BSR winter school
Big Software on the Run: Where Software meets Data

October 23-28, Ede, The Netherlands

Tutorials & Poster abstracts

Preface

The winter school on “Big Software on the Run: where Software meets Data” (BSR 2016) was held in Ede, the Netherlands, October 23-28, 2016. In this volume, you find the tutorials’ and posters’ abstracts presented at the winter school.

The BSR winter school is organized in the context of the 3TU.BSR project (`http://www.3tu-bsr.nl/`), a joint endeavor of the Netherlands’ three technical universities, Eindhoven University of Technology, Delft University of Technology and University of Twente. Six research groups, specialized in different areas of software analytics are joining their forces and conducting a multi-disciplinary research on the in vivo analysis of large software systems.

Software systems have grown increasingly large and complex in today’s highly interconnected world. Communication, production, healthcare, transportation and education all increasingly rely on *Big Software*. This increasing dependence makes reliable software systems a major concern and stresses the need for effective prediction of software failures. Since software is evolving and operates in a highly dynamic and changing environment, it becomes difficult if not impossible to anticipate all problems at design-time. Because trends like Devops blur the lines between development and deployment, system validation must shift from testing and verification at design time to monitoring at run time: only at run time do we know the application context and can we carry out the appropriate analyses and – if needed – interventions. This paradigm shift requires new forms of empirical investigation that go far beyond the common practice of collecting error messages and providing software updates.

The 3TU.BSR project is all about fully automatic monitoring and diagnosis. By logging all system events and comparing the traces thus obtained with a specification or reference model, we can check system or application correctness: does it behave according to the specification? The aim is to develