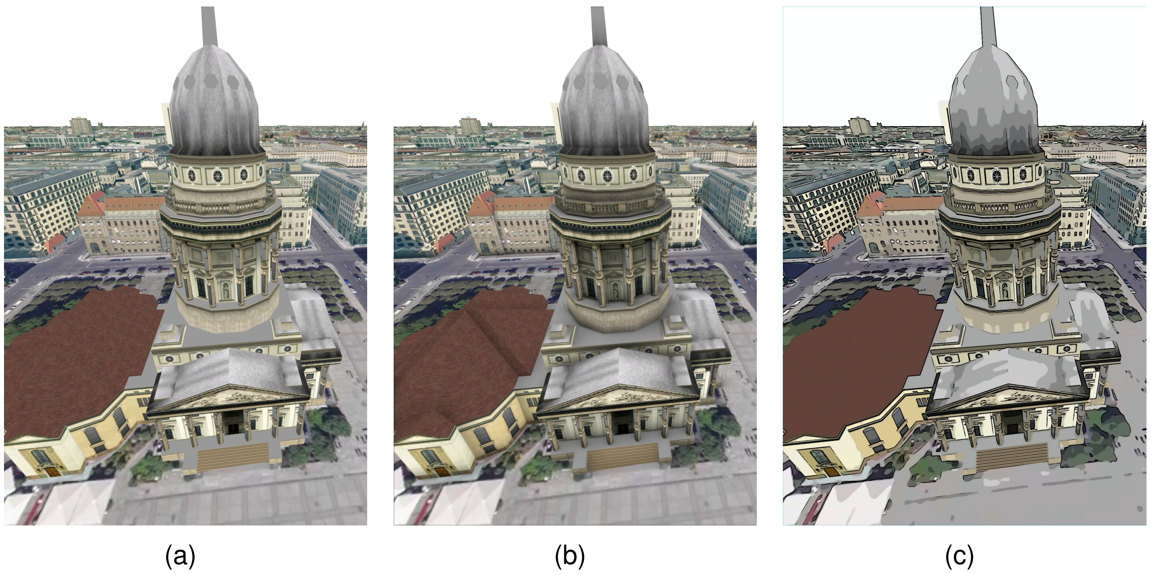


Figure 2: Examples for different stylization applied to a 3D scene: a) Without any explicit stylization b) With a self shadowing effect that improves depth perception c) an image abstraction effect, removing details from surface textures.

different image layers is used together with additional configuration options to configure the image stylization effect, e.g. the id of a scene object to be highlighted by a halo effect. Stylization effects are implemented efficiently using the graphics hardware. Unnecessary copy operations of source image-layers are avoided by reusing them on the graphics card without prior download into the main memory. This way, the graphics hardware of the rendering server is used in a very efficient way for image-based computations, e.g., non-photo-realistic rendering. Examples for image-based styling are depicted in Figure 2.

The image-based stylization has one major advantage compared to conventional techniques for 3D-stylization: The computational complexity does not depend on the model size and complexity. The computational costs are mainly dependent on the resolution of the source and target images and the complexity of the desired effect. Image-based techniques can also be used to visualize thematic data, e.g. through projection of image data or applying color values for certain objects encoding specific data values.

5 Thin-Clients for Map-Like Visualization of 3D Geodata

Integration of 3D visualization of geodata in a variety processes and application areas, e.g., in city planing, infrastructure management, tourism or city marketing, gains more and more importance. In contrast, the problem of provisioning, deployment and scaling 3D visualization applications for a large scale audience (e.g., thousands of concurrent users) in a cost efficient way remains a challenging task. Existing approaches for 3D visualization of geodata often require a client system that is capable of performing complex 3D rendering for massive amounts of data. The server-side requirements and especially the complexity of implementing high-quality rendering techniques for a large number of highly heterogeneous client hardware and software platforms makes it very hard to implement a usable 3D visualization component that can be integrated into third party applications.

Recently, we presented a client application that reconstructs a lightweight representation of a complex server-side model in order to allow and interactive user exploration and free navigation in 3D world [5]. While this kind of application addresses a lot of issues that arise when massive amounts of 3D geodata, such as virtual 3D city models, is used in mobile or web-based visualization applications, the issue of the complexity of 3D navigation in 3D geovirtual environments remains. To address this issue, and to provide a solution for exploring 3D city models in connection with related 2D and 3D application data, we created a map-like visualization that uses image data that can be pregenerated while still keeping the connection between underlying 3D geometry and the semantic data they represent. Therefore we use parts of the framework presented in Section 4, in particular preprocessing and rendering services, to generate tileable views of a 3D dataset. A restriction of user interaction, conceptually to a motion in parallel to the earth's surface and a modification of the distance between surface and camera, makes it possible to create sets of image tiles for applications that cover a specific spatial region. In contrast to earlier clients, which where using a conventional perspective projection for image generation, this application works with an orthographic projection. In this way, it is possible to generate we can create a specific image for each numbered spatial region (image tile). This way, each tile does exclusively contain a certain part of the overall model geometry. A script is used to generate all tiles for a spatial region, in our case the region covered by the city model of Berlin, in several discrete zoom levels (discrete distance steps for surface-camera distance).

The files for image tiles are organized in conformance to the Tiles Map Service, as a quasi standard for providing a simple access scheme for tiles georeferenced image data. The corresponding client application, depicted in Figure 3, requests tiles that are needed for the current zoom level and visible spatial region.

Besides the efficient visual presentation of 3D geodata, linkage of the visual entities and their underlying geodata plays a major role for visualization solutions to be applicable in every day working processes. Therefore, images containing object identifiers for an color image tile are generated. They can be used to get object identifiers for specific positions given in pixel coordinates for single tiles. The rendering service instance is then able to map these identifiers back to the original identifier in the source dataset. Besides these capabilities for data access, id images can be utilized for rendering effects on client side, e.g., highlighting of single objects.

6 Conclusions and Future Work

In this report I presented a service-based framework for 3D portrayal of 3D geodata implementing the 3D visualization process. Using the extended image generation capabilities of the 3D Rendering service, significant amounts of computational complexity

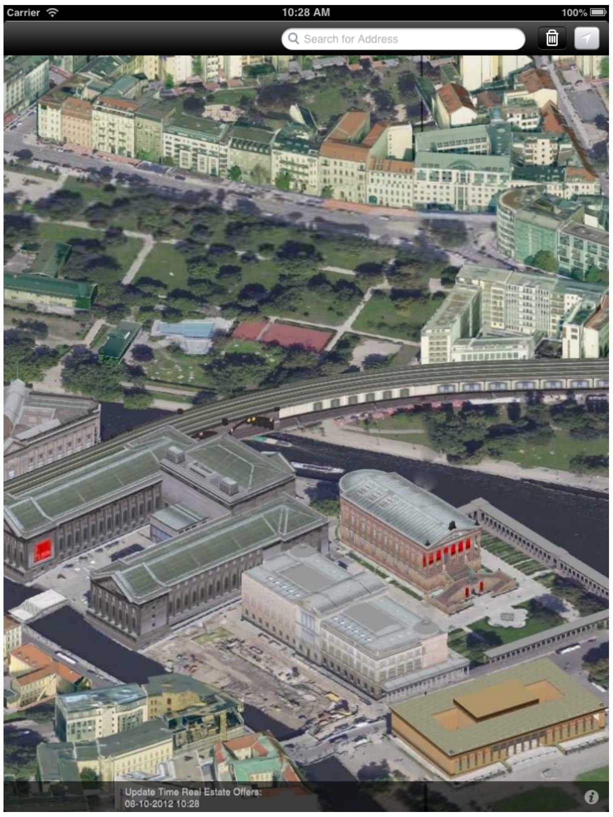


Figure 3: Example for a Web-based Visualization of the virtual 3D city model of Berlin created using the service pipeline. The created map-like visualization can be arbitrarily styled and customized. Underlying image data can be pregenerated in order to serve a large number of clients.

can be implemented on server-side, allowing for a more resource efficient implementation of client applications. Besides interactive, image-based, real-time 3D-clients presented earlier, map-like applications using pregenerated image-data provide further advantages with regard to server-side effort. Generated image-tiles can be served easily for a large number of users, e.g. by using highly scalable cloud storage services.

The next steps in research on our framework for service-based 3D visualization will be to perform performance tests and evaluations for several large-scale datasets in order to reveal potential for further optimizations and extensions to the overall system and explicitly to interactive client applications. Measuring the performance of the service based visualization system will be part of my work during the next month. Both, the performance of the rendering service itself, as well as the performance of the overall image-based visualization system need to be explored in more detail in order to identify possible bottlenecks, arising, e.g., from client accessing the rendering service concurrently.

Further we see, that 3D interaction and rendering techniques that support latency hiding seem promising in order to improve the overall user experience of image-based 3D visualization applications. Within the next term i am therefore going to focus on such techniques in connection with specific rendering technique for that purpose.

References

[1] A. Altmaier and T. H. Kolbe. Applications and Solutions for Interoperable 3D Geo-Visualization. In *Proceedings of the Photogrammetric Week 2003*, pages 251–267, Stuttgart, 2003. Wichmann.

[2] J. Basanow, P. Neis, S. Neubauer, A. Schilling, and A. Zipf. Towards 3D Spatial Data Infrastructures (3D-SDI) based on open standards experiences, results and future issues. In *Advances in 3D Geoinformation Systems*, Lecture Notes in Geoinformation and Cartography, pages 65–86, 2008. Springer.

[3] Y. Bishr. Overcoming the semantic and other barriers to GIS interoperability. *International Journal of Geographical Information Science*, 12(4):299–314, 1998.

[4] A. Boukerche and R.WN Pazzi. Remote rendering and streaming of progressive panoramas for mobile devices. In *Proceedings of the 14th annual ACM ...*, pages 691–694, 2006.

[5] J. Doellner, B. Hagedorn, and J. Klimke. Server-based rendering of large 3d scenes for mobile devices using g-buffer cube maps. In *Proceedings of the 17th International Conference on 3D Web Technology*, Web3D '12, pages 97–100, New York, NY, USA, 2012. ACM.

[6] Gerhard Gröger, Thomas H. Kolbe, Claus Nagel, and Karl-Heinz Häfele. OpenGIS City Geography Markup Language (CityGML) Encoding Standard Version 2.0.0. Technical report, Open Geospatial Consortium Inc., 2012.

[7] R. B. Haber and D. A. McNapp. Visualization Idioms: A Conceptual Model for Scientific Visualization Systems. In *Visualization in Scientific Computing*, pages 74–93. IEEE, 1990.

[8] B. Hagedorn. Web view service discussion paper, Version 0.6. 0. *Open Geospatial Consortium Inc*, 2010.

[9] D. Koller, M. Turitzin, and M. Levoy. Protected Interactive 3D Graphics via Remote Rendering. *Transactions on Graphics (TOG)*, 2004, ACM.

[10] F. Lamberti and A. Sanna. A streaming-based solution for remote visualization of 3D graphics on mobile devices. *IEEE transactions on visualization and computer graphics*, 13(2):247–60, 2007.

[11] D. Pajak, R. Herzog, E. Eisemann, K. Myszkowski, and H.-P. Seidel. Scalable Remote Rendering with Depth and Motion-flow Augmented Streaming. *Computer Graphics Forum*, 30(2):415–424, April 2011.

[12] G. Paravati, A. Sanna, F. Lamberti, and L. Ciminiera. An open and scalable architecture for delivering 3D shared visualization services to heterogeneous devices. *Concurrency and Computation: Practice and Experience*, 23(11):1179–1195, 2011.

[13] C. Portele. OpenGIS Geography Markup Language (GML) Encoding Standard, Version 3.2.1, July 2007, Open Geospatial Consortium.

[14] A. Schilling and T.H. Kolbe. Draft for Candidate OpenGIS Web 3D Service Interface Standard, Version 0.4.0, 2010, Open Geospatial Consortium.

[15] P. Schut. OpenGIS Web Processing Service, Version 1.0.0, 2007, Open Geospatial Consortium.

[16] P.A. Vretanos. OGC Web Feature Service Implementation Specification, 2005, Open Geospatial Consortium.

[17] A. Wessels, M. Purvis, J. Jackson, and S. (S.) Rahman. Remote Data Visualization through WebSockets. *2011 8th International Conference on Information Technology: New Generations*, pages 1050–1051, April 2011, IEEE.

Applications of Virtual Collaboration Monitoring in Software Development Projects

Thomas Kowark
thomas.kowark@hpi.uni-potsdam.de

This report summarises my research work of the last six months. The focus resided on the creation of a concept to include process metrics within Team Collaboration Networks (TCN) to enable reasoning about effects of virtual collaboration behaviour. The concept was prototypically implemented and applied in a classroom software engineering project to analyse the effects of conformance with software development principles on team progress. Based on the experiences of this case study, further applications in the analysis of existing open-source project data and experiments in software engineering research are outlined. The report concludes with initial insights from an ongoing test project that applies our prototype platform for virtual collaboration analysis in an industry setting.

1 Introduction

Virtual collaboration analysis aims to detect correlations between the way project members interact with groupware systems and different metrics that determine the state of a project. Two basic approaches are available for detecting such indicators and their effects on the measured process metrics.

In the top-down approach, researchers start by modelling potential indicators for the adherence-to or violation-of investigated collaboration activities. In the domain of software engineering, these activities usually relate to principles of the used development process. Secondly, data needs to be collected from the systems employed by the teams under investigation or by using an openly available data set. Finally, the indicators need to be translated into queries on the data and occurrences can be statistically tested for correlations with collected process metrics.

The bottom-up approach starts with the captured data and extracts frequently occurring usage patterns that correlate with process metrics. These patterns are subsequently described in a formal manner to enable their usage in a top-down analysis of other data sets. By that, project-specific phenomena can be separated from behaviour with general validity regarding its correlation with process metrics.

The main challenge for this kind of research is reproducibility. In case a freely available dataset has been used, it needs to be specified how the data was prepared (e.g., conversion of files in comma-separated-value (CSV) format into a database schema) and which queries have been carried out. In case a custom data set was created, e.g., by analysing an industry case study or conducting experiments, even more information is required. Tools employed by the team, programs used to capture data from the different groupware systems, data structures used to store the data, and queries that were executed on the data need to be made available for complete reproducibility. A simplification of this approach requires the following three components:

- open access to the investigated data,
- a common data format for storing traces of virtual collaboration activity,
- a standardised query language that allows to formally model the previously described indicators and reproducibly detect their occurrences across projects.

1.1 Data Repositories in Empirical Software Engineering

Researchers have previously acknowledged that data sharing is essential to promote rigour in empirical software engineering research. An approach like the PROMISE repository of empirical software engineering data [11] provides a quasi-standard data set that can be used to verify or falsify theories about the correlations between different software process metrics (e.g. [2]) across multiple projects. Thereby, such repositories provide a solution for the first challenge. In their current implementation, however, they do not cover the second challenge. Data is stored in a CSV format and potential semantic links are described in accompanying text files, leaving room for interpretation by users of the data.

Linked open data provides a solution to this challenge. Repositories like SeCold [6] are based on ontologies that describe the different concepts of the mined data sources, such as source code repositories and bug trackers, and consequently remove ambiguity in data handling. The named repository contains data from approximately 18,000 software development projects. Though the ontologies slightly differ from the ones created in the Team Collaboration Network (TCN) approach [12], the conceptual similarities confirm the viability of our approach and the provided data set will be used as an extension to the database created through data capturing in our own case studies [8–10] .

The third aspect is partly covered by SPARQL, which is the de-facto standard query language for Resource Description Framework (RDF) data. Queries created for a sample project can be applied to other projects without adoptions as long as the same ontologies are used. Query results, however, will have a project specific bias, as terminology, cardinalities, or timeframes can differ between projects. Hence, either queries have to be created in a very generic manner, or abstract descriptions of the monitored activities (e.g., using Business Process Modelling Notation (BPMN) or Event-Driven Process Chains (EPC)) are translated into SPARQL queries, taking into account the characteristics of the projects. We have presented a potential solution for this in [7].

1.2 Outline

An aspect that is currently not handled by the aforementioned approach is the explicit representation of process metrics within the linked data. This is required to not only detect recurring patterns in the collected collaboration activities but measure their effect on the underlying development processes. In this report, I present a prototypical integration of process metrics into TCN ontologies. Section 2 also discusses the costs of this approach with regards to storage requirements for TCN. Section 3 presents an initial application of our approach that investigates the impact of the adherence-to or violation-of the principle of Test-Driven Development in a sample software engineering project. The paper concludes with initial insights from an ongoing test project that applied our prototype platform for virtual collaboration analysis in an industry setting.

2 Team Collaboration Network Extensions

The definition of TCN as presented in [12] already includes support for attributed relations. Each Attribute A of the network was defined to be a 5-tuple $< s, r, o, t_{start}, t_{end} >$ where s, the entity of the graph the attribute is attached to, could either be a node $v \in V$ or an edge $e \in E$ of the TCN. Hence, the same attributes that could be assigned to a node could also be assigned to an attribute.

The OWL ontology created for the prototypical TCN implementation, however, did not include an explicit representation of this concept due to performance reasons. Since only temporal information was stored in addition to the source and target node of a relation, a workaround solution using custom table fields in the underlying SQL-database was created. As we intend to introduce additional information for relations within the graph, this solution is no longer feasible and an explicit representation of relations between graph nodes becomes inevitable. To this end, we introduced the class *AttributedRelation* to the TCN ontology. Figure 1 presents the respective ontology model using the graphical notation created in [12].

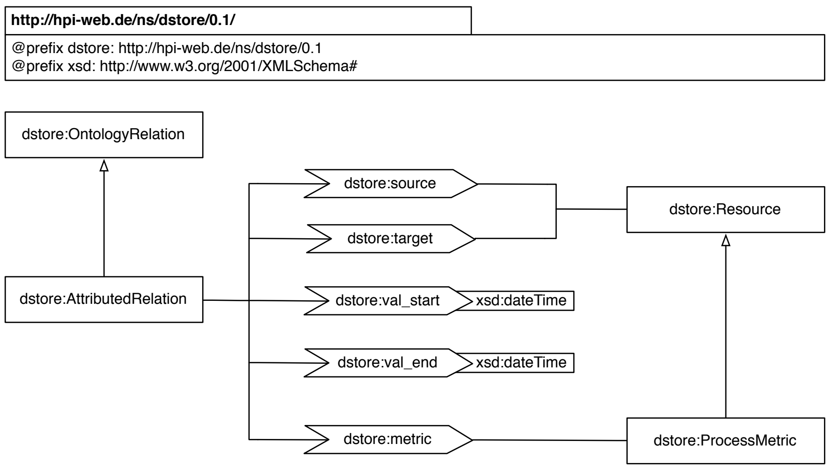


Figure 1: An ontology for the concept of attributed relations.

Temporal validity information is modelled as an attribute of the relation. Process metrics, on the other hand, are not included as attributes but need to be added as relations linking to the respective measurements. Since process metrics are generally captured through discrete samples, they themselves can be attributed with validity dates (e.g., the reported developer satisfaction over a given timeframe, velocity for a given sprint, etc.) and linking relations to those discrete samples reduces the overall statement count. Furthermore, not storing process metrics directly with relations allows rule-based inference of metrics, e.g., by specifying that relations are always linked to the measurement that was valid at the moment of their creation. By that, the statement count within the repository can be further reduced, however, at the cost of computational overhead required by the inference engine. In total, the statements describing the measurements plus one additional statement per measurement per relation (i.e., the link to the measurement) need to be added to include process metrics within TCN.

3 Classroom Application

In this section we present experiences from applying the system presented in [9] in a classroom software engineering setting. From this application we infer guidelines for future applications and outline experiment settings that can enhance rigour in the evaluations and minimise project specific influences.

3.1 Project Background

The lecture under investigation is a third year undergraduate software engineering class [8]. The 96 participants of the course were divided into two development groups, each consisting of eight teams. Those eight teams had the task of jointly developing a customer relationship management system (CRM). We used the approach presented in [7] to model indicators for violations of different practices prescribed in the course and continuously monitored the captured TCN for occurrences. A feedback system was installed to allow project tutors to provide subjective judgements about team progress and effort. After each meeting, they had to rank the perceived effort that the team put into the work and their progress (i.e., the fraction of planned work that was successfully completed) on a 10-point Likert scale (higher values indicate greater effort or progress). The collected metrics were included into the project TCN as described in Section 2.

3.2 Modelling Process Violations

Process violations were modelled according to a template created by Zazworka et. al [13]. It contains the process name, a categorisation, a textual description, data required to observe violations, and a description of investigated violations. The original template permits description of violations in a language of choice. To enhance reproducibility of analyses, we limit the possible languages to ones that can be mapped to SPARQL queries on TCN. Consequently, we use the notation presented in [7] to model indicators for potential violations of well-established software engineering principles [3]. Furthermore, the required data needs to be specified using TCN terminology and by defining types of resources, attributes, and relations that need to be present within the networks.

Process Name	Test-First
Process Focus	Correctness
Process Description	Version control revision history, test coverage for each revision of a file. * *Resources*: vcs:VcsRevision, vcs:FileRevision, vcs:Method, vcs:TestMethod * *Attributes*: vcs:test_coverage * *Relations*: vcs:successorOf, vcs:committedBy, vcs:includes, vcs:testedBy,
Process Violation	See Figures 2(a) and 2(b)

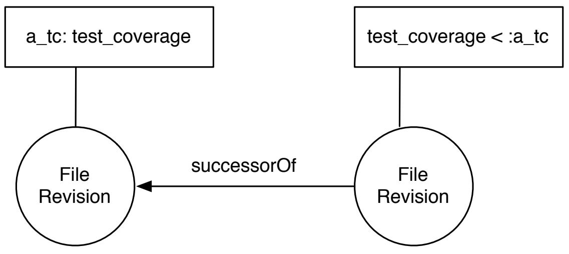
Table 1: Process Conformance Template (based on [13]) for Test-Driven Development.

We use the example of Test-Driven Development (TDD) [1] to outline the process of modelling indicators for process nonconformance. TDD requires developers to create a test case for a given implementation, first. Afterwards, an implementation is created that implements the tested functionality. A strict adherence to this principle would assure that no application code is being written without at least one test case that checks whether written functions generate the desired output for a given set of input data. This principle was tested in different related studies for its effects on metrics related to software development processes (e.g., team velocity, code quality, bug frequency in productive systems, etc.), but no consistent results emerged. In [14], Zazworka et al. identify differences in the way nonconformance with TDD is being recorded and measured as a potential source for differing results with regards to the impact of TDD on the measured metrics. Through tightening the constraints regarding allowed languages for violation descriptions and specification of required data, we aim at improving the template even further and help increasing repeatability of analyses of virtual collaboration activities.

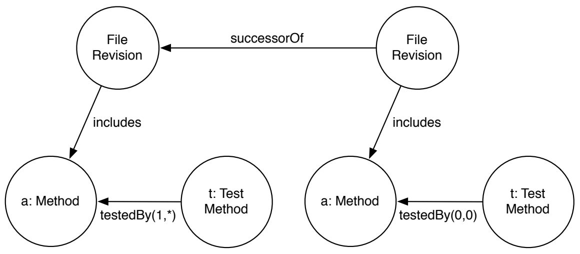
Table 1 presents the process conformance template for TDD. The violation indicator shown in Figure 2(a) denotes that test coverage, i.e., the fraction of source code that is covered by tests, decreases from one source code revision to the next. Due to limited tool availability for measuring test coverage in Ruby on Rails, only statement coverage could be captured. This is only possible in two cases: if a) code is being written without a test or if b) tests are deleted. Initially, only case a) was modelled. During the project, it became apparent that failing builds at the continuous integration server were sometimes “fixed” by deletion of tests. In some cases this deletion had effects only on branch coverage of the code, not statement coverage. Hence, a second indicator needed to be modelled to identify such violations, too (cp. Figure 2(b)).

3.3 Data Analysis

We utilised the detected violations in two ways. Firstly, email alerts were triggered upon each violation. Accordingly, the teaching team could review the revision and contact the team or person responsible, if necessary. This proved to be especially helpful with regards to the violation described in Figure 2(b) as a total of 5,275 commits was



(a) Creation of a source code revision that lowers test coverage.



(b) Removal of a test case for a given method.

Figure 2: TCN subgraphs representing violations of the TDD principle.

created, only 35 of which removed test cases. Given that through these commits a total of 123,166 revisions of the project's source code files were created, such violations could otherwise only have been detected with intensive manual efforts.

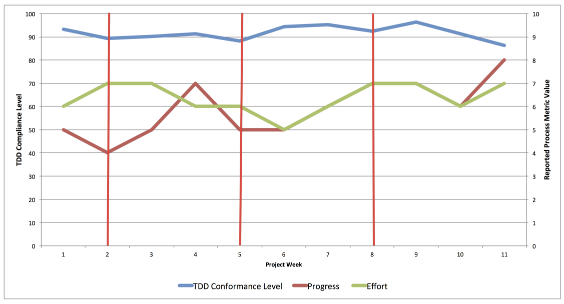
Secondly, we calculated a conformance level with the respective principles. Based on [14], we define the conformance level CL_{tdd} with the principle of TDD as follows:

$$CL_{tdd} = (\sum_{i=1}^{n} rev\_count(i) / \sum_{i=1}^{n} violation\_count(i)) * 100$$

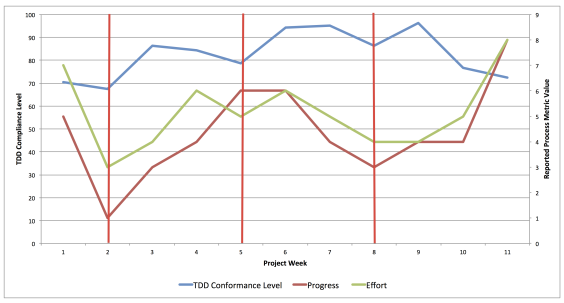
Whereas n is the number of developers under investigation, $rev\_count(i)$ the number of revisions created by developer i and $violation\_count(i)$ is the number of TDD violations created by developer i.

Figure 3 displays the calculated conformance level along with the recorded effort and progress measures for two example teams. The two teams presented here were part of different development groups but both worked on the customer management component of their respective CRM. Both teams showed a decline in conformance towards the end of the sprints, the most extensive one happening at the end of the last sprint, which also saw the highest effort and progress ratings for both teams. This behaviour can be attributed to the examination process of the course. Since students were judged equally on process conformance and project outcome, they were tempted to "get things done" at the end of each sprint and provide tests for the new functionality, afterwards.

The main difference between the two teams is the constancy in their work. Team alpha kept the TDD conformance at a high level throughout the project and showed only minor deviations in the effort they put into their work. Team beta on the other hand neglected the principle in the beginning and received low effort and progress ratings. After an intervention by the teaching team at the beginning of the second sprint, they increased the effort and paid closer attention to complying with TDD. Another interesting relation is the one between effort and progress, commonly referred to as productivity. In the first sprint, effort exceeded progress ratings for team alpha. In fact, the team noted that they needed to get accustomed to the development framework and underestimated the necessary overhead. Within the second sprint their progress ratings inclined without an incline in effort, which indicates that their productivity increased during this period. For team beta it is apparent that changes in their progress measures seemed to be coupled tightly to the effort they put into the project. This was



(a) Team alpha.



(b) Team beta.

Figure 3: Comparison of TDD compliance level, reported effort, and reported progress for two project teams. Vertical lines denote end of Scrum sprints

confirmed by the team, as they reportedly "liked the framework but simply did not find the time to work on the project". They reported the same lack of time at the end of the third sprint.

3.4 Lessons Learned

In summary, the observed setting was insightful with regards to the possibilities that virtual collaboration monitoring offers for project management purposes. The number of digital collaboration artefacts that required manual review could be reduced and allowed the tutors to provide immediate and focused feedback to the student teams. On the downside, we observed that digital collaboration analysis is prone to observer effects. If students know that the way they use groupware tools is analysed, it becomes more likely that they try to adapt their behaviour to generate supposedly ideal traces instead of focusing on doing what is best for the project.

Another point for improvement are the observed process metrics. They were mainly of subjective nature and need to be enriched with objective ones [5], such as, for example, exact measurements for development time and velocity of teams as a measure for their relative progress. However, obtaining such information reliably outside experimental setups is barely possible as the subjects under investigation have a motivation to report falsified values, namely their final grades in our project and potential bonuses and evaluations by superiors in industry settings. Hence, generalizable statements about the expressiveness of process conformance indicators and their correlation with process metrics should be obtained in specialised experiments that allow to verify the reported effort and progress measures objectively.

3.5 Future Applications

Based on the lessons learned throughout our case studies, future efforts will concentrate on the evaluation of data from open-source projects, as provided by the SeCold repository [6] or obtained through parsing open-source project platforms like github. Furthermore, we will perform smaller experiments that allow for an isolated evaluation of the correlations between presence of (non)conformance indicators for certain software development principles and objectively recorded process metrics.

Also, we will focus on the analysis of not only the structural properties of the captured TCN but the attributes of the contained nodes. A co-supervised master's thesis provided a first insight into possible applications of data mining techniques on the stored graphs [4]. By applying the statistical model Latent Dirichlet Allocation (LDA) on the captured TCN, we were, for example, able to determine which topics the project teams have dealt with and which project members were mainly involved with certain topics over which periods of time.

Building on these efforts, we want to investigate the application of different data mining techniques to identify the "regular" working behaviour of developers. This includes, amongst others, the following properties: People that developers have frequently collaborated with, frequently occurring subgraphs that reflect their usual working behaviour (e.g., regular ticket updates after check-ins), average properties of created artefacts (e.g., average size of source code revisions, average email length, etc.), and topics that developers have been involved with on a frequent basis. Based on this data, we can detect deviations from the regular working behaviour and reason about possible learning effects, potential problems within the project teams (e.g., unevenly distributed workload between team members), or impacts of newly prescribed methods.

4 Industry Application

AnalyzeD [9], our prototype platform for virtual team collaboration analysis, is currently deployed in an industry setting at SAP. The test project aims at identifying a general framework for future applications with regards to technical and legal aspects.

On a technical level, data privacy and scalability are primary concerns. By replicating a subset of the information originally stored in, for example, project emails, bug tracking systems, or documentation tools, potential data leaks of the platform can reveal secret information to unauthorised third parties. Various approaches for solving such issues exist and can be integrated into the platform but a complete evaluation is beyond the scope of my work. The platform design accounts for an intra-company deployment and application of analyzeD in various projects, simultaneously. The respective graphs are stored independent from each other, hence, dedicated databases for each team and a central directory that links to their addresses could provide a simple, yet effective, scale out strategy.

Legal aspects are pervasive in the application of the platform. On the one hand, data privacy needs to be assured to comply with non-disclosure agreements. On the other hand, the system – at least in Germany – must not enable the identification of people involved in the projects as it could be used to, theoretically, "judge people's performance". To avoid such issues in the test project, German users were automatically filtered from the data collection process. As such filtering reduces the expressiveness of the collected data, future work needs to include the creation of guidelines that allow platform application even under such regulations.

As a first evaluation, the manager of the test project confirmed that the system is a viable aid for his project management tasks. It, for example, helped him to detect potential information loss due to some of his team members sharing viable project-related information only via email instead of using the designated documentation system. Analysis of the captured data and interviews with project members are currently ongoing and will be finished by the end of 2012. Results will be summarised in a joint paper with the project partners at SAP.

5 Summary and Outlook

This report presented extensions to the TCN implementation that allow to explicitly include process metrics into the process of virtual team collaboration analysis. This approach was used to analyse the virtual collaboration behaviour of student teams and reason about the effects of process conformance on team progress. These evaluations will be extended towards analysis of open-source projects in future work. Finally, we presented insights gained within an industry application of our monitoring and analysis system. Based on this work, next efforts will concentrate primarily on the evaluation of the applicability of the platform as a project management tool. To this end, we will perform an experiment that requires a chosen set of test subjects to perform different project management tasks, such as detecting violations of the described principles or finding experts for different project topics. Our concept will then be tested along with other project management tools that offer comparable features and we will measure, for example, the average time for violation detection or the number of found violations.

Teaching Assistance

- Organisation and teaching support – *Global Team-Based Product Innovation and Engineering 2011/12*
- Assistance in corporate sponsor acquisition – *Global Team-Based Product Innovation and Engineering 2012/13*
- Preparation of exercise and lecture – *Softwaretechnik II WS 2012/13*
- Co-supervised master's thesis *Agile Methodologies for Large Enterprises - An Analysis of Scaling Strategies* by Markus Steiner
- Co-supervised master's thesis *Extracting Topics from Heterogeneous Digital Collaboration Artifacts of Software Engineering Teams* by Ralf Gehrer

References

[1] Kent Beck. *Test-Driven Development: By Example*. The Addison-Wesley Signature Series. Addison-Wesley, 2003.

[2] Nicolas Bettenburg, Meiyappan Nagappan, and Ahmed E. Hassan. Think locally, act globally: Improving defect and effort prediction models. In *MSR '12: Proceedings of the 9th Working Conference on Mining Software Repositories*, pages 60–69. IEEE, 2012.

[3] James O. Coplien and Neil B. Harrison. *Organizational Patterns of Agile Software Development*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2004.

[4] Ralf Gehrer. Extracting topics from heterogeneous digital collaboration artifacts of software engineering teams. Master's thesis, Hasso Plattner Institute for IT Systems Engineering, 2012.

[5] Lorin Hochstein, Victor R. Basili, Marvin V. Zelkowitz, Jeffrey K. Hollingsworth, and Jeff Carver. Combining self-reported and automatic data to improve programming effort measurement. *SIGSOFT Softw. Eng. Notes*, 30:356–365, September 2005.

[6] Iman Keivanloo, Christopher Forbes, Aseel Hmood, Mostafa Erfani, Christopher Neal, George Peristerakis, and Juergen Rilling. A Linked Data platform for mining software repositories. In *9th IEEE Working Conference on Mining Software Repositories, MSR 2012, June 2-3, 2012, Zurich, Switzerland*, pages 32–35, 2012.

[7] Thomas Kowark, Philipp Dobrigkeit, and Alexander Zeier. Towards a shared repository for patterns in virtual team collaboration. In *5th International Conference on New Trends in Information Science and Service Science*, 2011.

[8] Thomas Kowark, Jürgen Müller, Stephan Müller, and Alexander Zeier. An educational testbed for the computational analysis of collaboration in early stages of software development processes. In *Proceedings of the 44th Hawaii International Conference on System Sciences (HICSS)*, January 2011.

[9] Thomas Kowark and Hasso Plattner. AnalyzeD: A Shared Tool for Analyzing Virtual Team Collaboration in Classroom Software Engineering Projects. In *The 2012 International Conference on Frontiers in Education: Computer Science and Computer Engineering*, July 2012.

[10] Thomas Kowark, Matthias Uflacker, and Alexander Zeier. Towards a shared platform for virtual collaboration analysis. In *The 2011 International Conference on Software Engineering Research and Practice (SERP '11)*, 2011.

[11] Tim Menzies, Bora Caglayan, Ekrem Kocaguneli, Joe Krall, Fayola Peters, and Burak Turhan. The promise repository of empirical software engineering data, June 2012.

[12] Matthias Uflacker. *Monitoring Virtual Team Collaboration: Methods, Applications, and Experiences in Engineering Design*. PhD thesis, Hasso Plattner Institute for IT Systems Engineering, Potsdam, Germany, 2010.

[13] Nico Zazworka, Victor R. Basili, and Forrest Shull. Tool supported detection and judgment of nonconformance in process execution. In *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, ESEM '09, pages 312–323, Washington, DC, USA, 2009. IEEE Computer Society.

[14] Nico Zazworka, Kai Stapel, Eric Knauss, Forrest Shull, Victor R. Basili, and Kurt Schneider. Are developers complying with the process: an XP study. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '10, pages 14:1–14:10, New York, NY, USA, 2010. ACM.

Muscle Propelled Force-feedback: ultra-mobile haptic devices

Pedro Lopes
Human Computer Interaction Group, Prof. Patrick Baudisch
Hasso-Plattner-Institut
pedro.lopes@hpi.uni-potsdam.de

Unlike many other areas in computer science and HCI, force-feedback devices resist miniaturization, because in order to build up force they require physical motors. We propose mobile force-feedback devices based on actuating the user's muscles using electrical stimulation. Since this allows us to eliminate motors and reduce battery size, our approach results in devices that are substantially smaller and lighter than traditional motor based devices. We present a prototype device that we mount to the back of a mobile phone. It actuates users' forearm muscles via four electrodes, causing the muscles to contract involuntarily, so that users tilt the device sideways. As users resist this motion using their other arm, they perceive these counter forces as force-feedback. We demonstrate the interaction at the example of an interactive videogame in which users try to steer an airplane through winds rendered using force feedback. We evaluate our approach in three experiments. In Study 1, we measured that the device delivers up to 17.5N of force when applied to the palm flexor. In Study 2, participants correctly identified force feedback direction with 97.7% accuracy eyes-free. Finally, in Study 3 participants reported their experiences playing the video game described above.

1 Introduction

Force-feedback has been used to enable eyes-free targeting [15], to increase task realism [2], and to enhance immersion in video games [17]. For such applications, force-feedback is preferred over vibrotactile, because it provides physical forces that can counter the users' movements, providing a strong haptic sensation [16]. Recent research has started to create increasingly smaller force-feedback devices, such as deformable devices (e.g., SqueezeBlock [3]), motor-based haptics based on pulleys (e.g., FlexTensor [14]). What limits researchers' miniaturization effort, though, is that it involves physical actuators that are hard to scale down without losing force [16].

In this paper, we attempt to push this evolution forward with the ultimate goal of bringing force feedback to mobile devices. In order to achieve this, we explore using the user's own muscle power as a replacement for motors. We actuate the user's muscles using electrical muscle stimulation (EMS), a technique first explored in the 60's and 70's [11] and more recently in Possessed Hand [13].

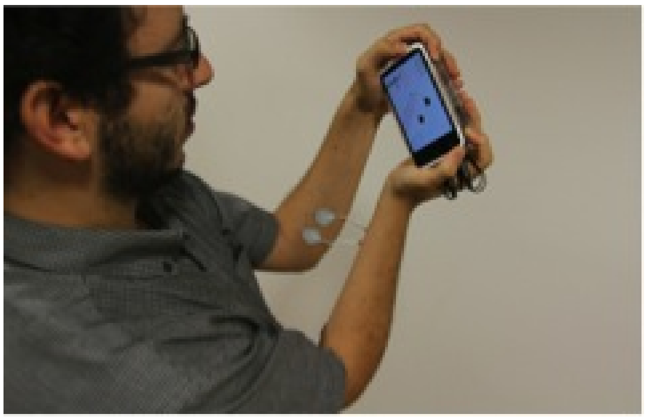


Figure 1: We developed a mobile force feedback device that electrically stimulates the user's muscles, instead of using mechanical actuators.

2 Mobile force-feedback device

Figure 1 illustrates the use of our mobile force feedback prototype, in a mobile gaming scenario. The device is mounted to the back of a mobile phone, and the player connects it using 2 skin-electrodes to the palm flexor muscles of each of his forearms. As shown in Figure 2, the game requires the user to steer an airplane through strong side winds. Our prototype renders these winds by tilting the user's arms sideways (Figure 2.a). It achieves this by stimulating muscle tissue in the user's arm though the electrodes, triggering an involuntary contraction. Since the airplane is controlled by turning the device like a steering wheel, the tilting derails the airplane. To stay on track, players counter the actuation using the force of their other arm (Figure 2.b). As we find in Study 3, players perceive this as force feedback.

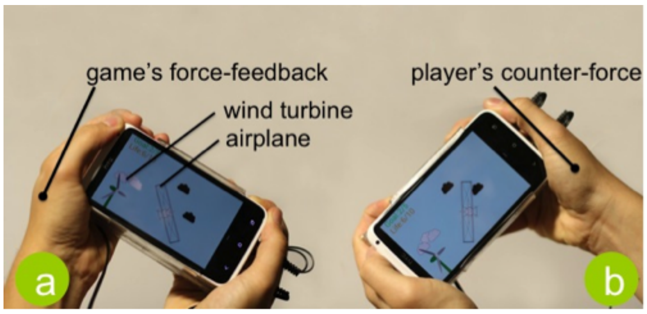


Figure 2: We demonstrate our prototype in a flight simulator game in which (a) the user's arm contracts as forces of side-winds drift the airplane, but rapidly (c) the user counter-forces and steers the plane against the wind.

2.1 Device hardware

Figure 3 shows a close-up of the hardware design. The device measures 133mm x 70mm x 20mm and weighs 163g. It is comprised of an arduino nano microcontroller, which communicates via USB or Bluetooth to its host device, such as a mobile phone, a battery-powered signal generator with medical compliant operational amplifiers which outputs a maximum current of 50V/100mA over a 500 ohm load, and reed relays. The device induces involuntary muscle contraction by generating a biphasic wave (frequency: 25Hz, pulse with of 290µs).

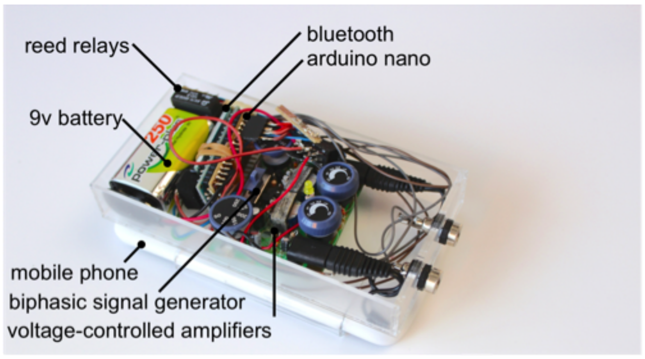


Figure 3: Hardware design specifications of the prototype.

3 Benefits and Contribution

Our main contribution is the concept of creating mobile force feedback using computer-controlled muscle stimulation. Our approach achieves miniaturization by (1) eliminating mechanical actuators, such as motors and (2) substantially reducing battery size, as it is two orders of magnitude more energy efficient to actuate a muscle (which receives its energy from the human body) than to drive a motor (we found muscles to require less than 100 mA to contract, while motors can use up to several Amps). With three simple user studies, we verify that our prototype creates sufficient force and that its effect is indeed perceived as force-feedback.

Limitations: the on-skin electrodes used by our prototype limit miniaturization [13] and deployment speed, in that users need to attach the device to their muscles before use. In addition to miniaturizing the circuit boards, future prototypes may overcome both of these limitations by using implanted intra-muscular electrodes [6].

4 Related Work

Force feedback is distinct from vibrotactile feedback in that it displaces the joints, with counter-forces. Force feedback devices administer force to body joints mechanically, by pulley systems [9], exoskeletons [14], and more recently deformable interfaces [3].

4.1 Motor-based Force-Feedback

An example of a pulley system is SPIDAR [9], which displaces the fingertip by pulling using four motors. Exoskeletons such as the Utah Dextrous Hand [5] or FlexTensor [14] require external apparatus to be mounted on the user. Furthermore, non-rigid actuation mechanisms include transmission of force by sound pressure [7] or air jets [12], even though these have not been shown yet to produce enough force to displace human joints.

4.2 Optimizing force-feedback for size and weight

Force-feedback devices that allow for a small form factor include deformable devices such as MimicTile [10], a flexible actuator placed on the side of a mobile phone that can dynamically regulate its stiffness using a shape memory alloy. SqueezeBlock [4] is a programmable spring device that provides force-feedback while grasped. InGen [3] is a self powered wireless rotary input device capable of force-feedback. Finally, Hemmert et al's device changes shape by moving a weight around [5].

4.3 Electrical Muscle Stimulation (EMS)

Recently, Tamaki et al. used EMS to actuate human fingers [13]. The technique targets situations where the user's input must be mediated or assisted, such as while learning to play a new instrument. With a similar approach, Kruijff et al. studied how haptic feedback through EMS is perceived on the biceps while gaming. In their studies, a human experimenter manually induces muscle contractions in participants playing videogames on a desktop computer [8].

5 Experiments

To validate our design, we conducted three user studies.

5.1 Study 1: Measurement of Generated Force

To determine whether the approach would deliver the sufficient force, we evaluated the force of the muscular output induced by our prototype.

5.1.1 Participants

We recruited 6 right-handed participants (all male), between 24 and 27 years old from our institution. They had no prior experience with EMS. Participants received a small compensation for their time.

5.1.2 Apparatus

As illustrated by Figure 4, the experimental apparatus actuated participant's wrist via disposable pre-gelled electrodes on the palm flexor muscles (flexor carpi radialis and partially the flexor digitorium superficialis) and measured the resulting force using a digital spring-scale.

First we applied a sequence of test patterns to get the participant acquainted with EMS. Then, we calibrated an intensity range per participant: minimum intensity with visible contraction up to maximum intensity within non-pain levels. These two values were linearly interpolated in a range of six intensities, numbered 1 (lowest) to 6 (highest).

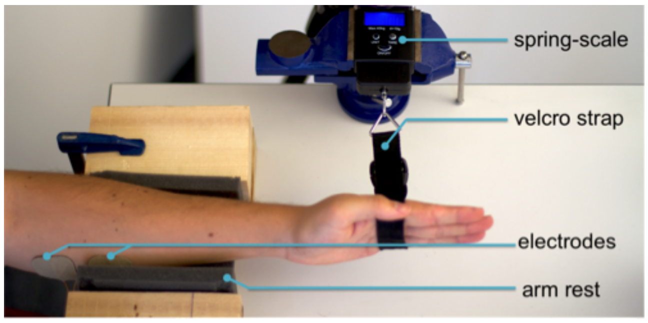


Figure 4: Apparatus for Study 1.

5.1.3 Task & Experimental Design

During the test, each participant was subjected to randomly distributed stimulation patterns that varied in intensity and duration: 6 intensities and 11 durations (two repetitions of each). Resulting in a total of 792 force readings: 6 (intensities) x 11 (durations) x 2 (repetitions) x 6 (participants).

5.1.4 Results and Discussion

Figure 5 shows the average of the peak force measured during each of the stimulation duration, across all users. As observed, force is directly related to the intensity level and duration. The maximal force was reached with 1s of stimulation at all users' highest intensity level, with an average of 1785g (17.5N). Furthermore, the lower bound of intensity seemed to normalize output force above 500ms stimulations. On the contrary, all other levels (2 to 6) showed an increase of force. For short durations (50-200ms) the variation of force is minimal, even amongst different intensities.

Finally, for all intensities levels above 2 and with stimulations longer than 400ms, participants reached more than 306g (3N), which is on par with the Phantom device [1]. These results suggest that our prototype creates sufficient force for real actuation.

5.2 Study 2: Proprioceptive and Force-Feedback

In this study, we investigated to what extent users perceive the force-feedback sensations generated by the prototype.

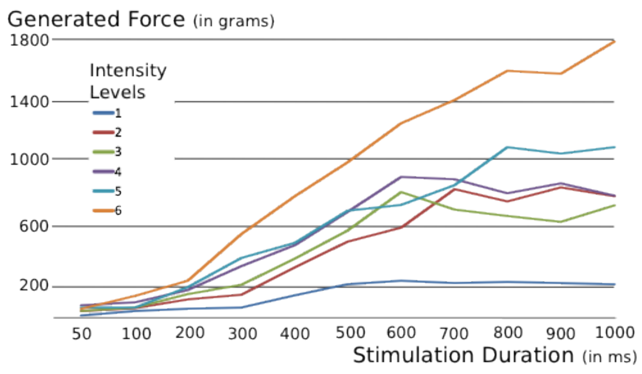


Figure 5: Average peak force (in grams) for palm flexion for different stimulation intensities (levels) and durations (in ms).

5.2.1 Participants

We recruited 8 new right-handed participants (all male) between 24 and 29 years old, from our institution. Participants received a small compensation for their time.

5.2.2 Apparatus

For this test the electrodes covered the palm flexor and extensor of one arm (palm moving inwards or outwards). To prevent participants from seeing their hands, which would bias the test outcome, we placed their hands under a tabletop, as shown in Figure 6. We instrumented the palm with reflective markers. OptiTrack motion capture system was used as a high-speed goniometer. For each participant we made sure that, below any noticeable pain-level, we achieved a palm displacement of 30 degrees, and we used that as the test intensity. Moreover, we logged the displacement of the joints during stimulations to further confirm if the joint in fact moved.

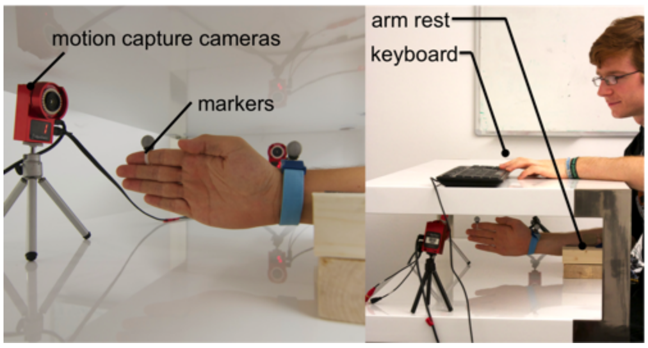


Figure 6: Apparatus for Study 2.

5.2.3 Controlling for tactile feedback

In EMS participants perceive actuation through proprioceptive and tactile feedback [8]. The latter is excited because the electric current is felt by the skin receptors for high stimulation levels. We minimized this effect by: lowering skin resistance with conductive gel and by lightly stimulating both muscles simultaneously below visible motion threshold, therefore causing the tactile sensation in both.

5.2.4 Experimental Design

Each participant was subjected to 44 trials of randomly distributed direction and duration (2 directions x 11 durations x 2 repetitions), a total of 352 samples for the study. Per trial, the participant entered on the keyboard the perceived direction (left or right).

5.2.5 Results and Discussion

Overall recognition rate of force-feedback direction was 97.73%, on average for all users. In detail, for brief stimulation (50ms), the recognition rate was 87.5%, for 100ms it was 90.63%, and 96.88% for 200ms. For all durations above 300ms, recognition rate was 100%, which pinpoints the potential of the approach for real mobile applications.

5.3 Study 3: Mobile force-feedback gaming

Finally, we examined actual use of the prototype. Participants played the game shown in Figure 1 including the force-feedback enhancements.

5.3.1 Participants

We recruited 10 participants (2 female) between 20 and 32 years old. Two of them had participated in Study 2.

5.3.2 Apparatus

We deployed the game from Figure 2 on an HTC One X. To win the game, players had to keep the airplane on screen while collecting white clouds and avoiding black clouds. Staying on screen required players to resist the wind turbines that âĂIJpushed the airplane off screenâĂİ. Participants steered the airplane left and right by tilting the device; and, touching the screen with the thumb allowed them to fly higher or lower.

5.3.3 Experimental Design

Participants played the videogame with and without force feedback for at least 5 minutes. Interface order was counterbalanced, i.e., half of the participants started with force-feedback. In the end of each condition they filled a questionnaire.

5.3.4 Questionnaire Results and Discussion

Participants rated the game as more enjoyable when playing with force-feedback (avg=4 of 5, sd=1.15), than without (avg=3.3 of 5, sd=0.95). While playing with muscle-propelled force feedback, participants stated to easily perceive the direction of the winds that derail the airplane in the game (avg=4 of 5, sd=1.63), confirming the results of Study 2. On the contrary, in the absence of the haptic sensation, subjects stated that wind direction and force was harder to perceive (avg= 2.7 of 5, sd=0.95). Also, participants rated the wind forces in the game as harder to counter when force-feedback was active (avg=3.8 of 5, sd=1.03), when compared to no force feedback (avg= 2.5 of 5, sd=0.85), which was expected after the force readings obtained in Study 1. After playing both games, all subjects expressed to prefer force-feedback to no force-feedback. Furthermore, all participants agreed that force-feedback sensations delivered by our prototype contributed to a positive gaming experience (avg=4.7 of 5, sd=0.48). Finally, we also registered several positive comments concerning sense of increased realism and excitement when gaming with force-feedback.

6 Conclusion

In this paper, we presented a mobile force-feedback device based on EMS. We demonstrated a prototype and illustrated its effect using a mobile gaming application. In three user studies we find that the device (1) generates up to 17.5N of force, (2) is correctly perceived 97.73% of the times by the users in eyes-free condition, and (3) contributes to an enjoyable mobile gaming experience. We believe our research points out a new direction of how force-feedback can effectively be miniaturized. We expect this to enable new application areas for mobile force-feedback in the future.

References

[1] Phantom senseable haptic device, http://www.sensable.com last accessed in 10/09/2012.

[2] Roland Arsenault and Colin Ware. Eye-hand co-ordination with force feedback. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI ’00, pages 408–414, New York, 2000. ACM.

[3] Akash Badshah, Sidhant Gupta, Gabe Cohn, Nicolas Villar, Steve Hodges, and Shwetak N. Patel. Interactive generator: a self-powered haptic feedback device. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI ’11, pages 2051–2054, New York, 2011. ACM.

[4] Sidhant Gupta, Tim Campbell, Jeffrey R. Hightower, and Shwetak N. Patel. Squeezeblock: using virtual springs in mobile devices for eyes-free interaction. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST ’10, pages 101–104, New York, 2010. ACM.

[5] John Hollerbach and Stephen C. Jacobsen. Haptic interfaces for teleoperation and virtual environments. In *in Proc. of First Workshop on Simulation and Interaction in Virtual Environments, Iowa City*, pages 13–15, 1995.

[6] Christian Holz, Tovi Grossman, George Fitzmaurice, and Anne Agur. Implanted user interfaces. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, CHI '12, pages 503–512, New York, 2012. ACM.

[7] Takayuki Iwamoto, Mari Tatezono, and Hiroyuki Shinoda. Non-contact method for producing tactile sensation using airborne ultrasound. In *Proceedings of the 6th international conference on Haptics: Perception, Devices and Scenarios*, EuroHaptics '08, pages 504–513, Berlin, Heidelberg, 2008. Springer.

[8] Ernst Kruijff, Dieter Schmalstieg, and Steffi Beckhaus. Using neuromuscular electrical stimulation for pseudo-haptic feedback. In *Proceedings of the ACM symposium on Virtual reality software and technology*, VRST '06, pages 316–319, New York, 2006. ACM.

[9] Jun Murayama, Laroussi Bougrila, Yanlinluo Katsuhito Akahane, Shoichi Hasegawa, Béat Hirsbrunner, and Makoto Sato. Spidar g&g: A two-handed haptic interface for bimanual vr interaction. In *Proceedings of EuroHaptics 2004*, pages 138–146, 2004.

[10] Yusuke Nakagawa, Akiya Kamimura, and Yoichiro Kawaguchi. Mimictile: a variable stiffness deformable user interface for mobile devices. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, CHI '12, pages 745–748, New York, 2012. ACM.

[11] Strojnik P, Kralj A, and Ursic I. Programmed six-channel electrical stimulator for complex stimulation of leg muscles during walking. *IEEE Trans Biomed Eng*, 26(2):112–116, 02 1979.

[12] Yuriko Suzuki, Minoru Kobayashi, and Satoshi Ishibashi. Design of force feedback utilizing air pressure toward untethered human interface. In *CHI '02 extended abstracts on Human factors in computing systems*, CHI EA '02, pages 808–809, New York, 2002. ACM.

[13] Emi Tamaki, Takashi Miyaki, and Jun Rekimoto. Possessedhand: techniques for controlling human hands using electrical muscles stimuli. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 543–552, New York, 2011. ACM.

[14] Dzmitry Tsetserukou. Flextorque, flextensor, and hapticeye: exoskeleton haptic interfaces for augmented interaction. In *Proceedings of the 2nd Augmented Human International Conference*, AH '11, pages 33:1–33:2, New York, 2011. ACM.

[15] Malte Weiss, Chat Wacharamanotham, Simon Voelker, and Jan Borchers. Fingerflux: near-surface haptic feedback on tabletops. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 615–620, New York, 2011. ACM.

[16] Alex Wright. The touchy subject of haptics. *Commun. ACM*, 54(1):20–22, January 2011.

[17] Lining Yao, Sayamindu Dasgupta, Nadia Cheng, Jason Spingarn-Koff, Ostap Rudakevych, and Hiroshi Ishii. Rope revolution: tangible and gestural rope interface for collaborative play. In *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology*, ACE '11, pages 11:1–11:8, New York, 2011. ACM.

Discovering SPARQL Query Templates for Dynamic RDF Triple Prefetching

Johannes Lorey
Information Systems Group
Hasso Plattner Institute

johannes.lorey@hpi.uni-potsdam.de

When issuing SPARQL queries against remote public endpoints, a user usually faces a number of limitations, e.g., regarding latency and availability. Especially when implementing a service relying on the information provided by the SPARQL endpoint, this can become a challenge as the service's quality directly depends on these factors. Here, it would be beneficial to detect similar requests and aggregate suitable information provided by the endpoint. Storing this data locally helps to respond to subsequent requests in a timely manner. In this work, we present an algorithm to detect similar queries and introduce the notion of query templates. Additionally, we illustrate several ways to augment SPARQL queries to prefetch data needed to answer subsequent queries. Lastly, we provide a brief evaluation of our approach based on real-world query logs.

1 Introduction

Public SPARQL endpoints provide valuable resources for a multitude of information needs, e.g., about drugs[1], government spendings[2], or cross-domain knowledge[3]. However, accessing this data is cumbersome as the capabilities of these endpoints are usually rather moderate. As most data is generated in a research context, the hardware resources available for hosting such a public infrastructure are limited, e.g., data is not replicated for fast access from different locations, query processing might take up to several seconds, or the endpoints suffer from availability restrictions.

For services querying data from SPARQL endpoints, these limitations may present a serious challenge. Processing retrieved data for future actions, such as integration or visualization, is delayed by at least the latency required to retrieve the results. This will impede interactive applications responding to dynamic user feedback, such as clicking on a point of interest on a map to retrieve information about a city. While circumventing some of these problems, downloading and integrating RDF data locally may also not be desirable (e.g., if there are frequent updates in datasets) or feasible (e.g., if such datasets are not provided) either.

Hence, there is a clear need for retrieving results for sequences of SPARQL queries in a more efficient way. In this work, we especially focus on detecting similar queries and sequences of such queries. We will use this information to rewrite queries early in the sequence to retrieve

[1]`http://www4.wiwiss.fu-berlin.de/drugbank/sparql`
[2]`http://govwild.hpi-web.de/sparql`
[3]`http://dbpedia.org/sparql`

(a subset of) the results of later queries. We can then store these results locally to reduce the latency for subsequent queries by querying this local RDF store.

The rest of this work is organized as follows: In Sec. 2 we present some preliminaries on SPARQL, the de facto query language for RDF data, and basic notations required for our approach. After that, we introduce an algorithm for detecting query templates, i.e., SPARQL queries that non-trivially subsume other SPARQL queries, in Sec. 3. In Sec. 4, we illustrate various rewriting strategies that can be applied to queries in a query session to retrieve results that may be useful to answer subsequent queries in the same session. We call this process query augmentation. To evaluate both contributions, i.e., template discovery and query augmentation, we conducted a number of experiments and present some results in Sec. 5. We conclude this work and comment on future work in Sec. 6.

2 SPARQL

SPARQL is the de facto standard query language for RDF triples. One central concept of a SPARQL query is that of a triple pattern $T = (s,p,o) \in (V \cup U) \times (V \cup U) \times (V \cup U \cup L)$ with V being a set of variables, U being a set of URIs, and L being a set of literals. A SPARQL query Q contains a number of graph patterns $P_1, P_2, \ldots$ which are defined recursively: (i) A valid triple pattern T is a graph pattern. (ii) If P_1 and P_2 are graph patterns, then P_1 `AND` P_2, P_1 `UNION` P_2, and P_1 `OPTIONAL` P_2 are graph patterns [5]. Notice that curly braces surrounding a graph pattern (i.e., $\{P\}$) are syntactically required for both P_1 and P_2 in a `UNION` statement, only for P_2 in an `OPTIONAL` statement, and are optional elsewhere. However, to increase readability, we will usually omit them in this work.

In terms of relational operations, the keyword `AND` represents an inner join of the two graph patterns, `UNION` unsurprisingly denotes their union, and `OPTIONAL` indicates a left outer join between P_1 and P_2. Whereas `UNION` and `OPTIONAL` are reserved keywords in actual SPARQL queries to indicate the corresponding connection between two graph patterns, the `AND` keyword is omitted. In [5], it is shown that there exists a notion of a normal form for SPARQL queries based on the recursive graph pattern structure presented earlier and the precedence of the operators connecting those graph patterns. Hence, for this work we assume a SPARQL SELECT query can always be expressed as a composition of graph patterns, connected either by `UNION`, `AND`, or `OPTIONAL`.

For this work, we define the three functions $\Theta_{\texttt{UNION}}(P)$, $\Theta_{\texttt{AND}}(P)$, and $\Theta_{\texttt{OPTIONAL}}(P)$ as follows (all $n \geq 2$):

$$\Theta_{\texttt{UNION}}(P) = \begin{cases} \{P_1, \ldots, P_n\}, & \text{iff } P := P_1 \texttt{ UNION } P_2 \ldots \texttt{ UNION } P_n \\ \emptyset, & \text{else.} \end{cases}$$

$$\Theta_{\texttt{AND}}(P) = \begin{cases} \{P\}, & \text{iff } P \text{ is a triple pattern} \\ \{P_1, \ldots, P_n\}, & \text{iff } P := P_1 \texttt{ AND } P_2 \ldots \texttt{ AND } P_n \\ \emptyset, & \text{else.} \end{cases}$$

$$\Theta_{\texttt{OPTIONAL}}(P) = \begin{cases} \{P_1, \ldots, P_n\}, & \text{iff } P := P_1 \texttt{ OPTIONAL } P_2 \ldots \texttt{ OPTIONAL } P_n \\ \emptyset, & \text{else.} \end{cases}$$

We also define the function $\Theta(P)$:

$$\Theta(P) = \begin{cases} \Theta_{\texttt{UNION}}(P), & \text{if } \Theta_{\texttt{UNION}}(P) \neq \emptyset \\ \Theta_{\texttt{OPTIONAL}}(P), & \text{if } \Theta_{\texttt{OPTIONAL}}(P) \neq \emptyset \\ \Theta_{\texttt{AND}}(P), & \text{else.} \end{cases}$$

We call $|P| = |\Theta(P)|$ the *size* of a graph pattern. Moreover, we call the graph pattern P in a query Q the *query pattern* P_Q of Q if for all graph patterns P_i in Q (including P itself) $P \notin \Theta(P_i)$. Note that every query has exactly one query pattern P_Q.

In addition, we also introduce a keyword function $\kappa(P)$:

$$\kappa(P) = \begin{cases} \texttt{UNION}, & \text{iff } \exists P_1 \in P_Q : P \in \Theta_{\texttt{UNION}}(P_1) \\ \texttt{OPTIONAL}, & \text{iff } \exists P_1, P_2 \in P_Q : P, P_2 \in \Theta_{\texttt{OPTIONAL}}(P_1) \wedge P_2 \texttt{ OPTIONAL } P \\ \texttt{AND}, & \text{else.} \end{cases}$$

3 Discovering Query Templates

In real-world applications, a large number of queries processed by a SPARQL endpoint exhibit similar structures and vary only in a certain number of resources. In this section, we present query templates that can be used to cluster these similar SPARQL query structures. To identify such query structures, we present a triple pattern similarity metric that is used for our graph pattern matching algorithm. In contrast to [6], we do not rely on knowledge of the underlying RDF graph for this.

3.1 Triple Pattern Similarity and Merging

We first define triple patterns that can be mapped to and merged with one another. To establish a mapping between two triple patterns $T_1 = (s_1, p_1, o_1)$ and $T_2 = (s_2, p_2, o_2)$, we try to match the individual elements of T_1 with the corresponding part of T_2, i.e., we align x_1 with x_2 for $x \in \{s, p, o\}$.

To calculate the distance of such mappings, we introduce the score $\Delta(x_1, x_2)$:

$$\Delta(x_1, x_2) = \begin{cases} \frac{1}{3} * \frac{d(x_1, x_2)}{max(|x_1|, |x_2|) + 1}, & \text{if } x_1 \in V \wedge x_2 \in V \\ \frac{d(x_1, x_2)}{max(|x_1|, |x_2|) + 1}, & \text{if } (x_1 \in U \wedge x_2 \in U) \vee (x_1 \in L \wedge x_2 \in L) \\ 1, & \text{else.} \end{cases}$$

Here, $|x|$ is the string length of x and $d(x_1, x_2) \to \mathbb{N}_0$ is a string distance metric with $d(x_1, x_2) = 0 \Leftrightarrow x_1 = x_2$. In our work, we use the Levenshtein distance. Notice that we apply the Levenshtein distance on the entire resource strings, i.e., including possible prefix definitions for URIs or types for literals.

To evaluate how easily two triple patterns can be merged, we define the triple pattern distance score $\Delta(T_1,T_2)$ as the sum of the triple parts individual distance scores, i.e.,

$$\Delta(T_1,T_2) = \Delta(s_1,s_2) + \Delta(p_1,p_2) + \Delta(o_1,o_2)$$

We also allow the calculation of distance scores between two graph patterns P_1,P_2 as follows:

$$\Delta(P_1,P_2) = \begin{cases} \Delta(T_1,T_2), & \text{if } \Theta(P_1) = \{T_1\} \wedge \Theta(P_2) = \{T_2\} \\ \infty, & \text{else.} \end{cases}$$

$\Delta(P_1,P_2)$ notation will mainly serve as a shorthand notation for analyzing graph patterns with size 1, i.e., graph patterns that constitute triple patterns.

Finally, we introduce the merge function $\pi(T_1,T_2) = \hat{T}$ that takes as input two triple patterns T_1, T_2 and merges them into one $\hat{T} = (\hat{s},\hat{p},\hat{o})$. It does so by simply replacing the non-equal triple pattern elements between $T_1 = (s_1,p_1,o_1)$ and $T_2 = (s_2,p_2,o_2)$ with variables. More formally, we first define $\pi(x_1,x_2)$ on the triple pattern parts with $x \in \{s,p,o\}$:

$$\pi(x_1,x_2) = \begin{cases} x_1, & \text{if } \Delta(x_1,x_2) = 0 \\ ?var_x, & \text{else.} \end{cases}$$

Here, $?var_x$ refers to an arbitrary, uniquely named variable in the triple pattern. Thus,

$$\pi(T_1,T_2) = (\pi(s_1,s_2), \pi(p_1,p_2), \pi(o_1,o_2))$$

In particular, this means that $\hat{T} = T_1$ iff $\Delta(T_1,T_2) = 0$, i.e., no merging is necessary, if the two triple patterns are identical. As with Δ, we use the shorthand notation $\pi(P_1,P_2)$, if $|P_1| = |P_2| = 1$.

3.2 Graph Pattern Matching

Using the triple pattern distance notion, we can now derive matches between graph pattern. We consider the task of finding these matches a variation of the stable marriage problem [4] which we solve recursively using Algorithm 1. The algorithm takes as arguments two graph patterns P_1,P_2, a maximum distance threshold Δ_{max} for mapping any two triple patterns, and an existing mapping between triple patterns. This mapping is initially empty and is established recursively by iterating over all graph patterns contained in P_1 and P_2. However, if no mapping can be derived, the final result of the algorithm is empty.

Any non-empty mapping resulting from Algorithm 1 can be considered stable in the sense that the included triple patterns have a minimal (local) distance score to their mapping partner with respect to the query structure. There might be cases where the algorithm detects two mappings T_i,T_j and T_k,T_l, where $\Delta(T_i,T_j) > \Delta(T_i,T_l)$ or $\Delta(T_i,T_j) > \Delta(T_k,T_j)$, but no mapping between T_i and T_l or T_k and T_j was discovered, even though the global distance score for these mappings would be lower. This happens, when T_i and T_l or T_k and T_j reside in different graph patterns that cannot be mapped to one another, e.g., because of different number of triple patterns or different keywords. If for any evaluated graph pattern no match could be determined, the overall return value of the algorithm is an empty set of mappings. Conversely, any non-empty mapping result is complete (or perfect) and therefore maximal (the size of non-empty mappings is determined by the number of triple patterns contained in the graph pattern).

Algorithm 1: GraphPatternMatching

```
Input  : $P_1, P_2$ : Two graph patterns
Input  : $\Delta_{max}$ : Triple pattern distance threshold
Input  : $mappings$ : Current triple pattern mappings
Output: $mappings$ : Triple pattern mappings between $P_1, P_2$
1  $S_1 \leftarrow \Theta(P_1)$
2  $S_2 \leftarrow \Theta(P_2)$
3  if $\kappa(P_1) \neq \kappa(P_2) \vee |S_1| \neq |S_2|$ then
4      return $\emptyset$
5  while $S_1 \neq \emptyset$ do
6      $P_1^i \leftarrow S_1$.pollFirst()
7      $foundMapping \leftarrow$ false
8      foreach $P_2^j \in S_2$ do
9          if $\left|P_1^i\right| = 1 \wedge \left|P_2^j\right| = 1$ then
10             $P_1^* \leftarrow mappings$.get($P_2^j$)
11             if $P_1^* = NIL$ then
12                 if $\Delta(P_1^i, P_2^j) > \Delta_{max}$ then
13                     continue
14                 $mappings$.put($P_2^i, P_1^i$)
15                 $foundMapping \leftarrow$ true
16                 break
17             else
18                 if $\Delta(P_1^i, P_2^j) < \Delta(P_1^*, P_2^j)$ then
19                     $mappings$.put($P_2^j, P_1^i$)
20                     $S_1$.add($P_1^*$)
21                     $foundMapping \leftarrow$ true
22                     break
23         else
24             $oldMappings \leftarrow mappings$
25             GraphPatternMatching($P_1^i, P_2^j, mappings$)
26             if $oldMappings \neq mappings$ then
27                 $foundMapping \leftarrow$ true
28     if $\neg foundMapping$ then
29         return $\emptyset$
30 return $mappings$
```

3.3 Query Templates and Clusters

Using the output of Algorithm 1, we can now discover *query templates*. The idea of query templates builds on the findings discussed in [7], where the authors mine SPARQL query log files to determine the behavior of agents issuing the respective query. We extend this approach by establishing a formal definition of what constitutes a query template and how to find it. In contrast to previous work, we also show a concrete application of query templates in the next section.

We use a maximum distance score of 1, i.e., two triple patterns may only differ in either their (non-variable) subject, predicate, or object. To determine a query template $\hat{Q}$, we evaluate the mappings generated by GraphPatternMapping$(P_{Q_1}, P_{Q_2}, 1, \emptyset)$ for two SPARQL queries Q_1, Q_2 with query graph patterns P_{Q_1}, P_{Q_2}, respectively. If the output of Algorithm 1 is empty or $Q_1 = Q_2$ (i.e., all triple pattern mappings $(T_1, T_1) \in map$ are trivial), no query template can be derived. Otherwise, we initialize the query template $\hat{Q}$ with the query Q_1 and replace all triple patterns T_1 in $\hat{Q}$ with the merged triple pattern $\hat{T}$ that resulted from $\pi(T_1, T_2)$ where $(T_1, T_2) \in map$.

All queries sharing a query template form a *query cluster*. Queries in a cluster have the same basic structure and differ in at least one triple pattern by a maximum of Δ_{max} non-variable triple pattern parts. We assume that for most queries in such a cluster a single resource or literal is replaced throughout all triple patterns by a machine agent as indicated by the findings in [7]. Note that query clusters may be overlapping.

4 SPARQL Query Augmentation

The core idea of our work is to rewrite SPARQL queries in such a way that the result set of the new query contains at least the same bindings as the result set of the old query. Ideally, results retrieved in addition to those of the original query can be materialized in a local triple cache and used to speed up future query evaluation by providing (a subset of) the results retrieved by subsequent queries. In this section, we introduce the concept of query augmentation[4], i.e., the rewriting of queries in the aforementioned way.

4.1 Augmentation Constraints

We present four different augmentation methods:

1. *Query Broadening*, i.e., the generalization of a query based on its query template.
2. *Random Triple Removal*, i.e., the removal of an arbitrary triple pattern.
3. *Selective Triple Removal*, i.e., the removal of the most selective triple pattern.
4. *Language Substitution*, i.e., removing filter and resource language restrictions.

4.2 Query Broadening

For query broadening, we simply issue the query template against the SPARQL endpoint instead of individual queries. Obviously, this requires that a query template has already been discovered, and typically this method results in a larger result set compared to the one for

[4]We use the term augmentation in the sense of semantic enhancement. Some of the augmentation types introduced here actually reduce the query length instead of augmenting it.

the original query. If another query was found previously and we were able to establish the query template, this other query potentially generated a result set, hence the result set of the discovered query template includes at least the results of the two queries sharing this query template.

Broadening a query will benefit any subsequent queries in a session that share a template with this query. This is usually the case when a query session contains a large amount of queries from machine agents, e.g., a service that retrieves information based on interactions with a user interface. Such a service typically issues queries using query blueprints and modifies them based on these interactions. For example, in our evaluation we discovered that a many queries from a specific source retrieved longitude and latitude information about cities.

4.3 Random Triple Removal

Removing random triples is a somewhat naïve, yet surprisingly successful augmentation methods. As the name indicates, we here randomly remove a triple pattern from a query pattern. The only requirement for this removal is that the triple pattern does not contain any projection variable that only occurs in this triple pattern. If we were to remove the only triple pattern containing a projection variable, the returned result set would not include the entire data the user queried for. In all other cases, removing a triple pattern will simply remove a constraint for one (or more) variables, which in turn will again result in a potentially larger result set.

In a somewhat hypothetical case, a query (pattern) might contain a triple pattern comprised only of non-variable subject, predicate, and object. While this is legitimate in a SPARQL query, the usage of such a construct would offer little benefit: If the triple pattern does not match in the RDF graph, the result set for the query (pattern) is empty, else the result set is equal to the query (pattern) without the triple pattern in question. Hence, when considering triple patterns to remove, we disregard any triple patterns containing no variable.

Removing random triples in a query might benefit the corresponding result set, if the restrictions contained in the query are too strict. This might be caused by a lack of data in the RDF graph the query is issued against. Here, even though a machine or human agent might have reason to believe that the triple pattern is part of the RDF graph, it cannot be matched. The reasons for this include spelling errors in the query or RDF data as well as ontology misusage and ontology evolution. For example, in previous work we discovered that the DBpedia ontology has a number of redundant properties (such as `http://dbpedia.org/ontology/Person/weight` and `http://dbpedia.org/ontology/weight`). Additionally, we found that the namespace of several properties had been adjusted over time (e.g., from `http://dbpedia.org/property/` to `http://dbpedia.org/ontology/`) [1].

4.4 Selective Triple Removal

This augmentation type can be considered a variation of the random triple removal introduced above. Instead of removing an arbitrary triple pattern, we identify the most selective triple pattern according to the heuristic presented in [8]. The basic notion here is that subjects are more selective than objects and objects are more selective than predicates. Using this heuristic, we calculate scores for all triple patterns based on whether the subject, predicate, or object is either a variable or not. A more sophisticated approach could incorporate ontology information and available metadata (e.g., provided by the Vocabulary of Interlinked Data and determined by our approach presented in [3]). However, this data might not be available in all cases.

Again, the same restrictions discussed for the random triple removal apply. We disregard any triple patterns containing no variables as they do not provide helpful selectivity information. Additionally, we do not remove any triple patterns that contain a projection variable which is used in no other triple pattern as doing so would reduce the amount of information returned to the user.

4.5 Language Substitution

There are essentially two methods to query for language-specific results in SPARQL: The user can either use a filter condition on a variable using the `langMatches` and `lang` keywords (e.g., `FILTER langMatches(lang(?abstract), "fr" ))` or indicate the language of a literal in a triple pattern (e.g., `?city rdfs:label "Berlino"@it`). In both cases, the language tags used are defined by the ISO 639 standard [2]. While in some large mixed-language knowledge bases such as DBpedia these language tags are provided explicitly, usually they are omitted when generating RDF data. Hence, when applying filter conditions using those language tags the retrieved results are empty. Moreover, the vast amount of language-specific facts is typically available in English, thus using any other language tag might severely reduce query results.

Thus, when discovering language tags or filter conditions in a query, there are two viable augmentation solutions. First, all language-specific restrictions may be removed, allowing to retrieve information provided without language annotation. Second, the language tags may be changed to English. While this second option might help to retrieve more results, it is prone to alter the semantics of a query. Consider the example used earlier: When changing the language tag in the triple pattern `?city rdfs:label "Berlino"@it` to “`@en`”, the query will not return the same (or any) results.

5 Evaluation

We analyzed the DBpedia 3.6 query log files contained in the USEWOD2012 Dataset[5] in the evaluation of our augmentation approach. We chose these particular log files for three reasons:

- The query intention is to some extent comprehensible to non-domain experts.
- We were able to locally set up a SPARQL endpoint containing the same data as the one the queries were originally issued against.
- All queries are assigned a source (hashed IP address) and timestamp (hourly granularity), allowing us to to identify query sessions.

To illustrate the last point, an excerpt of the query log file *2011-01-24.log* is shown in Listing 1. Each line starts with the hashed IP address of the issuing source followed by the timestamp and the actual query. As Listing 1 indicates the level of granularity of the query log is hours. For our experiments, we consider all queries from one user within one hour to

[5]`http://data.semanticweb.org/usewod/2012/challenge.html`

constitute a query session. For the remainder of this section, we present exemplary results for the query log file *2011-01-24.log*, which are similar for all files contained in the dataset.

```
237fbf63e8449c1ade56eb7d208ce219 − [24/Jan/2011 01:00:00 +0100] "/sparql/?query..."
f452f4195b4d2046c77ad98496c1b127 − [24/Jan/2011 01:00:00 +0100] "/sparql/?query..."
9b1d83195dd251275c55c12ac2efa43d − [24/Jan/2011 02:00:00 +0100] "/sparql/?query..."
f452f4195b4d2046c77ad98496c1b127 − [24/Jan/2011 02:00:00 +0100] "/sparql/?query..."
```

Listing 1: Excerpt from query log file *2011-01-24.log*.

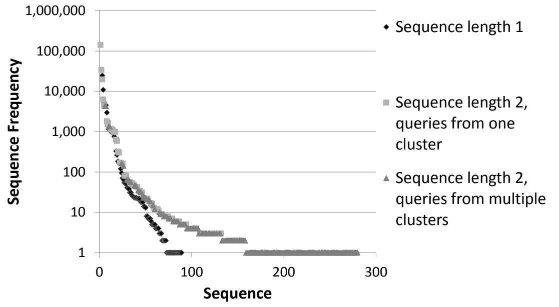
5.1 Query Session Analysis

For our first evaluation, we analyze variable-length sequences of queries within a query session and correlate the individual queries in such a sequence to the clusters they belong to. Here, we consider sequences of length 1, i.e., a single query within a query session, length 2, i.e., two successive queries in a session, and length n where n corresponds to the entire query session length.

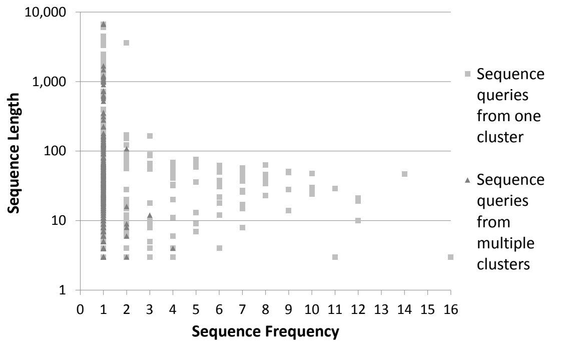
In Fig. 1(a), we illustrate the number of unique query sequences of length 1 and 2. As sequences of length 2 represent combinations of sequences of length 1, there are obviously more query sequences of length 2 (280 sequences) than of length 1 (89 sequences), but still far less than potential combinations of sequences of length 1 ($89 \times 89 = 7921$ sequences). Hence, we only observed a limited number of possible query sequence combinations. Additionally, we evaluated if the two queries in sequences of length 2 belong to the same or different clusters. We discovered that for sequences of length 2 occurring most often, both queries are from the same query cluster.

Figure 1(b) illustrates how long query sessions were compared to the frequency of the sequence they comprised. A large number (around 79.68%) of session sequences are unique, i.e., they were only observable once in the log. Additionally, a majority of query sessions (around 54.67%) have a length of 10 to 100 queries, whereas only about 23.97% have a length of over 100 queries, and approx. 21.34% are less than 10 queries long. While almost half the query sequences contain queries from only one cluster (about 48.53%), these sequences tend to be longer than those where the queries stem from multiple query clusters.

We also evaluated the conditional probability of sequences of length 2 for all query clusters discovered in the log. The results are presented in Fig. 2. Here, both query axes Q_i and



(a) Frequency of query sequences with length 1 and length 2.



(b) Frequency of query session sequences correlated with query sessions lengths.

Figure 1: Query sequence lengths and corresponding frequencies.

Q_{i+1} correspond to the query clusters, a single tick mark on each axis represents one cluster. Both axes contain the same query cluster references and are sorted in descending cluster size starting at the origin, where the first ten query clusters account for over 98% of all queries discovered in the log. The values illustrate the probability of observing a query from a certain cluster given the cluster of the previous query. A high value indicates that queries from two query clusters are likely to occur in sequence.

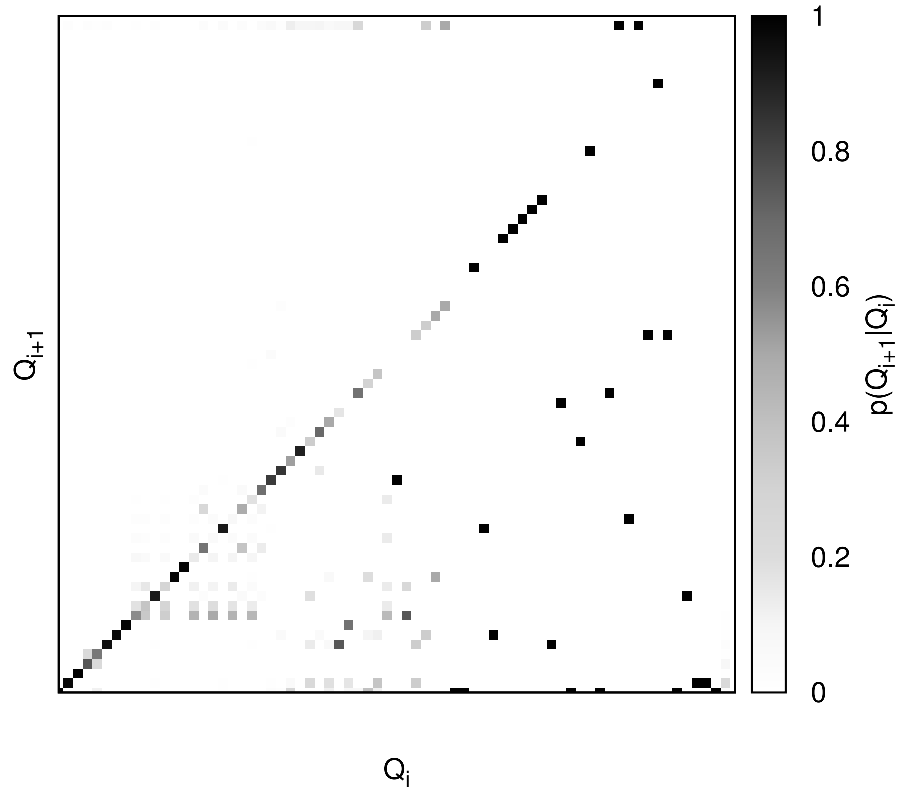


Figure 2: Conditional probabilities for sequences of length 2 for all query templates.

As can be observed in Fig. 2, the matrix of all conditional probabilities is sparsely populated, i.e., for a query belonging to any given query cluster discovered in the log, the subsequent query usually belongs to one of a limited number of clusters. Moreover, especially queries of the first nine query clusters, which in total account for nearly 97% of all queries, are most likely to be followed by queries from the same cluster (the conditional probability for this is 57.22% or more for all of them). Overall, the same applies to many queries analyzed as illustrated by the high correlation on the diagonal in Fig. 2. Similar results were presented in [7], where the author discusses that a large portion of real-world SPARQL queries are issued by machine agents and follow certain "blueprint" structures. Typically, between two subsequent queries issued by such an agent there are only small variations (e.g., in one resource or literal).

These findings have direct implications for our query augmentation approach. One important discovery is that many query sequences contain queries from the same cluster, i.e., queries that have the same basic structure, but differing in a limited number of triple patterns. Intuitively, this especially validates our query broadening augmentation notion. Instead of issuing n queries gathering only one result set at at time, we could send just one query, receive the combined result set, and thus circumvent the latency of $n-1$ individual queries.

Method	Sequences	Retrieved Triples
Query Broadening	17,885	33,446
Random Triple Removal	3,970	4,549
Selective Triple Removal	390	780
Language Substitution	8,258	15,402
Baseline	5,155	10,540
None	40,105	0

Table 1: Query augmentation evaluated on 75,763 query sequences of length 2.

5.2 Query Augmentation Analysis

Lastly, we illustrate some results of our augmentation methods. For this, we analyzed about 75,000 query sequences Q_i, Q_{i+1} of length 2 and evaluated how one of our augmentation method for Q_i results in retrieving more results for Q_{i+1} than the baseline approach. The results are presented in Tab. 1. Here, the second column indicates in how many cases the corresponding method retrieved the most results. For a large amount of the analyzed queries Q_i (around 52.9%), neither the original query nor the augmented queries could retrieve any results. This might be either because the DBpedia SPARQL endpoint we set up did not contain the same data as the endpoint the original queries were issued against or simply because there are no valid results for these queries in the DBpedia 3.6 dataset.

As is illustrated in Tab. 1, the best augmentation method is query broadening. For around 23.6% of all query sequences, we were able to retrieve more results by using this augmentation than the baseline approach (6.8%). The large amount of query sequences of length 2 for which both queries originate from the same query cluster benefit greatly from the broadening approach. Clearly, using one generic query to retrieve results for a number of more specific queries eliminates the need for issuing these queries.

For the triple removal methods, the random method seems more successful than the selective approach. However, in some cases both strategies result in the same augmented query (i.e., when the randomly removed triple pattern is the most selective one). In our evaluation, we only counted this towards the random removal approach, thus the overall score for the selective triple removal method might be higher in practice. Still, both methods are inferior to the language substitution approach, which accounts for around 10.9% of all query sequences. For the latter method, we substituted all language tags to English. Clearly, this approach benefits from the fact that the DBpedia contains a large amount of English-tagged literals compared to the other language versions. However, as illustrated in Sec. 4.5, the results might not be as useful to the query issuing agent.

6 Summary and Outlook

To support the consumption of Linked Data, there is a definite need to enhance the accessibility of these resources. As typically RDF data is queried using SPARQL, optimizing these requests can potentially increase the acceptance of service developers to take advantage of Linked Data as results may be provided faster and more reliably. In this work, we presented and evaluated two contributions for optimized access of SPARQL endpoints: Discovering query templates, i.e., detecting similar SPARQL queries, and query augmentation, i.e., rewriting queries to retrieve more results fitting for subsequent requests.

In future work, we plan on using the evaluation results presented in Sec. 5 to train a classifier to augment queries once they are issued by the agent. This will enable us to gradually enrich a local cache of RDF triples that can then be used to provide results for subsequent queries. For managing this cache, we will also focus on how query results from one agent can be used for other users and how cache results are invalidated. Additionally, we plan on implementing more augmentation methods better suited for the query sequences we found in our evaluation. For this, we want to analyze how human and machine agent requests differ, and how this knowledge can be used to find proper augmentation methods.

References

[1] Ziawasch Abedjan, Johannes Lorey, and Felix Naumann. Reconciling ontologies and the web of data. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, Maui, HI, USA, October 2012.

[2] Harald Alvestrand. RFC 3066 - tags for the identification of languages. Technical report, IETF, 2001.

[3] Christoph Böhm, Johannes Lorey, and Felix Naumann. Creating voiD descriptions for web-scale data. *Journal of Web Semantics*, 9(3):339–345, 2011.

[4] David Gale and Lloyd Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[5] Claudio Gutierrez Jorge Pérez, Marcelo Arenas. Semantics and complexity of SPARQL. *ACM Transactions on Database Systems (TODS)*, 34(3):16:1–16:45, September 2009.

[6] Jens Lehmann and Lorenz Bühmann. AutoSPARQL: let users query your knowledge base. In *Proceedings of the Extended Semantic Web Conference (ESWC)*, pages 63–79, Crete, Greece, 2011.

[7] Aravindan Raghuveer. Characterizing machine agent behavior through SPARQL query mining. In *Proceedings of the International Workshop on Usage Analysis and the Web of Data (USEWOD)*, Lyon, France, April 2012.

[8] Markus Stocker, Andy Seaborne, Abraham Bernstein, Christoph Kiefer, and Dave Reynolds. SPARQL basic graph pattern optimization using selectivity estimation. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 595–604, New York, NY, USA, April 2008. ACM.

The Role of Objects in Process Model Abstraction

Andreas Meyer
Business Process Technology Group
Hasso Plattner Institute
andreas.meyer@hpi.uni-potsdam.de

This report outlines my phd research project targeting the field of process model abstraction. While currently existing techniques allow the abstraction of control flow in various kinds, e.g., semantically or structurally, objects are not considered yet in the process of process model abstraction. Therefore, we will provide new means for process model abstraction involving object abstraction – as extension to existing control flow abstractions to enrich these techniques as well as as standalone approach without affecting control flow. However, the process model to be abstracted needs to be correct to provide meaningful and useful abstraction results. Again, for control flow, many correction criteria and checking methods do exist, e.g., various soundness checks and the field of process compliance, but object specific correctness is supported rarely. We will introduce the notion of weak conformance, which extends the existing conformance notion with means to handle not fully specified process models with respect to the given object life cycle as for instance abstracted process models. Thereby, object life cycles determine the actions allowed to be performed by an activity on the corresponding object based on the current state of the object. Finally, the proposed techniques for object abstraction and object correctness will be evaluated in a user study, for which process model collections are required, which provide process models with explicit object modeling. Unfortunately, such collections are very rare. Therefore, we will introduce an approach to derive object information from activity and event labels and to transform the given process model with no or incomplete object specification into one with a more complete explicit object specification.

1 Introduction

Business process management as "a systematic, structured approach to analyze, improve, control, and manage processes with the aim of improving the quality of products and services" [5] is an important approach to manage work in organizations, with process models being the key artifacts [20]. Process modeling usually comprises two aspects: Control flow and object flow. Control flow defines possible execution sequences of activities, whereas object flow describes the exchange of information between the activities by writing to and reading from these objects. An object can be formalized as set of object states and transitions between them, i.e., as labeled transition system, which is usually referred to as object life cycle. An object life cycle can be used to identify the current object state of the object and the set of reachable object states from the current one [3]. Additionally, objects have dependencies amongst each other, e.g., *is-a* and *part-of* relationships. These can be represented by using a class diagram – an object model.

The work managed by business process management is interesting for many different stakeholders of the organization, e.g., the board, the middle management, the clerk, or the IT staff, who is implementing the business processes to support the clerks with IT infrastructure while they execute their processes. Each of these groups has different requirements with respect to the information to be displayed in the corresponding process model, usually provided by the use of process model views [2,19]. One method to create such views is process model abstraction [17,18] as it allows the presentation of the very same business process in different views to various stakeholders by avoiding inconsistencies through several independent models; each view is adapted to the specific needs of the corresponding stakeholder by leaving out, aggregating, and/ or refining elements of the process model [6]. Thereby, existing techniques only consider control flow aspects for abstraction although objects are the driving force for process execution and important for process controlling and understanding as mentioned above. Therefore, we add object support to process model abstraction by extending the existing techniques relying on fragment-based control flow abstraction, e.g., [16]. We assume structurally sound process models and an explicitly defined object model. Object abstraction decisions are based on occurrences of objects in the process model and their relations in the object model as follows. Specifically, control flow and object behavior correspond to each other such that we provide a rule set, which allows object abstraction closely related to control flow abstraction. The utilized existing fragment-based control flow abstraction technique identifies the elements of the process models affected for a single abstraction step and based on this information, the objects get deleted, composed, or are preserved as is.

However, object abstraction requires correct object specifications in the source process model to be able to provide useful abstraction results with correctness referring to, for instance, safe execution of process models; i.e. the execution shall terminate properly. The execution semantics of process models are often described by Petri nets [14], which describe which activities can be executed based on activity enablement. In order to achieve safe execution, it must be ensured that every time an activity tries to access an object, the object is in a certain expected object state or is able to reach that expected object state from the current one via, for instance, implicitly or externally initiated object state transitions following the definitions of the corresponding object life cycle.

While there exists a large variety of correctness notions targeting control flow aspects of process models, only few exist to check whether objects are correctly used. The existing notions as, for instance, the conformance notion from [8], require fully specified process models such that only finalized process models on the lowest level of abstraction, the implementation level, can be checked. The conformance notion from [8] evaluates whether all object state transitions occurring in the process model can be mapped to a object state transition of the corresponding object life cycle. However process model creation is an iterative process and process models may exist on different levels of abstractions – or are intentionally created that way to cater for the specific needs of a stakeholder. Being able to check also these process models allows the identification of errors in early process development stages and in underspecified process models as well as the post-check whether object abstraction preserved the property of conformance. We introduce the notion of weak conformance between a process model and the object life cycles of the objects utilized in the process to ensure this correctness criteria for fully as well as underspecified process models.

For evaluation purposes, example sets of process models need to have attached object information. Unfortunately, the freely available process collections mainly lack such process models. Therefore, we provide means to derive information about object utilization from activity and event labels such that the resulting process model gets annotated with this object information. The derivation bases on findings from [10] as the authors state that each activity label can be decomposed in up to three components: An action, a object an action is performed upon, and an fragment providing further details (e.g., locations, resources or regulations). However, completeness cannot be guaranteed by only deriving object information from labels as, for instance, possible implicit dependencies are not comprised in all cases. Nevertheless, the approach provides a process model more complete than the source process model and therefore, allows or increases usefulness of a process model for evaluation application.

The remainder of this report is as follows. Section 2 describes the execution semantics of process models with focusing on objects dependencies. Afterwards, Sections 3 to 5 summarize the current state of three main contributions the phd research project will deliver before we conclude this report. First, we sketch the approach of enriching process models with explicit objects from labels followed by the notion of weak conformance and proposals of how to apply the notion. Finally, we introduce one object abstraction technique as extension to a subclass of the existing control flow abstraction techniques in Section 5.

2 Execution Semantics

This section introduces the semantics of process models presented by Petri nets. Each process model aligning with the definitions from [11] and [13] can be transformed into a Petri net by a set of rules. Thereby, we apply commonly used and accepted rules for control flow transformation, e.g., activities are transformed to transitions, they are labeled with the corresponding activity names, e.g., *check order*, and they are connected along there execution paths by edges via places to ensure the bipartite property. Additionally, we add further *nop* transitions – transitions not performing any work – which we will use for object and control flow synchronization as explained blow. Object access is usually considered as passive part of a process model and therefore represented by places in Petri nets (compare for instance [1]) – if at all. However, the process of reading and writing objects to respectively from, for instance, a database is active work. Objects to be stored must be prepared to meet the requirements of the corresponding object store as the data schema might be different to the one utilized in the process model. Similarly, objects to be read must also be preprocessed. Additionally, the ac-

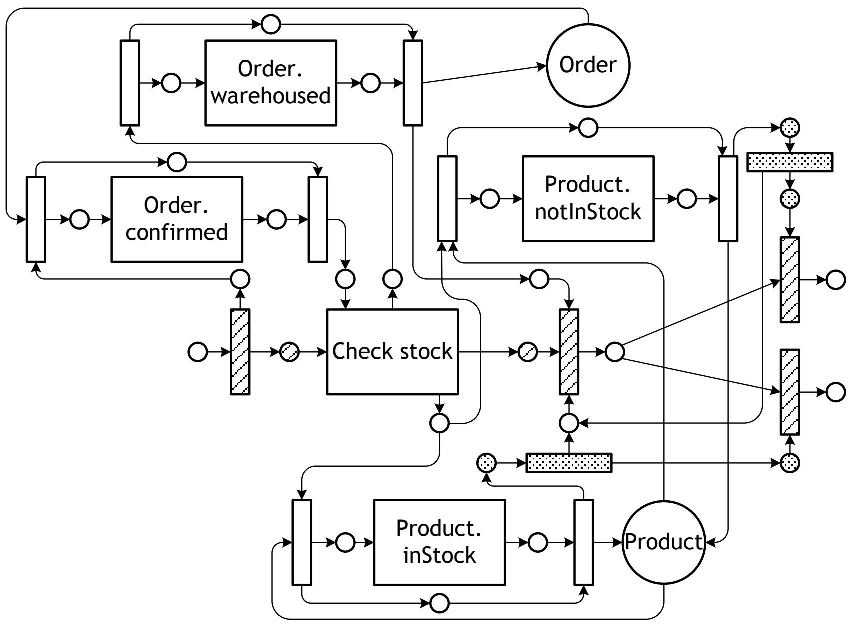


Figure 1: Semantics of process models presented by Petri net excerpt

tual object storage and retrieval comprise several atomic tasks like consistency checks and object access initiation including authentication. Persisting the object processed during process execution is necessary to allow, amongst others, fallback strategies in emergency cases. Therefore, we decided to represent objects as transitions as well in the resulting Petri net.

For space reasons, we omit the complete rule set but provide an overview about the transformation result. Assuming an order and delivery process, one of the activities undertaken during process execution might be *check stock*. The next execution step within the process is based on the outcome of the current activity one out of two alternatives. This activity requires the object *order* in object state *confirmed* as input requirements, i.e., as precondition to allow the activity to get executed. Executing this activity results in an object *Product* either in object state *inStock* or *notInStock* to indicate whether the ordered items are available or not. Additionally, the order object get transitioned into object state *warehoused*. Finally, it is also to be mentioned that not two activities access an object simultaneously as long as on of them is going to update the object for avoiding lost update issues (transaction property).

Figure 1 shows the Petri net excerpt describing the execution semantics. Thereby, the large places with labels inside are used as semaphores and their usage ensure the mentioned transaction property. The shaded transitions are the nop ones mentioned above for object and control flow synchronization. The dotted elements are used to define xor-split semantics deterministically based on object information; take the upper path, if the object state is *notInStock*, and take the lower path, if the object state is *inStock*. The remaining blank transitions and places describe the control and object flow of the process model. As described above, transition (or activity in the process model) *Check stock* can only fire (be executed), if the control flow reaches the places just before the transition and if object *Order* can be read in object state *confirmed*, i.e., earlier stages of the process must have provided the object in that object state, and if the object *Product* is currently not utilized by another transition, i.e., the corresponding semaphore place is marked. Completing transition *Check stock* results in two actions besides passing on the control flow. First, object *Order* is transitioned into object state *warehoused* and the object *Product* is transitioned either in object state *inStock* or object state *notInStock* depending on the execution of the corresponding business process activity. Second, the semaphore places for both data objects get marked again to signal data access completion. As the reader can see within this example, the nop transition before the activity in question is used to get all necessary objects in the required object states (read) while the nop transition after the activity is used to provide the updated objects in their new object states to the process context (write).

3 Extracting Objects and Object States

Usually, information about objects is hidden in activity labels such that process models covering control flow only can be enriched with these implicit information to provide insights about object usage. [10] describes that each activity label can be decomposed in up to three components: An action, an object an action is performed upon, and a fragment providing further details (e.g., locations, resources or regulations). For instance, the activity labels *order material from supplier* and *create order from template* encode the information, that objects *material* and *order* respectively are processed by the corresponding activity. The actions performed are *order* and *create* while the additional fragments provide information which additional resource is involved in the order action and that regulation stating a specific template is to be used has to be fulfilled. Following, we aim to enrich process models with information about the utilized objects in general and about the objct states of these objects specifically for each association between an object and and activity.

The algorithm sketched below can be generally applied to most process models from various notations as it focuses on core business process aspects as activities, gateways, and control flow edges as input for object information derivation. The only requirement is, that the notation of choice is capable of modeling objects, which basically all current notations allow [12], such that we can create the enriched process model within that notation. However, we also provide the opportunity to input process models in a notation not capable of modeling objects. But in these cases, the process model needs to be transformed in a canonical format – we use jbpt, a Java-based library containing techniques for managing and analyzing business processes. See `http://code.google.com/p/jbpt/` for details. The output will be a jbpt process model with explicit objects attached. Moreover, this algorithm can be easily adapted to be specifically tailored for a chosen notation. Thereby, additions are possible to benefit from all information a notation of choice can provide to increase the derivation result quality. We will provide (but omit here for space reasons) adaptations for the Business Process Model and Notation (BPMN) [15], Event-driven Process Chains (EPC) [7], and extended Event-driven Process Chains (eEPC), developed by ARIS and extending EPCs, where the latter two extend the algorithm to incorporate information provided by events preceding and succeeding the activities.

The basic algorithm comprises four steps to enrich the given process model. First, a preprocessing is applied to ensure that all labels match the so-called verb-object-style, where a verb is followed by a noun, which, in turn, may be followed by additional information as in the examples above. Using the algorithm proposed in [9], we receive the desired labeling style. Next, all activity labels of the process model get analyzed such that the noun is considered as object and the verb is considered as indicator for the state of that object after activity execution. The final two steps add the objects to the activities. The objects named in the label will get output objects to the corresponding activity. All objects being output of a directly preceding activity will get input objects to the corresponding activity. Therewith, the enriched process model is provided.

However, there are many opportunities to fine tune this approach. For instance, the resource assignments and object access rights can be taken into account such that only objects, which have been executed by a certain resource (role) get input object of an activity. Additionally, the access rights might also limit the number of objects being input to an activity as only the ones, which the resource executing the according activity is allowed to access, may get input objects.

4 Notion of Weak Conformance

Given a process scenario composed of a process model, a set of object life cycles utilized within this scenario, and a set of synchronization edges describing the dependencies between the object states of the different objects, we say that the process scenario satisfies weak conformance, if the process model satisfies weak conformance with respect to each of its objects. Weak object conformance is satisfied, if for each two directly succeeding states of an object in a process model there exists an execution sequence from the first to the second object state in the corresponding object life cycle and if the dependencies specified by synchronization edges with a target state matching the second object state of the two succeeding ones hold such that all dependency conjunctions and disjunctions are fulfilled. Two object states are succeeding in the process model, if either (i) they are accessed by the same activity with one being part of an input and one being part of an output object flow edge or (ii) there exists an execution sequence in the process model in which two different activities access the same object in two object states with no further access to this object in-between.

Computation of weak conformance can be performed using various techniques, as for instance the following ones. First, paths in the process model and the corresponding object life cycles may be analyzed and evaluated with respect to reachability of object states to decide about weak conformance. Second, the concept of soundness checking can be utilized. Therefore, the process model and the object life cycles are transformed into Petri nets and integrated into a single Petri net to be checked for soundness. Third, model checking can be used by checking computational tree logic formulas, derived from the object life cycles, against a Kripke structure derived from the process model. The first faces the issue of high complexity with respect to the synchronization edges. However, we will propose an algorithm, which allows to compute weak conformance via path analysis, if no synchronization edges are specified in a process scenario. The Petri net approach deals with very large and complex Petri nets, but thanks to existing services, e.g., LoLA (see `http://service-technology.org/tools/lola` for details), they can be handled automatically and no human needs to interact with these very complex Petri nets. However, there exist cases, when decidability relies on relaxed soundness but determination of accepting traces is complex. Model checking is also an expensive task but provides a result without compromises. The number of CTL formulas is exponential to the number of object states in the object life cycle as for each possible implicit, external and in-process object state transition, one formula must be derived and put into the correct combination with the others. This can be done completely automatically based on a set of rules. For three object states in sequence, formulas must be derived to cover the facts that we stay in the first forever, move to th second, move directly to the third, stay forever in the second, move from the second to the third, and stay in the third forever. Additionally, rules for the synchronization edges must be derived and incorporated in the model checking procedure. The increase of complexity is linear to the number of rules specified.

5 Object Abstraction

There exist two alternatives with respect to the relation between control flow and objects: Object abstraction follows control flow abstraction and object abstraction as standalone approach independently from control flow. In this report, we focus on the first option, i.e., we extend existing process model abstraction techniques with the capability to abstract objects as well. The only requirement, a technique must fulfill, is that it utilizes fragment-based control flow abstraction. Thereby, the complete process model gets fully partitioned into a set of process fragments with each only allowed to share entry or exit nodes with an other fragment. During the abstraction process, a selected fragment is abstracted to a single activity. We basically do not restrict the type of process model, which can be abstracted using the object abstraction extension, but as we require correct process models (see above), a process model to be abstracted must be structural sound, i.e., it has exactly one source and final node and each activity is on a path from the source to the final node. Additionally, also the notion of weak conformance must hold such that also the object part of the process model correct. Finally, a data model must be explicitly specified, which determines the relations and dependencies between the single objects utilized in the business process. The data model is a UML class diagram restricted generalization, aggregation, composition, and association relations; for further details, see [13].

The object abstraction is realized with a rule set consisting of six rules from three areas, which are applied iteratively. One deals with process quality and ensures that the granularity level between the activities and the objects is preserved in each abstraction step as well as that the all objects utilized on one abstraction level are on the same abstraction level, i.e., there exist no parent-child relations between any two utilized objects. Second, object combination aims to combine several objects into a single one. The third field, object preservation, deals with the decision whether an object is kept or removed within an process model abstraction iteration. A summary of all rule is provided in Table 1. More details about object abstraction including the complete formal specifications of the mentioned rule set can be found in [13].

1 Process Quality	
Abstraction Consistency	All representations of an object must be equally abstracted, if at least one of the representations gets abstracted. Therefore, this rule ensures abstraction consistency by enforcing identical object abstraction for all representations of one object.
Preserving Granularity Alignment	Assuming an alignment between object and control flow representations of a process model exists before abstraction, it must exist afterwards as well. The alignment is achieved by avoiding abstraction of objects being associated to activities, which have not been selected for control flow abstraction (except above rule requires it).
2 Object Combination	
Object Combination	This rule reduces the number of objects in the process model by combining them, if several objects being part of one fragment, which get abstracted, do have common ancestor nodes in the data model. These objects will be combined to the common ancestor node first found while traversing through the data model from the utilized objects.
3 Object Preservation	
Lower Bound of Objects	After data abstraction, an activity might not be associated to objects any more. But, objects are important in the context of process management. Therefore, this rule introduces a configurable number stating the minimum associations to exist after object abstraction.
Reoccurring Object	An object utilized in the fragment, which gets abstracted, and associated to at least one more activity, not part of this fragment, will be preserved as it is considered important for a larger part of the process (it is not used locally only in that fragment). Otherwise it is marked for removal.
Full Object Activity Association	Objects only utilized in a process fragment and nowhere else can still be of importance for the complete process model, if they are input or output to this process fragment. Therefore, all objects being input or output process fragment, which get abstracted, are preserved. All others get removed.

Table 1: Object abstraction rules overview

6 Conclusion

This report outlines the phd research project about the role of objects in process model abstraction and introduces the main contributions briefly. First, there will be a mapping of process models to Petri nets to explicitly and formally describe the execution semantics. This mapping also ensures the transaction property for object access allow true parallelism between various activities. Second, process models can be enriched with explicit information about object utilization. The proposed algorithm works for general process models and can easily be adapted to align with most existing process model notations as shown for BPMN and EPC. Further, the correctness ob object utilization during process execution can be checked via the notion of weak conformance, which also supports underspecified process models. Fourth, this research allows to extend existing process model abstraction techniques with means to abstract objects as well. These objects will be abstracted along with the control flow and remain aligned with the control flow for each abstraction step.

Publications 2012

Workshop paper: *Conformance of Process Models with respect to Data Objects* [11] and *A Platform for Research on Process Model Collections* [4]
Conference paper: *Data Support in Process Model Abstraction* [13]

Teaching Summer Term 2012

Master project on *Analysis of Complex Process Model Repositories*
Bachelor project on *Billing Process as a Service*
Master seminar on *Process Repositories* topic supervision

References

[1] A. Awad. *A Compliance Management Framework for Business Process Models*. PhD thesis, Hasso Plattner Institute, 2011.

[2] R. Bobrik, M. Reichert, and T. Bauer. View-Based Process Visualization. In *Business Process Management*, pages 88–95. Springer, 2007.

[3] G. Booch. *Object-Oriented Analysis and Design with Applications (3rd Edition)*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004.

[4] R.H. Eid-Sabbagh, M. Kunze, A. Meyer, and M. Weske. A Platform for Research on Process Model Collections. In *BPMN*, pages 8–22. Springer, 2012.

[5] D.J. Elzinga, T. Horak, C.Y. Lee, and C. Bruner. Business process management: survey and methodology. *IEEE Transactions on Engineering Management*, 42(2):119–128, 1995.

[6] R. Eshuis and P. Grefen. Constructing Customized Process Views. *Data & Knowledge Engineering*, 64(2):419–438, 2008.

[7] G. Keller, M. Nüttgens, and A. Scheer. Semantische Prozessmodellierung auf der Grundlage “Ereignisgesteuerter Prozessketten (EPK)”. Technical Report Heft 89, Institut für Wirtschaftsinformatik, University of Saarland, 1992.

[8] J. Küster, K. Ryndina, and H. Gall. Generation of Business Process Models for Object Life Cycle Compliance. In *BPM*, pages 165–181. Springer, 2007.

[9] H. Leopold, S. Smirnov, and J. Mendling. On the Refactoring of Activity Labels in Business Process Models. *Information Systems*, 37(5):443–459, 2012.

[10] J. Mendling, H.A. Reijers, and J. Recker. Activity Labeling in Process Modeling: Empirical Insights and Recommendations. *Inf. Systems*, 35(4):467–482, 2010.

[11] A. Meyer, A. Polyvyanyy, and M. Weske. Weak Conformance of Process Models with respect to Data Objects. In *ZEUS*, pages 74–80, 2012.

[12] A. Meyer, S. Smirnov, and M. Weske. Data in Business Processes. *EMISA Forum*, 31(3):5–31, 2011.

[13] A Meyer and M Weske. Data Support in Process Model Abstraction. In *Conceptual Modeling*, pages 292–306. Springer, 2012.

[14] T. Murata. Petri nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[15] OMG. Business Process Model and Notation (BPMN), Version 2.0, January 2011. `http://www.omg.org/spec/BPMN/2.0/` accessed August 5, 2013.

[16] A. Polyvyanyy, S. Smirnov, and M. Weske. The Triconnected Abstraction of Process Models. In *Business Process Management*, pages 229–244. Springer, 2009.

[17] A. Polyvyanyy, S. Smirnov, and M. Weske. Business Process Model Abstraction. In *Handbook on BPM 1*, pages 149–166. Springer, 2010.

[18] S. Smirnov. *Business Process Model Abstraction*. PhD thesis, Hasso Plattner Institute, 2012.

[19] A. Streit, B. Pham, and R. Brown. Visualization support for managing large business process specifications. In *BPM*, pages 205–219. Springer, 2005.

[20] M. Weske. *Business Process Management: Concepts, Languages, Architectures. Second Edition*. Springer, 2012.

Comprehensible 3D Maps and Building Panoramas

Sebastian Pasewaldt
Computer Graphics Systems Group
Hasso-Plattner-Institut
sebastian.pasewaldt@hpi.uni-potsdam.de

3D geovirtual environments (3D GeoVEs), such as 3D virtual city and landscape models, have become established tools to communicate geoinformation to experts and non-experts of different application domains. In contrast to 2D geovisualization tools (e.g., topographic and thematic maps) 3D GeoVEs primarily depict geoinformation utilizing a perspective view. A perspective view on geodata yields a number of advantages (e.g., immersion and improved spatial cognition due to depth cues) but also introduces numerous disadvantages, such as occlusion and perspective distortion. The advantages as well as the disadvantages of 3D geovisualization have been considered by cartographers and landscape artist and resulted in a different form of communication geoinformation: 3D panoramic maps.

Panoramic maps seamlessly combine multiple perspectives views in one image, providing additional information by revealing occluded geoobjects, or emphasizing relevant information (e.g., landmarks). The additional information supports a user at navigation and orientation tasks. This work presents two novel concepts and prototypical implementations that utilize MPVs for an effective visualization of 3D GeoVEs: Comprehensible 3D maps and building panoramas. Further, MPVs are classified with respect to their cartographic scale and conceptual integration. Finally, potentials and challenges of multi-perspective 3D GeoVEs are discussed.

1 Introduction

Geovisualization is a broad visualization discipline that utilizes images or image sequences to effectively communicate geodata (spatial data with a reference to the earth) to a user. Although 2D digital maps are still an established tool to communicate geodata, 3D GeoVEs have advanced in application domains like geoanalysis, disaster management and navigation systems. In contrast to 2D geovisualization tools, which encode the third dimension in an additional (visual) attribute, 3D GeoVEs offer a more natural access to geodata. This is further supported by utilizing a perspective view, which facilitates the human's 3D visual perception. A perspective view yields a number of advantages for the communication of geodata, such as immersion into- and an intuitive communication of 3D geodata, but also introduces disadvantages, such as occlusion, perspective distortion and multiple cartographic scales.

To cope with the disadvantages while still benefiting from the advantages, cartographers and landscape artists developed 3D panoramic maps. Panoramic maps seamlessly combine multiple perspectives views in one image, providing additional information by revealing occluded geoobjects, reducing the number of cartographic scales, or emphasizing relevant information. The concept of multi-perspective views (MPVs) has been revisited and adapted to 3D GeoVEs for focus+context visualization [2] of geodata with a large cartographic scale, such as 3D virtual city and landscape models, or virtual globes. For example Lorenz et al. (2008) and Möser et al. (2008) presented bended maps, which generates panoramic map-like visualization. The foreground (focus region) depicts a detailed 3D virtual city model, whereas the bended background (context region) contains an abstracted visual representation. The seamless combination of focus and context regions within one viewport is well suited for navigation tasks, because the detailed information eases navigation in the local neighborhood, while the context supports locating one's position in relation to landmarks.

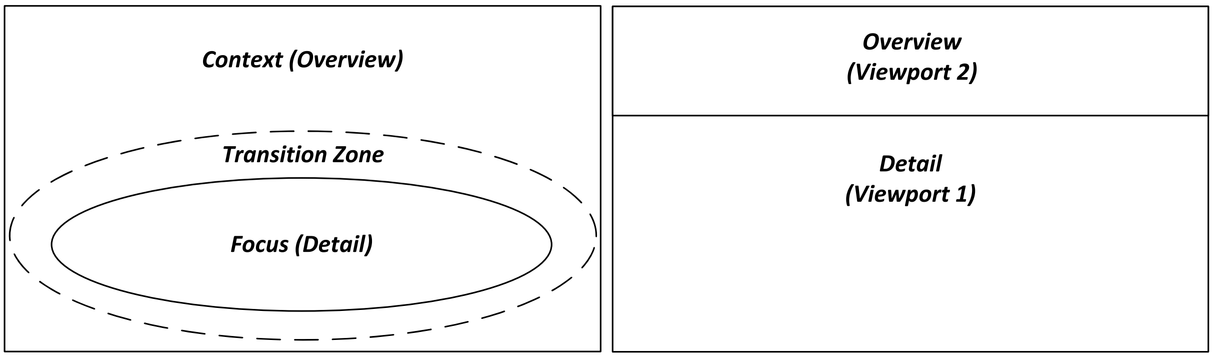


Figure 1: A focus+context visualization (left) seamlessly combines detail and context information within one viewport. In contrast, a overview+detail visualization physically separates detail and overview information, utilizing two viewports.

Another promising approach of utilizing MPVs in 3D GeoVEs are building panoramas. Building panoramas combine multiple perspective views of a building or a building block in one image (Section 3). Building panoramas visualize geodata of a small cartographic scale and utilize the concept of overview+detail visualization [24]. In contrast to focus+context, overview+detail presents the detail and the overview information in separated, but linked viewports (Figure 1). It enables the simultaneous exploration of building details in the context of the whole building, which would be occluded in a perspective 3D view.

Based on two prototypical applications, benefits and drawbacks of multi-perspective visualization for 3D GeoVEs are described. Section 2 briefly discusses drawbacks of current digital 3D maps and summarizes design guidelines that improve the comprehension of 3D maps. Further, it discusses to what extend MPVs are suited to implement these guidelines. The concept of building panoramas as well as applications examples are presented in Section 3. Section 4 summarizes the potentials of MPVs for 3DGeoVEs and gives a brief outlook.

2 Comprehensible 3D Maps

For centuries, maps have been essential resources for humankind to aid orientation, navigation, exploration, and analysis tasks. With the digital revolution in the last quarter of the 20th century, 2D digital maps started to foster an interactive communication of geoinformation, allowing, for the first time, to customize map contents to tasks and contexts based on a service-oriented architecture (SOA) [3]. Recent advancements in the acquisition, provision, and visualization of 3D geodata, such as virtual 3D city and landscape models, yield new possibilities for 3D mapping services. Today these services represent key components to a growing number of applications, like navigation, education, or disaster management. In particular mapping services based on 3D geovirtual environments (3D GeoVEs), such as Google Earth™ or the OGC's Web View Service (WVS) [7], provide users various information overlays to customize visualization to tasks and contexts, such as as thematic or topographic maps. The customizations often focus on embedding 2D overlays for points-(or areas-)-of-interest, for instance to highlight public transport networks, traffic information and landmarks, or to modify the map presentation (e.g., photorealistic or non-photorealistic). A photorealistic presentation tries to visualize geoobjects as realistic as possible, which eases the mental mapping between the visualization and the reality. By contrast, a non-photorealistic presentation abstracts from reality and provides the prerequisites to simplify and filter detailed elements as well as to clearly encode displayed information of complex geodata [4]. A specialized class of 3D GeoVEs is the digital 3D map (D3DM), which is based on a generalized data model and utilizes a symbolized (abstracted) visualization of 3D geoobjects.

In contrast to traditional 2D maps, D3DMs utilize a central perspective view, which is similar to the humans 3D visual perception, offering a natural access to geoinformation [10] and immersion [16] into the geodata, and thus increasing a user's effectiveness on spatial tasks [26]. During the last decade cartographers discussed and established guidelines to further increase the effectiveness and expressiveness of D3DMs (for example [6,9,20]). However, current products, systems, and applications providing D3DMs are still faced by a number of drawbacks that impact the comprehension of 3D map contents:

(D1) Occlusion. Due to overlapping of 3D geometric representations, occluded map content cannot be perceived in a perspective view.

(D2) Visual clutter. Because of perspective distortion, the size of distant map objects decreases. As a result, they are no longer recognized as single objects and are perceived as "visual noise" (e.g., because they occupy only one pixel).

(D3) Insufficient use of screen-space. In perspective views at pedestrian level, a large amount of the screen-space is occupied by the horizon or visual clutter. In this area no information can be communicated.

(D4) Unlimited number of cartographic scales. Due to perspective distortion, a 3D digital map includes an unlimited number of map scales. This complicates the estimation and comparison of spatial relationships.

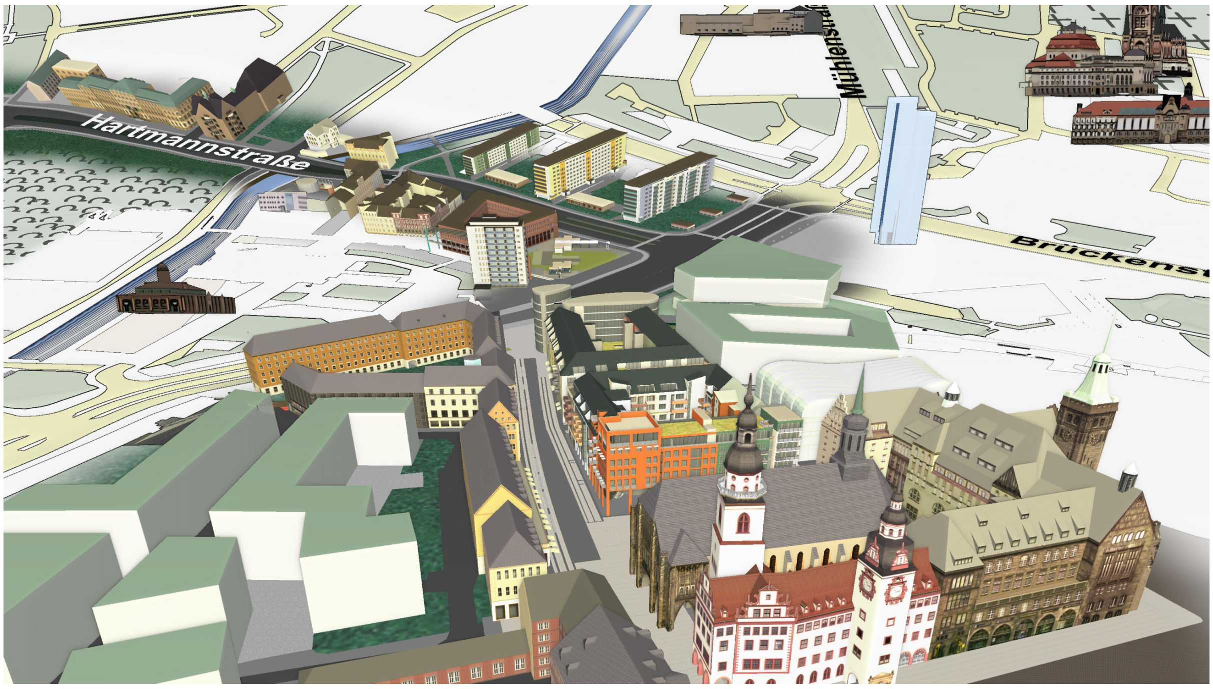


Figure 2: Digital 3D map of the virtual city model of Chemnitz showing a route: A view-dependent, multi-perspective view is utilized to increase the screen-space utilization by bending the background upwards. A cartograhpy-oriented stylization is applied off the route to lower visual complexity.

Previous work showed how view-dependent MPVs [18] and cartography-oriented visualization (COV) [25] can be used to partially overcome the drawbacks *D1–D4*. MPVs, on the one hand, enable the seamless combination of different 3D perspectives in a single view and thus feature less occlusion, a reduced number of cartographic scales, and an increased utilization of screen space. COV, on the other hand, facilitates guidance of a viewer's gaze to important or prioritized information, thus providing saliency-guided visualization. Both techniques adapt visualization to different contexts and contents with respect to user interaction or dynamically changing thematic information, but have not been applied concurrently in a single system.

In the following a prototype that combines MPVs with COV for comprehensible D3DMs (Figure 2) is discussed. For this purpose, cartographic design principles are identified that aim to increase the expressiveness and effectiveness of D3DMs. According to this classification, a prototypical implementation demonstrates the benefits of multi-perspective and non-photorealistic rendering techniques for the comprehension of D3DMs. In particular, the prototype enables (1) a seamless combination of cartography-oriented and photorealistic graphic styles while (2) increasing screen-space utilization and (3) simultaneously directing a viewer's gaze to important or prioritized information.

2.1 Design Principles

The International Cartographic Association (ICA) defines a map as "a symbolized image of geographical reality, representing selected features or characteristics, resulting from the creative effort of its author's execution of choices, and is designed for use when spatial relationships are of primary relevance" [8]. According to this definition, a perspective view in a 3DGeoVE is not a D3DM per se. Häberling et al. suggest that the term 3D map is applicable to 3DGeoVEs if the virtual environment is based on a generalized data model and utilizes a symbolized visualization of classified map objects [6] (e.g., by using the class model of CityGML [13]).

The aim of a map is to successfully communicate geoinformation between a cartographer (the map producer) and a user (the map consumer). Geoinformation are encoded using a semiotic model and transferred by the map. For a successful communication, the map consumer must understand the semiotic model to decode the information. Further, a map should satisfy the needs of the map consumer to improve the communication process: the map should be readable, comprehensible, and visualized in a way that the information can be memorized easily, and that not only rational but also emotional aspects [12] are addressed. According to Broderson the geocommunication process is successful if the map producer and the map consumer "agree" on aspects of location or space [1]. In order to achieve this agreement, cartographers developed different high-level design guidelines for D3DMs that are summarized in the following:

(A1) Decrease of visual complexity by classification, symbolization and abstraction.
Häberling et al. define the following three design steps [6] for D3DMs: (1) modeling, (2) symbolization and (3) visualization. For each design step the map producer can choose between different design aspects to configure the map to fit a user's needs and ease the communication process (Table 1). Modeling includes aspects of filtering raw geodata and mapping these to a 3D geometric representation of map objects suitable for rendering. The visual appearance of the 3D-geoobject (e.g., color, texture and degree-of-abstraction) are configured during symbolization. The visualization step defines the mapping of a 3D geometric representation to the presentation medium and is controlled by parameterizing the virtual camera (e.g., the field-of-view, and projection), as well as using scene specific parameters (e.g., lighting, shading, and atmospheric rendering effects). Based on this classification, Häberling et al. performed a user study to identify atomic design guidelines that assist the map producer to reduce visual complexity and improve comprehension.

(A2) Decrease of occlusion and visual clutter.
Although Häberling et al. proposed different design variables for parametrizing the projection of the virtual camera (e.g., orthographic and cylindric projection), the perspective projection is the most applied projection for digital D3DMs, mainly because it facilitates the human visual system and thus produces a familiar visualization. According to Jobst and Döllner (2008) the perspective view comprises a number of drawbacks, such as occlusion of map-objects and visual clutter due to perspective distortion in the distant parts of the D3DM, which reduces the effectiveness of geocommunication.

Table 1: Excerpt of design steps, aspects and variables proposed by [6].

Design Steps	Design Aspects	Design Variables
Modeling	Models of map objects	Model geometry Semantic attributes Position
Symbolization	Graphic appearance	Shape Size Color
	Textures	Pattern Pattern repetition rate Pattern orientation
	Animations	Size alteration Size alteration Texture alteration
Visualization	Perspective	Parallel projection Perspective projection
	Camera settings	Viewing inclination

To reduce occlusion, Häberling et al. suggest a viewing inclination of 45° [6] and generalization to minimize visual clutter. An alternative approach is used in panoramic maps. The landscape artists combine multiple perspectives in one image and distort (e.g., enlarge) map-objects [19].

(A3) Increase of user involvement.
The design process of maps can be described as a feedback loop between the map producer and the map consumer [21] where the map producer designs the map according to the consumer's feedback. Reichenbacher demands "the ability of flexible systems to be changed by a user or the system in order to meet specific requirements" [22]. An optimal map should "present as much information as needed (by a user) and as little as required" [23]. Service-based D3DMs fulfill this requirement, because an user's feedback is directly transformed into a new version of a map. This direct feedback-loop changes the strict separation between the role of the map producer and map consumer [11]. The consumer itself becomes the map producer. Instead of exposing all map design parameters, which possibly overwhelm a user, the D3DM should interactively react on a user's context. For example the map provides an overview while a user follows a navigation route. When a user is faced to make decisions, e.g, which road to travel at a cross road, the map focuses on the cross road, assisting with detail information.

(A4) Increase of screen-space utilization.
The map is presented using an information carrier, such as paper or digital displays. The size and resolution of the information carrier for D3DMs can vary between 3.5" with 330ppi (pixel per inch) for mobile devices up to 60" with 37ppi for monitors. The size and resolution in combination with the capabilities of the human's visual system defines the minimum size boundary of a map element. If the map element falls below this boundary, it becomes indistinguishable from its surroundings and, as a consequence,

the corresponding pixels cannot be efficiently used for communicating geoinformation. In order to prevent these "dead values" [10] a D3DM must be device aware.

To summarize, D3DMs need to be designed in a way that reflects the context and task of a map consumer to highlight prioritized or important information, i.e., using device-aware visual abstraction and symbolization that features less occlusion and utilizes screen-space efficiently. One possibility to achieve this, is to combine cartography-oriented visualization and multi-perspective views in a system approach. The following paragraph discusses how MPVs are utilized to implement design aspects A2-4. Since a detailed discussion of COV is not in the scope of this paper, the author refers to Semmo et al. (2012) for more details.

2.2 Multi-Perspective Views

Multi-perspective views are applied in panorama and landscape visualization, such as the panoramic maps of H.C. Berann [19]. Berann utilizes a progressive perspective, where a steep viewing angle in the foreground is progressivly interpolated to a flat viewing angle in the background. The foreground depicts an orthographic view on the environment showing the current position of a user whereas the perspective view of the horizon in the background assists a user to determine the viewing direction. By contrast, the degressive perspective applies a flat viewing angle in the foreground and a steep angle in the background. This leads to the impression that the virtual environment is bended towards the user, providing detailed information in the foreground combined with context information in the background (Figure 3).

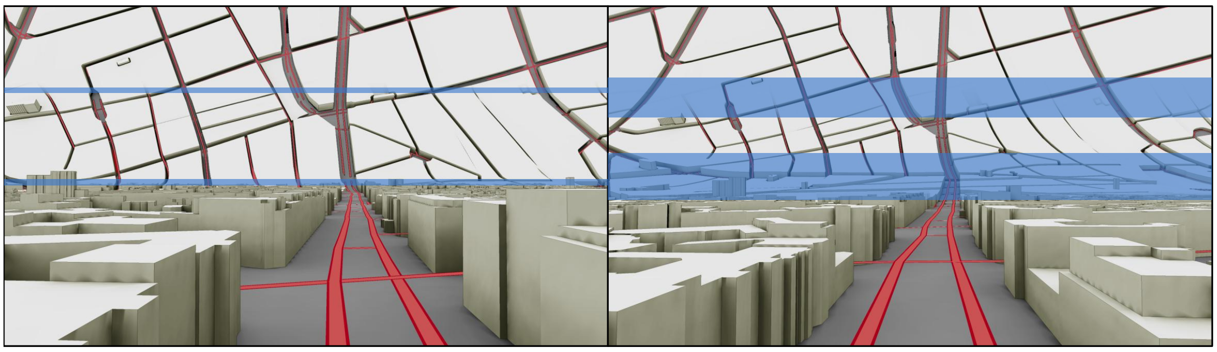


Figure 3: Comparison of two degressive MPV configurations that feature three viewport zones connected by transition zones (marked blue). Due to the minimized transition zones of the left configuration, the number of map scales are reduced. This eases object comparison in each zone [10].

Different real-time capable visualization techniques exist that are able to generate progressive, degressive as well as hybrid perspectives ([14, 17, 18]). The following concept and considerations are based on the view-dependent multi-perspective views of Pasewaldt et al. (2011). One key aspect of this approach is that one configuration of a MPV, a so called *preset* (P), is associated with an distinct viewing angle ϕ. The map producer can define multiple presets with different viewing angles. During map usage these presets are interpolated based on the current viewing angle. Thus, it is possible to utilize a degressive perspective for a flat viewing angle of the virtual camera and a progressive perspective for a steep viewing angle without having additional configurations during map utilization.

A preset is defined as $P = (C(t), \phi, s, e)$ with $C(t)$ being a parametric curve that is defined by a number of control points B_i. $C(t)$ is used to control the shape of the MPV (e.g., progressive or degressive shape). The scalars s and e define the start and end of the MPV with reference to a viewer's position.

Although the flexibility of the parametric curve can produce an arbitrary number of MPV configurations, especially the degressive and progressive perspective has been in the focus of cartographers. For example, Jobst and Döllner discussed to what extend MPV can increase the perception of 3D spatial relations by reducing occlusion and increasing screen-space utilization [10]. In their work they suggested a configuration that subdivides the view into distinct *viewport zones*, i.e., zones that are viewed with a distinct viewing angle (Figure 3). Since the number of cartographic scales are reduced in each zone, the comparison of map objects in each zone is eased. To further reduce the number of cartographic scales and enlarge the area of each viewport zone, the transition between two viewport zones is minimized.

Bending the D3DM towards a user replaces the horizon of a 3D perspective view by parts of the 3DGeoVE. This increases the screen space utilization (A4). The combination of different viewport zones can reduce occlusion (A2) and the number of cartographic scales. Further, perspective distortion, thus visual clutter is reduced (Figure 4). The presented concept enables the map producer to specify different MPV configuration that are interpolated based on a scalar ϕ. In the example, the viewing angle of the virtual camera is mapped on ϕ, but it is also reasonable to encode and map a user's context information (A3). MPVs are only one tool in the tool shelf of map producers for the production of comprehensible D3DMs. It can emphasize and complement the impact of existing visualization techniques, such as COV [25], for the expressive and effective communication of geoinformation.

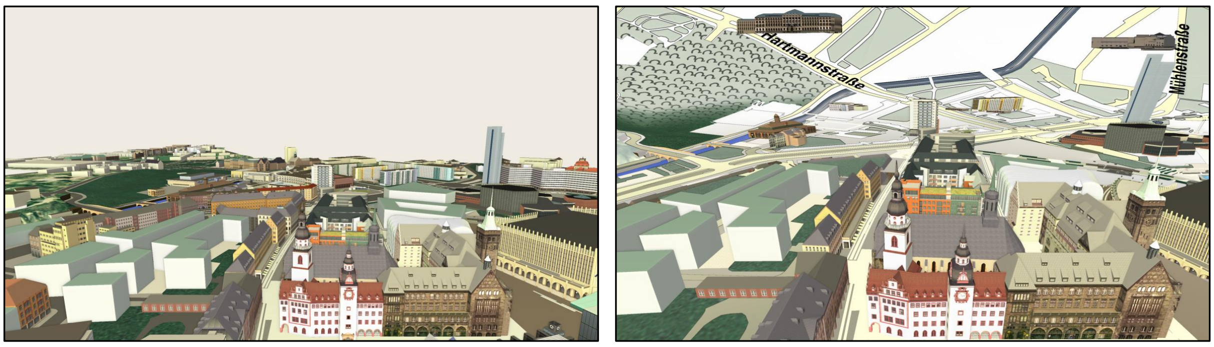


Figure 4: Exemplary perspective view of a virtual 3D city model (left) in comparison to a multi-perspective view (right) that is combined with cartography-oriented visualization (i.e., colorization, edge enhancement, labeling, waterlining, symbolization, landmark abstraction, and transformation)

3 Building Panoramas

Buildings models are dominant elements of a 3D virtual city models. They can be utilized as landmarks during navigation tasks and are in the focus of geo-analysis task such as solar-potential analysis. Due to the three-dimensional structure of a 3D virtual building model (3DBM) and the perspective view, a building model visualization has similar drawbacks as D3DMs (Section 2 D1-D4). For example, it is impossible to simultaneously explore or compare all building façades of one building because of occlusion (D1) and perspective distortion (D4). Although a photorealistic perspective 3D visualization depicts the 3DBM as a human would perceive it in reality, certain use-cases (e.g., architectural modeling or exploration of analysis results), demands alternative visualization with different requirements:

(R1) Overview of building façades. The texture and geometry of building façades define the characteristic and unique look of a building and occupy the majority of a building's surface. An overview of all building façades in one image enables their simultaneous exploration and eases analysis of building façades.

(R2) Preserved topology. The topology of the building façades in the overview must be preserved in order to ease the mental mapping to the 3D representation of the building model.

(R3) Minimized Distortion. A perspective view introduces distortion that complicates the estimation of spatial relations. For tasks such as measurement or building model refinement, the visualization of each façade should contain as less distortion as possible.

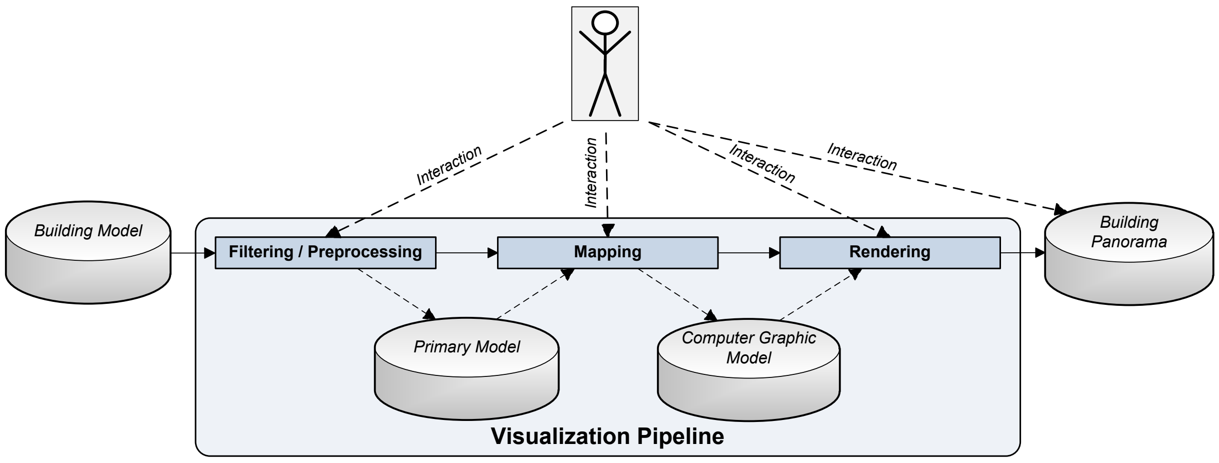


Figure 5: The conventional visualization pipeline is extended for building panorama image synthesis: Based on the geometric representation of the building model, a camera model description is generated in the preprocessing stage. The description is utilized in the rendering stage to generate the building panorama.

A multi-perspective view of the building model is a visualization that fulfills these requirement and further cope with occlusion. It aligns an orthographic view of each façades side-by-side and generates a 2D panoramic image of the building model. Previous work has shown how single orthographic views of building façades are utilized to communicate architectural design decisions or for an image-based refinement of building geometry [27]. Beside several advantages (Section 3.2), one drawback of the 2D panoramic image is that depth perception is decreased and thus the 3D structure of the building model can hardly be communicated. Thus, the panoramic image cannot be used as a substitute for the detailed perspective 3D visualization, but as an complementary overview in a overview+detail visualization [24].

3.1 Image Synthesis

The image synthesis of the building panorama is based on the visualization pipeline (Figure 5) and is subdivided into a filtering/preprocessing, mapping, and rendering stage. The user can modify every stage in order to customize the resulting building panorama. In the following building-panorama-specific extensions to the visualization pipeline are described in detail.

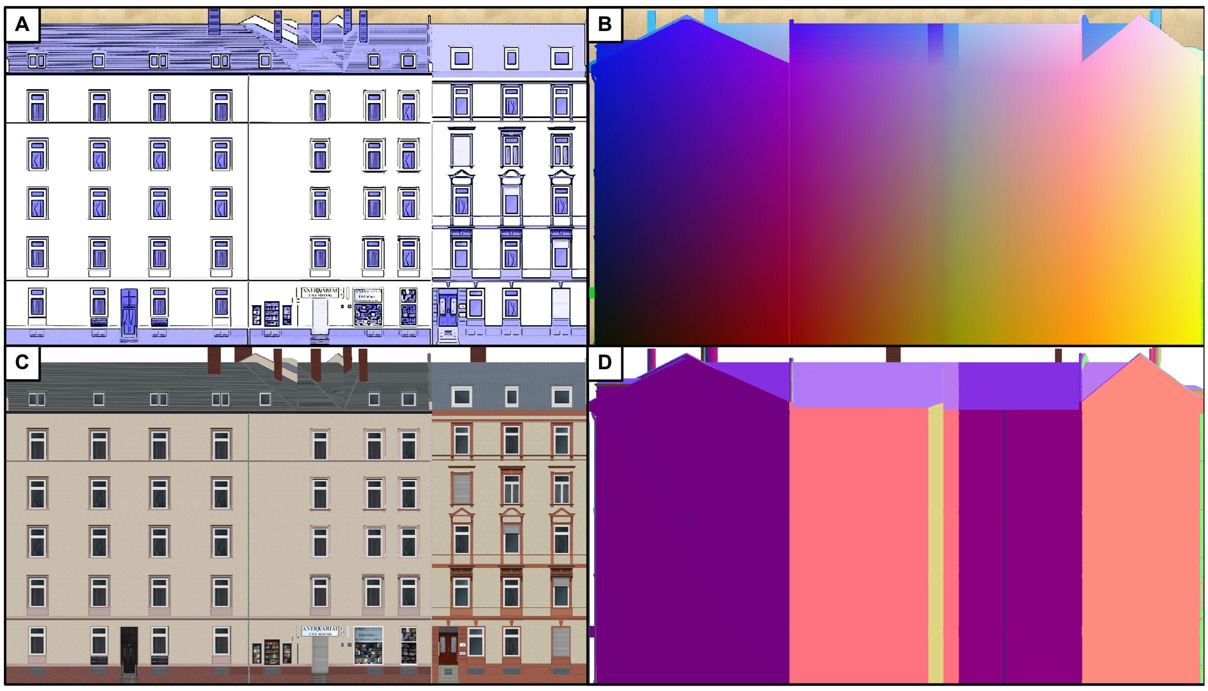


Figure 6: Different visual representations of a building panorama: A - An abstracted visualization. Compared to the conventional visualization (C), fine structures are emphasized using edge enhancement and semantic-based colorization. The vertex (B) and normal image (D) are utilized to link the panorama view with the 3D perspective view.

Preprocessing In the preprocessing stage the raw data (e.g., a City-GML [13]) description of the building model, is filtered and converted to a topology-preserving data structure (e.g., Half-Edge data structure) that is the input for a panorama-specific geometry analysis algorithm and the mapping stage. The algorithm determines a ordered list of building façades $\mathbf{F} = \{F_1, ..., F_n\}; n \in \mathbb{N}$. For each $F_i \in \mathbf{F}$, a camera description C_i is generated that exactly matches the camera view-frustum with the width of F_i. The height of the frustum equals the highest façade. The set of camera descriptions $\mathbf{C}$ is utilized during rendering to generate the building panorama. The user can configure the preprocessing stage to generate a panorama image containing a subset of $\mathbf{F}$.

Mapping In the mapping stage, the filtered and preprocessed building model is mapped to geometric primitives. Further, the visual appearance of the building model (e.g., color or texture) can be configured. If semantic information is available, it can be utilized for a semantic-based rendering such as façade abstraction [15], which generates a non-photorealistic visualization. The output of the mapping stage is a computer graphic representation of the 3DBM (computer graphic model).

Rendering In the rendering stage the computer graphic model is utilized for the image synthesis of the 3D perspective view as well as for the synthesis of the 2D panorama image. For each C_i, a rendering pass is performed, which generates an image of the F_i as part of the final panorama image. In addition to the color image, a vertex and a normal image are generated, which are utilized to link the 2D and 3D view (Figure 6). Since $\mathbf{F}$ is a list of order-preserving façades, the final panorama image fulfills *R2*. Instead of a perspective, a parallel projection is utilized, which eliminates perspective distortion (*R3*).

3.2 Application Examples

The 3D perspective view and 2D panorama image are utilized in a overview+detail visualization, i.e., the two images are organized in two separate viewports [24] (Figure 7). Between these viewports a bidirectional link is established that updates or synchronizes the viewports. Based on the application example, either the 3D perspective view or the 2D panorama image serves as overview. In the following, advantages as well as disadvantages are discussed using three applications examples.

Geometry Refinement Geometry refinement denotes operations that increase the complexity of a geometric representations. Xiao et al. (20008) utilize an orthographic image of a textured block building model to fully-automatic add openings (e.g., windows and doors), or exterior installation (e.g., stairs or rain pipes). The image is decomposed into distinct subsections ((e.g., one section per opening) using computer vision algorithms. If depth information is available (e.g., based on a 3D point cloud) these subsections are automatically extruded. Wanner et al. (2012) evaluate interaction metaphors that can assist the user in altering the sub section and thus configure the refinement process. The overview+detail visualization is well suited for an user-controlled geometry refinement, since refinement operations can be performed on the panoramic image (the detail view). Direct feedback on user actions is provided in the 3D perspective

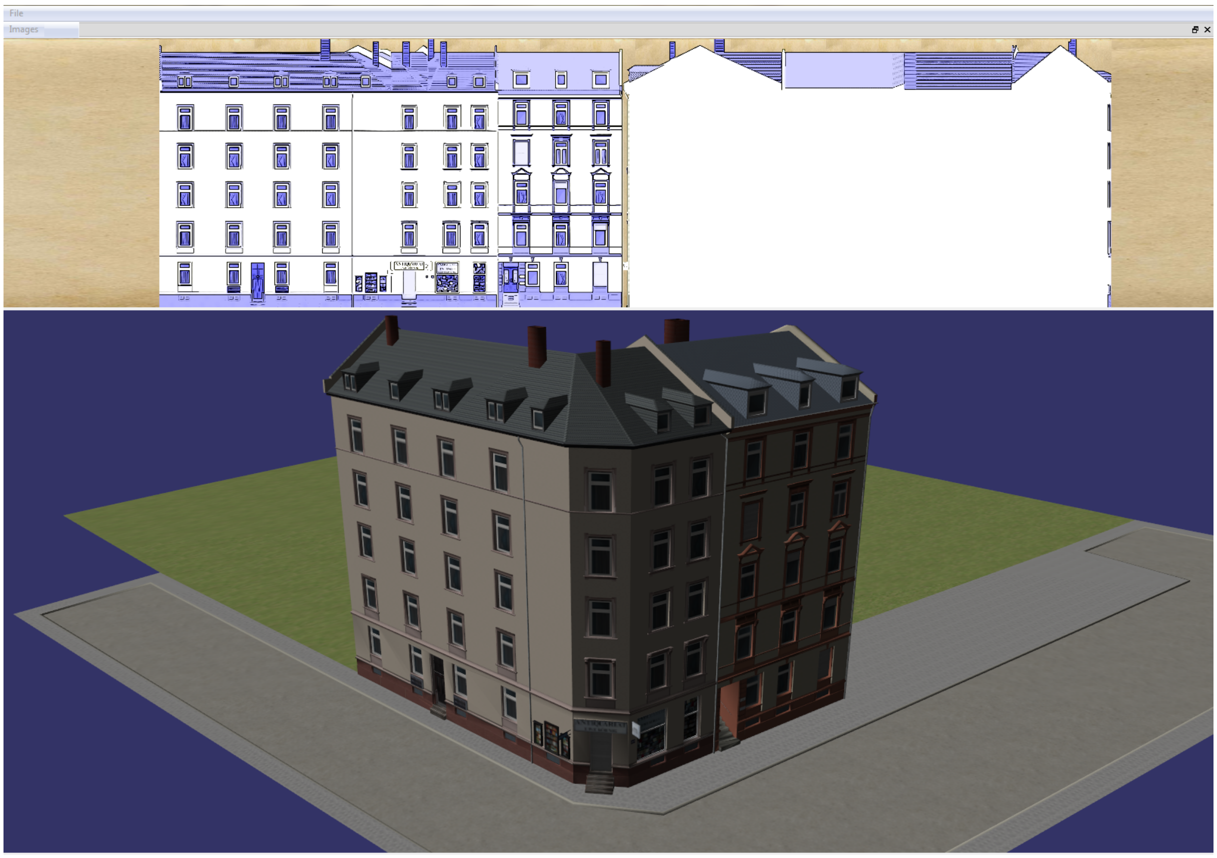


Figure 7: Screenshot of the prototype. The prototype consists of two separated viewports: The 2D building panorama (top) and the 3D perspective view (bottom). Based on the use-case, the 2D building panorama serves as detail or overview visualization for the 3D perspective view.

view (the overview). Further, the automatic panorama generation combined with semi-automatic geometry refinement techniques seems to be a promising approach for the refinement of 3D virtual city models.

Evaluation of Geoanalysis Results Geoanalysis denotes analysis operations that are performed on geodata, such as 3D virtual city models [5]. Popular examples are solar-potential or noise-pollution analysis. The results of the analysis are presented directly using the 3D virtual city model. To explore the results, the user has to control the virtual camera towards the object-of-interest using a 2D input device, which can be a cumbersome task. Building panorama can be a promising approach to reduce the exploration overhead, since interaction must only be performed in 2D. Due to *R1*, the building panorama enables the inspection of a single building façade in the context of the complete building, which is impossible in a 3D perspective view. In this application example the building panorama can act as detail, as well as overview visualization.

Comparison of Architectural Designs In the design process of a building, different variants are developed and discussed with multiple participants. These discussions are mainly based on architectural sketches, or the 3DBM. On the one hand, architectural sketches are well suited for a discussion, because they depict many relevant details and can be printed on paper, but on the other hand too many details and missing depth cues complicate discussion with non-experts. Further, sketches only depict the 3DBM from one distinct view, which complicates the generation of a mental map. In contrast, a 3D visualization offers a more realistic experience, even for non-experts but requires IT infrastructure to be utilized. A building panorama can bridge the gap between the 2D architectural sketches and the virtual 3D building model. The discussion can be based on a printed version of the panoramic image by using a overview+detail visualization. Due to the flexibility of the rendering pipeline, a photorealistic as well as an architecture-oriented stylization can be generated.

4 Summary & Outlook

A multi-perspective view is an expressive and effective approach to communicate 3D geoinformation. The seamless combination of multiple perspectives views in one image reduces occlusion and enables new insights into geodata. Applied to digital 3D maps using focus+context visualization techniques, it increases screen-space utilization and reduces visual clutter. Compared to traditional perspective views, more pixels are used to communicate geoinformation, thus increasing the efficiency. Further, the seamless combination of focus and context regions reduces context switches, which lowers a user‘s mental load. As a result, the user can focus on the task. Drawbacks of multi-perspective maps are the unnatural view and the computational complexity. A first preliminary user study pointed out that users prefers MPV 3D maps over traditional 3D maps [18] for navigation task, but whether the efficiency is increased must be proved in future studies.

The application of MPVs as part of an overview+detail visualization of a small cartographic scale, such as buildings and building blocks, seems a promising approach to simplify user interaction and thus reduce a user’s burden on tasks, such as geometry refinement. The presentation of detailed information, while maintaining an overview of the complete data set, may ease visual analysis tasks and helps a user to gain new insights. A drawback of the overview+detail approach is context switches because of the two separate views. A user study will show whether the benefits outweigh the drawbacks.

Beside an evaluation of the presented approaches, future research will focus on the integration of comprehensible digital 3D maps in a service-oriented architecture. Therefore, different engineering challenges, e.g., the flexible combination of different shader programs, or a simple but versatile service interface, as well as conceptual challenges have to be addressed. The building panoramas are in a very early but promising prototypical stage with open research questions concerning the image synthesis and stylization.

References

[1] Lars Brodersen. Paradigm shift from cartography to geo-communication. In *XXIII International Cartographic Conference*. ICA, 2007.

[2] Andy Cockburn, Amy Karlson, and Benjamin B. Bederson. A review of overview+detail, zooming, and focus+context interfaces. *ACM Comput. Surv.*, 41(1):2:1–2:31, 2009.

[3] Jeff de la Beaujardiere. OpenGIS Web Map Server Implementation Specification, Version 1.3.0. OpenGIS Implementation Specification, 2006.

[4] Jürgen Döllner and Jan Eric Kyprianidis. Approaches to Image Abstraction for Photorealistic Depictions of Virtual 3D Models. In *Cartography in Central and Eastern Europe*, pages 263–277, 2010.

[5] Juri Engel and Jürgen Döllner. Approaches towards visual 3d analysis for digital landscapes and its applications. *Digital Landscape Architecture Proceedings 2009*, pages 33–41, 2009.

[6] Christian Häberling, Hansruedi Bär, and Lorenz Hurni. Proposed Cartographic Design Principles for 3D Maps: A Contribution to an Extended Cartographic Theory. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 43(3):175–188, 2008.

[7] Benjamin Hagedorn. OGC Web View Service. OGC Discussion Paper, February 2010.

[8] ICA. International Cartographic Association, Organisation and activities 1999 – 2003. Netherlands. Cartographic Society., 2000.

[9] Markus Jobst. Überlegungen für perzeptive Grössen in multimedialen 3D Karten. *Kartographische Schriften*, 10:117–126, 2006.

[10] Markus Jobst and Jürgen Döllner. Better Perception of 3D-Spatial Relations by Viewport Variations. In *Proceedings of the 10th International Conference on Visual Information Systems, VISUAL '08*, pages 7–18, 2008.

[11] Markus Jobst and Jürgen Döllner. Neo-Cartographic Influence on Map Communication in LBS. In *Location Based Services and TeleCartography II, From Sensor Fusion to Context Models*, pages 207–219, 2008.

[12] A Kolacny. Cartographic Information–a Fundamental Concept and Term in Modern Cartography. *Cartographic Journal*, 6:47–49, 1969.

[13] Thomas H. Kolbe. Representing and exchanging 3d city models with citygml. In *Proceedings of the 3rd International Workshop on 3D Geo-Information*, Lecture Notes in Geoinformation & Cartography, page 20, Seoul, Korea, 2009. Springer.

[14] Haik Lorenz, Matthias Trapp, Jürgen Döllner, and Markus Jobst. Interactive Multi-Perspective Views of Virtual 3D Landscape and City Models. In *Proc. AGILE Conference*, pages 301–321, 2008.

[15] Aniruddha Loya, Neeharika Adabala, Amitav Das, and Pragyan Mishra. A practical approach to image-guided building facade abstraction. *Computer Graphics International*, 2008.

[16] Alan M. MacEachren, Robert Edsall, Daniel Haug, Ryan Baxter, George Otto, Raymon Masters, Sven Fuhrmann, and Liujian Qian. Virtual environments for geographic visualization: potential and challenges. In *Proceedings of the 1999 workshop on new paradigms in information visualization and manipulation (NPIVM)*, pages 35–40. ACM, 1999.

[17] Sebastian Möser, Patrick Degener, Roland Wahl, and Reinhard Klein. Context Aware Terrain Visualization for Wayfinding and Navigation. *Computer Graphics Forum*, 27(7):1853–1860, 2008.

[18] Sebastian Pasewaldt, Matthias Trapp, and Jürgen Döllner. Multiscale Visualization of 3D Geovirtual Environments Using View-Dependent Multi-Perspective Views. *Journal of WSCG*, 19(3):111–118, 2011.

[19] Tom Patterson. A View From on High: Heinrich Berann's Panoramas and Landscape Visualization Techniques For the US National Park Service. In *Cartographic Perspectives*, number 36, pages 38–65. NACIS, 2000.

[20] Dave Pegg. Design Issues with 3D Maps and the Need for 3D Cartographic Design Principles, 2012.

[21] Michael P. Peterson. *Interactive and animated cartography*. Prentice Hall, 2005.

[22] Tumasch Reichenbacher. Adaptation in mobile and ubiquitous cartography. In *Multimedia Cartography*, chapter 27, pages 383–397. Springer, 2007.

[23] Tumasch Reichenbacher. The concept of relevance in mobile maps. In *Location Based Services and TeleCartography*, pages 231–246. Springer, 2007.

[24] Jonathan C. Roberts. State of the art: Coordinated & multiple views in exploratory visualization. In *Proceedings of the Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization*, CMV '07, pages 61–71, Washington, DC, USA, 2007. IEEE Computer Society.

[25] Amir Semmo, Matthias Trapp, Jan Eric Kyprianidis, and Jürgen Döllner. Interactive Visualization of Generalized Virtual 3D City Models using Level-of-Abstraction Transitions. *Computer Graphics Forum*, 31(3):885–894, 2012. Proceedings EuroVis 2012.

[26] Desney S. Tan, Darren Gergle, Peter G. Scupelli, and Randy Pausch. Physically large displays improve path integration in 3D virtual navigation tasks. In *Proc. ACM CHI*, pages 439–446, 2004.

[27] Jianxiong Xiao, Tian Fang, Ping Tan, Peng Zhao, Eyal Ofek, and Long Quan. Image-based façade modeling. *ACM Trans. Graph.*, 27(5):161:1–161:10, December 2008.

Demonstrating Test-driven Fault Navigation

Michael Perscheid
Software Architecture Group
Hasso-Plattner-Institut
michael.perscheid@hpi.uni-potsdam.de

In this paper, we present a demonstration of our test-driven fault navigation approach. This interconnected guide for debugging reproducible failures analyzes failure-reproducing test cases and supports developers in following infection chains back to root causes. With the help of a motivating error from the Seaside Web framework, we show how to reveal suspicious system parts, identify experienced developers for help, and debug erroneous behavior and state in the execution history.

1 Introduction

The correction of ubiquitous software failures can cost a lot of money [19] because their debugging is often a time-consuming development activity [3, 7]. During debugging, developers largely attempt to understand what causes failures: Starting with a test case, which reproduces the observable failure, they have to follow failure causes on the infection chain back to the root cause (defect) [22]. This idealized procedure requires deep knowledge of the system and its behavior because failures and defects can be far apart [8]. Unfortunately, common debugging tools are inappropriate to systematically investigate such infection chains in detail [9]. Thus, developers have to primarily rely on their intuition and the localization of failure causes takes up a lot of time [10]. To prevent debugging by trial and error, experienced developers apply the scientific method and its systematic hypothesis-testing [10,22]. However, even when using the scientific method the search for failure causes can still be a laborious task. First, missing expertise about the system makes it hard to understand incorrect behavior and to create reasonable hypotheses [11]. Second, contemporary debugging approaches still provide little or no support for the scientific method. For these reasons, we summarize our research question as follows:

> How can we effectively support developers in creating, evaluating, and refining failure cause hypotheses so that we reduce debugging costs with respect to time and effort?

In this paper, we present a demonstration of our *test-driven fault navigation* that guides the debugging of reproducible failures [13, 14, 20]. Based on the analysis of passing and failing test cases, we reveal anomalies and integrate them into a breadth first search that leads developers to defects. This systematic search consists of four specific navigation techniques that together support the creation, evaluation, and refinement of failure cause hypotheses for the scientific method. First, structure navigation [14] localizes suspicious system parts and restricts the initial search space. Second, team navigation [14] recommends experienced developers for helping with failures even if defects are still unknown. Third, behavior navigation [14, 15] allows developers to follow emphasized infection chains of failing test cases backwards. Fourth, state navigation [5, 6] identifies corrupted state and reveals parts of the infection chain automatically. We implement test-driven fault navigation in our *Path tools framework* [12, 14, 15, 20] for the Squeak/Smalltalk development environment and limit its computation costs with the help of our *incremental dynamic analysis* [5, 12, 15]. This lightweight dynamic analysis ensures a feeling of immediacy when debugging with our tools by splitting the run-time overhead over multiple test runs depending on developers' needs. Hence, our test-driven fault navigation in combination with our incremental dynamic analysis answers important questions in a short time: where to start debugging, who understands failure causes best, what happened before failures, and which program entities are in question.

The remainder of this paper is organized as follows: Section 2 explains the motivating example for demonstrating our approach. Section 3 briefly introduces our test-driven fault navigation. Section 4 demonstrates our approach and how it supports debugging of the motivating example. Section 5 concludes and presents next steps.

2 Motivating Example: Typing Error in Seaside

We introduce a motivating example error taken from the Seaside Web framework [2, 16] that serves as a basis for demonstrating our test-driven fault navigation approach in the following sections.

Seaside[1] is an open source Web framework implemented in Smalltalk [4]. The framework provides a uniform, pure object-oriented view of Web applications and combines a component-based with a continuation-based approach [17]. With this, every component has its own control flow which leads to high reusability, maintainability and a high level of abstraction. Additionally, it is written in Smalltalk that allows developers to debug and update applications on the fly. It provides a layer over HTTP and HTML that let you build highly interactive Web applications that come very close to the implementation of real desktop applications. Finally, Seaside consists of about 650 classes, 5,500 methods and a large test suite with more than 700 test cases.

We have inserted a defect into Seaside's Web server and its request/response processing logic (`WABufferedResponse` class, `writeHeadersOn:` method). Figure 1 illustrates the typing error inside the header creation of buffered responses. Once a client opens a Seaside Web application, its Web browser sends a request to the corresponding Web server. This request is then processed by the framework leading to a corresponding response to the browser. Depending on the Web application, this response is either a streamed or buffed response object. While the first transfers the

[1] `www.seaside.st`

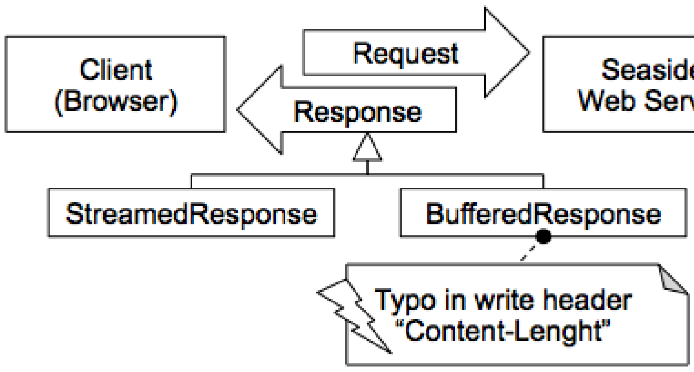


Figure 1: An inconspicuous typo in writing buffered response headers leads to faulty results of several client requests.

message body as a stream, the latter buffers and sends the response as a whole. During the creation of buffered responses, there is a typo in writing the header. The typo in "Content-Leng<u>ht</u>" is inconspicuous but leads to invalid results in browser requests that demand buffered responses. Streamed responses are not influenced and still work correctly. Although the typo is simple to characterize, observing it can be laborious. First, some clients hide the failure since they are able to handle corrupted header information. Second, as the response header is built by concatenating strings, the compiler does not report an error. Third, by reading source code like a text, developers tend to overlook such small typos [18].

3 Anomalous Guide to Localize Causes in Failing Test Cases

Based on test cases that reproduce the observable failure [20], we introduce a novel systematic top-down debugging process with corresponding tools called test-driven fault navigation. It does not only support the scientific method with a breadth-first search [21] but also integrates hidden test knowledge for guiding developers to failure causes. Starting with a failure-reproducing test case as entry point, we reveal suspicious system parts, identify experienced developers for help, and navigate developers along the infection chain step by step. In doing so, anomalies highlight corrupted behavior and state and so assist developers in their systematically hypothesis-testing. Figure 2 summarizes our complete test-driven fault navigation process and its primary activities:

Reproducing failure: As a precondition for all following activities, developers have to reproduce the observable failure in the form of at least one test case. Besides the beneficial verification of resolved failures, we require tests above all as entry points for analyzing erroneous behavior. For this activity, we have chosen unit test frameworks because of their importance in current development projects. Our approach is neither limited to unit testing nor does it require minimal test cases as proposed by some guidelines [1].

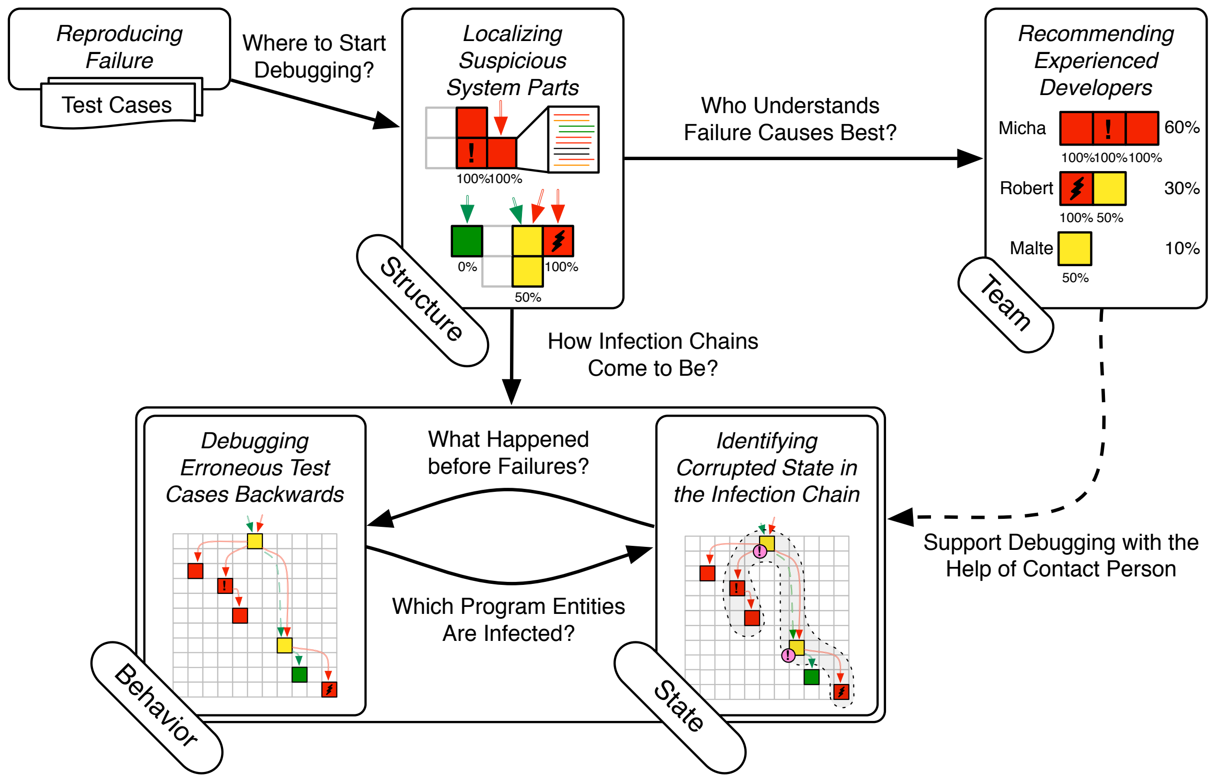

Reproducing Failure
Test Cases
Where to Start Debugging?
Localizing Suspicious System Parts
100%100%
0%
100%
50%
Structure
Who Understands Failure Causes Best?
Recommending Experienced Developers
Micha 60%
100%100%100%
Robert 30%
100% 50%
Malte 10%
50%
Team
How Infection Chains Come to Be?
Debugging Erroneous Test Cases Backwards
What Happened before Failures?
Which Program Entities Are Infected?
Identifying Corrupted State in the Infection Chain
Support Debugging with the Help of Contact Person
Behavior
State

Figure 2: Our test-driven fault navigation debugging process guides developers with interconnected advice to reproducible failure causes in structure, team, behavior, and state of the system under observation.

Localizing suspicious system parts (*Structure navigation*) Having at least one failing test, developers can compare its execution with other test cases and identify structural problem areas that help in creating initial hypotheses. By analyzing failed and passed test behavior, possible failure causes are automatically localized within a few suspicious methods so that the necessary search space is significantly reduced. We have developed an extended test runner called *PathMap* that supports spectrum-based fault localization within the system structure. It provides a scalable tree map visualization and a low overhead analysis framework that computes anomalies at methods and refines results at statements on demand.

Recommending experienced developers (*Team navigation*) Some failures require knowledge of experts to help developers in creating proper hypotheses. By combining localized problem areas with source code management information, we provide a novel *developer ranking metric* that identifies the most qualified experts for fixing a failure even if the defect is still unknown. Developers having changed the most suspicious methods are more likely to be experts than authors of non-infected system parts. We have integrated our metric within PathMap providing navigation to suitable team members.

Debugging erroneous test cases backwards (*Behavior navigation*) For refining their understanding of erroneous behavior, developers experiment with the execution and state history of a failing test case. To follow the infection chain back to the defect, they choose a proper entry point such as the failing test or one of its suspicious methods and start *PathFinder* our lightweight back in time debugger. If anomalies are available, we classify the executed trace and so allow developers to create proper hypotheses that assist the behavioral navigation to defects.

Identifying corrupted state in the infection chain (*State navigation*) Besides the classification of executed behavior with spectrum-based anomalies, we also highlight parts of the infection chain with the help of state anomalies. We derive state properties from the hidden knowledge of passing test cases, create generalized contracts, and compare them with failing tests. Such dynamic invariants reveal state anomalies by directly violating contracts on the executed infection chain and so assist developers in creating and refining hypotheses. For this state navigation, our *PathMap* automatically harvests objects and creates contracts while our *PathFinder* integrates the violations into the execution history.

Besides our systematic top down process for debugging reproducible failures, the combination of testing and anomalies also provides the foundation for interconnected navigation with a high degree of automation. All four navigation activities and their anomalous results are affiliated with each other and so allow developers to explore failure causes from combined perspectives. An integration supports developers in answering more difficult questions and allows other debugging tasks to benefit even from anomalies. Linked views between suspicious source code entities, erroneous behavior, and corrupted state help not only to localize causes more efficiently but also to identify the most qualified developers for understanding the current failure. Our *Path tools* support these points of view in a practical and scalable manner with the help of our incremental dynamic analysis. With a few user interactions, we split the expensive costs of dynamic analysis over multiple test runs and varying granularity levels so that we can provide both short response times and suitable results. Thus, developers are able to answer with less effort where to start debugging; who understands failure causes best; what happened before failures; and which program entities are infected.

4 Example: Debugging Seaside's Typo

With respect to our debugging process, we demonstrate each navigation step with the help of Seaside's typing error in more detail. Therefore, we start with the implementation of a failing test case that reproduce the observable failure. In the case of our example, developers have to implement a simple server request waiting for a corrupted response that cannot be parsed correctly. After that, they are able to apply our approach and its specific navigations as follows:

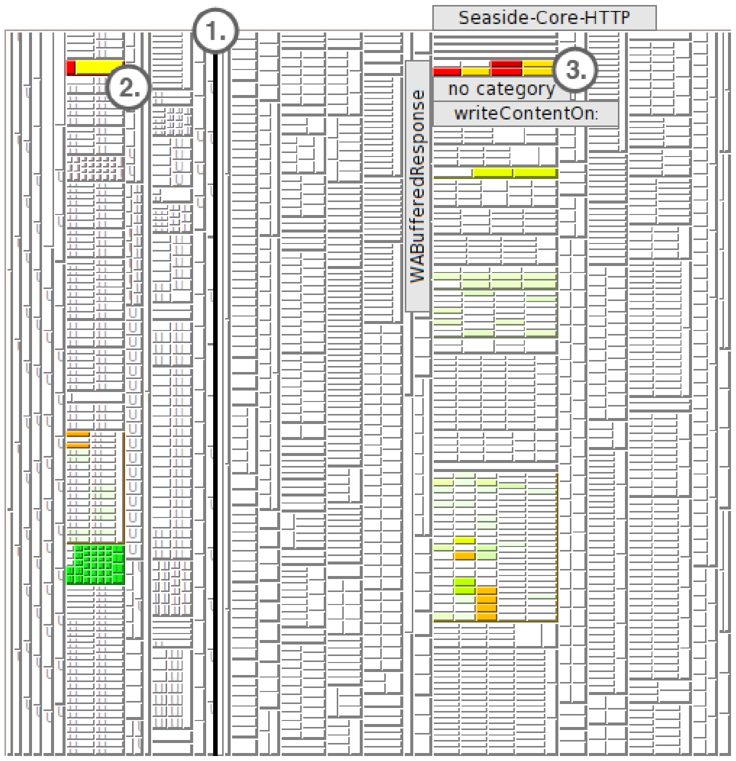
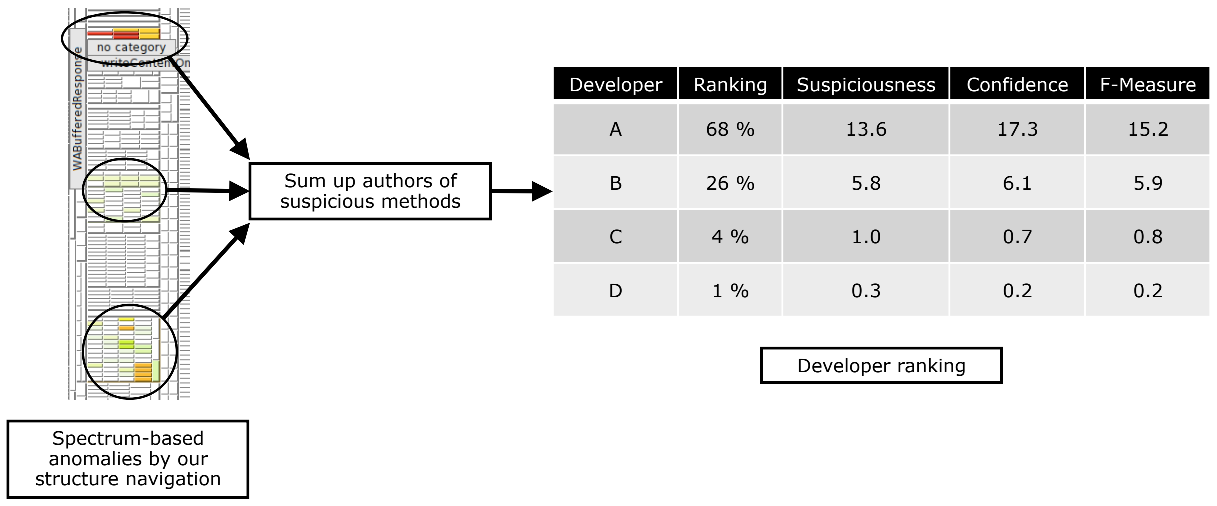


Figure 3: PathMap: In our Seaside example, our structure navigation restricts the search space to a few very suspicious methods in buffered responses.

4.1 Structure Navigation: Localizing Suspicious Response Objects

In our motivating typing error, we localize several anomalies within Seaside's response methods. Figure 3 presents the tree map visualization of Seaside with test classes on the left side and application classes on the right side (1). After running Seaside's response test suite with the result of 53 passed and 9 failed tests, our structure navigation colorizes the suspiciousness scores of methods and reveals anomalous areas of the system. For example, the interactively explorable yellow box (2) illustrates that all nine failing tests are part of the buffered test suite. In contrast, the green box below includes the passed streaming tests. The more important information for localizing failure causes is visualized at the upper right corner (3). There are three red and three orange methods providing confidence that the failure is included in the `WABufferedResponse` class. To that effect, the search space is reduced to six methods. However, a detailed investigation of the `writeContentOn:` and `content` method shows that they shares the same characteristics as our failure cause in `writeHeadersOn:`. At this point, it is not clear from a static point of view how these suspicious methods are related to each other. Developers need further help in order to understand how the failure comes to be.


no category
writeContentOn
WABufferedResponse
Sum up authors of suspicious methods
Developer ranking
Spectrum-based anomalies by our structure navigation

Developer	Ranking	Suspiciousness	Confidence	F-Measure
A	68 %	13.6	17.3	15.2
B	26 %	5.8	6.1	5.9
C	4 %	1.0	0.7	0.8
D	1 %	0.3	0.2	0.2

Figure 4: PathMap: Our developer ranking points out (anonymized) experts. Based on authors of spectrum-based anomalies, we create a ranked list of possible experts that understand failure causes best.

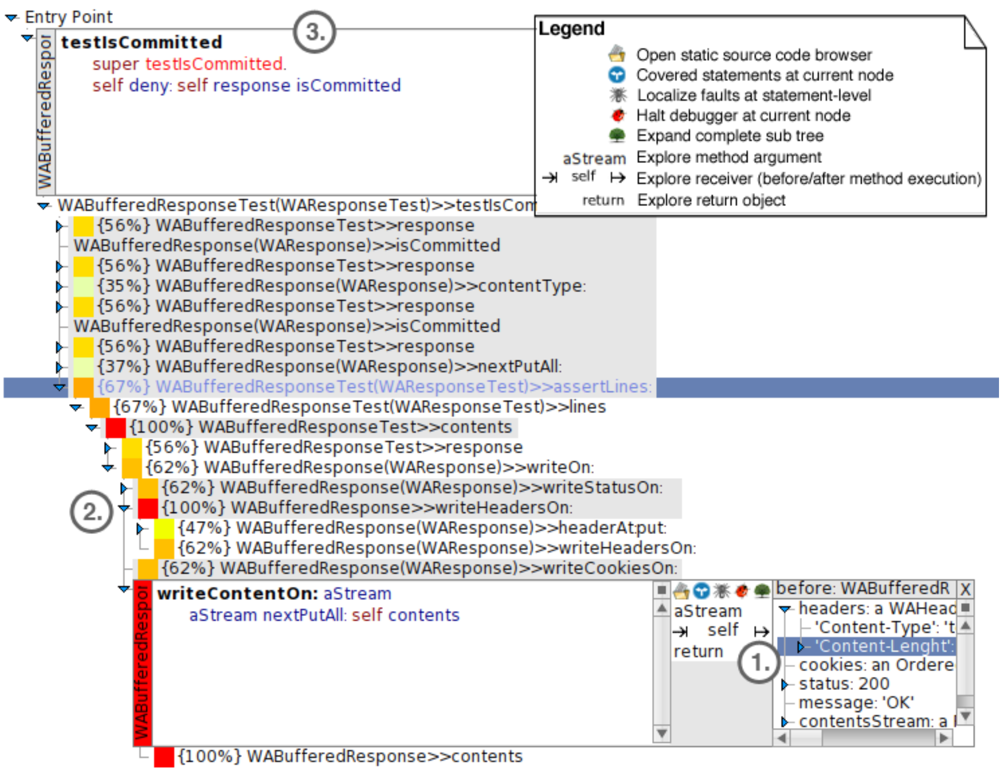
4.2 Team Navigation: Finding Experienced Seaside Developers for Help

With respect to our typing error, we reduce the number of potential contact persons to 4 out of 24 Seaside developers, whereby the author of the failure-inducing method is marked as particularly important. The table in Figure 4 summarizes the (interim) results of our developer ranking metric and suggests Developer A[2] for fixing the defect by a wide margin. Compared to a coverage-based metric, which simply sums up covered methods of failing tests per developer, our results are more precise with respect to debugging. A's lead is shrinking (only 55 %), C (24 %) changes the place with B (19 %), and the list is extended with a fifth developer (1 %). It should be noted that our team navigation does not blame developers. We expect that the individual skills of experts help in comprehending and fixing failure causes more easily and thus might reduce the overall costs of debugging.

4.3 Behavior Navigation: Understanding How the Failure Comes to Be

In our Seaside example, we highlight the erroneous execution history of creating buffered responses and support developers in understanding how suspicious methods belong together. Following Figure 5, developers focus on the failing `testIsCommitted` behavior and follow the shortest infection chain from the observable failure back to its root cause. They begin with the search for executed methods with a failure cause probability larger than 90 %. The trace includes and highlights four methods matching this query. Since the `writeContentOn:` method (1) has been executed shortly before the failure occurred, it should be favored for exploring corrupted

[2]Developers' names have been anonymized.



```
Entry Point
  testIsCommitted
      super testIsCommitted.
      self deny: self response isCommitted
  WABufferedResponseTest(WAResponseTest)>>testIsCom
    {56%} WABufferedResponseTest>>response
    WABufferedResponse(WAResponse)>>isCommitted
    {56%} WABufferedResponseTest>>response
    {35%} WABufferedResponse(WAResponse)>>contentType:
    {56%} WABufferedResponseTest>>response
    WABufferedResponse(WAResponse)>>isCommitted
    {56%} WABufferedResponseTest>>response
    {37%} WABufferedResponse(WAResponse)>>nextPutAll:
    {67%} WABufferedResponseTest(WAResponseTest)>>assertLines:
      {67%} WABufferedResponseTest(WAResponseTest)>>lines
        {100%} WABufferedResponseTest>>contents
          {56%} WABufferedResponseTest>>response
          {62%} WABufferedResponse(WAResponse)>>writeOn:
            {62%} WABufferedResponse(WAResponse)>>writeStatusOn:
            {100%} WABufferedResponse>>writeHeadersOn:
              {47%} WABufferedResponse(WAResponse)>>headerAt:put:
              {62%} WABufferedResponse(WAResponse)>>writeHeadersOn:
            {62%} WABufferedResponse(WAResponse)>>writeCookiesOn:
            writeContentOn: aStream
                aStream nextPutAll: self contents
              {100%} WABufferedResponse>>contents
```

```
aStream
⇥ self ↦
return

before: WABufferedR
  headers: a WAHead
    'Content-Type': 't
    'Content-Lenght':
  cookies: an Ordere
  status: 200
  message: 'OK'
  contentsStream: a
```

Legend

	Open static source code browser
	Covered statements at current node
	Localize faults at statement-level
	Halt debugger at current node
	Expand complete sub tree
aStream	Explore method argument
⇥ self ↦	Explore receiver (before/after method execution)
return	Explore return object

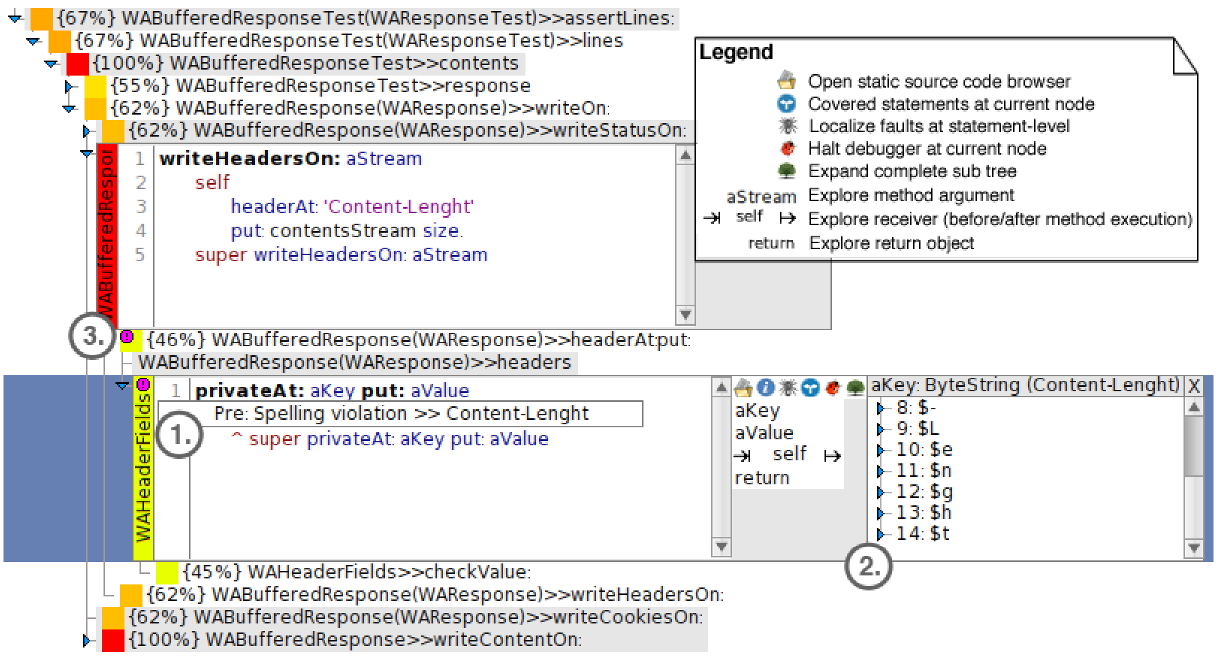
Figure 5: PathFinder: The classified execution history of our Seaside typing error.

state and behavior first[3]. A detailed inspection of the receiver object reveals that the typo already exists before executing this method. Following the infection chain backwards, more than three methods can be neglected before the next suspicious method is found (2). Considering `writeHeadersOn:` in the same way manifests the failure cause. If necessary, developers are able to refine fault localization at the statement-level analogous to our structure navigation and see that only the first line of the test case is always executed, thus triggering the fault (3).

4.4 State Navigation: Come Closer to the Typing Error

In our Seaside typing error, our state navigation is able to reveal two anomalies close by the root cause. First, we run all passing tests from the still working streamed responses and collect type and value ranges of their applied objects. Among others we check all string objects if they are spelled correctly or not. Second, we derive common invariants from the concrete objects and create corresponding contracts. Thus, we propagate the implicit assertions of the response tests to each covered method and automatically generate assertions for pre-/post-conditions and invariants of the corresponding class. Each assertion summarizes common object properties such as types, value ranges of numbers, and permissions of undefined objects. Third, we execute the same failing test case as in our behavior navigation but now with enabled contracts. As soon as a

[3]The simple accessor method `contents` can be neglected at this point.



```
{67%} WABufferedResponseTest(WAResponseTest)>>assertLines:
  {67%} WABufferedResponseTest(WAResponseTest)>>lines
    {100%} WABufferedResponseTest>>contents
      {55%} WABufferedResponseTest>>response
      {62%} WABufferedResponse(WAResponse)>>writeOn:
        {62%} WABufferedResponse(WAResponse)>>writeStatusOn:
        WABufferedRespon
writeHeadersOn: aStream
    self
        headerAt: 'Content-Lenght'
        put: contentsStream size.
    super writeHeadersOn: aStream
3.      {46%} WABufferedResponse(WAResponse)>>headerAt:put:
          WABufferedResponse(WAResponse)>>headers
            WAHeaderFields
privateAt: aKey put: aValue
1.  Pre: Spelling violation >> Content-Lenght
    ^ super privateAt: aKey put: aValue

aKey
aValue
⇥ self ↦
return

aKey: ByteString (Content-Lenght)  X
  8: $-
  9: $L
  10: $e
  11: $n
  12: $g
  13: $h
  14: $t
2.
          {45%} WAHeaderFields>>checkValue:
        {62%} WABufferedResponse(WAResponse)>>writeHeadersOn:
        {62%} WABufferedResponse(WAResponse)>>writeCookiesOn:
        {100%} WABufferedResponse>>writeContentOn:
```

Legend

	Open static source code browser
	Covered statements at current node
	Localize faults at statement-level
	Halt debugger at current node
	Expand complete sub tree
aStream	Explore method argument
⇥ self ↦	Explore receiver (before/after method execution)
return	Explore return object

Figure 6: PathFinder: State anomalies highlight the typing error and reveal the infection chain in the near of the defect.

contract is violated, we mark the corresponding exception in the execution history and so reveal for our `testIsCommitted` two state anomalies that are close by the defect.

Figure 6 summarizes the result of our state navigation. We mark method calls triggering a violation with small purple exclamation marks (1). Developers can further inspect these violations and see that a precondition fails. There is a spelling violation in the first argument of this method—all streamed responses used correctly spelled identifier keys for their header information. The corrupted state is opened for further exploration on the right (2). As our typo in "content-lenght" is automatically revealed, our state navigation gives developers helpful advice about the real failure cause. Another spelling violation is close by and developers can easily follow the infection chain back (3). Finally, the next very suspicious spectrum-based anomaly at `writeHeadersOn:` highlights the last step to the root cause. Following both state and spectrum-based anomalies directly guides developers to the defect of our Seaside typing error and also allows them to understand what causes the failure.

5 Summary and Next Steps

In this paper, we presented a demonstration of our test-driven fault navigation by debugging a small example. Starting with the structure navigation, we restrict the initial search space and lower speculations about failure causes. Based on this information, we are able to recommend experienced developers that can further help with debugging this failure. After that, developers apply our behavior and state navigation and follow the highlighted infection chain back to its root cause.

Future work deals with finishing the dissertation. So far, there is a first complete draft including 180 pages in total with 135 pages of content. We plan to submit the thesis at the end of this year.

References

[1] K. Beck. *Test-driven Development: By Example*. Addison-Wesley Professional, 1st edition, 2003.

[2] S. Ducasse, A. Lienhard, and L. Renggli. Seaside: A Flexible Environment for Building Dynamic Web Applications. *IEEE Software*, 24(5):56–63, 2007.

[3] M. Eisenstadt. My Hairiest Bug War Stories. *Communications of the ACM*, 40(4):30–37, 1997.

[4] A. Goldberg and D. Robson. *Smalltalk-80: The Language and its Implementation*. Addison-Wesley, 1st edition, 1983.

[5] M. Haupt, M. Perscheid, and R. Hirschfeld. Type Harvesting A Practical Approach to Obtaining Typing Information in Dynamic Programming Languages. In *Proceedings of the Symposium on Applied Computing*, SAC, pages 1282–1289. ACM, 2011.

[6] R. Hirschfeld, M. Perscheid, C. Schubert, and M. Appeltauer. Dynamic Contract Layers. In *Proceedings of the Symposium on Applied Computing*, SAC, pages 2169–2175. ACM, 2010.

[7] Z. Li, L. Tan, X. Wang, S. Lu, Y. Zhou, and C. Zhai. Have Things Changed Now?: An Empirical Study of Bug Characteristics in Modern Open Source Software. In *Proceedings of the Workshop on Architectural and System Support for Improving Software Dependability*, ASID, pages 25–33. ACM, 2006.

[8] B. Liblit, M. Naik, A. Zheng, A. Aiken, and M. Jordan. Scalable Statistical Bug Isolation. In *Proceedings of the Conference on Programming Language Design and Implementation*, PLDI, pages 15–26. ACM, 2005.

[9] H. Lieberman and C. Fry. Will Software Ever Work? *Communications of the ACM*, 44(3):122–124, 2001.

[10] R. Metzger. *Debugging by Thinking - A Multidisciplinary approach*. Elsevier Digital Press, 1st edition, 2003.

[11] N. Palix, J. Lawall, G. Thomas, and G. Muller. How Often Do Experts Make Mistakes? In *Proceedings of the Workshop on Aspects, Components, and Patterns for Infrastructure Software*, ACP4IS, pages 9–15. Technical report 2010-33 Hasso-Plattner-Institut, University of Potsdam, 2010.

[12] M. Perscheid, D. Cassou, and R. Hirschfeld. Test Quality Feedback - Improving Effectivity and Efficiency of Unit Testing. In *Proceedings of the Conference on Creating, Connecting and Collaborating through Computing*, C5, pages 60–67. IEEE, 2012.

[13] M. Perscheid, M. Haupt, R. Hirschfeld, and H. Masuhara. Test-driven Fault Navigation for Debugging Reproducible Failures. In *Proceedings of the Japan Society for Software Science and Technology Annual Conference*, JSSST, pages 1–17. J-STAGE, 2011.

[14] M. Perscheid, M. Haupt, R. Hirschfeld, and H. Masuhara. Test-driven Fault Navigation for Debugging Reproducible Failures. *Journal of the Japan Society for Software Science and Technology on Computer Software*, 29(3):188–211, 2012.

[15] M. Perscheid, B. Steinert, R. Hirschfeld, F. Geller, and M. Haupt. Immediacy through Interactivity: Online Analysis of Run-time Behavior. In *Proceedings of the Working Conference on Reverse Engineering*, WCRE, pages 77–86. IEEE, 2010.

[16] M. Perscheid, D. Tibbe, M. Beck, S. Berger, P. Osburg, J. Eastman, M. Haupt, and R. Hirschfeld. *An Introduction to Seaside*. Software Architecture Group (Hasso-Plattner-Institut), 1st edition, 2008.

[17] C. Queinnec. The Influence of Browsers on Evaluators or, Continuations to Program Web Servers. In *Proceedings of the International Conference on Functional Programming*, ICFP, pages 23–33. ACM, 2000.

[18] G. Rawlinson. *The Significance of Letter Position in Word Recognition*. PhD thesis, University of Nottingham, 1976.

[19] RTI. The Economic Impacts of Inadequate Infrastructure for Software Testing. Technical report, National Institute of Standards and Technology, 2002.

[20] B. Steinert, M. Perscheid, M. Beck, J. Lincke, and R. Hirschfeld. Debugging into Examples: Leveraging Tests for Program Comprehension. In *Proceedings of the International Conference on Testing of Communicating Systems*, TestCom, pages 235–240. Springer, 2009.

[21] I. Vessey. Expertise in Debugging Computer Programs: A Process Analysis. *International Journal of Man-Machine Studies*, 23(5):459–494, 1985.

[22] A. Zeller. *Why Programs Fail: A Guide to Systematic Debugging*. Morgan Kaufmann, 2nd edition, 2009.

Migrating Traditional Web Applications into Multi-Tenant SaaS

Eyad Saleh
Internet Technologies and Systems - Prof. Dr. Christoph Meinel
Hasso-Plattner-Institut
eyad.saleh@hpi.uni-potsdam.de

Software-as-a-Service (SaaS) is emerging as a new model of delivering a software, where users utilize software over the internet as a hosted service rather than an installable product. Multi-tenancy is the principle of running a single instance of the software on a server to serve multiple companies (tenants). Large number of applications have been built into non-SaaS mode, re-engineering such applications from scratch into SaaS requires tremendous efforts in terms of cost, manpower, and time. Thus, Migrating existing applications into SaaS mode becomes a requirement. In my first technical report [25], I introduced several challenges in SaaS and multi-tenancy, one of those challenges is software migration into SaaS. In this report, I highlight this challenge, discuss the related work and existing approaches, then I introduce my proposed approach.

1 Introduction: Multi-tenancy Evolution

History has shown that advances in technology and computing changes the way software are designed, developed, and delivered to the end users. These advances yield to the invention of personal computers (PCs) and graphical user interfaces (GUIs), which in turn adopted the client/server architecture over the old big, super, and expensive mainframes. Currently, fibers and fast internet connections, Service-Oriented Architectures (SOAs), and the high-cost of managing and maintaining on-premises dedicated applications raised the flag for a new movement in the software industry, and the result was the introducing of a new delivery model called Software-as-a-Service (SaaS) [1].

SaaS provides major advantages to both service providers as well as consumers. Service providers can provision a single set of hardware to host their applications and manage thousands of clients (tenants). As an alternative, they can easily install and maintain their software using any IaaS provider. As for consumers, they can use the application anywhere and any time, they are relieved from maintaining and upgrading the software (on-premises scenario), and benefit from cost reduction by following the pay-as-you-go model [2].

Multi-tenancy is a requirement for a SaaS vendor to be successful (Marc Benioff, CEO, Salesforce) [3]. Multi-tenancy is the core of SaaS; it is the ability to use a single-instance of the application hosted by a provider to serve multiple clients (tenants).

Software applications have been built for decades into non-SaaS mode, reengineering or re-designing such applications from scratch requires tremendous efforts in terms of cost, manpower, and time. Therefore, researchers have been proposing several approaches to migrate such applications into SaaS mode.

While migrating non-SaaS applications into SaaS mode, certain issues need to be considered, such as database architecture, security isolation, UI customizations, data-model extensions, business logic configuration, and workflow management. In this report, I propose an approach to facilitate the migration process with the focus on business logic configuration and workflow customization.

After this introduction, Section 2 discusses the related work and Section 3 outlines our proposed approach. In Section 4, we describe the use case followed by the implementation. Finally, we close with future work in Section 5.

2 Related Work

Migrating legacy applications to the cloud has received considerable attention recently as SaaS is becoming more popular. Existing approaches [4, 5, 7, 10–12] typically focus on UI customization and basic database modifications. The goal of such approaches is rather the ability to convert a legacy application into SaaS; they didn't consider to which level the migrated application will cope with changing business and users' requirements. For example, Cai et al. [4] and Nitu [10] concentrate on the look-and-feel and some basic configuration options; critical points such as business logic, workflow customization and data-model extension are not considered. An interesting approach for migrating traditional web applications into SaaS automatically without changing the source code is propose by Song et al. [5]. They adopt several technologies to accomplish this goal: (1) page template to fulfill configurability, (2) memory in thread to maintain tenant-info, and (3) JDBC proxy to adopt the modified database. Practically, migrating an application from non-SaaS mode into SaaS without having or changing the source code is very hard to achieve. Another transparent approach is introduced by Cai et al. [6], they focus on intercepting web requests and deriving tenant context, then carry this context with a thread in the web container. Additionally, they categorise several isolation points to achieve multi-tenancy.

Experienced reports are also reported in the literature, one example is Bezemer et al. [7], they report on a use case of converting an industrial, single-tenant application (CRM) for a company called Exact[2] into a multi-tenant one. They propose a pattern to migrate the application taking into account hardware sharing, high degree of configurability, and shared database instance. Another example is Chauhan et al. [8], they report on their efforts in migrating Hackystat [9], an open-source legacy application to the cloud.

Back-end customization was the focus of Müller et al. [11], they identify different categories of customization, such as desktop integration and UI customization. Additionally, they identify two cornerstones for a customizable, multi-tenant aware infrastructure, namely, dynamic instance composition and abstraction from the persistency layer.

[2]http://www.exact.com

Regarding performance and availability, several approaches tackle this challenge [21, 22]. Service Performance Isolation Infrastructure (SPIN) [22] aims at achieving efficient isolation and better resource sharing at the same time. SPIN detects instable states, identifies aggressive tenants (mainly based on abnormal resource usage), and removes bad impacts on other tenants. The main limitation of this approach is the manual identification and removal of the aggresive tenants, which is not applicable in large-scale SaaS environments. While Guo et al. [21] introduce a hybrid isolation between tenants. This isolation is accomplished in two steps. First, it categorizes the tenants into groups based on their SLA and behavior during run-time. Second, it applies one or more of the following isolation techniques: by tenant resource reservation, by tenant resource admission control, or tenant-oriented resource partition. Although this approach can achieve a degree of isolation between tenants, it decreases the level of resource sharing, because dynamic sharing between tenants is not possible.

From security perspective, security and data privacy on the SaaS level needs to be carefully considered, because the information about companies (tenants) is stored outside their control; therefore, they need to make sure that all required mechanisms to secure their data are in place. Current approaches, such as Lin et al. [23] propose a data protection framework consists of mainly three components, policy ranking, policy integration, and policy enforcement. Policy ranking aims to match the service provider who provides privacy policies similar to the client requirements. The integration module takes the requirements from the previous module as input, and generates privacy policies that satisfy involved parties. Finally, the enforcement module is responsible for ensuring that the policies are correctly applied in the environment. Another approach is proposed by Shen et al. [24], that achieve data privacy protection based on privacy constraints. They propose three kinds of privacy constraints to support multi-tenancy: (1) Attribute Privacy Constraint, (2) Value Privacy Constraint, and (3) Dependency Privacy Constraint. Additionally, a hybrid fragmentation algorithm is introduced.

3 Our Proposed Approach

In this report, we propose an approach for migrating traditional web applications into SaaS mode as shown in Figure 1. The proposed approach facilitate the migration process to produce a multi-tenant version of the application. The core component of our approach is the configuration and customization layer, by which, the UI components, such as logos and colors, the business logic, and workflow configuration data for the current tenant are restored and passed to the application server. The DB configuration data will be passed to the DB server for query transformation as well. Further, the application server receives the above specified data from the upper layer and pass it to the run-time customization engine, which integrates all components and lunches the application instance. Furthermore, a log service is used to record the application actions and store them in text files. Finally, a dedicated monitoring service is used to monitor the performance and status of the application, and detects any faults or bad resource usage. Next, we describe our approach in details.

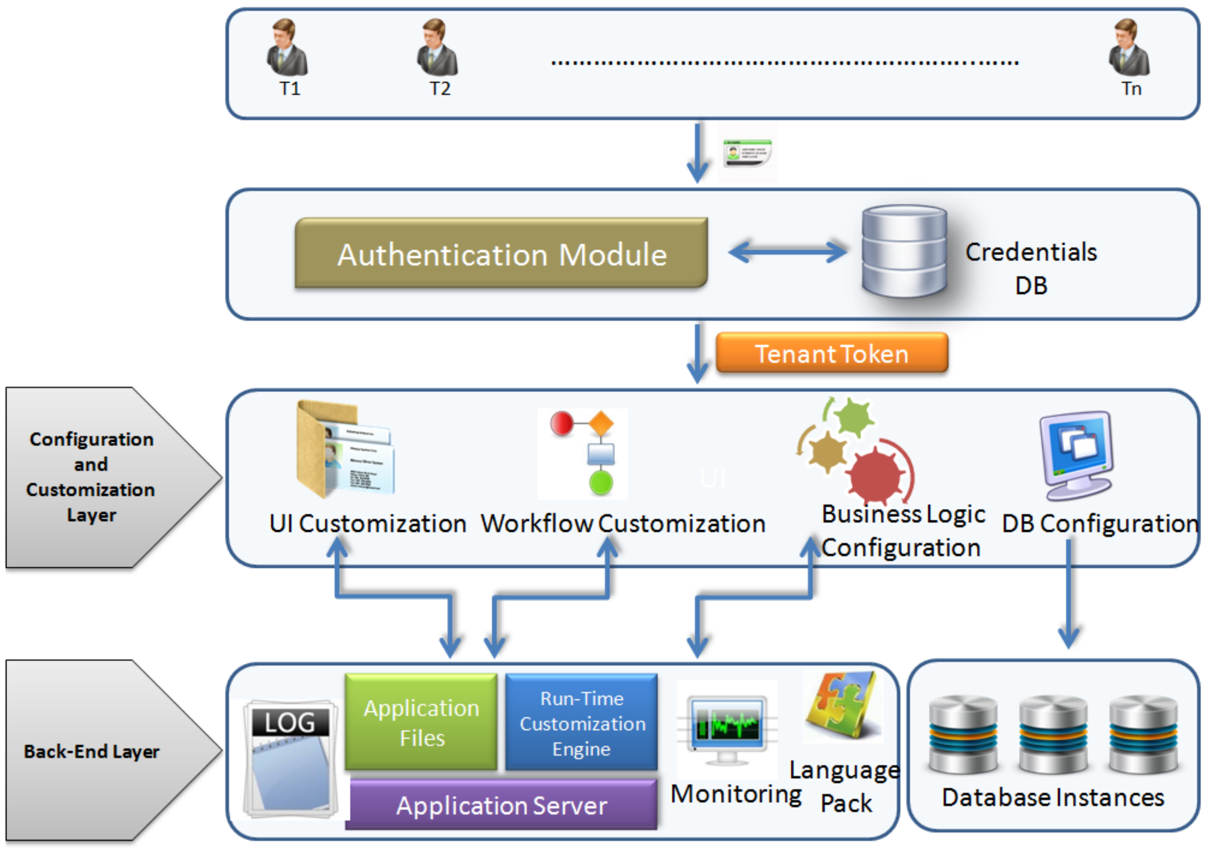


Figure 1: The Architecture of our Proposed Approach

3.1 The Back-End Layer

The main components of this layer are:

A) Monitoring means getting feedback from the usage of the current software, which leads to enhance and improve the current version of the software. This service monitors the performance of the software, for example, which queries respond slowly, what are the most heavily-used components of the application, which tenant is overusing the resources, etc. This collective data will enable the vendor to enhance (upgrade) the software and better isolate tenants to improve the performance.

B) Log files are important to several applications, and more importantly to the multi-tenant ones. They can be used for many reasons, such as monitoring the performance of the application, figuring out processing bottlenecks, discovering software bugs in the early stages of the release and fix them immediately. Thus, we can use data extracted from those files to improve the performance of the application.

C) A multi-tenant application is used by several tenants, and they might be from different domains or having specific language requirements. Therefore, a language pack is an additional component the tenant can use to personalize the language settings. This component is responsible for managing language files, and providing the settings that correspond to the tenant preferences to the run-time customization engine. Several languages could be defined in the language pack such as English, Arabic, German, Chinese, etc.

3.2 The Configuration and Customization Layer

A) **User Interface Customization:** UI customization means changing the look-and-feel of the application to be tenant-specific. This includes general layout, logos, buttons, colors, and locale settings. To utilize this customization, we propose the usage of Microsoft's ASP.NET master page concept [16].

ASP.NET master page allows the developer to create a consistent look for all pages (group of pages) in the application; one or more master pages could be created for each tenant and used in the application. The master page provides a shared layout and functionality for the pages of the application, when users request any page, ASP.NET engine merges the master page that contains the layout with the requested content-page, and sends the merged page to the user as shown in Figure 2.

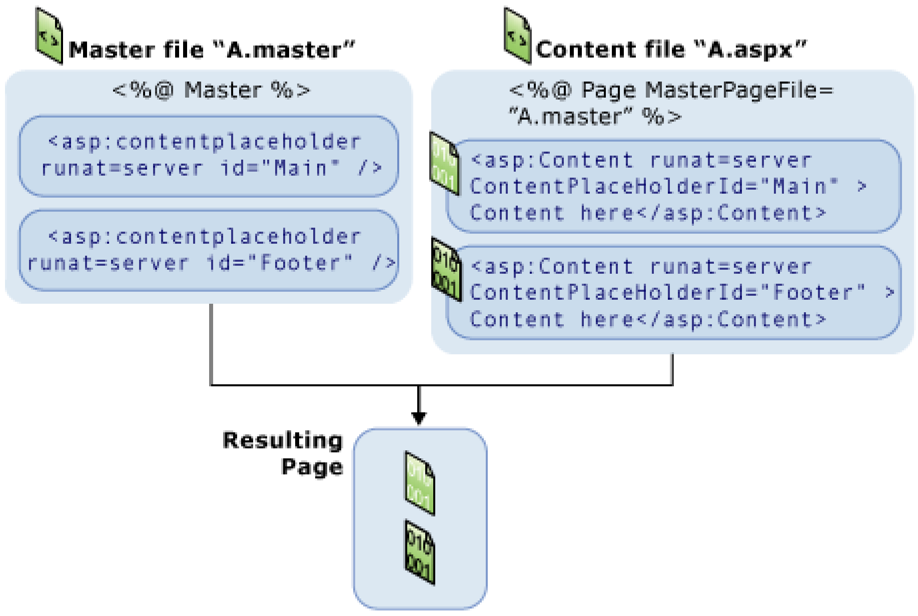

Master file "A.master"
<%@ Master %>
<asp:contentplaceholder runat=server id="Main" />
<asp:contentplaceholder runat=server id="Footer" />
Content file "A.aspx"
<%@ Page MasterPageFile= "A.master" %>
<asp:Content runat=server ContentPlaceHolderId="Main" > Content here</asp:Content>
<asp:Content runat=server ContentPlaceHolderId="Footer" > Content here</asp:Content>
Resulting Page

Figure 2: ASP.NET Master Page [16]

Application developers can define a master page for each tenant by applying the master page technique, which contains the required layout, color, buttons, and other design components. Moreover, several master pages could be defined for each tenant. Therefore, tenants will have the chance to get benefit of using dynamic look-and-feel.

B) **Workflow Customization:** The workflow of the application might vary from one tenant to another, for instance, a recruitment agency (Tenant A) might wait until they receive a request for a specific vacancy (from a company looking for employees), then start looking for applicants, while another agency (Tenant B) would collect applications, interview applicants, and then short-list them according to their potential, and have them ready for any vacancies from companies looking for employees (Figure 3). Therefore, we consider that customizing the workflow of the multi-tenant software is important. In order to achieve this customization, two steps are required. First, identify the components of the software that need to be customized, second, change the design of these components to be loosely-coupled, thus they can be easily replaced by other versions and integrated with other components, and therefore, each tenant can have his own version of the same component.

Worth to mention that changing the design of the entire application into loosely-coupled components is a difficult task, on the other hand, customizing the complete workflow of the application may not be necessary since the majority of the application components are normally common among all tenants. Therefore, our approach, as mentioned earlier, is to identify the components of the software that need to be customized first, then proceed with the customization process.

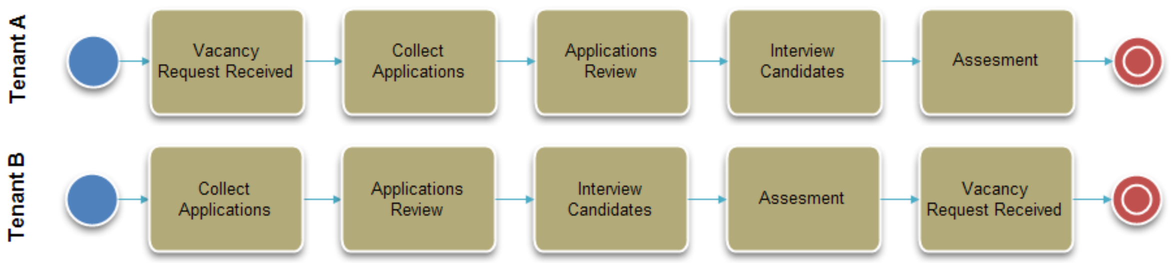


Figure 3: Different Workflow for two recruitment agencies

C) **Business Logic Configuration:** In software engineering, multi-tier architecture enables developers to divide the application into different tiers to maximize application's re-usability and flexibility. One of the most common implementations of multi-tier is the three-tier architecture, which divides the application into three-tiers, namely, the presentation layer (PL), the business logic layer (BLL), and the data access layer (DAL).

Business rules are part of the business logic layer (BLL), and these rules vary from one organization to another. For instance, in a travel agency, if a reseller or a client exceeds his credit limit, all his upcoming purchases are rejected, while another agency may apply a different rule that state if the reseller exceeds his credit limit for three weeks without any payment, he is blacklisted.

In order to achieve and maximize multi-tenancy, we propose that these business rules need to be tenant-specific; the tenant should have the ability to design, apply, and re-configure his own rules at run-time. Therefore, a tool that offers this feature is needed as a part of the proposed approach.

D) **Database Configuration:** Database design is considered as one of the most critical issues in multi-tenant SaaS because multiple tenants are mapped into one physical database. Therefore, a robust database architecture must be modeled and implemented.

Consolidating multiple tenants into one database requires changes to the design of the tables and the queries as well, thus, a query transformation is required. For instance, in a traditional hospital management system, a simple query to fetch a patient record would be "select * from patient where SSN=1234", while in a multi-tenant system, this will not work, since the "patient" table would have information for many tenants (i.e., hospitals). Therefore, the query should be changed to something similar to "select * from patient where tenant_id=12 and SSN=1234", in this case, the patient record that belongs to the tenant 12 (i.e., hospital 12) is retrieved. Based on that, the rules for query transformation should be stored in the database configuration files and restored by the transformation engine.

4 Use Case and Implementation

In Jan 2011, Forrester Research published a report with the title "Which Software Markets Will SaaS Disrupt?" [18], one of the conclusions was that Human Resource Management Software (HRMS) is a good candidate for SaaS. Thus, we decided to select an HRMS as our use case. Orange Human Resource Management (OrangeHRM) is the world's most popular open source HRM application with more than one million users globally and more than 600,000 downloads [17]. OrangeHRM released under the GNU General Public License and targets small and medium enterprises. The system functionality includes employee information management, absence and leave management, recruitment management, performance evaluation, etc. From technical point of view, OrangeHRM is implemented using PHP and MySQL, and its relational data model contains about 115 tables. The implementation of our approach is described below.

The main components of our implementation can be classified as follows:

A) **Tenant Subscription:** A dedicated module is responsible of handling tenants subscription. The tenant fills the required information in a simple form and receives access to the system within 10 minutes.

B) **Authentication Module:** Two additional tables are created, to store tenant's information and credentials. The first table stores general information about each tenant, while the second table stores the usernames and passwords for all tenant's users.

C) **UI Customization:** UI Customization is achieved by separating the code from the view layer, then allowing the tenants to customize the view layer to match their needs. OrangeHRM follows the three-tiers architecture we proposed in our architecture, therefore, in order to accomplish UI customization, we will develop a tool to allow tenants to configure the Presentation Layer (PL) according to their needs. thus, the tenant is able to change the look-and-feel of the system, such as general layout, colors, images, font-type, menu design, etc.

D) **Business Logic Configuration:** As mentioned earlier, OrangeHRM is built on top of Symfony, and Symfony was developed using standard OO techniques, and therefore, all the business logic is encapsulated in classes. As a result, customizing the business logic means having different classes for different tenants.

First, for each unit of business logic, a base class is created which contains the common business logic shared by all tenants. Second, a second class is created for every tenant which inherits from the base class, thus all data members and member functions are available for this child class. A dedicated tool will be developed to allow the tenant to customize the code of the child class, it will be similar in concept to the query builder in known DBMSs such as Oracle or SQL Server. The tenant will be able to define new variables, insert different programming statements such as if-else or loops, use HTTP objects, such as Request and Response, etc. It is worth mentioning that the tenant will not be able to write any single line of code manually, simply for validation purposes. It will be very complex to validate code entered by tenants manually.

E) **Database Configuration:** Several schema-mapping techniques are available for multi-tenancy and SaaS in general [13,15,20]. In terms of schema management and DDL, these techniques can be classified in two types, first, the database owns the schema, such as private tables, extension tables, and sparse columns (in Microsoft SQL Server), second, the application owns the schema, such as XML and pivot tables.

We decided to select the second option where the application owns the schema based on our assumption that the application needs to be customized as much as possible, especially in terms business logic. The XML technique is ideal for our work because it allows us to dynamically (i.e., on the fly) customize the database by attaching an XML document to every table we feel appropriate. In this XML document, we can define custom fields belong to some tenants over the others.

F) **Tenant Termination:** This module has four different cases where the tenant account can be terminated: (1) the duration of the subscription period has ended, and the tenant is not intended to renew, (2) the tenant decides to voluntarily terminate his subscription, (3) the vendor detects bad behaviour from the tenant during service monitoring, such as bad resources usage, and (4) automatic termination of tenant account without notification, and this option might be available in some cases. However, this should be mentioned in the availability SLA and terms of use policies.

G) **Tenant Management:** In this module, the vendor is able to list all the tenants, view their details, such as subscription date, expiry date, etc. Also the vendor is able to view the specific customizations done by each tenant. Moreover, the vendor can monitor the performance of the tenants and check their resources usage. contact with tenants is also done using this module.

5 Summary and Future Work

In this report, we explored the challenge of migrating existing web applications into SaaS. We discussed the related work and introduced our approach to handle such a challenge. We propose a new approach to facilitate the migration process. Primarily, we explored the configuration and customization of the application from several layers, such as UI, business logic, workflow, and database design. The future work will focus on applying our approach on OrangeHRM, and conduct several experiments to validate the proposed approach. Additionally, we will conduct a survey to understand the users' experience and to highlight key requirements from their perspective.

References

[1] T. McKinnon: *The Force.com Multitenant Architecture: Understanding the Design of Salesforce.com's Internet Application Development Platform*, White Paper, USA, 2008.

[2] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and M. Zaharia: *Above the Clouds: A Berkeley View of Cloud Computing*. Technical Report, University of California, Berkeley, USA, 2009.

[3] D. Woods: Salesforce.com Secret Sauce, Forbes, January 2009. [Online]. Available: http://www.forbes.com/2009/01/12/cio-salesforce-multitenancy-tech-cio-cx_dw_0113salesforce.html [retrieved: 10, 2012]

[4] H. Cai, K. Zhang, M. J. Zhou, W. G., J. J. Cai, and X. Mao: *An End-to-End Methodology and Toolkit for Fine Granularity SaaS-ization*. In IEEE International Conference on Cloud Computing, Bangalore, India, 2009.

[5] J. Song, F. Han, Z. Yan, G. Liu, and Z. Zhu: *A SaaSify Tool for Converting Traditional Web-Based Applications to SaaS Application*. In IEEE 4th Int. Conf. on Cloud Computing, Washington, DC, USA, 2011.

[6] H. Cai, N. Wang, and M. Zhou: *A Transparent approach of enabling SaaS multi-tenancy in the cloud*. In IEEE 6th World Congress on Services, Miami, Florida, USA, 2010.

[7] C. Bezemer, A. Zaidman, B. Platzbeecker, T. Hurkmans, A. Hart: *Enabling multi-tenancy: An industrial experience report*. In the 26th IEEE International Conference on Software Maintenance, Timisoara, Romania, 2010.

[8] M. Chauhan and M. Babar: *Migrating service-oriented system to cloud computing: An experience report*. In IEEE 4th Int. Conf. on Cloud Computing, Washington DC, USA, 2011.

[9] Hackystat website. [Online]. Available: http://code.google.com/p/hackystat [retrieved: 10, 2012]

[10] Nitu: *Configurability in SaaS (software as a service) applications*. In the 2nd India Software Engineering Conference, Pune, India, 2009.

[11] J. Müller, J. Krüger, S. Enderlein, M. Helmich, and A. Zeier: *Customizing Enterprise Software as a Service Applications: Back-end Extension in a multi-tenancy Environment*. In the 11th International Conference on Enterprise Information Systems, Milan, Italy, 2009.

[12] D. Yu, J. Wang, B. Hu, J. Liu, and L. Zhang: *A Practical architecture of cloudification of legacy applications*. In IEEE 7th World Congress on Services, Washington DC, USA, 2011.

[13] F. Chong, C. Gianpaolo, and R. Wolter: *Multi-Tenant Data Architecture*, Microsoft Corporation, http://www.msdn2.microsoft.com, 2006.

[14] W. Tsai, Q. Shao, Y. Huang, X. Bai: *Towards a scalable and robust multi-tenancy SaaS*. In the Second Asia-Pacific Symposium on Internetware, Suzhou, China, 2010.

[15] S. Aulbach, T. Grust, D. Jacobs, A. Kemper, and J. Rittinger: *Multi-Tenant Databases for Software as a Service: Schema-Mapping Techniques*. In the 2008 ACM SIGMOD international conference on Management of data, Vancouver, Canada, 2008.

[16] ASP.NET Master Page on Microsoft.com. [Online]. Available: http://msdn.microsoft.com/en-us/library/wtxbf3hh.aspx [retrieved: 10, 2012]

[17] OrangeHRM Website. [Online]. Available: http://www.orangehrm.com [retrieved: 10, 2012]

[18] A. Bartels, L. Herbert, C. Mines, and Sarah Musto: Forrester Research. [Online]. Available: http://www.forrester.com/Which+Software+Markets+Will+SaaS+Disrupt/fulltext/-/E-RES57405 [retrieved: 10, 2012]

[19] Symfony Website. [Online]. Available: http://www.symfony-project.org/ [retrieved: 10, 2012]

[20] S. Aulbach, D. Jacobs, A. Kemper, and M. Seibold: *A Comparison of Flexible Schemas for Software as a Service*. SIGMOD Conference (SIGMOD 2009), Providence, Rhode Island, USA, 2009.

[21] C. J. Guo, W. Sun, Y. Huang, Z. H. Wang, and B. Gao: A framework for native multi-tenancy application development and management. In *The 9th IEEE Int. Conf. on E-Commerce Technology and The 4th IEEE Int. Conf. on Enterprise Computing, E-Commerce and E-Services*, Tokyo, Japan, 2007.

[22] X. H. Li, T. C. Liu, Y. Li, and Y. Chen: SPIN: Service Performance Isolation Infrastructure in Multi-tenancy Environment. In *the 6th International Conference on Service-Oriented Computing*, Sydney, Australia, 2008.

[23] Dan Lin and Anna Squicciarini: Data protection models for service provisioning in the cloud. In *the 15th ACM symposium on Access control models and technologies*, Pittsburgh, Pennsylvania, USA, 2010.

[24] Y. Shen, W. Cui, Q. Li, Y. Shi: Hybrid Fragmentation to Preserve Data Privacy for SaaS. In *the 2011 Eighth Web Information Systems and Applications Conference*, Chongqing, China, 2011.

[25] Eyad Saleh. “Multi-tenans SaaS: Challenges and Approaches". Technical report, Hasso-Plattner-Institut, Potsdam, Germany, April 2012.

Challenges in the Visualization of Large Multi-Dimensional Hierarchies

Sebastian Schmechel
Computer Graphics Systems Group
Hasso-Plattner-Institute
sebastian.schmechel@hpi.uni-potsdam.de

The visualization of hierarchically ordered data has an important role in the research field of information visualization. In extension to the parent-child-relationship between nodes in such hierarchy, the encoding of additional attributes and their evolution over time results in a major challenge in this research area. To handle challenges like large data visualization, the stability of layout algorithms, to support user's pattern recognition and through it the memorability of visualization parts, and the interaction and configuration depending on users' needs, new rendering, interaction and visualization techniques are needed. This report presents the main challenges of multidimensional hierarchy visualization and discusses special issues like constrained hierarchies as well as time-dependent changes and their consequences.

1 Introduction

The visualization of hierarchicall ordered data has an important role in the research field of information visualization. Its primary goal, supporting a user to achieve a fast visual extraction of the underlying data-structure, is enabled either explicit, through node-link diagrams, implicit, through recursive partitioning of a given root-space, or in a hybrid way [10]. The commonality of these three ways is the creation of geometrical representations of a hierarchy's nodes — the visualization artifacts. In general, the hierarchically ordered nodes, represented by those artifacts, contain additional information dimensions. Using various visual variables, e.g., size, color or position (a detailed overview can be found in [2]), as a basis for the visualization artifacts the possibility to encode these additional dimensions is given.

For example, a file-system visualized by a simple Treemap algorithm (Slice'n Dice Treemap [11]) containing folders and files (shortcuts are not included due to the destruction of the basic tree structure), represents the hierarchical data with files as leafs and folders, that can recursively contain folders and files, as nodes. In addition to the hierarchical parent-child relationship information of files and directories meta information for both types about the label, creator or editing authors and permission details are attached, as well as specific information for leaf nodes (files), e.g., size or type-specifier. The aforementioned Slice'n Dice Treemap algorithm partitions a given

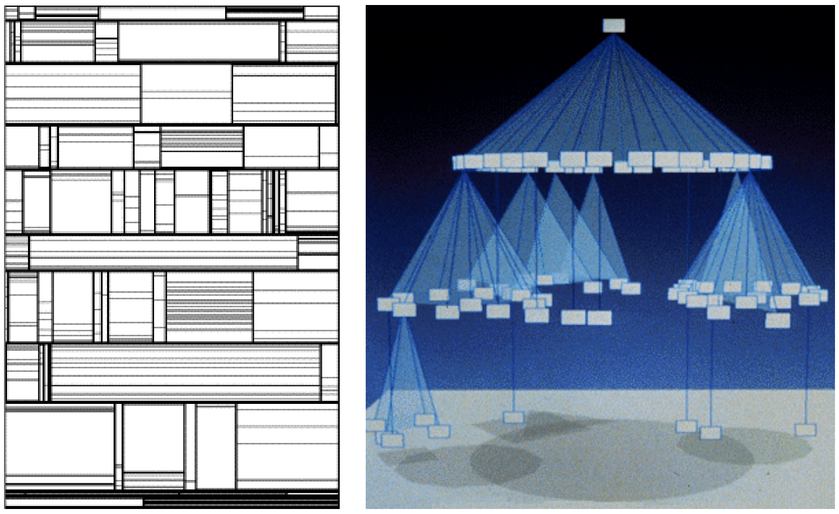

(a) Slice 'n dice Treemap [11] (b) Cone-Tree [8]

Figure 1: Two layout approaches for hierarchy visualization

rectangular bounding area, representing the root node of the hierarchy, into smaller partitions — the roots' children — with respect to their percentage size of its parent. The bounding rectangle is split alternating in either horizontal or vertical way depending on their depth-level in the hierarchy. This results in a classical 2D-Treemap, encoding the depth-level of nodes and their size attribute.

Several other layout algorithms exists, all with their specific advantages and disadvantages. One aspect to focus is the stability and readability of these algorithms. More over to visualize multidimensional hierarchically ordered data additional mappings are needed and common problems of large data-set and hierarchies with additional constraints have to be discussed.

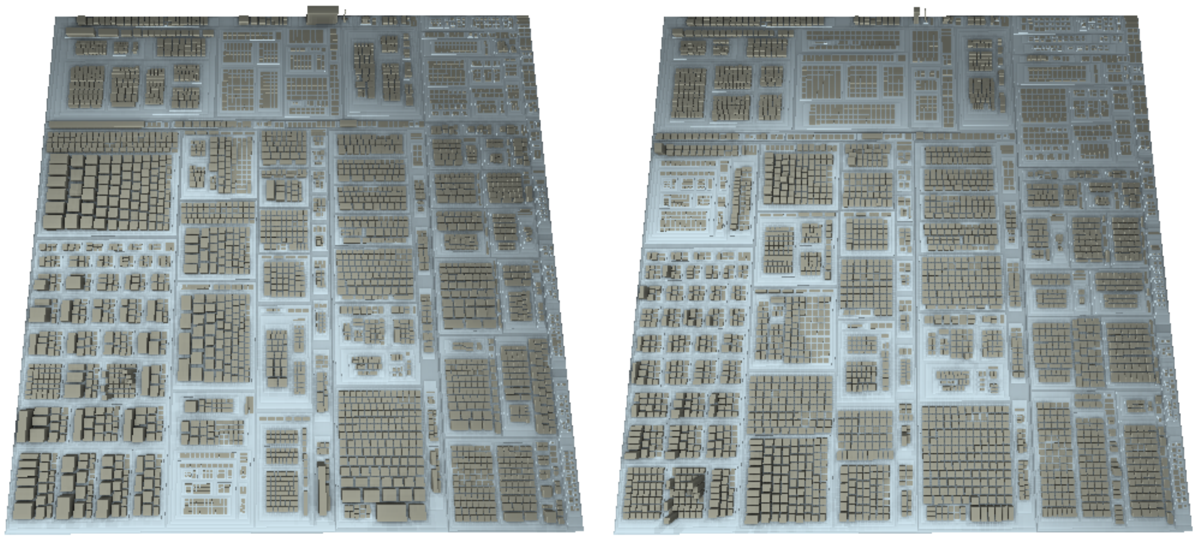
2 Challenges

This section summarizes and discusses the existing challenges of large multidimensional hierarchy visualization in more detail and gives an idea of how to create a multi-dimensional hierarchy visualization.

2.1 Layout Algorithms

The described Slice'n Dice algorithm is just one example to create an initial layout that visualizes the parent-child relationships and weights of the hierarchy nodes. Various other layout algorithms and approaches exist, e.g., PieTree [3], Fractal Tree [7] or Sunburst [12]. Schulz et al. [10] describe the design space of hierarchy visualization by three main aspects:

- Dimensionality (2D / 3D / hybrid)



(a) Software System Files at Revision x (b) Software System Files at Revision x+n

Figure 2: Visual effect of time-dependent changes in the underlying data attributes

- Edge Representation (explicit / implicit / hybrid)
- Node Alignment (radial / axis-parallel / free)

For Example: The aforementioned Slice'n Dice algorithm of Johnson and Shneiderman is a 2D (dimensionality), axis-parallel (node alignment) layout with an implicit edge representation (Figure 1(a)), whereas the Cone-Tree [8] of Robertson et al. is a 3D, radial layout, which visualizes its edges explicitly (Figure 1(b)).

This huge variation of algorithms that are used to create an initial layout brings up the question of comparability with respect to the users needs. For it, there are two main properties of layout algorithms:

2.1.1 Readability

The readability describes the cognitive effort of a user to create a mental model of the underlying hierarchical relationship and their attributes. It involves the ability to compare items' attributes with each other. The aforementioned Slice'n Dice algorithm for example, has a bad readability for item size comparison because of its alternating change of the slice-axis in horizontal and vertical way. In it, it is possible to create items with equal area size but totally different aspect ratios.

2.1.2 Stability

The degree of stability of a layout algorithm is determined by its visual change due to changes in the hierarchy and the nodes' attributes. If small changes in the hierarchy, e.g., adding or removing a node, have a large effect on the visual appearance of the resulting visualization one speaks off instable layouts. This property gets important in

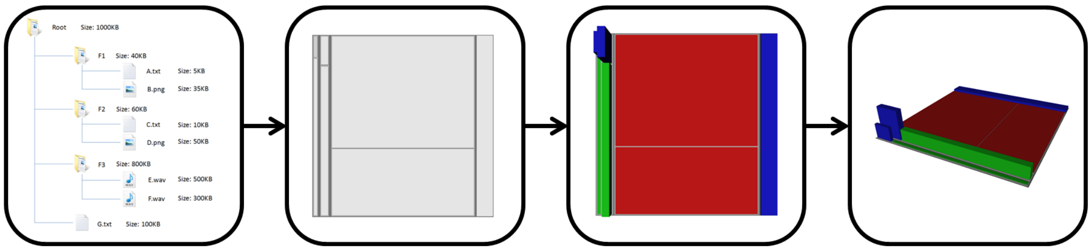


Figure 3: Slice'n Dice Treemap with additional encoding (file/directory size mapped to bounding area, file-type to color and number of editors to height)

case of visualizing evolutions of hierarchies instead of just looking at snapshots at a single time (Figure 2).

2.2 Additional Attribute Encoding

To encode additional attributes in the previous example other dimensions of the visual artifacts' design space are used. The number of a nodes' authors can be mapped to the node height. Furthermore color can be used to show the depth level of directory artifacts and the type of files (see Figure 3). Besides the dimensions of hierarchical nodes' attributes, which makes it possible to show snapshots at a specific time, the dynamic changes over time form another important dimension. One important requirement to create a visualization that shows the evolution of hierarchies and their attributes is the aforementioned layout-stability. Another attribute dimension, which will not be discussed, is the connection between leaf-nodes, visualized for example in Holten's BundleView [5].

2.3 Large Hierarchical Data-Sets

Another big advantage in hierarchical data visualization is the size of the datasets. Going back to the directory example, hierarchies with more than 100k items are quite common. The visualization of those huge datasets affects two main aspects. First, an effective representation of the given data (hierarchy and attributes) is needed to make sure the needed memory stays minimal. Second, computer graphics solutions for complex geometry rendering is needed due to the additional mappings. Through these facts techniques like level-of-detail, server-side rendering and hardware rendering have to be focussed. Examples for such datasets are SAP-Software systems structure as well as file-systems, the icd-code base and the tree of life.

2.4 Constraints in Hierarchies

The last aspect to focus on are hierarchies that are constrained in a several way. Going back to the SAP-Software System example, the given hierarchy is represented by the structure packages as group nodes and development objects as leaf nodes. Besides that, there are 2 other hierarchies that work as constraints and should be able to be displayed in a visualization — the maintenance component hierarchy and the application component hierarchy. They are logical hierarchical representations created by grouping elements of the structure package hierarchy, which can originate from different levels and parents. These constraints, that do not effect the structure package hierarchy itself but the positioning of their nodes require an extra handling.

3 Related Work

Schulz et al. give a good overview about hierarchical visualization [9]. The huge variety of visualization listed there shows two things. The visualization techniques do not differ a lot for the last decade and there are still no visualizations that handle all of the aforementioned challenges. One of the main problems is that visualizations are often too static to bring a solution for these complex problems. For it, the use of hybrid approaches is a good entry point.

Steinbrückner and Lewerentz introduced a metaphor-based method to create a stable hierarchy visualization with additional informations about the hierarchies evolution [13]. A road network created by using a fractal algorithm is created that represents the hierarchy nodes. Leafs are mapped onto building located on the side of their road (parent node). The elevation buildings standing on represents a node's age in the graph. Nodes that are removed or moved in the hierarchy are painted with a low alpha value while the rest of the items has color mapping.

Balzer and Deussen present another approach to create hierarchy visualization with a higher stability — Voronoi treemaps [1]. They use Voronoi diagrams as bases for the recursive layout algorithm. A random point set, representing the nodes children, is initally used to create a Voronoi diagram. After that the centroid of the created cells are calculated and used as new sites. The Voronoi diagram is iteratively calculated until a threshold of movement for these new calculate centroid sites is reached. Several other publications improve the calculation time by using either hardware accelaration, parallel processing or totally different algorithms like [14], [4]or [6].

4 Approaches

To create visualizations with higher layout stability the use of Voronoi diagrams or their generalization — power diagram — is a well proved way. However no 3D visualizations based on Voronoi diagrams are published yet. One of the main problems in existing approaches is the start with a random point set as basis for the layout and through it the need for an iterative calculation of it until a defined stability threshold is reached. By using an algorithm that creates a unique pointset to a given polygon this will be not needed anymore.

For the visualization of constrained hierarchies only explicit edge representations are useful. The use of implicit edge representation would result in recursive partitioning of a given parent space. Through it, the rearrangement of nodes from different parents as group become impossible without destroying the readability of the basic hierarchy. To handle this, a layout algorithm, similar to the one Steinbrückner and Lewerentz used, but with configurable curves as group nodes is one possible solution. Users then can get an idea of the hierarchy by viewing the structural road network and deform the visualized roads to make the relations or constraints clear by creating regional groups. Another more automatically way to create the regional groups is to use a force metaphor, similar to the dust and magnet visualization technique for multi-variate data [15]. The computation of the constraint regions can then be done by mapping a gravity force value onto the distance of a node to a specified region.

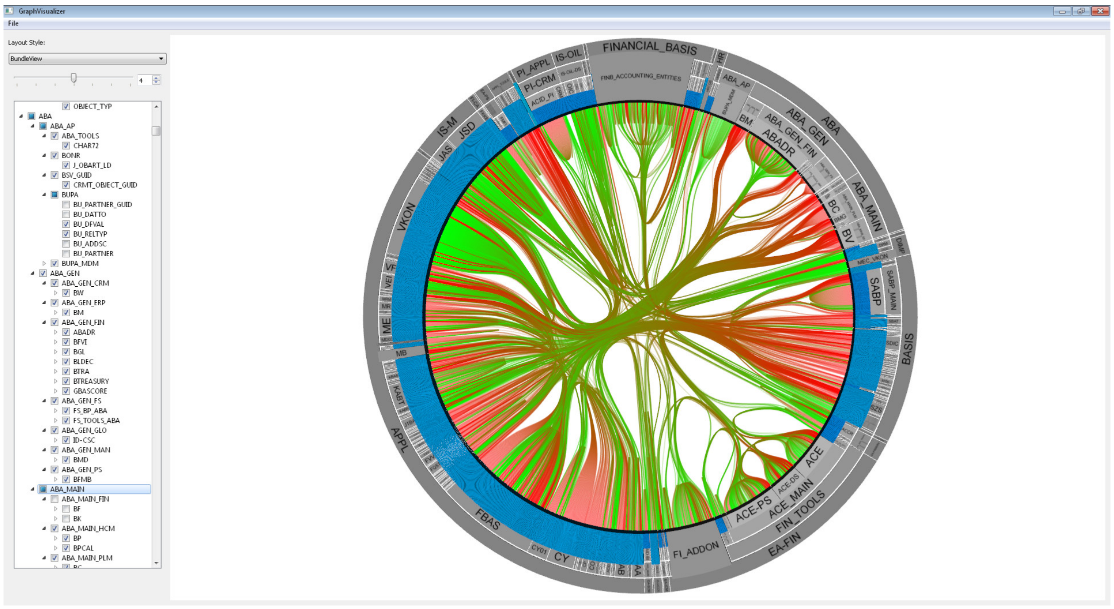
The challenge of rendering high complex 3d scenes with simple items, e.g., boxes in Treemaps, polygons in Voronoi-Treemaps, can be solved by an approach we submitted at the GRAPP conference. For it, we use a geometry shader, that handles the creation of simple shapes (triangle strips), with its desired shape attributes as input values (size / height). This approach also comes with another advantage, the representation of the complex scenes can be simplified to point plus attribute representations to decrease the needed memory and through it makes it possible to visualize millions of items.

5 Next Steps

As next steps I'm planning to prototypical implement the aforementioned aspects with primary focus on stable layouting, constraints in hierarchies and configurability of layouts. After that I want to focus more on interaction topics in 2.5D or 3D Voronoi treemaps like rearrangement of nodes and testing focus and content techniques in it. Furthermore the use of other visual variables like shape or texture has not been focused on yet.

6 Further Activities

In the last semester I co-organized the seminar for software visualization and analysis. There I supervised a student working on an image-based approach for Treemap-comparison to identify and visualize changes in multiple Treemap layouts generated from snapshots of a hierarchy at different evolution steps. More over I was teaching assistant in the "Visualization" lecture. Furthermore I'm taking part of the SAP Innovation Center / HPI Cooperation for Software Visualization. The main goal in it, is to create



Layout Style: BundleView

Figure 4: Visualization of SAP Development Object Relationships

visualizations that help SAP-IC identify structural and performance problems with SAP-related data (see Figure 4). At least I co-submitted a paper showing a new technique for a faster 3D-Treemap rendering technique at the GRAPP conference.

References

[1] Michael Balzer and Oliver Deussen. Voronoi treemaps. In *Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*, INFOVIS '05. IEEE Computer Society, 2005.

[2] J. Bertin and M. Barbut. *Sémiologie graphique: les diagrammes, les réseaux, les cartes*. Mouton Paris, 1967.

[3] A. Dix, R. Beale, A. Wood, et al. Architectures to make simple visualisations using simple systems. In *AVI: Proceedings of the working conference on Advanced visual interfaces*, volume 2000, pages 51–60, 2000.

[4] D. Gotz. Dynamic voronoi treemaps: A visualization technique for time-varying hierarchical data. Technical report, Technical Report RC25132, IBM, 2011.

[5] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):741–748, 2006.

[6] A. Nocaj and U. Brandes. Computing voronoi treemaps: Faster, simpler, and resolution-independent. In *Computer Graphics Forum*, volume 31, pages 855–864. Wiley Online Library, 2012.

[7] T.J. Ong, J.J. Leggett, and U. Yun. Visualizing hierarchies and collection structures with fractal trees. In *Computer Science, 2005. ENC 2005. Sixth Mexican International Conference on*, pages 31–40. IEEE, 2005.

[8] G.G. Robertson, J.D. Mackinlay, and S.K. Card. Cone trees: animated 3d visualizations of hierarchical information. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, pages 189–194. ACM, 1991.

[9] H.J. Schulz. Treevis. net: A tree visualization reference. *Computer Graphics and Applications, IEEE*, 31(6):11–15, 2011.

[10] H.J. Schulz, S. Hadlak, and H. Schumann. The design space of implicit hierarchy visualization: A survey. *Visualization and Computer Graphics, IEEE Transactions on*, 17(4):393–411, 2011.

[11] B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on graphics (TOG)*, 11(1):92–99, 1992.

[12] J. Stasko and E. Zhang. Focus+ context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In *Information Visualization, 2000. InfoVis 2000. IEEE Symposium on*, pages 57–65. IEEE, 2000.

[13] F. Steinbrückner and C. Lewerentz. Representing development history in software cities. In *Proceedings of the 5th international symposium on Software visualization*, pages 193–202. ACM, 2010.

[14] A. Sud, D. Fisher, and H.P. Lee. Fast dynamic voronoi treemaps. In *Voronoi Diagrams in Science and Engineering (ISVD), 2010 International Symposium on*, pages 85–94. IEEE, 2010.

[15] J.S. Yi, R. Melton, J. Stasko, and J.A. Jacko. Dust & magnet: multivariate information visualization using a magnet metaphor. *Information Visualization*, 4(4):239–256, 2005.

Memory Management in a Many-Core Distributed Hypervisor

Jan-Arne Sobania

Operating Systems and Middleware
Hasso-Plattner-Institut
jan-arne.sobania@hpi.uni-potsdam.de

Computer architecture for mainstream desktop and server machines is changing. What began with single-core machines and is mostly multi-core today will be *many-core* in the future – much more processor cores, and consequentially overall different system architecture and interconnect. This provides challenges for designers and engineers to provide backwards compatibility for existing software on modern machines: hardware may no longer provide features, like cache coherency, programmers have been accustomed to and software relies on. This report proposes a solution in form of virtualization layer, consisting of distributed hypervisors, to provide those features the real machines lack, thus allowing coexistence of old and new applications on modern platforms.

1 Introduction

Due to advancements in processor manufacturing processes over the last years, the classic architecture of “mainstream” computers – especially, those based on the Intel 32-bit architecture and its descendants – is changing. What first started with the single-processor 8086 as the IBM PC is now one of the most successful architectures in the market, ranging from small laptops to mid- and high-end servers. All these systems are *multi-core* machines today, following the well-known shared-memory symmetric multi-processing (SMP) model.

However, SMP has its drawbacks. With an increasing number of cores, inter-core communication overhead increases as well. If each core needs to communicate with each other, for example to maintain cache coherency (a necessary prerequisite for SMP operation), it places a natural limit on how many cores can be integrated. Even though these limits can be increased by using well-known techniques from cluster machines [12] – like directory nodes to restrict the set of cores that need to receive cache-related messages – this merely delays the problem. Therefore, researchers are striving for other architectures to build future *many-core* chips; processors with *far* higher numbers of cores than today’s, and a potentially very different system and interconnect architecture.

An example of a potential prototype for such an architecture is the Intel Single-chip Cloud Computer (SCC) [5], a 48-core research processor developed at Intel Labs Braunschweig, Germany. Even though individual cores still use the well-known x86 instruction set and can execute a standard operating system like Linux, the chip does not support an SMP OS (like a single standard Linux kernel instance running concurrently on more than *one* core) due to missing cache coherency.

A different architecture also poses different problems, though: there is much existing software for x86-based systems, and new architectures would need to be able to execute it to be successful in the market. If the hardware does not offer features required by old applications, these would need to be added in software. We propose to use *virtualization* to transparently add such features to the platform.

The report is organized as follows: section 2 presents a model of the potential future many-core processor we want to virtualize. Section 3 discussed how basic SMP emulation via a hypervisor can be achieved. Section 4 details memory management necessary for such a hypervisor. Section 5 discusses related work, and section 6 concludes the report.

2 System Model

Before we can go into details on the architecture of a hypervisor for upcoming many-core architectures, it is necessary to describe its underlying system first. Our model for new many-core architectures is based on the following principles:

1. All processors are identical in computation capabilities. That is, they support the same instruction set and the same set of features in the processor architecture (e.g., pipeline stages, caches, model-specific registers and so on).

2. Processors are grouped into *multi-core islands*. A single multi-core islands resembles a traditional symmetric multi-processing (SMP) chip with all required internal communication, like for cache coherency and inter-processor interrupts.

3. All islands form *nodes* on an on-chip network, over which all external communication is carried out. The network is formed of message routers. We model all links as having infinite bandwidth and no congestion. Messages use fixed routes only, and router latency is finite and constant.

4. Multi-core islands do not communicate with each other, unless explicitly requested by software running on them. Specifically, if processors from different islands exchange messages, other islands that happen to be connected to network routers on that path do not take note of the other communication, and are not affected by it in any way.

5. A restricted form of *backwards compatibility* is designed into the system. If the network and routers are configured appropriately, arbitrary software (including an operating system kernel) running on the new machine behaves as it did on the predecessors, as long as it runs on *only a single multi-core island*. The only notable difference could be timing of memory accesses, because these could target memory connected to different message routers.

6. All processors can access all memory, but not necessarily at the same time; the processor's physical address space is smaller than the one for the many-core. Instead, we assume an additional component (topologically between the processor and the on-chip network) to translate addresses, in addition to and independent of the regular address translation performed by the processors themselves.

Essentially, the new many-core could be build by taking an existing multi-core processor design and connecting its external (e.g., front-side-bus) interface to a *protocol converter* for generating and receiving packets suitable for the on-chip network. The protocol converter would also be responsible for performing secondary address translation; the target address for memory-access packets (for example, the network address of the message router the memory controller is connected to) could be taken from the same set of configuration registers.

In the list of principles above, we only mentioned that *computation capabilities* are assumed to be identical for all processors, but we did not make any restriction in regards to I/O devices. This is intentional; we explicitly allow for different nodes to be connected to different devices. For example, some nodes may contain more processors cores, but not connect to any devices at all in our model, whereas others contain fewer processor cores but special I/O devices like graphics cards or other accelerators.

An example for a many-core of this architecture is depicted at a very high-level in figure 1.

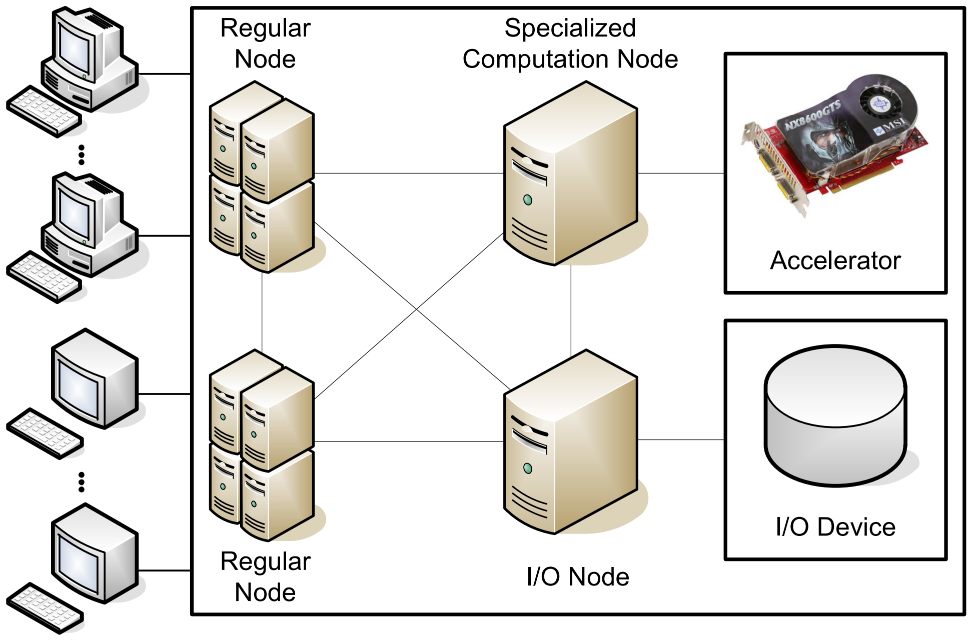


Figure 1: High-Level System Overview

This system consists of four islands of two different types: two "big" 4-way SMP islands without I/O devices on the left-hand side, and two "small" single-processor islands with I/O devices on the right-hand side, one with an accelerator device and one with a hard disk. For the following sections, we concentrate only on the processors in the system, on both kinds of islands; mentioned I/O devices as well as the monitors and consoles on the far left side are future work.

3 SMP-Emulation via Virtualization

When faced with a many-core system as discussed above, the question of how to write corresponding software arises. One possibility would be to treat the many-core as a collection of independent processors (i.e., a “cluster-on-a-chip”) and use established programming techniques for such an environment; for example, sockets (over emulated on-chip Ethernet links) or the *Message Passing Interface* (possibly over an optimized processor-specific communication mechanism).

However, cluster programming has a set of drawbacks, especially for the area of “mainstream computing” – PCs or compatible systems based on the Intel 32-bit architecture (known as IA32) and its extension to 64-bit. These architectures dominate the desktop as well as the entry- to mid-level server markets, support a large base of existing programs and developers, and have a long-standing tradition of backwards compatibility. “Traditional” x86-based systems all follow the *symmetric multi-processing* (SMP) model with shared memory, which programmers for these platforms have also been accustomed to, but unfortunately there is no straight-forward way of porting programs between the two models. Therefore, a future many-core which does not support the SMP model would be likely to fail in this market, as bringing existing programs to the new architecture would require a complete rewrite; given the huge amount of time and money that went into the development of the old programs, this does not seem to be an option for most developers.

As a solution to this dilemma, we use a virtualization layer on the new many-core to emulate a traditional SMP system. This way, the new system can not only execute *applications* written for the old architecture – which would also be possible by using a single-system image (SSI) operating system known from traditional cluster computing – but an entire *SMP operating system* as well. This also allows for applications that have dependencies on operating system implementation details to make use of the new architecture, something which may not be easily possible by just re-implementing operating system functions.

For virtualization, one can distinguish two broad architectures: the control program (or *virtual machine monitor* (VMM), as it is commonly called today) can either run as the lowest-level system software, directly on the hardware (“bare-metal” or *Type 1 VMM*), or as a regular application on top of an underlying operating system (“hosted VMM” or *Type 2 VMM*). We chose the hosted approach for our system, which we named “RockyVisor”. The resulting architecture is depicted in figure 2. The term “hypervisor” is used as a synonym for “virtual machine monitor” in this work.

All processors (or, more general: multi-core islands) run within their own private memory space – therefore resembling our model of a future many-core system above – and run a base operating system called the *Level 1 Operating System* (LV1). Within LV1, the hypervisor container runs as a regular application (Type 2 VMM). The set of hypervisor processes then provide a virtual machine, with virtual CPUs running inside a shared memory block and executing the guest operating system, which we call the *Level 2 Operating System* (LV2). LV2 is a regular SMP OS like Linux that can then run arbitrary SMP applications.

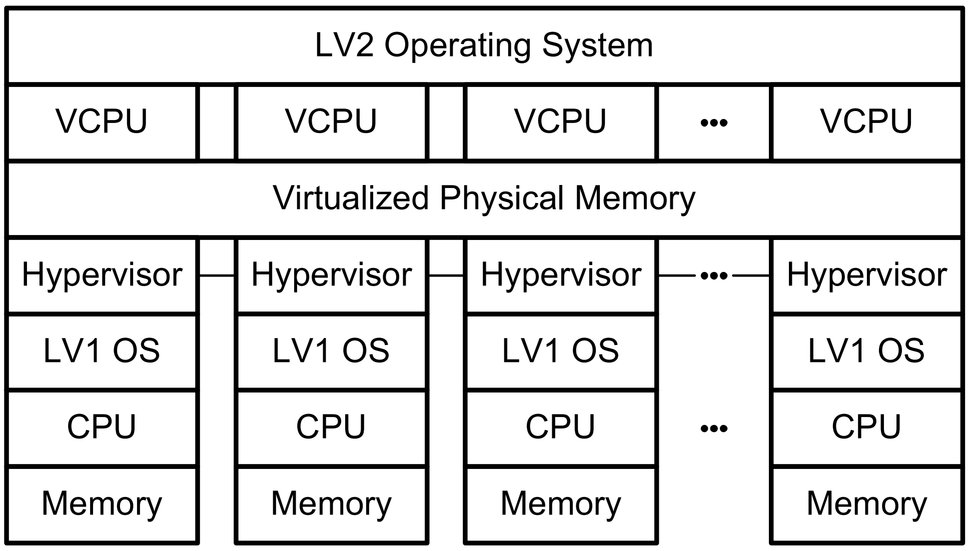


Figure 2: RockyVisor with Guest OS

This layering allows LV1 to manage resources (like memory) assigned to the SMP VM just like it does for any other application as well. In addition, memory can also be *shared* between the container process and other processes; this allows us to move any necessary bookkeeping or I/O emulation into another process. If a single LV1 instance runs on an SMP island, I/O can therefore run concurrently to the guest, with all necessary communication happening via shared memory. For single-core islands, the I/O processes would either compete with the container for CPU resources, or we could partition the many-core into hypervisor islands (for running the guest's virtual processors) and I/O islands (for I/O emulation processes that never execute any guest code).

In our architecture, the hypervisor container processes are responsible for emulating any SMP features the guest needs. If the underlying many-core supports some features directly, they may simply be passed through (e.g., direct device access if the router network can be configured accordingly). Other features that may not be present on the physical hardware itself must be emulated in the hypervisor processes (like shared-memory obeying a consistency model the guest can run under). This requires the hypervisors to be able to communicate to each other, but as they are applications on top of LV1, they can use the corresponding LV1 capabilities.

Even though LV1 is used as the host for the hypervisors, it is not restricted to this role. For example, if an application shall be run on the many-core that does *not* need SMP emulation – like an application specifically written for the system, or more general, for cluster environments – it can run on top of LV1 directly. Therefore, by introducing the hypervisors, the many-core system would not be restricted to running old-style SMP applications; it rather allows for them to coexist with new applications, so both users and programmers can choose for the task at hand which application or programming model provides the better fit.

For SMP simulation, the hypervisor processes are primarily responsible for three tasks:

- *CPU core virtualization*, for running guest code inside the virtual machine. This can be implemented like processor virtualization on traditional systems; e.g., via trap-and-emulate (if the cores support it), binary translation, or para-virtualization.
- *SMP interconnect simulation*, so the virtual CPUs effectively appear as in a normal SMP system. Since the many-core does not support all necessary features directly, the hypervisors must intervene and emulate them. For this, certain hardware features (like, for example second-level address translation like Intel's *Enhanced Page Tables* (EPT) [6, 7], or AMD's *Nested Page Tables* [1]) can also be used to reduce runtime overhead.
- *Device I/O redirection*. In the most simple model, the many-core's network can be configured such that devices are accessible from all cores, and the virtual machine can access them directly. If this is not possible (because, for example, a device's DMA operations would interfere with the memory management performed by the hypervisors), I/O can be intercepted by the hypervisors and redirected to an appropriate emulation process on one of the participating LV1 instances.

For the current report, we concentrate on memory management. This mainly relates to the first (page table manipulation, which interacts with MMU virtualization) and second category (for the memory consistency model).

4 Memory Management in the Hypervisor

4.1 Memory Consistency Model

The memory consistency model of an x86-SMP is not formally defined in the processor manual [6, 7]. Instead, it is described in informal prose, as the result of several interacting settings and mechanisms from the processor core and support hardware – like memory caching attributes, cache organization, write-combining buffers, instruction reordering and so on. Owens et al. have interpreted the descriptions from the manuals and constructed a formal model, which was later refined by the same authors to better match the behavior of real processors [11]. However, we did not base the memory consistency model implemented by our hypervisor on their work; we use the natural consistency of the underlying hardware instead, where possible, and simple sequential consistency [9] otherwise, for reasons outlined below.

By construction of our many-core, each multi-core island already fulfills a memory consistency model considered "correct" for the guest operating system; this is guaranteed by the backwards compatibility property mentioned above. Therefore, when running multiple hypervisor processes on one multi-core island, each of them providing one virtual processor to the guest, and each virtual processor having mapped a particular page, the guest also behaves correctly: the hypervisors are not involved in any memory access after page tables have been set up, and the processors just perform regular memory accesses under their "native" consistency model. For these cases, a formal treatment is not necessary.

4.2 Coherency Simulation

Now consider interacting multi-core islands. Because, by construction, they only communicate if explicitly requested by software, it is an error to map the same page of system memory to both islands. All local caches on the islands will work as usual, but due to missing coordination (or *cache coherency*), cache lines may become stale if any island is allowed to perform writes. Even worse, if more than one island is allowed to write, conflicting modified cache lines can be produced, and the eventually-occurring write-backs can lead to data loss.

To solve these issues, our system simulates coherency in software, by coordinating changes to page tables. This is comparable to *distributed shared-memory* (DSM) known from cluster computing, but with an important difference: we do not need to send *page contents* over the interconnect ourselves, as each island's memory interface and protocol converter will generate corresponding packets automatically. We only need to concentrate on coordinating *content of page tables*.

Shared memory support is based on two principles: a state vector for each shared (physical page) in shared memory, and a remote page-table invalidation mechanism. The state vector denotes, for each participating islands, which accesses are allowed to the page without inducing the possibility of stale cache entries. If an island requires another access that it has not been granted, it requests the owner(s) of the page to invalidate their own page tables and potential local copies in their caches; the requesting island is only to allowed to map the page once it receives confirmation that all other conflicting mappings have been removed. Figure 3 shows the state transition diagram for each page and island.

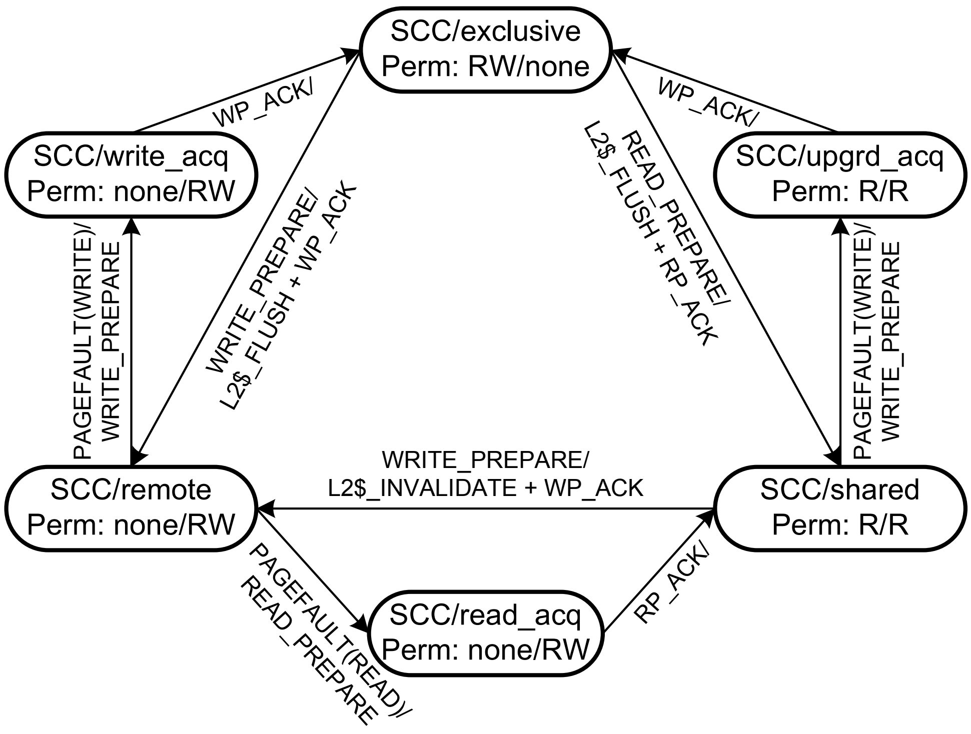


Figure 3: Shared Page States

As discussed above, arbitrary page mappings are valid, as long as no two different islands have mapped a page for write access. Concurrent reads do not present an issue, because the corresponding cache lines will never be marked *modified* and thus never trigger a write-back. Therefore, read access can be acquired by an arbitrary number of islands at the same time. However, write accesses result in modified cache lines, so they potentially present two problems:

1. A cache line may reside in *modified* state on one island, and in *unmodified* (either *shared* or *exclusive* in terms of the MESI protocol, depending on whether the cores reside on a single- or multi-core island) state on another. After eviction of the unmodified cache line and write-back of the modified one has occurred, the second island may be able to observe the changed value after a non-deterministic amount of time and other local behavior.

2. A cache line may reside in *modified* state on two different islands, which can result in a lost update to main memory, or (if an observer's timing is right) even a non-deterministic rollback of a memory change.

To solve both problems, we require that, once an island attempts to acquire write access, *all* other islands cease using the page. That is, their local page tables are modified to mark relevant entries as invalid, and caches are flushed and invalidated, before the acquire-write request is granted. Similarly, when an island requests read access to a page another island was allowed to write, its page tables are modified to write-protect the page, and cache contents are written back to memory, before the request can be granted.

Our shared memory support is integrated into the LV1 operating system, so it works transparently for all applications running under this kernel, including the hypervisor container and I/O emulation processes. If the process uses only regular memory access instructions to access the page, no further cooperation is possible – this is the case for most operations, even when the hypervisor needs to access guest memory. There is one notable exception, though: processes manipulating their own page tables.

In order to support MMU virtualization for the guest, the hypervisor container may need to create its own page tables if the underlying processor core does not support nested address translation. To prevent invalid cache lines from occurring, these page tables must be incorporated into the coherency mechanism in the same way the regular (OS-maintained) ones have been; specifically, offending mappings must be removed when a remote island requests conflicting page accesses, and such access must be acquired from remote islands before inserting page table entries and allowing the virtual CPU to run. For this, we integrated a callback interface that allows arbitrary kernel-mode code, to interact with the coherency driver.

4.3 MMU Virtualization

Several algorithms have been developed for MMU virtualization, partially because x86 processors until a few years ago did not support hardware virtualization assists, so also did not provide nested paging. The purpose of these algorithms is to map a two-level page translation scheme to the one level the hardware supports natively. In the two-level scheme, the “outer” (lower) level represents the address translation of the host (which runs the hypervisor as an application), whereas the “inner” (higher) level represents the address translation of the guest. This two-level scheme is shown in figure 4.

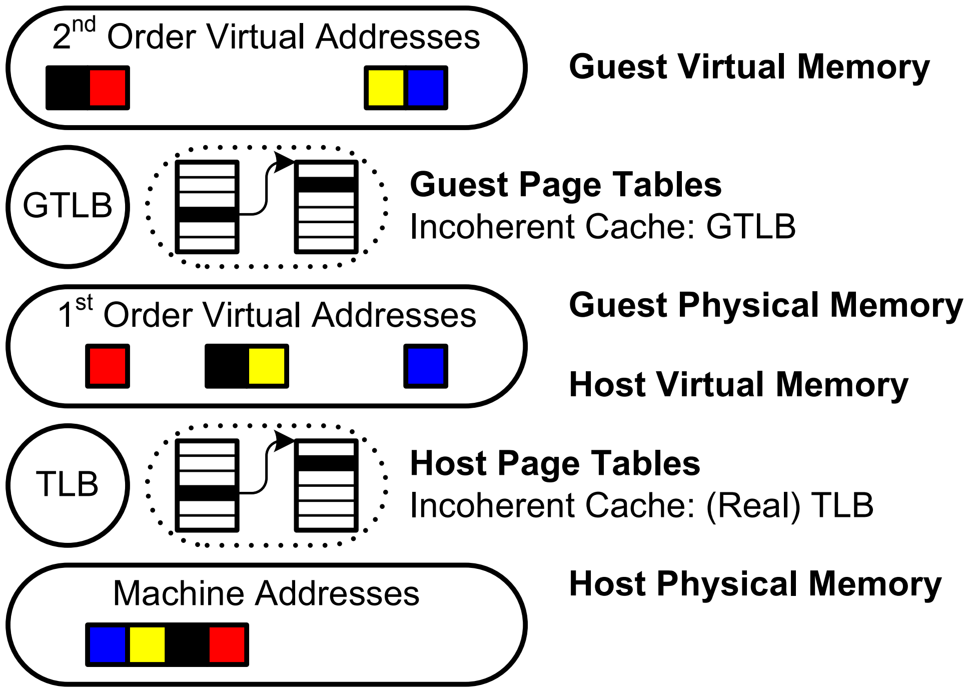


Figure 4: Two-Level Address Translation with TLBs

Each level of page tables has an associated (physical or virtual) *Translation Lookaside Buffer* (TLB). The physical TLB is controlled by the host, and must be invalidated according to the processor manual; it is an incoherent cache for the real page tables. For example, on multi-core islands, if certain changes to page tables are made by one core, but another core is referencing these page tables as well, it may have conflicting information in its TLB. Therefore, the operating systems must perform a *TLB shootdown* whenever such a change is performed.

The need for TLB shootdowns lies in the processor architecture, so relevant code is present in *both* the host and guest operating system – at least if the latter one is able to run on non-virtualized machines as well. Our method for MMU virtualization, which will be discussed later, uses this to its advantage.

4.3.1 Classic MMU Virtualization Approaches

Of the several MMU virtualization approaches, the *Emulated TLB* algorithm is probably the most simple one. As the name implies, it emulates the guest TLB, by changing the real page tables as the guest runs. The virtual CPU starts with an empty (real) page table. When page fault occurs, the hypervisor interprets the guest’s page tables, and constructs corresponding mappings as needed. When the guest switches to another address space, the real page table is cleared again; therefore mimicking the operation of a hardware TLB.

The main downside of the emulated TLB is a large number of “hidden” page faults: page faults that occur due to virtualization, not due to non-existent or restrictive page table entries installed by the guest. Hidden page faults constitute overhead, so approaches for eliminating them have been proposed and implemented. We mention two of them *Shadow Page Tables*, and the approach pioneered by XEN [2].

For *Shadow Page Tables*, the hypervisor saves a set of page tables, and fills them as needed from page faults. However, when the guest switches to another address space, the previous page table is not cleared. Instead, it is retained and can be reused later, if the guest decides to switch back to this address space. This saves hidden page faults, but has another major drawback: the guest may, now that its previous page table is unused, make arbitrary changes to it without informing the hypervisor. Therefore, the contents of the guest page table and corresponding shadow page table may run out of sync, unless the hypervisor recognizes such changes and reacts accordingly. The VMware hypervisor uses a technique called “traces”: it marks guest pages containing page table data as read only, so any write attempt by the guest will trigger a page fault. The page fault handler then recognizes that the fault is due to a page table change, and can take down the shadow page table if it still exists.

XEN [2], if running on hardware without virtualization support, uses a different approach. If running under XEN, the guest needs to be aware of the real (machine) addresses of its pages, and it computes the outer page mapping itself. The guest OS then simply installs the real machine address in its page tables; the hypervisor simply needs to check that these addresses are in the desired range (thus preventing the guest from accessing memory it is not supposed to), but it can save on costly software page table walks.

4.3.2 Our Approach: Cooperative Shadow Page Tables

For our many-core hypervisor, we use a different MMU virtualization scheme that is based on shadow page tables. We call it “Cooperative Shadow Page Tables”, because it is based on the concept shortly described above, but also requires cooperation between the hypervisor and guest.

Cooperative Shadow Page Tables also involves a set of cached real page tables in the hypervisor, which are activated when the guest switches to the corresponding address space. However, invalidation of these caches is handled differently. Following the assumption that the guest is an SMP operating system, it is aware of the (G)TLB that functions as an incoherent cache of its page tables. We now intercept the TLB flushes of the guest to know exactly when certain shadow page tables need to be invalidated.

On a real machine, the guest would need to perform TLB shootdowns if it modified page tables in certain ways. This involves a message (or, more specifically: an IPI) to *all* CPUs that use the page table. Because this may influence a potentially very large number of CPUs, and IPIs are relatively “expensive” runtime-wise, the corresponding code paths in operating systems are highly optimized. No IPI is sent out unless absolutely necessary, which means *for real machines* the hardware is actively using the target page table; i.e., the corresponding address space is active.

In our approach, we now modify the notion of when an address space is considered *active* in our guest (LV2) OS. On regular machines, an address space is active if it is referenced by the hardware; that is, if the hardware page-table walker can access it, because the page table base address is stored in the corresponding control register. Therefore, a context switch from A to B on processor X happens in three steps:

1. Mark address space B as *in-use* by processor X.
2. Store base address of address space B in control register.
3. Mark address space A as *not in-use* by processor X.

The mark operations are carried out in this order to prevent a possible race condition: modifying the address space on another core between unmarking and changing the base address. Therefore, up to two address spaces may be marked in-use by any processor at any point in time.

For our guests, we perform steps 1 and 2 as usual, but omit step 3. Therefore, when a virtual CPU switches to a new address space, it is marked to be in-use on the target processor, but the old address space still remains as being marked this way. This is the desired operation, because the corresponding hypervisor may still have cached the shadow page table for the old address space.

If the address space is later changed in a way that would require a TLB flush, the TLB shootdown code examines the list of processors on which it is active, and informs them to flush their cached mappings. Consequentially, in our architecture, the flush would be executed on all processors that had the address space active somewhere in the past. Therefore, all hypervisors are guaranteed to be informed whenever the shadow page tables need to be invalidated.

Up to now, changing address spaces works as on regular hypervisors, without any "cooperation" with the guest. The "cooperative" part of the approach's name handles a certain subtle point: as of now, address spaces can only be marked active, but never unmarked. Therefore, even if a hypervisor decided to delete a shadow page table, it would still receive TLB flush IPIs. To prevent these unneeded IPIs from occurring, we inform the hypervisor, during the context switch, where the *in-use marking area* of the guest is located. Having this knowledge, a hypervisor can then perform unmarking when it deletes the corresponding shadow page table. This eliminates any unwanted and unneeded flush IPIs, safe the ones occurring due to races between guest's modifications and hypervisor page table deletes.

5 Related Work

Our RockyVisor is an extension of lguest [13], which has been developed by Rusty Russel. It is a minimal, but fully-functional hypervisor that is included with the Linux kernel. It supports para-virtualization only, so the guest kernel must be changed accordingly; guest implementations exist for Linux and Plan 9. Unlike XEN [2] or KVM [8], lguest is meant as a platform for research and experimentation; specifically, lguest favors readability of the hypervisor code over performance wherever possible.

vNUMA [3], developed by Matthew Chapman as part of his Ph.D. thesis, is a distributed hypervisor for IA-64 processors that simulates an SMP system on networked workstations, using Gigabit Ethernet as node interconnect. NEX [15] by Xiao Wang et al. is a similar effort based on the open-source XEN hypervisor, but requires hardware extensions for virtualization. Versatile SMP [14] by ScaleMP is a commercial product that claims to support up to 1024 processors (with up to 8192 cores) on standard computers, interconnected via Infiniband, in a virtual SMP system.

Finally, *MetalSVM* [10] is another project aiming at hypervisor-based SMP on the SCC. MetalSVM itself is a small operating system kernel, for which Jacek Galowicz has ported the lguest driver and user-mode code as part of his B.Sc. thesis [4]. It is therefore comparable to the RockyVisor, both being Type 2 VMMs that run on top of an underlying operating system. The main difference, at the time of writing, is that the RockyVisor is functional for 2-way SMP, whereas the MetalSVM virtualization layer only supports single-processor guests, just like the lguest hypervisor.

6 Conclusion and Future Work

We have presented a model of a many-core processor that consists of smaller multi- and single-core islands, interconnected by an on-chip network. Whereas each individual processor core follows an established architecture (like IA-32) and can execute existing software on its own, such software cannot run on the entire system, due to missing hardware features like memory coherency.

To bridge this gap, we presented an architecture of a distributed hypervisor, named *RockyVisor*, that can not only be used to run an application, but an entire shared-memory symmetric multi-processing (SMP) operating system on the system. Because an operating system for the predecessor architecture now runs on the new many-core processor, this also allows arbitrary applications to work on the new system. We have shown details on memory management and virtualization of the processor's memory-management unit in such a system, and outlined our *cooperative shadow page table* mechanism, in which the hypervisor works together with the guest operating system to coordinate views of processes' address space across the many-core processor.

Our architecture also allows for a peaceful coexistence of applications written for the old and new machine. Existing applications do not need to be ported or re-written for the new cluster-like environment, they can simply run unchanged in a virtual machine. Furthermore, because we use a hosted VMM approach, native applications can run side-by-side with older ones in the new environment.

A prototype of our hypervisor has been implemented on the Intel Single-chip Cloud Computer (SCC) [5]. In its current state, the RockyVisor runs on two single-core tiles of the SCC. It uses a Linux kernel as its host (Level 1) operating system that has been extended with our memory coherency driver. The guest (Level 2) operating system is another Linux kernel, extended with a corresponding sub-architecture layer that implements the para-virtualization interface for the RockyVisor.

Further work involves finalizing the I/O emulation and redirection infrastructure, as well as a specification of the para-virtualization interface between the guest and hypervisor.

References

[1] Advanced Micro Devices, Inc. AMD-V Nested Paging, July 2008.

[2] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *SIGOPS Oper. Syst. Rev.*, 37:164–177, October 2003.

[3] Matthew Chapman. *vNUMA: Virtual Shared-Memory Multiprocessors*. PhD thesis, Computer Science and Engineering, The University of New South Wales, 2008.

[4] Jacek Galowicz. Design and Implementation of a Virtualization Layer for the Operating System Kernel “MetalSVM”. Master’s thesis, 2012.

[5] Jason Howard, Saurabh Dighe, Yatin Hoskote, Sriram Vangal, David Finan, Gregory Ruhl, David Jenkins, Howard Wilson, Nitin Borkar, Gerhard Schrom, and et al. A 48-Core IA-32 message-passing processor with DVFS in 45nm CMOS. *2010 IEEE International SolidState Circuits Conference ISSCC*, 9:58–59, 2010.

[6] Intel Corporation. *Intel Architecture Software Developer’s Manual, Volume 3a: System Programming*, 2010.

[7] Intel Corporation. *Intel Architecture Software Developer’s Manual, Volume 3b: System Programming*, 2010.

[8] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, and Anthony Liguori. KVM: The Linux virtual machine monitor. In *Ottawa Linux Symposium*, pages 225–230, July 2007.

[9] Leslie Lamport. How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programs. *IEEE Trans. Comput.*, 28:690–691, September 1979.

[10] Stefan Lankes. MetalSVM: A Bare-Metal Hypervisor for Non-Coherent Memory-Coupled Cores. `http://www.lfbs.rwth-aachen.de/content/metalsvm`, 2011.

[11] Scott Owens, Susmit Sarkar, and Peter Sewell. A better x86 memory model: x86-TSO (extended version). Technical Report UCAM-CL-TR-745, March 2009.

[12] Gregory F. Pfister. *In search of clusters*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2 edition, 1998.

[13] Rusty Russel. lguest: Implementing the little Linux hypervisor. In *OLS '07: Proceedings of the Linux Symposium*, volume 2, pages 173–178, June 2007.

[14] ScaleMP. Versatile SMP (vSMP) Architecture. `http://www.scalemp.com/architecture`.

[15] Xiao Wang, Mingfa Zhu, Limin Xiao, Zhonglin Liu, Xiao Zhang, and Xiangshan Li. NEX: Virtual Machine Monitor Level Single System Support in Xen. In *International Workshop on Education Technology and Computer Science*, volume 3, pages 1047–1051, Los Alamitos, CA, USA, 2009. IEEE Computer Society.

Interleaving Programming Tasks: Challenges for Interruption Handling in Programming Environments

Marcel Taeumel

Software Architecture Group
Hasso-Plattner-Institut
marcel.taeumel@hpi.uni-potsdam.de

Whenever programmers stop working on one task to start or continue another, there is a time-consuming overhead. Such task switches are often initiated by information needs to gain a better understanding about the software system. If questions are answered, the interrupted task will be resumed hence switching will happen again. Thus, designers of programming environments are challenged to either reduce those interruptions or support task resumption efficiently.

We propose several ideas considering interruption handling in programming tools and describe our concept for environments that reduces interruptions and supports task resumption for code-centric comprehension activities. Our current research addresses direct manipulation environments and tool-supported software classification.

1 Introduction

Programming is a difficult activity that involves huge mental effort for transforming intentions originated in natural human language into a representation that can be processed by computers [4]. When doing so, working with computer interfaces does not just mean to write source code in a programming language, but also to interact with interfaces of programming tools and environments.

During programming tasks, programmers need to access, comprehend, and apply different kinds of information [12]. For example, this could be abstract documentation to a module interface, concrete information about program behavior, or more general system design rationals. Unfortunately, such information is often not immediately available. Accessing it involves several sub-steps and hence interrupts the current task.

Interruptions increase the time to complete tasks. Altmann et al. [1] described two delays that occur in case of an interruption: (1) the interruption lag and (2) the resumption lag. Before an interrupting task can be started, programmers need time to finish the current step and prepare for resumption, e.g., take some notes [14]. After the interrupting task is finished, programmers need time to regain focus for the interrupted task along with reminding task-related details and understanding existent tool state.

This is where programming tools can support programmers:

- Provide more direct access to information access to make task switching be less interrupting, i.e., reduce interruption lags
- Support immediate interruption recovery for any kind of tool-driven activity that shifts programmers' focus to a different kind of information, i.e., reduce resumption lags

Traditional programming environments like Eclipse[1] already connect various sources of information. For example, tooltips embed valuable documentation, context menus provide links to associated source code artifacts, or panels show items in shared repositories (e.g., bug trackers, version control systems). Thus, tool developers are improving static information access in traditional environments for quite some time. However, programmers often have difficulties in understanding program run-time behavior and hence require more than just access to static information. Unfortunately, accessing information about program run-time is time-consuming and interrupting (e.g., using breakpoint debugging) [15].

Current research reveals many new ideas for tools that aim for directly answering relevant questions [25] and hence satisfying information needs of programmers [12]. Mostly, those questions target program run-time. Prominent examples include direct support for Why-questions [13], direct access to run-time context using tests [26] and ideas for environments that immediately reflect run-time effects as source code is written[2]. By doing so, the chance for interruptions, which involve a noticeable overhead between task switches, is reduced because the feedback loop is shortened.

Other research projects try to improve interruption recovery by providing support for externalizing the mental model [31] of programmers into the environment. This compares with manual note taking but should shorten the interruption lag and the resumption lag. For example, using spatial context can be very supportive when reminding tasks [17]. Thus, Bragdon et al. allow programmers in Code Bubbles [5] to freely arrange source code artifacts. Kersten et al. propose a degree-of-interest model [11] to only show task-relevant artifacts and avoid cluttering of information on the screen. This also shortens the amount of time that is needed to recover from an interruption.

However, there are still many shortcomings in addressing interruptions. Programming environments do not sufficiently reflect the conceptual model that programmers have in mind when thinking about object-oriented programs. On the one hand, this makes information access often feel interrupting and on the other hand interruption recovery time-consuming.

Thus, our current research focuses on the following topics:

- A feasible approximation of the conceptual model that programmers have in mind when thinking about object-oriented programs
- A concept for programming environments that reflects this model and hence reduces noticeable interruptions while supporting interruption recovery

[1]`http://www.eclipse.org`

[2]`http://worrydream.com/LearnableProgramming`, accessed on 2012-10-04

The remainder of this paper is structured as follows. Section 2 explains the gap between programmers' goals and programming environment interfaces. It also describes what direct information access means and when interruption recovery becomes difficult. Section 3 describes our current research progress in the field of interruption handling in programming environments. A new concept for accessing and arranging static and dynamic program information should address several challenges for handling interruptions when comprehending programs in a code-centric fashion. Finally, section 4 sketches open hypotheses and next steps.

2 Why Interruptions are Time-consuming

Programmers are frequently interrupted due to information needs. They cannot know every detail about a system part but need to query available information sources to verify assumptions and extend system knowledge. There are two reasons, which make those queries be interrupting and time-consuming: (1) programmers need to map their information goals to tool-specific intentions thus involving unnecessary task switches and (2) programmers often do not notice task switches thus missing to prepare optimized task resumption.

2.1 False Assumptions

If programmers should describe programming environments, they would think of a set of tools that allow for finding, modifying, executing, and sharing source code while exchanging intermediate results. They would think of tasks like enhancing the program with a new feature, debugging the program to fix a defect, or refactoring the code base to reduce technical debt [8]. Designers of such environments create an appropriate conceptual model (design model) that captures all common scenarios and implement this model. Having this, programmers use the implementation and form their conceptual model (user's model) using the given features and inferring purposes, which hopefully match the designer's ones. As Norman [18, pp.189] explained, designers only communicate their models to users via such a materialized system image. Having this fact in mind, programming environments like Eclipse are improved continuously with each new release considering user feedback.

However, programmers think of object-oriented programs in a different way, thus rendering the basic (and stable) assumptions of traditional programming environments invalid. Programmers do not think of writing statements into files. They do not think of setting breakpoints to interrupt control flows for state inspection. They do not think of committing code snippets into a repository for shared use. They think of objects. Before any code is written, programmers think about collaborating objects to achieve their goals–at least when using object-oriented languages. In general, programmers think in terms of the *language model* underlying the programming language that is appropriate for the given problem. Hutchings et al. [10] coined the term *semantic distance*, which describes the mismatch between the user's goals and the features an interface provides. To overcome this distance, the user has to restate the goals into fine-granular

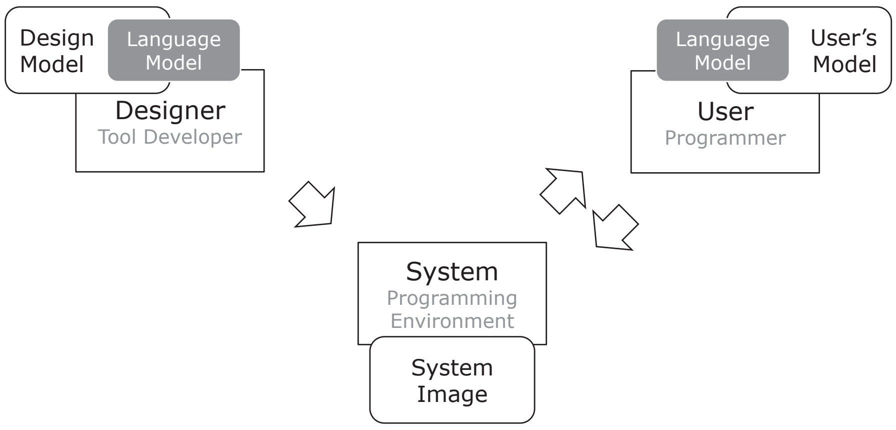


Figure 1: Designer and user only communicate through the system. Fortunately, the language model is known to both the designer and the user. Thus, the design model should include the language model to create more usable and less interrupting programming environments. Graphics adapted and modified from [18, p.190].

intentions that match with the interface. This mapping is time-consuming and likely to cause an interruption. Hence, the designer of programming environments should include the language model in the design model as illustrated in Figure 1. Programmers could then be able to interact with tools in a less interrupting manner by transferring thoughts into actions more directly.

2.2 Unnoticed Task Switches

Programmers have difficulties in noticing task switches triggered by sub-conscious or unintended actions. Actually, they are aware of a distracting task that approaches the current one if its signals are obvious: the telephone rings, the co-worker knocks the door. There is time to prepare to optimize task resumption later on. However, programming environments introduce task switches associated with information access, which are not necessarily noticed by programmers and hence miss preparation for interruption recovery, e.g.:

Sub-conscious Whenever a test fails, the experienced programmer instantly switches mentally into a "debugging mode" and thinks about where to set breakpoints and inspect program state–without noticing that she starts to forget how far the new feature was implemented.

Unintended The programmer just wanted to run all tests, which normally takes about 10 seconds, before committing the changed source code, but it took too long and she went for a coffee after waiting some minutes–without noticing that she starts to forget what kind of change she exactly made.

It is helpful to know in advance, whether an action will trigger an interrupting task and hence whether to prepare for optimized resumption. Especially if the state of the programming environment does not align with the programmer's mental model of the current task, unprepared interruption recovery can be time-consuming. Unfortunately, this is the case in many situations where programmers need to explore available information–documentation, source code, program behavior–to verify assumptions and extend system knowledge.

Fleming et al. [9] applied the Information Foraging Theory (IFT) to programming tasks that cover debugging, refactoring and code reuse. This theory is about humane search behavior: evaluating information patches and navigating valuable links (known as cues) between patches to achieve an information goal. The difficulty lies in estimating cost and value of cues and choosing the helpful ones. The authors use the IFT and its applications to try to generalize requirements for supportive information processing systems by introducing common patterns.

Programming environments should to consider the IFT to better guide programmers when exploring information. Environments should either support quick interruption recovery automatically or make programmers aware of interrupting task switches to aim for a semi-automatic trade-off.

2.3 Opportunities for Programming Environments

Programming environments should consider important questions about interruption handling and the nature of interleaving tasks caused by the vast amount of supportive information, which makes programming a difficult activity requiring high mental effort:

- When do interruptions occur?
- How to make programmers aware of unnoticed task switches?

- What do programmers externalize before switching tasks?
- How do programmers resume interrupted tasks?

- How to reduce the interruption lag with tool support?
- How to reduce the resumption lag with tool support?

Using research results about humane working and information processing behavior [17] [31] [9], we believe that designers of programming environments should solve the following problems:

- Support scenarios that align with programmers' mental model of the program and its building blocks, i.e., consider the language model (see Figure 1)
- Provide direct access to information using simple queries, i.e., consider temporal, spatial, and semantic immediacy [30] [9, "Cue Decoration"]
- Allow for free collection and arrangement of information to support interruption recovery using spatial context [17] [9, "Gather Together"]
- Avoid cluttering of information to support interruption recovery [17]
- Prepare for frequent focus changes and design appropriate task reminders that form a trade-off between reminding and interrupting [17]

Hutchings et al. [10] explain that, at best, programmers do not recognize using distinct tools, but are engaged with a direct manipulation interface, which allows for triggering actions to directly achieve desired goals without distracting intermediate steps. Having this, the presence of tools becomes transparent to programmers and thus programming feels less interrupting.

3 A Less Interrupting Programming Environment

In this section, we present our current research progress. It is a concept for programming environments that reduce the number of context switches when accessing run-time information during program comprehension activities. Within this environment, programmers stay focused in their task-/problem-oriented thinking and do not have to make tool-driven decisions that are distracting and hence time-consuming. A clear, consistent user interface abstracts from technical details and integrates with programmers' activities in a user-centric way by directly supporting answering questions in understanding whenever they arise.

Unger et al. identified three different types of *immediacy* [30] that programming environments should support to keep programmers focused on their task: (1) Temporal immediacy addresses the delay between performing an action and receiving a feedback in the environment, (2) spatial immediacy addresses the visual distance of related information on-screen, and (3) semantic immediacy addresses the number of user interactions (e.g., mouse clicks) needed to access a desired information.

At first, we address the problem of interruptions through tool-driven transformation of intentions by explaining our notion of run-time information and how to capture and provide it automatically. This corresponds to temporal immediacy. Then, we address the problem of externalizing thoughts in the environment and present a visualization based on a scrollable tape with embedded editors to display needed information in a simple, clear, and predictable way. Hence, spatial and semantic immediacy are ensured.

3.1 Capturing Example Run-time Information

Our notion of run-time information encompasses exemplified program behavior to support code-centric comprehension tasks. In the strict sense, we want to collect information about method calls–namely object states (i.e., callers, callees, arguments, results) and behavioral traces (i.e., call trees). We do not target concrete debugging scenarios where defects have to be found; the awareness of specific failing tests would be important for that [19]. Furthermore, examples of indifferent origin should help to map abstract source code to concrete program behavior and hence to verify and extend system knowledge at an exemplary but valuable level.

Tests are well-suited program entry points that produce representative control flows and hence valuable information about program behavior [26] [20] [19]. In fact, writing tests is known to be supportive during software development [3] [2]. By having these defined program entry points, dynamic analysis techniques [6] [22] [20] are able to capture run-time information without requiring programmers' attention. Programmers can focus their comprehension activity and query run-time information directly when needed–assuming that tests cover the method of interest.

There is a shift of responsibility from programmers to the environment considering run-time information access. Hence, tests need to be deterministic. Until now, reproducibility of run-time information has been more important from a programmer's perspective than from a technical perspective. Programmers may need to recall the same information several times during step-wise refinement of comprehension questions depending on their mental capacity. In our concept, programmers do not have to think of ways to achieve reproducible results anymore but implementations of such environments do have to. Direct information access means that programmers are aware of the number of, for example, mouse clicks they have to perform and hence they will notice how long the *response times* are, until the desired information becomes visible on the screen. The size of this time frame has an impact on when to lose focus on the current activity[3]. The problem is that dynamic analysis can be expensive considering time- and memory-consumption [16]. Hence, implementations of our concept need to pay attention to performance issues and may consider *partial tracing* approaches [28] [21] [20]. Therefore, reproducible results rely on deterministic tests.

All kinds of program comprehension questions that we address can be reduced to automatic capturing, querying, and post-processing of object states and behavioral traces in the context of a specific method call. Post-processing varies from simply accessing example data to aggregating all information for providing ranges of possible variations in a broader scope.

3.2 Displaying Source Code and Run-time Information

The visualization part in our concept tries to mask the presence of dedicated subtools and hence tries to combine source code and run-time information in a way that directly integrates with programmers' comprehension activities. For this, the desktop metaphor, which tries to imitate real-world artifacts and activities in graphical user interfaces, is considered as inappropriate because programming environments have no representations of artifacts or activities in the real-world [23].

We put each self-contained portion of information (e.g., a class' methods, an object state, or a call tree) into one rectangular view–called *editor*. Editors are arranged on a horizontal unbounded *tape* side by side. Connections between visible information are displayed via *overlays*.

Horizontal Tape Modern wide-screen monitors offer an image ratio of 16:10 or 16:9. Having this, the primary (since largest) screen axis is the horizontal one and programmers need to think about how to make efficient use of the available screen space. Since source code lines are rarely longer than 100 characters, this kind of information tends to spread along the vertical axis leaving much whitespace to the left or to the right. Traditional programming environments surround this central code area with

[3]Shneiderman et al. [24, p.445] argue that a response delay of 1 second does still not distract users from simple and frequent tasks.

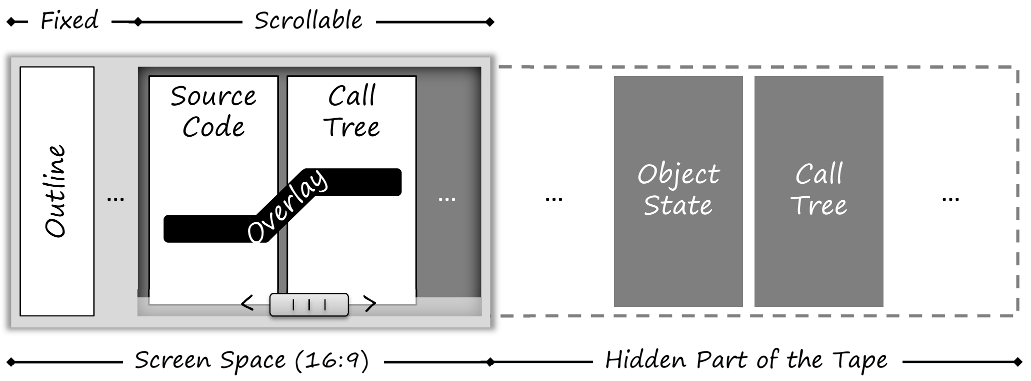


Figure 2: Our concept for integrated programming environments. Editors are arranged from the left to the right either in a fixed portion of the screen space or on a horizontal boundless tape that is accessible through a scrollable container. Overlays visualize relations between editors. Any kind of editor can be placed multiple times on the tape.

freely-arrangeable views for, e.g., system outlines, documentation, or run-time information to make use of whitespace. This leaves both screen dimensions open for different kinds of information that programmers may have to look for.

Our concept proposes a *horizontal unbounded tape* that is embedded into a scrollable area as shown in Figure 2 to make efficient use of wide-screen monitors. On this tape, editors are freely-arrangeable from the left to the right. This assigns a clear level of information granularity to each screen axis: the horizontal is reserved for different kinds of information (e.g., source code, call trees, object states) and the vertical exposes details for each kind (e.g., chronologically ordered call nodes). Hence, programmers should be able to recall information more quickly and thus reducing the resumption lag when recovering from an interruption.

Besides the tape, part of the screen space is reserved for editors that should always be visible: the *fixed area*. Having this, the environment organizes information in a two-level hierarchy: (1) Is the information always visible or potentially hidden? (2) Is the information to the left or right of the current view? Still, these constraints allow for an unrestricted exploration of the system while avoiding programmers to get distracted when positioning information on the screen. Additionally, new editors that are about to appear can be positioned in a more predictable manner for programmers. Hence, navigating to supporting information should feel less interrupting.

Simple Editors Each editor contains details for one primary kind of information. This can be displayed in a central list, table, tree, or other visualization. For example, class editors can show a list of open methods, system overviews can show tree-like outlines of captions, call trees can show concrete behavioral traces, object explorers can compare object states before and after an exemplary method call.

In addition, pop-up menus and tooltips can reveal other (secondary) kinds of information that are directly associated. For example, the tooltip for each node in a call tree can show the called method's source code. Having this, programmers can directly connect abstract source code with concrete run-time information and hence verify/extend their current system knowledge.

Different to views in traditional programming environments, all editors in our concept are of equal priority for program comprehension activities. Editors for source code are not more important than editors for run-time information and vice versa.

Connecting Overlays Programmers open and arrange editors on the tape during comprehension activities. Thus, there is a relation between open editors corresponding to the navigation history and hence the programmers' mental model of the system. Our concept uses *overlays* as a third technique to illustrate those relations and hence allows programmers to recall information more quickly while reducing the cognitive load and the task resumption lag.

3.3 Interacting within the Environment

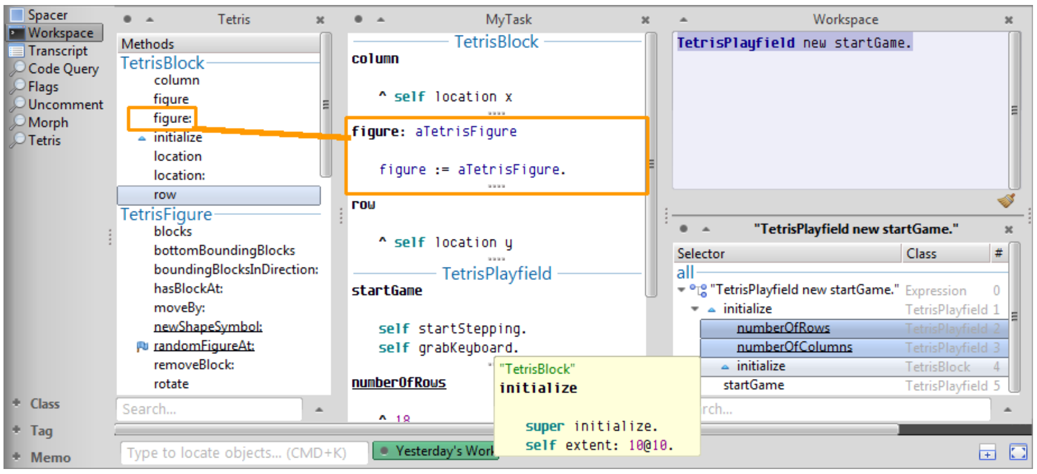
In our concept, direct access to run-time information means to provide supportive information to an arising comprehension question with as few user interactions (e.g., mouse clicks) as possible. Having this, we believe that programmers will keep on exploring the program without noticing the environment as a distracting intermediate. To achieve this, our concept considers common *starting points* for comprehension activities and *simple queries* using a consistent vocabulary in pop-up menus to navigate between different kinds of information.

Starting Points Code-centric program comprehension starts with source code reading and looking for promising beacons [31]. This could mean to browse overviews of system parts or detailed sources of methods. To achieve this, editors that show the appropriate information need to be directly accessible using search mechanisms. In the first place, a text-based search is sufficient because programmers start with looking for identifiers (e.g., class names or method signatures) that seem to correspond with domain concepts when exploring systems [25]. When getting more knowledgeable with the system, this search could be extended to make use of run-time information.

However, our concept tries to reduce the complexity of queries. Programmers should not have to translate rather complex comprehension questions into the complicated vocabulary of environments. We want to keep the semantic distance [10] low for code-centric program comprehension tasks.

Simple Queries Simplicity is important when directly accessing run-time information. At first, programmers need to transform their question into one out of three elementary purposes: Browse Code, Explore Object, View Trace.

This transformation is supposed to be straightforward because programmers are aware of the language model that underlies object-oriented programs as illustrated in Figure 1. This simple vocabulary should be visualized with *pop-up menus* and integrated into all editors consistently. By doing so, programmers are free to decide whether, for example, focused pieces of run-time information benefit from additional run-time data or source code.



```
column

   ^ self location x

figure: aTetrisFigure

   figure := aTetrisFigure.

row

   ^ self location y

startGame

   self startStepping.
   self grabKeyboard.

numberOfRows

   ^ 18
```

```
"TetrisBlock"
initialize

   super initialize.
   self extent: 10@10.
```

```
TetrisPlayfield new startGame.
```

Selector	Class	#
"TetrisPlayfield new startGame."	Expression	0
initialize	TetrisPlayfield	1
numberOfRows	TetrisPlayfield	2
numberOfColumns	TetrisPlayfield	3
initialize	TetrisBlock	4
startGame	TetrisPlayfield	5

Figure 3: Screenshot of our prototypical implementation called VIVIDE. The column-oriented layout allows for simple collection and arrangement of important information. Overlays visualize navigation history. Static and dynamic information are comparable side-by-side and hence should reduce interruption and resumption lags.

4 Conclusions and Next Steps

Programmers experience frequent interruptions due to switches between unfinished tasks during programming activities. Since programming tasks interleave, the occurring lags form an overhead that disturbs the workflow. Most interruptions are initiated by the programmers themselves when accessing required information. Thus, programming tools could either simplify this access to avoid context switches completely or ease the steps between interrupted and interrupting task to minimize overhead.

We implemented a prototype of our concept for programming environments–called VIVIDE [27]. It supports code-centric comprehension activities by simplifying access to run-time information (see Figure 3). For evaluation purposes, students created a small Tetris game with it. Observations revealed that programmers' ways to explore object-oriented programs are still different to how editors present information. Additionally, cluttering of information will become a problem if the environment is overpopulated with source code, object explorers, and call trees.

At the moment, we are investigating how to better support querying static and dynamic information to provide immediate starting points in the environment and hence support interruption recovery. This also involves the entire field of *software classification* [7] because modularity issues are still present in programming languages [29] and tools [11]. Actually, programmers are way more flexible in grouping artifacts mentally, hence forming abstractions to ease comprehension, than they can do with such tools or languages.

References

[1] E. M. Altmann and J. G. Trafton. Task Interruption: Resumption Lag and the Role of Cues. In *Proceedings of the 26th Annual Conference of the Cognitive Science*, 2004.

[2] K. Beck. *Test-driven Development: By Example*. Addison-Wesley, 2003.

[3] K. Beck and C. Andres. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 2004.

[4] A. F. Blackwell. What is Programming. In *Proceedings of the 14th Workshop of the Psychology of Programming Interest Group*, pages 204–218. PPIG, 2002.

[5] A. Bragdon, R. Zeleznik, S. P. Reiss, S. Karumuri, W. Cheung, J. Kaplan, C. Coleman, F. Adeputra, and J. J. LaViola Jr. Code Bubbles: A Working Set-based Interface for Code Understanding and Maintenance. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems*, pages 2503–2512. ACM, 2010.

[6] B. Cornelissen, A. Zaidman, A. van Deursen, L. Moonen, and R. Koschke. A Systematic Survey of Program Comprehension Through Dynamic Analysis. *IEEE Transactions on Software Engineering*, 35(5):684–702, September/October 2009.

[7] K. De Hondt. *A Novel Approach to Architectural Recovery in Evolving Object-oriented Systems*. PhD thesis, Vrije Universiteit Brussel, 1998.

[8] M. Doernhoefer. Surfing the Net for Software Engineering Notes. *ACM SIGSOFT Software Engineering Notes*, 37(3):10–17, 2012.

[9] S. Fleming, C. Scaffidi, D. Piorkowski, M. Burnett, R. Bellamy, J Lawrence, and Kwan I. An Information Foraging Theory Perspective on Tools for Debugging, Refactoring, and Reuse Tasks. *ACM Transactions on Software Engineering and Methodology*, 2012 (to appear).

[10] E. L. Hutchins, J. D. Hollan, and D. A. Norman. Direct Manipulation Interfaces. *Human-Computer Interaction*, 1(4):311–338, 1985.

[11] M. Kersten and G. C. Murphy. Mylar: A Degree-of-Interest Model for IDEs. In *Proceedings of the 4th International Conference on Aspect-oriented Software Development*, pages 159–168. ACM, 2005.

[12] A. J. Ko, R. DeLine, and G. Venolia. Information Needs in Collocated Software Development Teams. In *Proceedings of the 29th International Conference on Software Engineering*, pages 344–353. ACM/IEEE, 2007.

[13] A. J. Ko and B. A. Myers. Debugging Reinvented: Asking and Answering Why and Why Not Questions about Program Behavior. In *Proceedings of the 30th International Conference on Software Engineering*, pages 301–310. ACM/IEEE, 2008.

[14] A. J. Ko, B. A. Myers, M. J. Coblenz, and H. H. Aung. An Exploratory Study of How Developers Seek, Relate, and Collect Relevant Information During Software Maintenance Tasks. *IEEE Transactions on Software Engineering*, 32(12):971–987, December 2006.

[15] T. D. LaToza and B. A. Myers. Developers Ask Reachability Questions. In *Proceedings of the 32nd International Conference on Software Engineering - Volume 1*, pages 185–194. ACM/IEEE, 2010.

[16] B. Lewis. Debugging Backwards in Time. In *Proceedings of the 5th International Workshop on Automated Debugging*, Ghent, Belgium, September 2003.

[17] Y. Miyata and D. A. Norman. Psychological Issues in Support of Multiple Activities. *User Centered System Design*, pages 265–284, 1986.

[18] D. A. Norman. *The Design of Everyday Things*. Basic Books, 1988.

[19] M. Perscheid, M. Haupt, R. Hirschfeld, and H. Masuhara. Test-driven Fault Navigation for Debugging Reproducible Failures. *Journal of the Japan Society for Software Science and Technology*, 29, 2012.

[20] M. Perscheid, B. Steinert, R. Hirschfeld, F. Geller, and M. Haupt. Immediacy through Interactivity: Online Analysis of Run-time Behavior. In *Proceedings of the 17th Working Conference on Reverse Engineering*, pages 77–86. IEEE, 2010.

[21] D. Röthlisberger, M. Denker, and É. Tanter. Unanticipated Partial Behavioral Reflection: Adapting Applications at Runtime. *Computer Languages, Systems & Structures*, 34(2-3):46–65, July–October 2008.

[22] D. Röthlisberger, O. Greevy, and O. Nierstrasz. Exploiting Runtime Information in the IDE. In *Proceedings of the 16th International Conference on Program Comprehension*, pages 63–72. IEEE, 2008.

[23] D. Röthlisberger, O. Nierstrasz, and S. Ducasse. Autumn Leaves: Curing the Window Plague in IDEs. In *Proceedings of the 16th Working Conference on Reverse Engineering*, pages 237–246. IEEE, 2009.

[24] B. Shneiderman and C. Plaisant. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley, 5th edition, 2009.

[25] J. Sillito, G. C. Murphy, and K. De Volder. Asking and Answering Questions During a Programming Change Task. *IEEE Transactions on Software Engineering*, 34(4):434–451, July 2008.

[26] B. Steinert, M. Perscheid, M. Beck, J. Lincke, and R. Hirschfeld. Debugging into Examples. *Testing of Software and Communication Systems*, pages 235–240, 2009.

[27] M. Taeumel, B. Steinert, and R. Hirschfeld. The VIVIDE Programming Environment: Connecting Run-time Information With Programmers' System Knowledge. In *Proceedings of the 11th Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*. ACM, 2012 (to appear).

[28] É. Tanter, J. Noyé, D. Caromel, and P. Cointe. Partial Behavioral Reflection: Spatial and Temporal Selection of Reification. In *Proceedings of the 18th Conference on Object-oriented programing, systems, languages, and applications*, pages 27–46. ACM SIGPLAN, 2003.

[29] P. Tarr, H. Ossher, W. Harrison, and S. M. Sutton Jr. N Degrees of Separation: Multi-dimensional Separation of Concerns. In *Proceedings of the 21st International Conference on Software Engineering*, pages 107–119. ACM, 1999.

[30] D. Ungar, H. Lieberman, and C. Fry. Debugging and the Experience of Immediacy. *Communications of the ACM*, 40(4):38–43, April 1997.

[31] A. Von Mayrhauser and A. M. Vans. Program Comprehension during Software Maintenance and Evolution. *Computer*, 28(8):44–55, 1995.

Workload Prediction and Utilization Change Detection in Virtualized Data Centers

Ibrahim Takouna
Internet-technologies and Systems
Hasso-Plattner-Institute
ibrahim.takouna@hpi.uni-potsdam.de

Cloud computing as a consolidation environment assists in reducing energy consumption by computing services. Cloud's providers provide on-demand computing services where customers pay based on the actual resource usage. In this context, capacity planning becomes a significant issue to handle the trade-off between performance and energy.

Thus, this report presents an implementation of an adaptive workload prediction by which we can predict the number of active VMs for the next planning period. Then, it presents an online CPU-utilization state change detection approach that helps to take efficient decisions to perform any action (e.g., VM migration). Furthermore, it presents the modeling and implementation of memory bandwidth demand of the NAS Parallel Benchmark suite. In this report, we discuss the results of each contribution. The results demonstrate the efficiency of our approach for capacity planning in virtualized data centers. Finally, as future work, we investigate improving the accuracy of CPU-utilization prediction and increasing the number of lookahead steps. Additionally, we implement a robust optimization technique for capacity planning exploiting our proposed workload prediction approach.

1 Introduction

In the last retreat, we presented a framework for energy-aware resource management in virtualized data centers where many Clouds providers such as Amazon EC2 leverage virtualization technologies to increase the utilization of physical servers and reduce the number of active physical servers. Realization of an efficient resource management in cloud data centers implies determining the number of the required servers for hosting the active VMs and the number of VMs to be co-hosted (i.e., server consolidation) on a physical server.

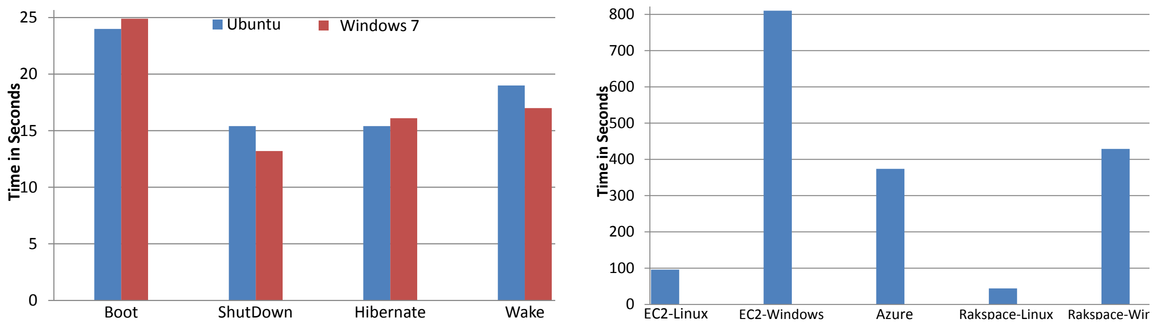
Although much research work has been done in this context [1] [2] [3], many researchers did not consider the overhead of changing the power-state of servers and the energy wastage due to this action. A normal server takes time to go from power-state to another, during this time the server consumes energy without performing any useful work (e.g., execution workload). Figure 1-(a) shows the time spent by a normal PC to switch from power-state to another with different types of operating system.

Turning on/off a server takes the longest time compared to the other saving power states. This is because the boot process includes hardware components check, which depends on the server types and its components.

Furthermore, Mao et al. [5] found that a VM in Amazon EC2 takes time to boot and be ready to execute workload as depicted in Figure 1-(b). Mao et al. [5] observed that the average startup time of a VM varies with its OS type (e.g., Linux or Windows) and its image size. For instance, the average VM startup time of EC2-Linux and EC2-Windows are 96.9 and 810.2 seconds, respectively. Thus, we need a pro-active optimization solution that reacts before the real event occurred. For example, if we can predict the number of the requested VMs in the next planning period, we can prepare these VMs images and the physical server in advance. To this end, we conducted an analysis of historical data for the number of active VMs per unit time (e.g., 5 min). Furthermore, we developed an adaptive prediction algorithm that estimates the number of active VMs for the next 5 minutes. The proposed algorithm is discussed in Section 3.

However, we found that predicting the number of VMs is not enough to develop an energy-aware resource management algorithm. Thus, we decided to study the historical VMs CPU-utilization. Unfortunately, VMs' CPU-utilization fluctuates highly. In this case, taking the raw CPU-utilization as a trigger is unsuitable to perform any action. In this context, we develop two algorithms: i-CPU-utilization prediction; ii-CPU-utilization state change detection. These two algorithms are presented in Section 4.

Finally, we found that using only the historical CPU-utilization is inefficient to guarantee applications' performance in consolidation environment. For instance, we found that the utilization of other shared resources particularly shared memory bus utilization has high impact on applications performance [16]. Thus, we implemented a queuing model that reflects the influence of the memory-bus utilization on co-hosted applications' performance. This is presented in Section 5. However, next we present a few details about the workload traces from Planet Labs, which were used to evaluate our proposed algorithms.



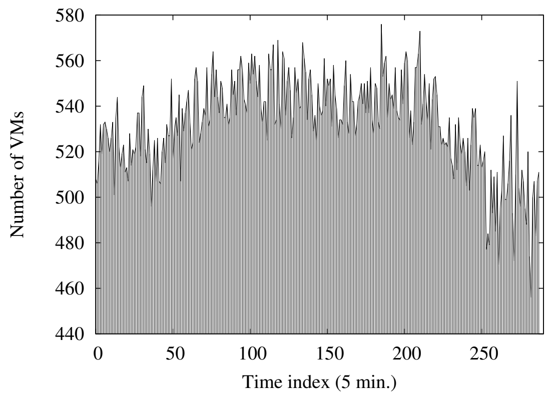
(a) Power state change time (b) VM startup time

Figure 1: Power state change time and VM startup time [4] [5]

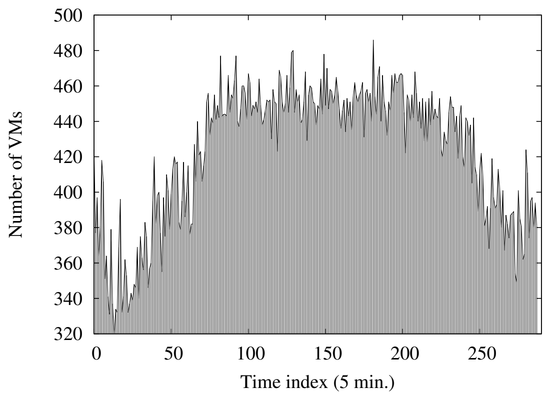
2 Planet workload traces

The monitoring infrastructure project of PlanetLab provides traces of historical data for CPU utilization, which measured every 5 minutes. These traces are for more than a thousand VMs running in more than 500 locations around the world. Here, we present data for four days that have different workload fluctuations. The number of VMs in the traces is constant. However, to simulate the on-demand concept of the cloud computing environment (i.e., the open system behavior), we terminate the VMs with less than 5% CPU utilization. In other words, we considered it as being destroyed and exited the system. Then, when the trace shows a VM with a CPU utilization higher than 5%, we consider a new request for provisioning a VM.

Table 1 shows statistical information of four days of the traces. The average utilization of VMs is around 22% with low standard deviation. On the other hand, the traces show different averages of the number of VMs and levels of fluctuation. For instance, the average number of active VMs for day03 is 532 VMs with 20 VMs as standard deviation. The standard deviation reflects the workload fluctuation around the average value. Thus, day03 shows lowest fluctuation compared to the other days. On the other hand, day09 experiences the highest fluctuation workload where the standard deviation is 60 VMs. Finally, traces of day06 and day22 show medium fluctuation with different average 423VMs and 723VMs, respectively.



(a) Day03



(b) Day06

Figure 2: Planet workload traces

Table 1: Statistical analysis for 4days of Plantlab traces

Trace	Mean CPU Util.	Std. CPU Util.	Mean No. VMs	Std. No. VMs
Day03	22.43	0.87	532	20
Day06	22.02	0.77	423	37
Day09	21.13	1.28	487	60
Day22	17.43	1.09	723	43

3 Workload prediction with an adaptive window

In this section, we present a prediction approach with an adaptive window-size algorithm to predict the number of demanded VMs in a data center. Our approach consists of three stages: i-selecting the historical window size based on the statistical test, t-test/p-value; ii-smoothing the values of the selected historical window; and iii-predicting the next number of active VMs and its minimum and maximum range. In our approach, we estimate the range based on the standard deviation value of the historical selected window. Thus, the range becomes wider when the fluctuation of the values in the selected historical window is high. For this reason, we try to find the suitable backward values to the next value. This allows a higher accuracy of prediction.

Typically, point value prediction techniques might not cover the workload fluctuation (i.e., number of demanded VMs). The approaches solve the problem as a deterministic optimization (i.e., pro-active), which assume the precise knowledge of the workload demand. Furthermore, optimization based on the mean-value or the max-value of the workload can produce low provision or high provision which is costly in both cases.

Furthermore, we plan to solve the optimization problem using a robust optimization approach that applied on a range of values. We choose robust optimization because it allows performing pro-active and reactive decisions. A pro-active method aims to provide an initial off-line decision that is robust to uncertainties during run-time. On the other hand, a reactive method reacts to uncertainties during run-time. The implementation of this approach is our next step.

Most of the related work in the context have been done for grid computing [6] [7] [8] [9]. For example, Wu et al. [6] have proposed an adaptive prediction of grid performance with a confidence window for the historical values. They used an auto-regression to find a model for the historical interval by which predicts the future workload. They claimed that their approach could predict more than 20 steps ahead, which equals to 100 minutes. In our point of view, predicting 100 minutes in advance is not reasonable in virtualized data centers. As shown in Figure 2, the number of VMs shows random behaviour, which cannot be accurately modeled.

3.1 Historical window-size selection

We introduce in Figure 3 a workload prediction approach. This approach uses an adaptive window-size of historical values to provide a high accurate prediction range. The measured workload values are shown by a series of line-dots up to time t. On the other hand, the gray dot represents the predicted workload value. Our interest is to predict the number of VMs for the next 5 minutes from the historical window HW. The historical window-size is determined based on the P-value of both F-test and T-test to filter out the values that are very unlikely to be in the same window.

We used F-test and T-test to probe the significance of the change in variance and mean between two samples of populations, respectively. F-test and T-test give P-value, which indicates whether the two samples have almost the same variance and the same mean. The P-value of F-test is the probability of getting an extreme value under the null hypothesis (i.e., H0:$\sigma_1^2 = \sigma_2^2$). For example, after performing F-test, if we find out

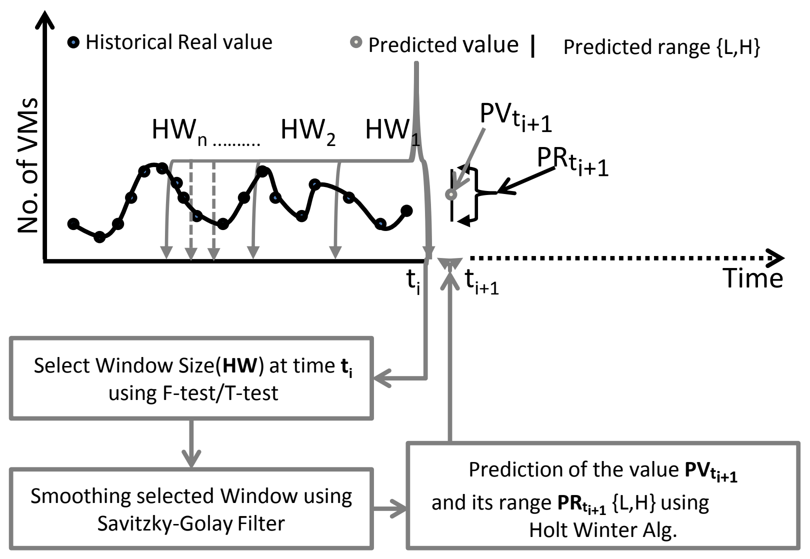

Historical Real value
Predicted value
Predicted range {L,H}
No. of VMs
HW_n .... HW_2 HW_1
$PV_{t_{i+1}}$
$PR_{t_{i+1}}$
t_i
t_{i+1}
Time
Select Window Size(**HW**) at time t_i using F-test/T-test
Smoothing selected Window using Savitzky-Golay Filter
Prediction of the value $PV_{t_{i+1}}$ and its range $PR_{t_{i+1}}$ {L,H} using Holt Winter Alg.

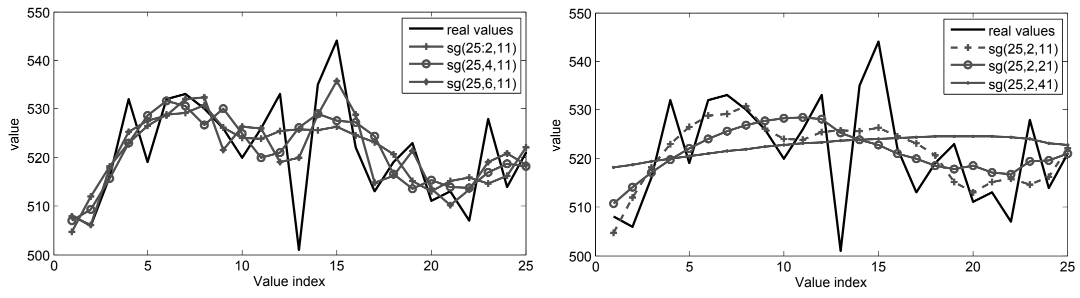
Figure 3: Adaptive window-size prediction approach

that the P-value is less than $\alpha = 0.05$, we reject the null hypothesis. This means that these values do not belong to the same historical window. On the other hand, the P-value of T-test indicates that whether we can accept the null hypothesis (i.e., H0: $\mu_1 = \mu_2$). In other words, the P-value shows how well the two samples can be in the same historical window. The smaller the P-value is the stronger the confidence to reject the null hypothesis H0. Thus, a higher fluctuated workload, a smaller window-size is selected.

3.2 Smoothing the selected window's historical values

Using prediction algorithms with the historical values causes errors. Thus, we used a smoothing filter to remove noise and prevent its influence on the prediction algorithm. There are many smoothing filters, but we selected Savitzky-Golay filter. From literature, Savitzky-Golay filter is effective in keeping the peak values and removing the spikes, which can be considered as noise. Typically, a long polynomial or a moderate order polynomial allows a high level of smoothing without attenuation of data features.

Savitzky-Golay filter has two significant parameters that guide the smoothing process: the frame size and polynomial degree. In our approach, the frame size is not constant, and it equals to the selected historical window-size. In contrast, Wu et al. [6] fixed the frame size (i.e., frame size=51). Regarding the polynomial degree, after conducting some experiments as shown in Figure 4, we found that using the second degree is efficient in our work. Figure 4 shows the output of Savitzky-Golay filter with different settings: S-G(25,2,11), S-G(25,4,11), and S-G(25,6,11).



(a) Fix frame size with different poly. degree (b) Poly. degree size with different fix frame

Figure 4: Savitzky-Golay filter settings influence on time series

3.3 Prediction using HoltWinter algorithm

We used Holtwinter implemented in R-tool, because it dynamically determines the parameters α and β as shown in Equations 1 and 2 that influence the level and the trend of the time series, respectively. For example, a low value of α (e.g., 0.3) indicates that the estimation of the level at the current time point depends mainly on the recent observations. On the other hand, the value of β shows the trend component.

$$\bar{y}_t = \alpha \bar{y}_t + (1-\alpha)(\bar{y}_{t-1} + F_{t-1}) \tag{1}$$

$$F_t = \beta(\bar{y}_t - \bar{y}_{t-1}) + (1-\beta)F_{t-1} \tag{2}$$

Importantly, we determine the predicted range PR based on the single predicted point value PV and the standard deviation of the selected window σ_{HW}. The predicated range PR $\{R_L\ , R_H\}$ equals $\{PV - \sigma_{HW}\ , PV + \sigma_{HW}\}$.

3.4 Implementation and results

We implemented the proposed approach using Java programming language with integration of R-tools, which consists of many statistical functions and the required filters. Here, we present the results of our approach. Figure 5 shows the predicted range for each value of workload (i.e., number of VMs). The low predicted R_L is shown by a red dashed-line meanwhile the high predicted R_H is represented by a blue dashed-line. The purple sold-line represents the single point predicted value.

Figure 6 shows the CDF of the relative error for the proposed adaptive historical window-size selection and different fixed values. From Figure 6, we notice the accuracy of our proposed approach compared to the fixed value. Importantly, we can notice that the range covers the real measured values that revealed by time. Furthermore, the range increases proportionally with workload fluctuation. In contrast, the historical window size increases inversely proportional with workload fluctuation. For instance, with high workload fluctuation the historical window-size is relatively small and the predicted range is relatively wide as shown in Figure 5 and Figure 7. Figure 7 shows the cumulative distribution function (CDF) of the selected historical window-size for each day. Clearly, from Table 1, Day03 has the less fluctuation workload compared to other

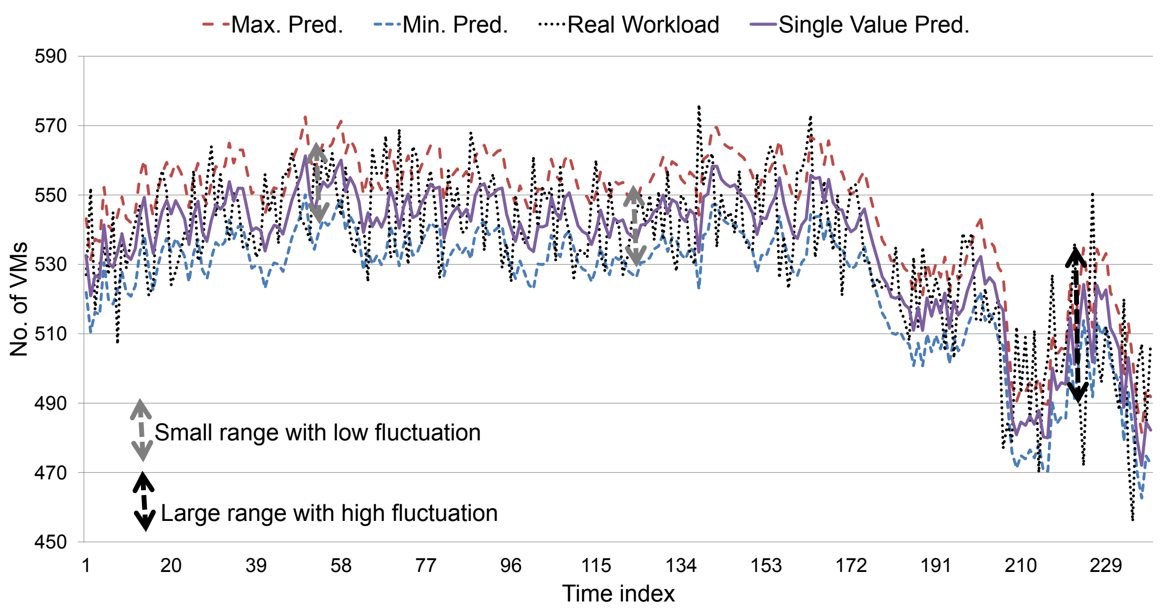


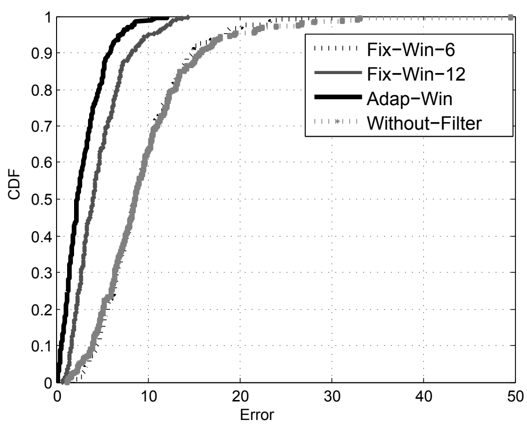
Figure 5: Results of the proposed approach

days. From Figure 7, the historical window-size of Day03 is relatively large most of the day. In contrast, the historical window-size of Day09 is relatively small most of the day.

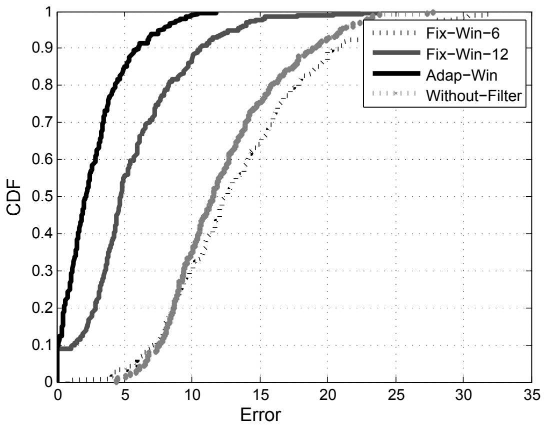
4 CPU-utilization state change detection

Online dynamic resource management algorithms require a trigger to initiate an action. The trigger can be a change in CPU-utilization of a virtual machine, and the action is the virtual machine migration from a server and turning off the server. In this case, we have one trigger and two actions. Thus, the efficiency of taking the best action depends highly on state change of CPU utilization. Unfortunately, CPU-utilization is not always in steady state, but it always fluctuates. Taking an action based on the real measured value of CPU-utilization has two drawbacks: i- delaying the action ii-making the system unstable. Hence, this section presents our approach to conceal or mitigate the two drawbacks.

We propose an online detection of CPU utilization state change approach. This approach consists of three stages as shown in Figure 8, CPU utilization prediction, CPU utilization state representation, and CPU utilization state change detection. Each of these stages is discussed in the following sections. In the context there is much related work [10] [11] [12]. However, these approaches assume a preliminary knowledge about the statistical characteristic of the time series. Thus, they used these filters: Kalman filter, sequential Monto Carlo method or particle filtering. In our work, we do not use this assumption. Therefore, we used the recent proposed ROBUST.FILTER [13].



(a) Day03



(b) Day06

Figure 6: Cumulative distribution function of relative error for the single predicated value: adaptive historical window-size vs fixed window-size

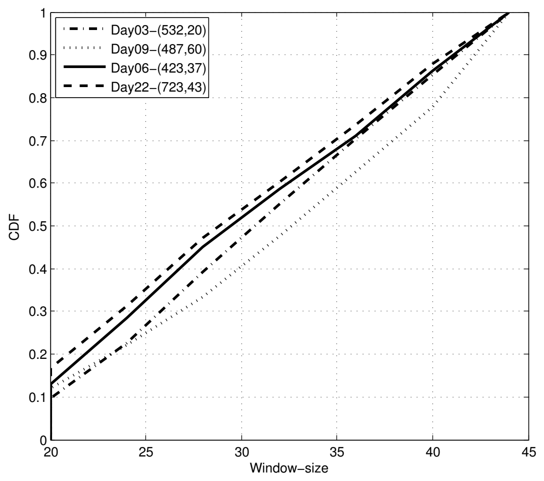


Figure 7: Cumulative distribution function of window-size for the 4 days

4.1 CPU utilization prediction

CPU utilization prediction is the first stage in our proposed approach. This stage allows predicting the future CPU utilization of a virtual machine. Consequently, we can detect the change in advance before that might occur. Furthermore, resource management algorithms can work pro-actively to provide a suitable solution instead of waiting the measured time series of utilization TS_m to be revealed. To this end, we used the prediction approach that was presented in Section 3.

4.2 CPU utilization state representation

After prediction of a time series of CPU utilization TS_p, it passes to the second stage converting TS_p to TS_r state-representation that represents the changes in the time series. Thus, we used ROBUST.FILTER that proposed in [13]. ROBUST.FILTER was

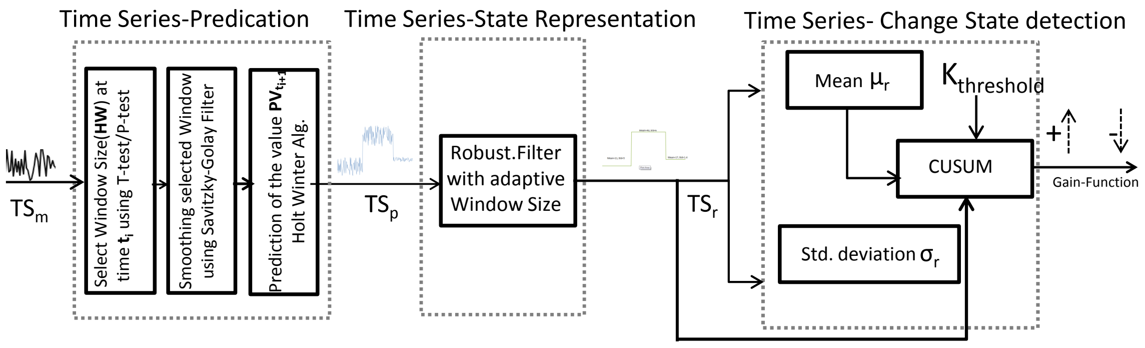

Time Series-Predication
Time Series-State Representation
Time Series- Change State detection
Select Window Size(HW) at time t_i using T-test/P-test
Smoothing selected Window using Savitzky-Golay Filter
Prediction of the value $PV_{t_{i+1}}$ Holt Winter Alg.
TS_m
TS_p
Robust.Filter with adaptive Window Size
TS_r
Mean μ_r
$K_{threshold}$
CUSUM
Std. deviation σ_r
+
-
Gain-Function

Figure 8: CPU utilization prediction and state changes detection

proposed for a fast and reliable filtering of the data. It allows distinguishing artifacts from clinically relevant changes in the patient's condition. Fried [13] used robust regression functional for the local approximation of the trend in a moving time window. Furthermore, ROBUST.FILTER allows online outlier replacement and trimming based on robust scale estimators. The output of this stage gives a squared signal as shown in Figure 9, which further can be used to detect CPU utilization state changes.

4.3 CPU utilization state change detection

Once we have the squared signal TS_r, we used a Cumulative Sum Control Chart (CUSUM) technique for monitoring change detection. This technique developed by E. S. Page of the University of Cambridge. TS_r has a known mean μ_r and standard deviation σ_r. When the value of gain function G exceeds a certain threshold value K, a change in value is detected. The equation 3 detects the positive changes meanwhile the equation 4 detects the negative changes. The output of this stage as depicted in Figure 8 is a pulse function to identify the change and its direction (i.e., positive or negative).

To detect changes, the two gain function (i.e., G^+ and G^-) are applied simultaneously. The initial values of G_0^- and G_0^- are 0. Furthermore, ω_p and ω_n represent the weight of the increase detection G^+ and the decrease detection, respectively. The values of ω_p and ω_n equal to $(\mu_r + K)$ and $(\mu_r - K)$,respectively. In our case, the threshold value K was set to 5 where the change is detected when CPU utilization is increased or decreased by 5. This stabilizes the system in particularity when CPU utilization fluctuates highly. The gain function shows high values at changes either in positive or negative as shown in Figure 10. These high values will be used for triggering any action (e.g., VM migration). Importantly, by using Equations 5 and 6, we update the new mean of the new level of the representation state. The value of *L* specifies the sensitivity of the gain function to the change.

$$G_{i+1}^{+} = max\{0, G_i^{+} + TS_r - \omega_p\} \tag{3}$$

$$G_{i+1}^{-} = min\{0, G_i^{-} + TS_r - \omega_n\} \tag{4}$$

$$\mu_i = \mu_{i-1} + K + G_i^{+} \qquad if \qquad G_i^{+} > L \tag{5}$$

$$\mu_i = \mu_{i-1} - K - G_i^- \qquad if \qquad G_i^- > L \tag{6}$$

4.4 Implementation and results

We implemented the proposed approach using Java programming language with integration of R-tools. Here, we present the results of our approach. Figure 9-(a) shows a time series with three different means and standard deviations to illustrate the efficiency of our proposed approach in detecting the significant change in CPU utilization. Figure 9-(b) depicts the gain function at each value. Clearly, we can distinguish between the significant shift in CPU utilization from low to high mean around 60 and 70 time unit. This is indicated by a high gain of the increase-detection value in the blue colour in Figure 9. On the other hand, the gain of the absolute value of decrease-detection is relatively high around 150 and 153 time unit. This detects the shift in CPU utilization from high mean to low mean. Furthermore, we can exploit the value of standard deviation for efficient consolidation. Here, the standard deviation illustrates the workload intensity where a higher value is a more intense. Thus, we can choose VMs with a low CPU utilization standard deviation to be scheduled with other VMs showing a high CPU utilization standard deviation.

Figure 10-(a) shows the gain function compared to the measured time series values TS_m. This shows many noisy gains with low values. To reduce this noise, we decided to use the representative time series TS_r as shown in Figure 10-(b). In contrast to Figure 10-(a), Figure 10-(b) clearly distinguishes the shifts of the CPU utilization's mean 11 to 40.

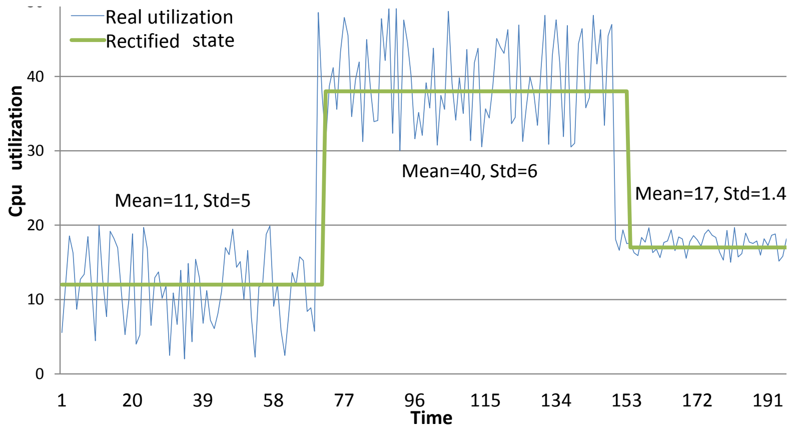
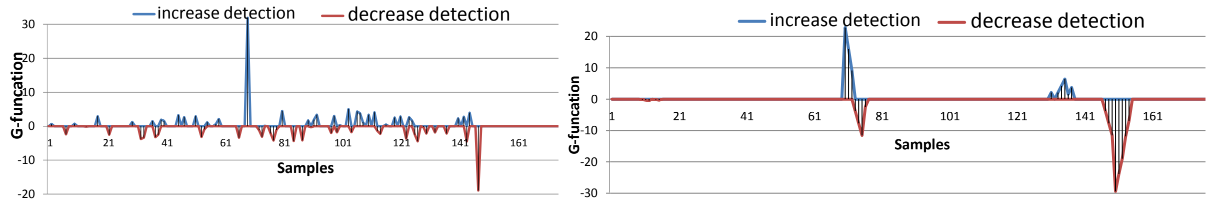


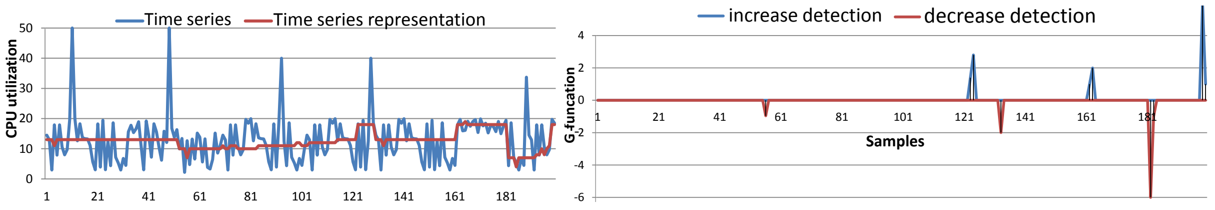
Figure 9: An example of CPU utilization detection and the output of the gain function of CUSUM

Furthermore, we examined the proposed approach against a time series with spikes as shown in Figure 11-(a). The blue line is the output of the CPU utilization state representation stage. Using ROBUST.FILTER in this stage assisted in removing the effects of spikes. Additionally, it was capable to generate state representation at each significant change in CPU utilization. Figure 11-(b) depicts the output of the gain function.



(a) With the measured time series (b) With the rectified time series

Figure 10: An example of CPU utilization detection and the output of the gain function of CUSUM



(a) CPU utilization time series with spikes (b) CPU utilization change detection

Figure 11: An example of CPU utilization detection and the output of the gain function of CUSUM

The results show that our approach distinctly detected the changes that occurred over the K threshold value (i.e., $K = 5$).

5 Modeling and implementation of memory bandwidth demand and VM migration

Simulation tools also have a trade-off between using a detailed simulation which accurately reflects the real environment and simple simulation which models the basis of the real environment. In consolidation environments such as Clouds, it is very important to use a detailed simulation because an inadequate detail in the model representation can lead to misleading or wrong results [14].

Thus, in this paper, we provide a detailed analysis and simulation of different aspects that concern consolidation and migration. First, we simulate memory-bus that connects a processor with a memory-subsystem showing the significance of monitoring the utilization of memory-bus for predicting and enhancing applications' performance.

Second, we simulate different communication techniques of VMs including shared-memory for multi-threaded applications (i.e., OpenMP applications) and network for multi-processes applications (i.e., MPI applications). Thus, we can show the influence of memory-bus's utilization on multi-threaded applications and network utilization on multi-processes applications.

Third, we give an analysis for NAS Parallel Benchmarks (NPB) benchmarks suite enclosing three different implementations: Serial, OpenMP, and MPI. A simulation of the memory demands behavior and communication patterns of each benchmark are also presented. To validate simulation, we compare the results with the real experiments that have been presented in [15].

Finally, many migration algorithms consider only CPU utilization to trigger live migration, but they do not consider the consequences of migration's overheads on the other server resource such as memory bus and network. Hence, we illustrate the effects of migration on CPU, memory bus, and network and their consequences on migration time and power consumption. The details of this work are presented in [16].

6 Summary and Next step

In this report, we presented our work that included implementation of an adaptive workload prediction, CPU utilization state changes detection, and modeling and implementation of memory bandwidth demand of the NAS Parallel Benchmark suite. Furthermore, we presented the results of our work for the adaptive workload prediction and CPU utilization state changes detection. These results showed the efficiency of our approach to be used for capacity planning in virtualized data centers.

Regarding our next step, we will investigate improving the accuracy of CPU utilization prediction and increasing the number of lookahead steps. Furthermore, we will conduct more experiments to study the efficiency of our algorithm in server consolidation based on CPU utilization state changes detection. As we mentioned earlier, we can use the standard deviation of the CPU utilization state to identify the running job intensity instead of just using the mean value. This will be investigated within the next 6 months.

As we implemented the prediction approach based on a range not a single value, we will study the implementation of robust optimization for capacity planning. Robust optimization deals with optimization problems where robustness is sought against uncertainty or deterministic variability in the value of a parameter of the problem (i.e., the workload). The principle of robust optimization considers point prediction meaningless and it replaced by range prediction. Thus, robust optimization addresses data uncertainty by assuming that uncertain parameters belong to a bounded range.

In our approach, we avoid the assumption that considers the precise knowledge of the workload demand in the planning horizon where many proposed solutions have solved the problem as a deterministic optimization (i.e., pro-active) [1] [2] [3].

References

[1] D. Kusic, JO. Kephart, JE. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control, " *Proceedings of the 5th IEEE ICAC*. Chicago, USA, June 2008.

[2] A. Verma, P. Ahuja, A. Neogi, "pMapper: Power and migration cost aware application placement in virtualized systems," *Proceedings of the 9th ACM/IFIP/USENIX*

International Conference on Middleware (Middleware 2008), Springer, Leuven, Belgium, 2008; 243-264.

[3] A. Gandhi, M. Harchol-Balter, R. Das, C. Lefurgy, ”Optimal power allocation in server farms," *Proceedings of the 11th International Joint Conference on Measurement and Modeling of Computer Systems*, ACM New York, NY, USA, 2009; 157-168.

[4] BOOTING-TIME.[Online]. Available:http://www.tomshardware.co.uk/ubuntu-oneiric-ocelot-benchmark-review,review-32377-15.html.[Accessed: 1-Jun-2012].

[5] M. Mao and M. Humphrey, ” a performance study on the VM startup time in the cloud," Proc. of fifth IEEE International Conference on Cloud Computing, 2012.

[6] Y. Wu, K. Hwang, Y. Yuan , and W. Zheng, ”Adaptive Workload Prediction of Grid Performance in Confidence Windows," *IEEE Trans. on Parallel and Distributed System*, 2009.

[7] M. J. Clement and M.J. Quinn, ”Analytical Performance Prediction on Multicomputers," J. Supercomputing, pp. 886-894, 1993.

[8] J. Dongarra, I. Fister, G. Fox, W. Gropp, K. Kennedy, L. Torczon, and A. White, eds. Kaufman Publishers, 2002.

[9] I. Foster and C. Kesselman, ”The Grid: Blueprint for a New Computing Infrastructure," Kaufmann Publishers, 2004.

[10] M. Andreolini, S. Casolari, M. Colajanni, ”Models and framework for supporting run-time decisions in web-based systems," ACM Tran. on the Web 2 (3)(2008).

[11] P. Dinda, D. O'Hallaron, ”Host load prediction using linear models," Cluster Computing 3 (4) (2000) 265-280.

[12] E. Hartikainen, S. Ekelin, ”Enhanced network-state estimation using change detection," in: Proc. of *the 31st IEEE Conf. on Local Computer Networks*, Nov. 2006.

[13] R. Fried, ”Robust filtering of time series with trends," Journal of Nonparametric Statistics, vol. 16, no. 3-4, pp. 313-328, Jun. 2004.

[14] D. Cavin, Y. Sasson, and A. Schiper, ” On the accuracy of manet simulators," In POMC '02: *Proc. of the Second ACM Int. Workshop on Principles of Mobile Computing*, pages 38-43, New York, NY, USA, 2002. ACM Press.

[15] U. K. Medisetty, V. Beltran, D. Carrera, M. Gonzalez, J. Torres, and E. Ayguade, ”Efficient HPC application placement in Virtualized Clusters using low level hardware monitoring," [Online]. Available:http://www.udaykiranm.com/hpcvirt.pdf

[16] I. Takouna, W. Dawoud, and Ch. Meinel, ”Analysis and Simulation of HPC Applications in Virtualized Data Centers,"*IEEE GreenCom*, 2012.

Understanding Code with Execution Traces

Arian Treffer
Enterprise Platform and Integration Concepts
Hasso-Plattner-Institut
arian.treffer@hpi.uni-potsdam.de

This report outlines research ideas that were formed over the last few months. We propose to use an in-memory database to store and access execution traces, especially of unit tests. Based on this, IDEs can be extended to search and visualize the data in the context of its source. This might help to increase developer productivity when understanding code or searching for bugs.

1 Introduction

Most developers spend much time understanding existing code. Especially in large systems, another important task is finding the code in the first place [4]. Programmers search code for various reasons, for instance,

- finding code that implements a feature,
- finding existing solutions to similar problems, or
- understanding how a code unit is used.

At development time, runtime information is not available. However, unit tests are a source for reproducible runtime data. In a well-tested system, the tests should provide enough runtime information of common executions paths, as well as anticipated excpetional cases, to answer typical questions about a program's behavior.

There are several different approaches to work on runtime data of unit tests. Some development tools collect only parts of the data, and discard information that is no longer regarded relevant [1]. Others re-execute tests repeatately, collecting only the data that is currently needed [5].

However, for a system-wide search these approaches cannot be used, as all data is needed at once. A third, and maybe the most obvious approach, is to collect and store everything that happens during the execution. This approach has been successfully used to implement omniscient debuggers [2]. However, it is generally assumend that, especially for large systems, this approach creates more data than can be handled in a reasonable manner [1, 2, 5].

We want to challenge this assumption and propose that a modern database is capable of handling the trace data of unit tests even for business size applications. Recent development in databases allows the fast execution of analytical queries on large amounts of data [6]. This technology was already succesfully applied to static code search [3]. However, this system does not include runtime information.

2 Research Idea

Execution traces of unit tests can be stored in a database. Then, an extension to the developer's IDE can allow searching this data in a meaningful way. It does not seem reasonable to expect the developer to formulate an exact query (e.g., in an SQL-like language) of what she is looking for. Instead, a graphical interface can help with the construction of the query. The developer begins with browsing the results of a very broad query, which then can be refined incrementally, until the result set is small enough so that each result can be examined individually. Using an in-memory will ensure the fast response time that is necessary for performing incremental searches.

The research will show which data layouts are optimal for efficiently answering developers' questions, how user interfaces can be designed to incrementally formulate queries on execution traces, and how complex queries are that developers want to ask.

3 Further Applications

There more useful applications to trace data, that can build upon a fast analytical search. For instance, the IDE can quickly visualize execution paths and variable values that a developer should expect in a given method, show the typical history of an object's state, or provide a usage-search for virtual methods that is more precise than the current search, based solely on static analysis.

Furthermore, it should be possible to implement an omniscient debugger on top of the database. This debugger would not only allow forward and backward stepping, but could also perfom semantic steps, such as

- "step to the next invocation of this method,"
- "find another invocation of this method (maybe even in a different trace), where one parameter is different," or
- "go back to the last step where this field was read or written."

Until now, we only considered unit tests as a source for reproducible execution traces. However, when the entire trace is stored, it does not have to be reproducible. Thus, the proposed system could also be used to find non-deterministic bugs, for instance, bugs that are caused by racing conditions or arbitrary output of external systems.

4 Summary

Searching code using runtime information can help to find code that implements a feature of interest or to locate the source of a bug. An incremental approach should allow to formulate complex queries that will yield a sufficiently narrow result set. Using an in-memory database should ensure good response times even for large systems.

Once a prototype is implemented, user studies will have to show which questions are asked, how the construction of queries can be supported, and to which extend this helper can increase developer productivity.

References

[1] Christoph Hofer, Marcus Denker, and Stéphane Ducasse. Design and implementation of a backward-in-time debugger. *NODe 2006*, pages 17–32, 2006.

[2] Bil Lewis. Debugging Backwards in Time arXiv : cs / 0310016v1 [cs . SE] 9 Oct 2003. (September):225–235, 2003.

[3] Oleksandr Panchenko, Jan Karstens, Hasso Plattner, and Alexander Zeier. Precise and Scalable Querying of Syntactical Source Code Patterns Using Sample Code Snippets and a Database. In *2011 IEEE 19th International Conference on Program Comprehension*, pages 41–50. IEEE, June 2011.

[4] Oleksandr Panchenko, Hasso Plattner, and Alexander Zeier. What do developers search for in source code and why. *Proceeding of the 3rd international workshop on Search-driven development: users, infrastructure, tools, and evaluation - SUITE '11*, pages 33–36, 2011.

[5] Michael Perscheid, Bastian Steinert, Robert Hirschfeld, Felix Geller, and Michael Haupt. Immediacy through Interactivity: Online Analysis of Run-time Behavior. In *2010 17th Working Conference on Reverse Engineering*, pages 77–86. IEEE, October 2010.

[6] H. Plattner and A. Zeier. *In-Memory Data Management: An Inflection Point for Enterprise Applications*. Springer, 2011.

Duplicate Decision for Data Quality Web Services

Tobias Vogel
Forschungskolleg
Hasso-Plattner-Institut
tobias.vogel@hpi.uni-potsdam.de

Data Quality is an abstract measure of how well data can be utilized. Data Cleansing is the process of establishing this quality. This report describes this process and shows how to reduce the computational effort while keeping up the accuracy. Moreover, means to further automate the duplicate decision process are outlined.

1 Data Quality Services

"The only thing constant in life is change"[1] as the socio-economic change of the last 200 years shows. Developing from an agrarian society over the industrial and service era, mankind has reached the information society age. Many companies now base on and trade data. The enormous amount of data is continuously rising as well as the rate of interchange between different stakeholders, whether companies, governmental institutions, or divisions within.

To enable this interchange and to generally manage the mass of data, the information's data quality has to be high. This renders the data comparable, interchangeable, searchable, etc. Data quality refers to the data being complete, relevant, trustworthy, accessible, . . . , and *duplicate-free*. Data cleansing helps assuring these properties. Moreover, this activity shall be fast, cheap, and accurate.

Automatic data cleansing services promise to serve these requirements. A data cleansing service identifying multiple representations of same real-world objects (*duplicates*) adheres to the following workflow shown in Figure 1.

1.1 Analyze Input

In the first phase, preparative tasks are performed. This comprises, for example, the removal of noise (special or non-printable characters) or stop words. If the dataset originates from different sources, the different schemas might be aligned to make the datasets comparable. With that, attributes may have to be joined together (such as street addresses) or be separated (such as given and family names). Records and their properties have to be separated and sorted to be individually accessible. The following sequence is applied.

[1] Heraklit of Ephesus

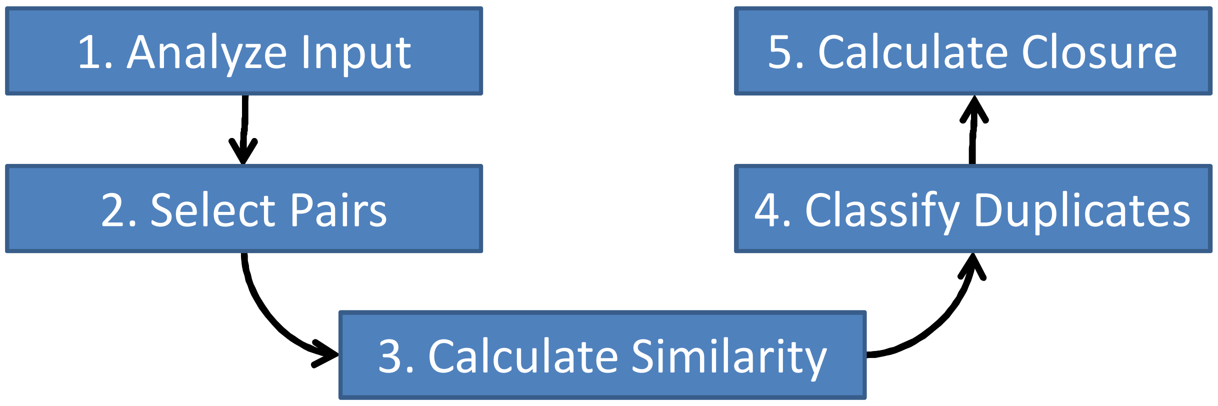


Figure 1: Overall data cleansing workflow

1. The overall file type is examined. Depending on the type, record separators are single characters such as newlines (plain-text, CSV) or further parsing has to be performed (JSON, XML formats). In case of non-machine-readable formats, information extraction techniques have to be applied, for example, to extract genuine, relational tables from HTML documents [8] and similar. The same is true for the attributes of the records.

2. Once the attributes are separated, they might have different schemas, e. g., if they were structured and schema-free data such as XML files. To be able to compare corresponding attributes of several records with each other, the attributes have to be aligned, i. e., the schemas have to be matched [5].

3. The duplicate detection can be made even more effective if the datatypes of the attributes are known. This is only rarely the case. Since instance data is available, the semantics of the attributes can be estimated [6]. All values of an attribute are matched against reference data values. For very clean attributes this might already suffice, for more polluted attributes or attributes for which no reference data are available, features are extracted from these values and compared to the features extracted from example data with known semantics. With this knowledge attached to the attributes, specialized similarity metrics can be used. However, also the previous schema matching step can benefit from these techniques. Same attribute values are likely to have the same datatype.

Until here, no duplicate detection has been performed.

1.2 Select Pairs

Comparing pairs of records is a highly parallelizable process. With the ability of SaaS application to easily scale up, more processing power can be offered in short time. Yet, the complexity of duplicate detection is quadratic (each record has to be compared to each other record) which still renders it infeasible or at least ineffective (and too expensive) to work on all records.

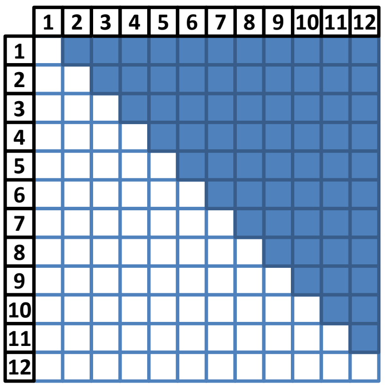


Figure 2: Naive comparison: the baseline for saving comparisons (only dark comparisons are actually performed)

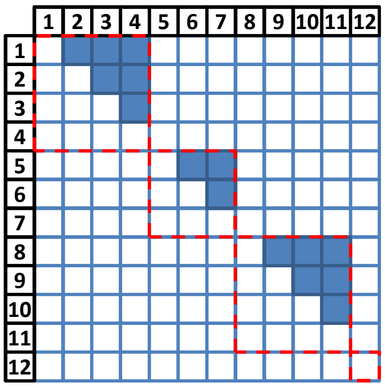


Figure 3: Blocking: only elements within each disjoint block (e. g., red) are compared

Fortunately, the “duplicate” relation is symmetric and reflexive. Thus, only less than the half of all comparisons have to be performed, because two records do not need to be compared twice and a record does not need to be compared to itself. Figure 2 shows an illustration. Only the dark colored combinations have to be examined.

To further reduce the amount of comparisons, there exist comparison pair selection algorithms that create disjoint or overlapping partitions of the data and also intentionally ignore many pairs. To do that, they sort the elements regarding to a specific key and put all elements with the same key into one cluster. Such a key could be, for example, the (first two letters of the) family name for an address dataset or the year of first manufacturing in a product database. It is common practice to generate different keys to create different smaller clusters and to perform several runs for preserving the recall. The precision depends solely on the similarity measures and the subsequent decision on whether a pair is a duplicate or not. Figure 3 illustrates a blocking with disjoint partitions.

Parameters such as the blocking key are usually tuned and developed by human experts. Humans use their domain knowledge about the nature of the dataset and then decide for the blocking key, e. g., some digits of the ZIP code. In the services world, experts are not available and a service has to be self-configuring. Bilenko et al. [1] present an algorithm that automatically proposes different blocking keys, but it relies on the availability of positive and negative examples due to the machine learning techniques employed.

The service will most likely not be in possession of such examples. However, to find key candidate attributes, a service can make use of having many users with datasets from similar domains. Blocking keys for one dataset perform comparatively well on other datasets from the same domain, thus, they have to be retrieved only once per domain.The automatic generation of blocking keys is described in Section 2.

1.3 Calculate Similarity and Classify Duplicates

In this phase, the similarity between the records is calculated, that have been marked as promising in the pair selection phase (Section 1.2). Once the pairs are selected, specific similarity measures are applied on each pair of attribute values. There are several general-purpose similarity measures such as Edit Distance, Jaro-Winkler, and Jaccard similarity. However, knowing about the very nature of an attribute, common misspellings, and – more important – acceptable differences (known from the analysis phase (Section 1.1)), strongly increases the significance of the similarity measure.

After similarity scores for all relevant pairs are calculated, they have to be used to decide, whether a pair of records should be regarded as duplicate. This can be done via weighted sums or decision trees, for example. Section 3 outlines how the blocking information can be re-used to take the duplication decision.

1.4 Calculate Closure

With the previous step, a list of duplicate pairs was declared. However, it is not clear whether there are only pairwise duplicates. There are probably clusters of duplicates from which only a small fraction is explicitly listed, up to this point. Thus, the pairs have to be joined to clusters. For example, the transitive closure can be calculated or some other clustering means has to be applied.

2 Blocking

It is possible to find suitable blocking keys (see Section 1.2) automatically for a dataset equipped with a gold standard, the *training dataset*. Those blocking keys can be re-used for datasets from similar domains lacking a gold standard [7], the *test dataset(s)*. To be general and to support a large variety of data types, blocking keys are created based on unigrams.

2.1 Problem Formalization

The formalized problem is as follows: Given a dataset and its schema, find a valid blocking key (or a set of k valid blocking keys) that achieves the optimal trade-off between pairs completeness and efficiency. The details are explained below. A blocking key consists of a set of unikeys. Each unikey is a combination of an attribute (e.g., ZIP code) and a position within this attribute. Applying such a unikey on actual attribute values yields said unigrams.

Validity

Usually, a dataset different from the training dataset will comprise other attributes. A given blocking key is called *valid* in a test dataset, iff all of its unikeys are available in the test dataset, both regarding the availability of the schema attributes as well as the attribute lengths. The attribute length is defined by the schema (e. g., a `CHAR(100)` in SQL) or is infinite for other data sources (e. g., CSV files).

Pairs completeness

The pairs completeness [2, 4] is the measure of how many of the duplicates can be found for a blocking key, i. e., how effective the blocking key is.

A blocking key is used to create a partitioning to pre-classify duplicate records. Subsequently, a similarity measure is applied on each possible pair within each partition. If the pair's similarity is above a given threshold, it is treated as a duplicate, otherwise as a non-duplicate. The ratio of actual duplicates among the declared pairs divided by all duplicates is called recall and serves as the pairs completeness. In our experiments, we replace such a similarity measure by a lookup in the true matches.

Efficiency

A blocking key is efficient if it uses relatively few comparisons to achieve a given pairs completeness. Thus, the measure for efficiency is the average number of performed comparisons for each found duplicate.

In practice, however, the number of comparisons should not exceed a fixed threshold θ. We express efficiency by normalizing the number of comparisons c according to θ and subtract it from 1 to align it to the pairs completeness. Thus, efficiency is defined as $1 - (\frac{c}{\theta}) \in [0, 1]$, assuming $c \leq \theta$.

This measure resembles the term *Filtered Reduction Ratio* [2]. Yet using the actual number of potential comparisons ($5 \cdot 10^{10}$ for 100,000 tuples) in the denominator would usually create a value close to 1. Therefore, we adapt the notion by Gu and Baxter, but instead of a filtering step, we give the efficiency in relation to a baseline approach. In our case this is the number of comparisons, the Sorted Neighborhood approach [3] would have created.

Overall Blocking Key Quality (BQ)

A good blocking key should be effective and efficient. Therefore, we define the *Overall Blocking Key Quality* BQ as the harmonic mean between pairs completeness and efficiency ($BQ = \frac{2 \cdot PC \cdot Ey}{PC + Ey}$), where PC is pairs completeness and Ey is efficiency.

2.2 Key Generation Workflow

Automatic blocking key generation is performed in two steps. First, for a training dataset with a given gold standard, all combinatorially possible blocking keys are evaluated. Second, for a test dataset, typically lacking a gold standard, the previously created list of blocking keys is iterated to find the best valid blocking key.

Training Phase

As the first step, good blocking keys are identified:

1. Generate all possible unikey combinations (i. e., blocking keys).
2. For each blocking key perform a duplicate detection experiment on the reference dataset:
   (a) If the number of comparisons exceeds the threshold θ, discard this blocking key.
   (b) Else, calculate the achieved overall blocking key quality (BQ) for the blocking key.
3. Sort all non-discarded blocking keys descendingly by BQ.

Production Phase

The keys from the training phase can subsequently be used to find duplicates in test datasets of similar domains.

1. For each blocking key in the previously calculated list, check for validity for the current dataset.
2. For each remaining valid blocking key (still ordered by BQ), start a duplicate detection run.
   (a) If the number of comparisons exceeds a certain threshold, abort the run, keeping the so-far detected duplicates.
   (b) Else, finish the duplicate detection run until one of the following abortion criteria is fulfilled: the desired number of passes have been executed, the total number of actually performed comparisons over all runs exceeds a threshold, the overall efficiency sinks below a given threshold (i. e., no or not enough new duplicates are found), or the number of detected duplicates is sufficient. Note that the thresholds might be domain dependent or given by a user.

2.3 Evaluation

The experiments were performed on different random samples of two address datasets examining 6 million blocking keys. Table 1 shows the most successful blocking keys with regard to the number of found duplicates, comparisons, pairs completeness, efficiency, and BQ. To compare, an “expert guess” – the ad-hoc blocking key `[city-0, familyname-0, givenname-0, zip-0]` a human expert might have come up with – only found 274 of the 804 duplicates to find. However, achieving a very high efficiency value is typical for user-provided blocking keys.

Just a bit of derivation in the attribute positions as in `[city-0, familyname-0, givenname-3, zip-1]` had found seven more duplicates comparing one fifth fewer records. The most successful blocking key found 86.69% of the duplicates, but used – on average – 12,521 comparisons for each duplicate. In contrast, the most efficient blocking key only performed 27 comparisons per duplicate revealing only a small fraction of all the duplicates. Finally, the overall best key (`[familyname-0, familyname-1, zip-0, zip-1, zip-2]`) was both, effective and efficient and achieved very good results in both disciplines. The respective maximum values in the table are emphasized.

Description	Blocking key	Found duplicates	Comparisons	Pairs Completeness	Efficiency	BQ
Expert guess	`[city-0, familyname-0, givenname-0, zip-0]`	274	258,077	34.08%	97.39%	50.49%
Most duplicates and maximum pairs completeness	`[zip-0, zip-1, zip-2, zip-3]`	*697*	8,727,009	*86.69*%	11.80%	20.78%
Least comparisons per duplicate and most efficient	`[city-0, familyname-0, family-name-3, givenname-3, street-3]`	214	*5,781*	26.62%	*99.94%*	42.04%
Overall best	`[familyname-0, familyname-1, zip-0, zip-1, zip-2]`	672	407,232	83.58%	95.88%	*89.31*%

Table 1: Selected outstanding blocking keys

To evaluate the ability for domain transfers of blocking keys between two datasets from similar domains, we took a sample of another dataset. We chose the 300 best blocking keys (according to their BQ) from the training dataset and performed duplicate detection runs on them.

The absolute number of found duplicates vastly increased, because there are much more duplicates in the test dataset. The overall numbers of comparisons stayed in the same order of magnitude (remember that there is a cut-off at 10 million comparisons). Only 131 blocking keys were valid, however the first invalid blocking key had rank 50, thus the best blocking keys did actually work also on the test dataset. The average overall blocking key quality is 94.29% due to a generally higher pairs completeness.

This means that the duplicate characteristics resemble the blocking keys very well, even with data from different languages and domains. Table 2 shows key figures for the first 10 blocking keys.

3 Duplication Decision

It is common to weigh the similarities concerning to their relevance (for example, gender might not be as relevant as family name) to calculate a weighted sum. If this sum is above a given threshold, the pair is regarded as being duplicate, otherwise not. However, global thresholds are too inflexible. Different partitions might have completely different similarity distributions. For example, John Does from a large city will be frequent and only have tiny differences in their pairwise similarities. In contrast, another partition (say) containing inhabitants of a smaller city will have very diverse similarities. Applying a global high threshold will keep the John Does apart but will result in missing all the duplicates in the other partition. Applying a global low threshold will work well in the second partition but will not classify any pair of John Doe as duplicate. A global fixed threshold is easy to set by a user but will result in poor duplicate decision capabilities.

Threshold-based Duplicate Decision The global threshold has to be adapted for each block. Let there be two blocks of records. Block 1 contains pairwise similarities between 80% and 95%, block 2 contains similarities between 30% and 90%. A theoretical, global threshold of 70% would result in an actual threshold of 90.5% for block 1 and 72% in block 2. Thus, the different similarity distributions within the specific blocks are taken into consideration.

Blocking-key-based Duplicate Decision More flexibility is offered by a rule-based approach. In this case, the similarities are used as predicates within a disjunctive normal form or in a decision tree. With that, specific irregularities in the dataset can be covered. For example, if the date and name similarities are above high thresholds, the pair is regarded as a duplicate, ignoring, say, the city similarity. Also negative rules are possible: if the dates of birth do not match, the pair is classified as non-duplicate regardless of the (possibly high) similarity for family names.

Duplication classifiers need these parameters as input to aggregate a given set of similarities between attributes of a duplicate candidate into a duplication decision. Typically, these parameters change over domains, datasets, and even parts of the data and a global parameter setting is too inflexible. For example, address records might share the same ZIP code. For ZIP codes belonging to large cities, the city name is irrelevant as well as parts of the phone number whereas for small cities, the city name is very distinctive, because several smaller cities might share the same ZIP code. Consequently, there should be different parameters for different partitions of the dataset.

In the blocking phase (Section 2), partitions were generated whose participants share specific characteristics. Those partitions (basing on blocking keys) can be reused for the duplicate decision step. Following the pre-requisite, that a gold standard for a training dataset is available, this standard does also serve for learning appropriate rules for the duplication decision. Again, this knowledge can be transferred to other datasets from similar domains. I. e., if the described city example is a general principle, it will generally hold for other datasets. Creating a mapping from blocking keys to rule sets will enable an automated duplicate detection service to autonomously decide on duplication in new, unknown datasets.

4 Conclusion

The process of data cleansing in general and duplicate detection in particular poses many research opportunities towards increasing effectiveness and efficiency and automating it as a whole. While it has been subject of innumerable efforts since more than 40 years now, the Software-as-a-Service community did not yet fully investigate the potential of the particularities of this new approach, namely having many different customers and datasets from different domains as well as being used by unexperienced users who cannot tune and tweak all the necessary parameters.

I presented the overall duplicate detection workflow (Section 1) and particularly showed how to achieve a good blocking to reduce the computation effort while maintaining accuracy (Section 2). Once these blocks are known, pairwise similarities can be calculated. The next challenge is to make use of these similarities to actually classify a pair of records as being duplicate. I presented two means to do that automatically without human intervention (Section 3). The evaluation of this duplicate decision phase is left as a next step.

References

[1] Mikhail Bilenko, Beena Kamath, and Raymond J. Mooney. Adaptive blocking: Learning to scale up record linkage. In *6th IEEE International Conference on Data Mining (ICDM)*, 2006.

[2] Lifang Gu and Rohan Baxter. Adaptive filtering for efficient record linkage. *Proceedings of the SIAM International Conference on Data Mining (SDM)*, 2004.

[3] Mauricio A. Hernández and Salvatore J. Stolfo. The merge/purge problem for large databases. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, 1995.

[4] Matthew Michelson and Craig A. Knoblock. Learning blocking schemes for record linkage. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, 2006.

[5] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 2001.

[6] Tobias Vogel and Felix Naumann. Instance-based "one-to-some" assignment of similarity measures to attributes. In *Proceedings of the International Conference on Cooperative Information Systems (CoopIS)*, 2011.

[7] Tobias Vogel and Felix Naumann. Automatic blocking key selection for duplicate detection based on unigram combinations. In *Proceedings of the International Workshop on Quality in Databases (QDB)*, 2012.

[8] Yalin Wang and Jianying Hu. Detecting tables in html documents. In *Proceedings of the International World Wide Web Conference (WWW)*, 2002.

Blocking key	Training					Test				
	Found Duplicates	Comparisons	Pairs Completen.	Efficiency	BQ	Found Duplicates	Comparisons	Pairs Completen.	Efficiency	BQ
[familyname-0, familyname-1, zip-0, zip-1, zip-2]	672	407,232	83.58%	95.88%	89.31%	7,151	270,706	100%	97.26%	98.61%
[street-1, street-4, zip-0, zip-1, zip-2]	670	396,567	83.33%	95.99%	89.21%	6,721	405,490	93.98%	95.90%	94.93%
[zip-0, zip-1, zip-2, street-0, street-1]	668	455,230	83.08%	95.39%	88.81%	6,753	617,040	94.43%	93.76%	94.09%
[zip-1, zip-2, street-1, street-4]	670	914,682	83.33%	90.75%	86.88%	6,717	465,847	93.93%	95.29%	94.60%
[familyname-0, familyname-1, zip-1, zip-2]	672	1,082,543	83.58%	89.05%	86.23%	6,721	432,902	93.98%	95.62%	94.79%
[street-0, street-1, zip-1, zip-2]	668	1,060,006	83.08%	89.28%	86.07%	6,717	379,379	93.93%	96.16%	95.03%
[street-1, street-4, zip-0, zip-2]	670	1,129,843	83.33%	88.58%	85.87%	6,731	545,091	94.12%	94.49%	94.30%
[familyname-0, familyname-1, title-3, zip-1, zip-2]	661	1,060,663	82.21%	89.28%	85.60%	6,736	993,987	94.19%	89.95%	92.02%
[familyname-0, familyname-1, title-2, zip-1, zip-2]	661	1,060,666	82.21%	89.28%	85.60%	6,727	1,241,669	94.07%	87.45%	90.64%
[familyname-0, familyname-1, title-4, zip-1, zip-2]	661	1,060,668	82.21%	89.28%	85.60%	6,717	388,312	93.93%	96.07%	94.99%

Table 2: Comparison of the key figures for the first 10 best blocking keys in the training dataset applied on the test dataset.

Integrated Software Development for Embedded Robotic Systems

Sebastian Wätzoldt
Systems Analysis and Modeling Group
Hasso Plattner Institute
sebastian.waetzoldt@hpi.uni-potsdam.de

In the recent years, improvements in robotic hardware have not been matched by advancements in robotic software and the gap between those two areas has been widening. To cope with the increasing complexity of novel robotic embedded systems an integrated and continuous software development process is required supporting different development activities and stages being integrated into an overall development methodology, supported by libraries, elaborated tools and toolchains. For an efficient development of robotic systems a seamless integration between different activities and stages is required. In the domain of automotive systems, such an overall development methodology, consisting of different development activities/stages and supported by elaborated libraries, tools and toolchains, already exists. In this report, we show how to adapt an existing methodology for the development of automotive embedded systems for being applicable on robotic systems.

1 Introduction

In novel robotics applications steady improvements in robotic hardware is not matched by advancement in robotic software leading to an increasing gap between those two areas. The increasing complexity of modern robotic systems requires to further support several different software development activities such as modeling, simulation and testing that allow the incremental development of robot systems, starting with a single sensor and resulting in a complex application. Elaborated tools and toolchains are required to support the different activities and integrate them into an overall and well structured development methodology. To realize an efficient software development process, on the one hand, one has to provide libraries supporting individual development activities at different levels, e.g., at the level of individual sensors and control functions or at the level of systems or sub-systems, being incrementally composed. On the other hand, a seamless migration between individual development activities and stages has to be achieved. Furthermore, one crucial aspect that needs to be considered for a large portion of robotic systems is real-time behavior.

Accordingly, the following aspects need to be considered for bridging the gap between hardware and software development in novel robotic systems: (I) An overall methodology is required that supports (II) different development activities like modeling, simulation and testing at (III) different stages, e.g., simulation, prototyping and (pre-)production. Such a methodology has to be supported by (IV) elaborated tools and (V) libraries integrated into (VI) an overall toolchain allowing a seamless migration between the different development stages and artifacts. (VII) Simulation and testing support is required for the stages, allowing to validate created functionality, developed sub-systems or systems, e.g., by providing executable functional models, simulation environments and plant models. (VIII) Last but not least, real-time constraints need to be reflected.

As an example, in the automotive domain large complex real-time embedded systems are developed using different development stages, e.g., simulation, prototyping, and pre-production. Advanced tools and libraries have emerged during the recent years, integrated into sophisticated toolchains supporting different development stages as well as a seamless migration between them. To deal with the increasing complexity and to further reduce software development costs as well as time, advanced frameworks for the distributed and component-based development have been developed. In this report, we propose adapting the existing software development methodology used in the domain of automotive embedded systems to support the software development of novel, complex embedded robotic systems. The proposed methodology includes an overall development process consisting of tools included into an overall toolchain as well as libraries. We apply this existing approach to the domain of robotic systems and evaluate as a proof of concept, which modifications have to be made. The approach is evaluated using a mobile robot developed according to the adapted methodology. Special attention is given to real-time constraints that need to be considered in a slightly different way than in the case of automotive real-time embedded systems. Therefore, we show a new approach for combining hard and soft real-time behavior in the existing automotive framework.

The remainder of this report is organized as follows. Section 2 briefly discusses the foundations of robotic as well as automotive systems and introduces a running example for this report. Section 3 describes our development approach including different stages and highlights our used tools as well as simulation and verification possibilities. The report discusses related work in Section 4 and concludes in Section 5.

2 Foundations – Robotic and Automotive Systems

2.1 Robot Laboratory

For the evaluation of our research activities, we use our CPSLab[1] robot laboratory consisting of three Robotino robots.[2] The robots can be equipped with several sensors (e.g., laser scanner, infrared (IR) distance sensors, GPS like indoor navigation systems) as well as different actuators (e.g., servo motors, omnidirectional drive, gripper). The general idea of our evaluation scenario is the realization of a variable production setting, where robots are capable of transporting small pucks (representing goods in a production system) to different locations. Robots have to fulfill different requirement,

[1]`www.cpslab.de`
[2]`www.festo-didactic.com`

e.g., they have to provide basic functionality like moving and avoiding obstacles in hard real-time (reacting on obstacles within a few milliseconds). Further, the robots have to reflect high level goals, e.g., energy saving of the battery, short routing to the destination points and optimizing the throughput while transporting the pucks. While basic functionalities, such as obstacle avoidance, have to be realized in hard real-time, we use existing libraries to realize higher functionalities such as path planning or creating a map by evaluating measured distance values. The latter can rarely be realized under hard real-time constraints because of insufficient libraries.[3] Furthermore, we run a RTAI Linux operating system[4] on the robot to enable hard real-time execution.

As a running example, we use a single robot with the following hardware/ software configuration: The robot has three wheels realizing an omnidirectional drive. The drive unit provides an incremental encoder to realize odometry functionality, which calculates the relative position over time according to the drive speed and the orientation of the omnidirectional drive of the robot. Due to the fact that this odometry calculation becomes more and more imprecise over time, we use an additional GPS like (*NorthStar*[5]) indoor navigation system to correct the position in the long run. IR distance sensors are used to avoid obstacles during movement. A more complex navigation logic uses these sensors for maintaining a map[3] as well as computing an appropriate route for the robot while avoiding obstacles.

2.2 Automotive Development Process

A commonly applied development process for the development of automotive embedded real-time systems according to [4] is depicted on the left in Fig. 2. The development process includes three different stages, namely the simulation, prototyping and pre-production stage. During the simulation stages models are extensively used for realizing control functionality as well as for representing the environment. At the prototyping stage, a transition from a model-based to a software centric development approach is realized. Often, this is achieved by using code generators that automatically derive source code from the models used in the previous stage. In the pre-production stage, more and more aspects of the real system are involved, e.g., by using prototyping HW including the processor type (with additional debugging support) that is later used. Furthermore, parts of the real plant enable a realistic validation of the real-time behavior.

2.3 AUTOSAR

The **AUT**omotive **O**pen **S**ystem **AR**chitecture was invented to further support the development of complex and distributed systems. AUTOSAR[6] is the new de facto standard in the automotive domain. It defines a layered architecture, standardized communication mechanism and a whole development methodology. Furthermore, it supports

[3]For path planning and creating a map the MRPT library is used (`www.mrpt.org`).

[4]`www.rtai.org`

[5]`www.evolution.com/products/northstar/`

[6]`www.autosar.org`

the interaction between different car manufactures and suppliers. Figure 1 gives an overview of the layered AUTOSAR architecture. The layer at the bottom represents

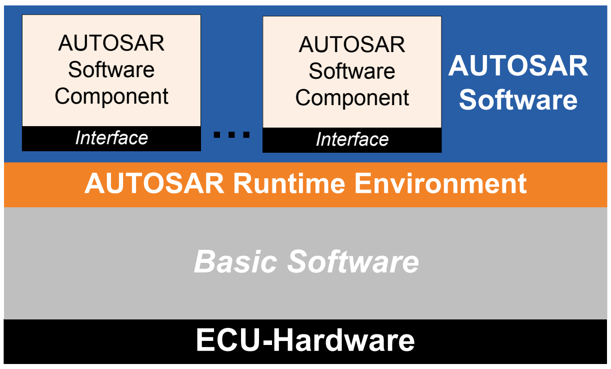

AUTOSAR Software Component
Interface
...
AUTOSAR Software Component
Interface
AUTOSAR Software
AUTOSAR Runtime Environment
Basic Software
ECU-Hardware

Figure 1: The layered AUTOSAR architecture according to the specification in [12].

the real hardware including microcontroller and communication busses. An abstraction layer on top of the real hardware, included in the basic software layer, offers standardized interfaces for accessing the HW. Further functionality realizing the OS behavior as well as functionality for realizing communication is included in the basic software layer. The AUTOSAR runtime environment (RTE) is responsible for realizing the communication from and to the top software application layer. Software components (SWCs) realize application functionality at the layer on top. There, the architecture style changes from a layered to a component based approach [12]. SWCs communicate over well-defined ports using AUTOSAR interfaces, which are realized by the RTE layer. Each SWC consists of an arbitrary number of so-called *Runnables* that specify the behavior entities of each component.[7] Such Runnable entities are mapped on OS tasks, which are scheduled and handled by the operation system included in the basic software layer.

2.4 Automotive vs. Robotic Systems

In an automotive embedded system, usually applications are developed in such a fashion that hard real-time capable functionalities are separated from soft real-time applications. For example, it is quite common to deploy soft and hard real-time functionality on disjoint execution nodes and direct communication between them is avoided.

For robotic systems it is quite common to combine soft and hard real-time behavior into one application. For example, a mobile robot needs to avoid obstacles under hard real-time during navigation while calculating a route and updating a map. Both functionalities need to be combined while predicting the execution time, e.g., of a route planing algorithm, is often not possible.[8] Thus, one difference between automotive and robotic systems concerning the real-time behavior is, that soft and hard real-time capable functionalities need to be more closely linked in robotic systems.

[7]The functionality of a Runnable can be realized by a C/C++ function.
[8]Execution time depends on the size of the map, which is usually not known before runtime.

3 Development Environment

In this section, we describe our development environment, the tools and libraries used in the different development stages as well as our test and verification possibilities during system development. According to [4], we distinguish three development stages at different levels of abstraction targeting specific key aspects, namely simulation, prototyping and pre-production. Validation and verification activities are applied in each stage according to the given abstraction level. On the left in Figure 2, the overall process including the different stages is shown. In the following, we describe the applied validation and verification activities of each stage in the form of the libraries, methods and tools used. Furthermore, we show how to achieve an AUTOSAR conform system realizing the complex behavior of the robot incrementally developed, validated and verified during the different development stages.

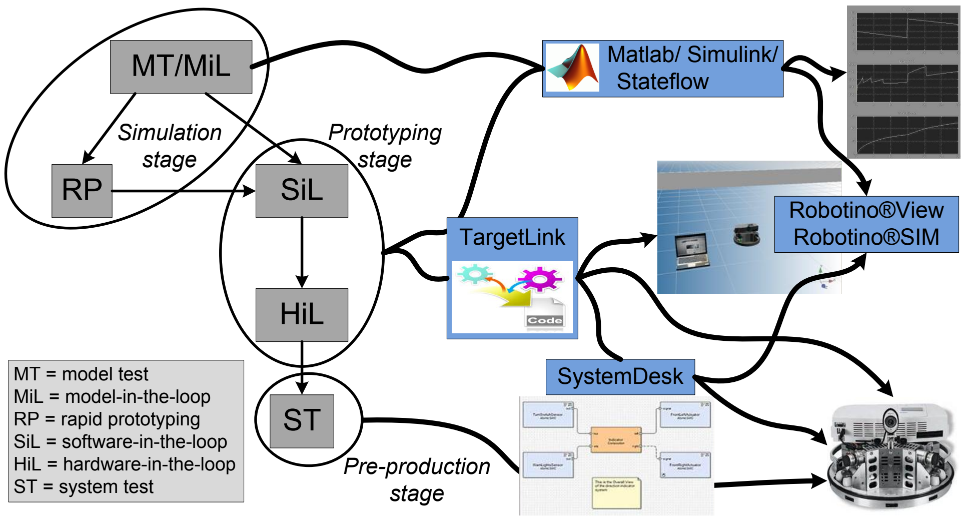


Figure 2: On the left are the three development stages according to [4] in combination with our toolchain during software and system development on the right.

3.1 Simulation Stage

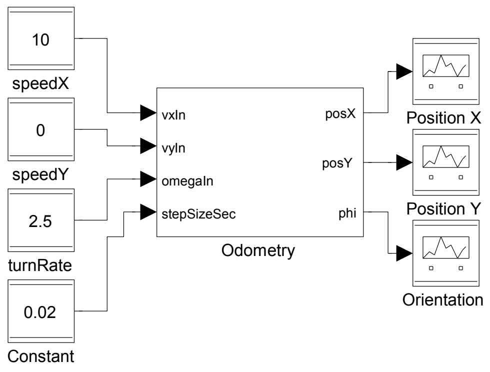


Figure 3: Odometry in MATLAB, which calculates the position from fix drive speed and turn rate.

Individual functions as well as composed behavior, resulting from multiple individual functionalities, are the subject of the simulation stage. Data flow models in the form of block diagrams (e.g., MATLAB/Simulink) usually in combination with control flow models like Statecharts (e.g. Stateflow) are used [6]. Normally, function development is done independent from platform specific limitations (memory capacity, floating point calculation or effects resulting from discretization). Additionally, environment specific signals and other real sensor values (e.g. produced by A/D, D/A converter or specific communication messages) are ignored for the sake of simplicity. The goal of the simulation stage is to prove that the functional behavior can work and as a result provides a first proof of concept for control algorithms.

As depicted in Fig. 2 and according to the aspect (IV), we mainly use the MATLAB tool suite including the Simulink and Stateflow extension in this development stage. Let us consider the MATLAB model shown in Fig. 3, as an example modeling the functionality of an odometry. It reads data from moving sensors to calculate changes in the position over time according the actual orientation and movement speed of the robot. In the simulation stage, such a model is used to apply a so-called *model test (MT)*, where individual functionalities can be simulated sending static input values to the model (e.g., drive speed and turn rate of the robot as in Fig. 3) and plotting the computed output values as shown in Fig. 4. These one-shot/ one-way simulations are typical for the MT step and do not consider the interaction with the environment or a plant model. More complex behavior is constructed and validated in the form of individual functionalities and running *model-in-the-loop (MiL)* simulations [4] including preliminary environment models of the plant. At this point in time, feedback simulations validate the developed functionality considering the dynamic behavior of the environment. Outputs are sent to the plant model, which itself gives feedback used as input for the function blocks in the next iteration of the MiL simulation. In such a manner, the overall control law can be validated concerning basic constraints like stability, safety or reliability of the system (VII).

In the case of robotic systems, such a plant model can be represented at different levels, e.g., by using models representing a single sensor, the behavior of a single robot using multiple sensors or in the case of a complex simulation realizing the behavior of multiple robots as well as relevant parts of the logical and/or physical environment. Using such a plant model in the context of a MiL simulation, we must bridge the gap between our MATLAB models and the provided model of the plant (VI). For this purpose, on the one hand, we use the *RobotinoSim* simulator in combination with the graphical *RobotinoView* environment[9] to create plant models (cf. the upper path from the simulation stage in Fig. 2). Therefore, we implemented a block library for MATLAB in our development environment, which allows access to sensors (e.g., distance sensors, bumper, incremental encoder, electrical motors) and actuators according to requirement (V). The sensors and actuators can be accessed individually inside a MiL simulation supporting the validation of the models (VII). The RobotinoSim simulator provides optimal sensor values excluding effects such as sensor noise. Therefore, on the other hand, we can access the HW of the robot directly via a wireless LAN connection. Due to the fact that we use the concrete HW in this simulation setting, we could verify our functionalities and control algorithm with real sensor values including measure errors and sensor noise.

To sum it up, on the right in Fig. 2, one can follow the toolchain used via the flow arrows.[10] However, we are not limited to the RobotinoSim tool in our development approach. We use this tool to show the proof of concept, but in general it is possible to create block libraries in MATLAB or use existing ones[11] for other robots, simulation frameworks or individual sensors/ actuators.

3.2 Prototyping Stage

The focus of this stage changes from design to implementation. While in the simulation stage models are the main artifacts, in this stage the source code plays a major role. In the following, we show how to support the prototyping stage at the level of more isolated functional parts as well as at the level of the system behavior by using the professional, commonly used tools of the automotive domain.

Function Level – TargetLink: In the automotive domain, code generators are commonly used to derive an implementation for the specific target platform. Usually, the models from the simulation stage are directly used or refined until a code generation step is possible. In our development environment, the tool *TargetLink* from dSPACE is fully integrated into MATLAB and can automatically derive the implementation from behavior models in form of C-Code. In this step, we use the same MATLAB blocks as discusses in Section 3.1. So, we are able to seamlessly migrate (VI) our functions and control algorithm from the model level, realizing continuous behavior, to the implementation level, realizing a discrete approximation of the original continuous behavior.[12] We can configure several characteristics of the desired target platform/ HW.

[9] In the following, we only mention the simulator, but we always use both tools together in combination. Tools see: `www.festo-didactic.com`

[10] The described RP flow to the real robot is not shown in the figure.

[11] For example this toolbox: `http://petercorke.com/Robotics_Toolbox.html`

[12] Discretization is applied at different levels. E.g., fixed point variables are used for the implementation at the data level or time continuous differential equations are mapped to discrete execution intervals at the timing level. For further details compare [4].

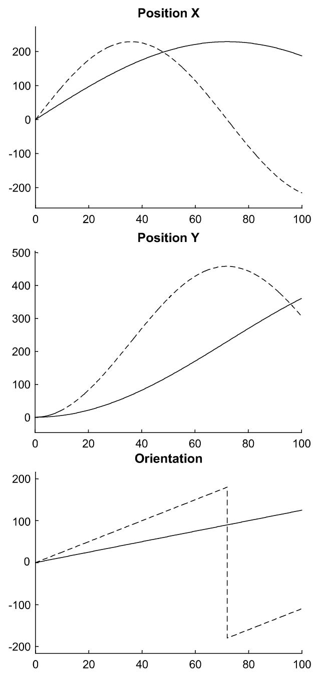


Figure 4: MiL (dashed line) and SiL simulation values of the odometry block.

Software-in-the-loop (SiL) simulation is a first step from the pure model execution to a code-based testing. Certain assumptions can be validated by replacing more and more models with code. While still executing the software on a host pc and not on the real HW, different effects can be analyzed, which result from chosen configuration parameters during code generation. Just as in the MiL simulation case, a SiL simulation can be applied in MATLAB using the generated source code instead of the original model. The developer can switch between the MiL and SiL simulation mode in MATLAB. Therefore, he can easily compare the simulation results. Fig. 4, for example shows the monitored results of the position as well as the orientation from the MiL and SiL simulation runs of the odometry. The simulations run against the RobotinoSim simulator. In the MiL run (dashed line), appropriate values for the actual position and orientation are calculated. Because of rounding (discretization) effects in the SiL run, the calculated values are much too low. So, the difference between pure model simulation and code generation becomes visible.

The problem in this special example could be fixed by choosing different values for the discretization over time. Calculating the position each 0.02 time units (corresponds to a scheduling with a period of 20 ms, cf. the constant value in Fig. 3) leads to very small offsets in the position, which is often rounded to zero due to discretization. After we identified the problem, we could easily fix it in the model. Instead of a 20 ms period, we double it to 0.04 time units for calculating the position. After generating code again, we could validate our assumption, which leads to a new requirement to trigger the functionality of the odometry with a period of 40 ms. Using code generators for automatically deriving the implementation realizing the behavior of initially created models support the seamless migration from the model level to the implementation level as well as allow to analyze effects arising from the implementation. Therefore, we cover the aspects IV, VI, and VII developing robotic systems at this point.

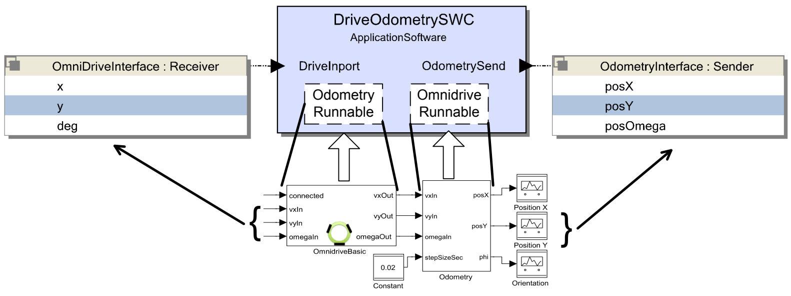


Figure 5: Mapping from MATLAB models to SWCs.

System Level – SystemDesk:

For more complex system behavior resulting from the composition of multiple individual functionalities, we use the component-based architecture provided by the AUTOSAR framework. Individual functionalities provided by the MATLAB models are mapped on components such as those depicted in Fig. 5. The generated source code from TargetLink is mapped into the AUTOSAR SWC in the form of so-called *Runnables*.

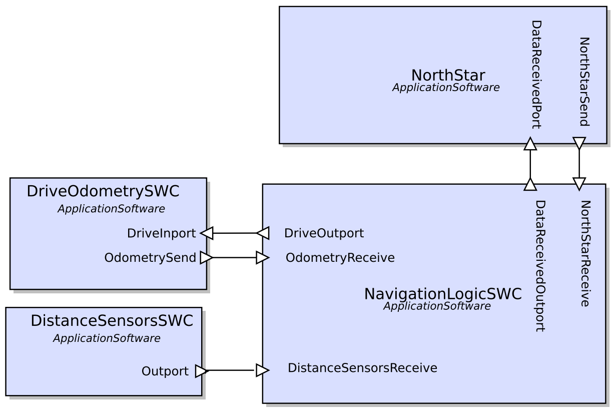


Figure 6: SWC architecture in AUTOSAR.

So, the same C-Code as in the SiL simulation is used and thus, a seamless integration (VI) of individual functions into the overall system behavior is achieved. In our example, we split the MATLAB model into two Runnables, namely *OdometryRunnable* and *OmnidriveRunnable*.[13] The SWC communicates to other ones over well defined ports. Furthermore, the input and output values are mapped to AUTOSAR interfaces with data entries and types respectively.

The AUTOSAR architecture consists of four SWCs[14] (see Fig. 6). It realizes the autonomous movement of the Robotino robot and includes the SWCs *DriveOdometrySWC*, *DistanceSensorsSWC*, *NorthStar* and *NavigationLogicSWC*. Each SWC provides the functionality such as that described previously in Section 2.1.

System Configuration: In addition to the architecture modeling and the separation of functions in different SWCs, SystemDesk supports a task specification for the underlying operating system. Runnables can be mapped to different tasks. Furthermore,

[13]This separation allows us to trigger the two Runnables with different periods.

[14]Due to a better understanding, we choose this simple excerpt of a larger architecture.

several task activation events including periodic and sporadic ones are supported and additional scheduling information like periods and priorities can be modeled.

For a system simulation, one has to specify a concrete AUTOSAR conform system configuration, which includes 1) a set of tasks, each consisting of one or more Runnables, 2) one or more electronic control units, which are specialized processors, and 3) communication capabilities (buses) with a concrete mapping of messages, which have to be exchanged. In the following, we describe the first point in more detail using our running example.

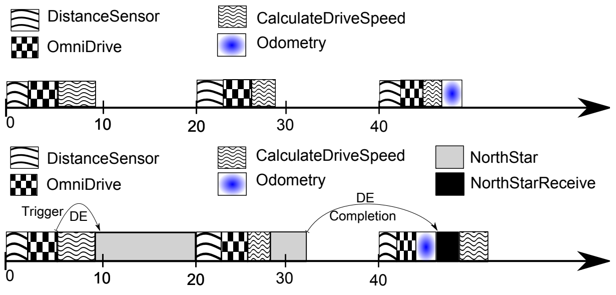


Figure 7: Upper time line: scheduling of hard real-time functions. Lower time line: combined hard and soft real-time scheduling.

The Runnables *DistanceSensor*, *OmniDrive* and *CalculateDriveSpeed* are mapped to an OS task, which is executed with a period of 20 ms. A second task with the derived period of 40 ms contains the Runnable *Odometry* (cf. Section 3.2). The resulting execution of the Runnables and the schedule of the tasks is depicted in the upper time line of Fig. 7. These four basic functions run under hard real-time constraints, so we can be sure that all deadlines are met.

After adding more information to satisfy points 2) and 3), SystemDesk can realize a system simulation. It automatically generates the required simulation framework code according to the AUTOSAR standard, e.g., the RTE, messages, task bodies and trigger events. Furthermore, existing source files, generated by TargetLink (from the MATLAB models), are compiled and linked into the tasks. The complete system runs in a special simulation environment inside the SystemDesk tool and considers the HW configuration as well as OS task specifics. Again, this simulation is executed on a host PC and thus belongs to the prototyping stage. As depicted in Fig. 2,we can validate the overall system behavior in the three following scenarios considering the aspects (VI, VII, and VIII): First, we can monitor different output values, messages and variables inside the simulation environment itself. Second, we can connect the Robotino simulation environment as a plant model, which interacts with the SystemDesk tool. Finally, we are able to replace the plant simulator with the real robot. Therefore, we have to establish a W-LAN connection for the communication and to access the real sensors as well as actuators. Unfortunately, this unpredictable connection can destroy the timing behavior of the simulation, although the simulator tries to keep all deadlines. If we find errors during our validation processes, we can change the configuration, architecture or communication possibilities in SystemDesk and run our simulations again. Furthermore, we are able to re-import SWCs into MATLAB and therefore, switch between the different development stages.

According to the stage description in [4], *Hardware-in-the-loop (HiL)* simulations can be applied in the prototyping stage too. In these kind of simulations, the "unlimited" execution and testing hardware is often replaced by special evaluation boards with additional debugging and calibration interfaces, which are similar to the final hardware configuration. Due to limitations of our robot laboratory and missing evaluation boards, we do not use such HiL simulations. However, the integration of such boards can be carried out easily in the SystemDesk tool by changing the HW specification during the system configuration step.
Adaptation to Robotic Systems: In contrast to classic (hard) real-time applications in the domain of automotive embedded systems, robotic systems must realize functionalities, for which worst-case execution times (WCETs) are hard or impossible to predict. As a result, the integration of such behavior can only guarantee soft real-time constraints. In our application example, we use the NorthStar sensor, which is accessed via a serial USB port. Due to the fact that we use the default Linux OS driver, the timing behavior is unpredictable for that port. Additionally, we implement the navigation logic, which uses this NorthStar sensor, with library function from the MRPT library (cf. Section 2.1) for maintaining the map information of the explored topology. This includes the dynamic instantiation of an unknown number of C++ objects (classes) at runtime, what hinders the WCET estimation, too. Therefore, the WCET can rarely be estimated at the range of a few milliseconds.

Due to te fact that AUTOSAR does not directly support such a combination of soft and hard real-time behavior, we need to adapt the framework to realize it in such a way that: (1) the schedule guarantees the preservation of hard real-time constraints for the basic functionality and (2) the communication between soft and hard real-time functionality is achieved as such that only consistent data is read.

In the first step, we separate the hard and soft real-time functions/ Runnables and map them onto different OS tasks. A soft real-time task can be configured with a lower priority in such a way that it will be interrupted by all hard real-time tasks with a higher priority. Following this development guideline achieves the first requirement (1). For the second one, we use special data events (DE) in combination with Sender/Receiver-interfaces of the AUTOSAR standard. Such events can be used to trigger the execution of Runnables inside an OS task. A DE is sent from the hard real-time task (resp. Runnable) to trigger the execution of the soft real-time Runnable. The interruptible soft real-time function produces another DE, iff, the requested output data is in a consistent state (2). The hard real-time task can read the data in its next period and triggers the soft real-time function again if required.

The lower time line in Fig. 7 illustrates the combined scheduling of soft and hard real-time tasks. The soft real-time task is triggered via a DE generated by the *OmniDrive* Runnable. During execution, it is preempted in order to ensure the timing deadlines of the other hard real-time Runnables. After the *NorthStar* Runnable has finished its execution, it sends another DE to indicate completion, which includes that the consistent data results can be read in the next period of the *OmniDrive* Runnable.

Our described development approach supports the prototyping stage of robotic systems very well. We are able to incrementally refine more and more information to specify the system while seamlessly integrating artifacts of the previous stages (VI). Activities like function development and system configuration can be applied in a round-trip engineering approach (I, II). First, we develop the control functions in MATLAB (II). Afterwards, we generate code using the TargetLink code generation capabilities (IV). At this point, we can manually integrate additional, arbitrary functionality in C/ C++or use existing libraries (V). As soon as sufficient code artifacts and libraries are provided, we are able to use the code generation and simulation capabilities of the SystemDesk tool (IV, VII). Existing SWCs, e.g., developed in a previous project can be seamlessly integrated into the system architecture and new components can be exported as library elements for other projects. Additionally, we have shown the idea of creating a combination of hard and soft real-time tasks using the AUTOSAR framework during this stage (VIII).

3.3 Pre-Production Stage

Within the pre-production stage, usually, a prototype of the real system is built. This prototype is tested against external environmental influences (such as temperature, vibration or other disturbances). The goal of this stage is to prove whether all requirements and constraints are still met on the real HW. During this last integration of all components and system parts, upcoming problems should be fixed as early as possible and before the final production of the product starts [4]. In our setting, we did not built any HW prototypes. Instead, we integrate the overall functions, components as well as the generated RTE and tasks to a complete system, compile and run it on the target processor of the robot[15]. So in this last step, we have no simulation semantic and W-LAN connection to other tools. We can fully operate the behavior of the robot in hard real-time. For verification, we use some hard real-time logging mechanism of the robot OS. Furthermore, we can change the hardware composition of the robot by adding or removing special sensors and actuators (see Section 2.1).

4 Related Work

Tackling the complexity of robotic and other embedded systems, we found a great deal of previous work covering partial aspects of developing such systems. According to our found aspects in Section 1 and our focus on the automotive domain, we combine the existing development methodology from [4] and the AUTOSAR standard. We evaluate our approach in a robotic production scenario using the component-based AUTOSAR architecture [12]. Other frameworks often cover only parts of the found aspects. RT-Middleware [1, 2] and ORCA [5] focus on the specification of components including interfaces and ports (aspects IV, V, and VII). They lack the integration of an overall methodology as well as architecture specification. Very similar to the AUTOSAR approach but with the focus on the robotic domain is the MOAST framework [3], which covers the points II, III, IV, and partially VII. However, a seamless integration of an overall methodology and the support for different tools is missing. A good comparison with other frameworks can be found in [8].

[15]We can automatically transform AUTOSAR compliant applications to the RTAI Linux.

In the embedded world, testing and simulation are the major activities to verify the behavior of the system [4]. We have made intensive use of the MATLAB, Robotino and SystemDesk simulators. However, other simulators like Gazebo [7] or Webots [9] are applicable as well.

Furthermore, there are other tools for modeling and simulation of AUTOSAR conform parts of the system architecture as the Real-Time Workshop(RTW) (MATLAB extension) from the MathWorks company.[16] The RTW extension is limited to component functionality and interfaces. The overall system architecture description is needed beforehand [11]. Parts of this description can be built by the Volcano Vehicle Systems Architect[17] (VSA), which can import and export AUTOSAR conform architecture description [10]. However, all these different tools can be used instead of the tools presented in this report, if the integration in the overall methodology as well as the support for the different development stages is guaranteed.

Considering real-time constraints and combining hard and soft real-time tasks are important because of the support for library functionalities in different use-cases. For example, our navigation logic in this report cannot be done in a predictive amount of time. Combining soft and hard real-time guarantees (1) a basic hard real-time behavior of the robotic system and (2) supports the development of complex algorithm and higher system components. Existing robotic frameworks as the Robot Operating System[18] or Microsoft Robotics Studio[19] are well established for developing complex robotic systems. They have drawbacks concerning the integration of hart real-time constraints.

5 Conclusion

We have shown in this report an overall methodology (I) along with different exemplary development activities as well as artifacts on different levels of abstraction (II, III). We know that not all tasks can be executed in HRT. Therefore, we have shown the idea of combining different hard and soft real-time tasks into the overall system using the AUTOSAR approach (VIII). Furthermore, we are able to integrate several tools and external libraries into our overall toolchain (IV, V, VI). However, we are not limited to the tools we show in this report. This provided flexibility is stabilized by a clear structure of different development stages (III) allowing a round-trip engineering for different functions, the integration of components as well as the simulation and testing of the development artifacts to the point of the complete system on the target platform. Therefore, we adapt ideas of the automotive domain to the development of robotic systems.

As future work, we want to build a complex robot production scenario applying the proposed methodology of this report and evaluate the interaction of soft and hard real-time system parts.

[16] `www.mathworks.com/embedded-systems/`
[17] `www.mentor.com/`
[18] `www.ros.org`
[19] `www.microsoft.com/robotics/`

Acknowledgment

This report is based on joint work with Stefan Neumann and Falk Benke. For more information see [13].

References

[1] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, and Y. Woo-Keun. RT-middleware: distributed component middleware for RT (robot technology). In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3933–3938, 2005.

[2] Noriaki Ando, Takashi Suehiro, and Tetsuo Kotoku. A software platform for component based rt-system development: Openrtm-aist. In *Proceedings of the 1st International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, SIMPAR '08, pages 87–98, Berlin, Heidelberg, 2008. Springer.

[3] Stephen Balakirsky, Frederick M. Proctor, Christopher J. Scrapper, and Thomas R. Kramer. A mobile robot control framework: From simulation to reality. In *Proceedings of the 1st International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, SIMPAR '08, pages 111–122, Berlin, Heidelberg, 2008. Springer.

[4] Bart Broekman and Edwin Notenboom. *Testing Embedded Software*. Wesley, 2003.

[5] A. Brooks, T. Kaupp, A. Makarenko, S. Williams, and A. Oreback. Towards component-based robotics. In *International Conference on Intelligent Robots and Systems*, pages 163–168, 2005.

[6] Holger Giese, Stefan Neumann, Oliver Niggemann, and Bernhard Schätz. Model-Based Integration. In *Model-Based Engineering of Embedded Real-Time Systems, Dagstuhl Castle, Germany*, volume 6100 of *LNCS*, pages 17–54. Springer, 2011.

[7] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2149–2154, 2004.

[8] Luis Manso, Pilar Bachiller, Pablo Bustos, Pedro Núñez, Ramón Cintas, and Luis Calderita. Robocomp: a tool-based robotics framework. In *Proceedings of the 2nd international conference on Simulation, modeling, and programming for autonomous robots*, SIMPAR'10, pages 251–262, Berlin, Heidelberg, 2010. Springer.

[9] O. Michel. Webots: Professional Mobile Robot Simulation. *International Journal of Advanced Robotic Systems*, 1:39–42, 2004.

[10] G. Sandmann and M Seibt. AUTOSAR-Compliant Development Workflows: From Architecture to Implementation - Tool Interoperability for Round-Trip Engineering and Verification and Validation. Technical Report 2012-01-0962, SAE International, 2012.

[11] G. Sandmann and R Thompson. Development of AUTOSAR Software Components within Model-Based Design. Technical Report 2008-01-0383, SAE International, 2008.

[12] `http://www.autosar.org/`. AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf, 2011. page id: 94ju5.

[13] Sebastian Wätzoldt, Stefan Neumann, Falk Benke, and Holger Giese. Integrated Software Development for Embedded Robotic Systems. In *Proceedings of the 3rd International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, volume 7628, pages 335–348, October 2012.

ISBN 978-3-86956-256-8
ISSN 1613-5652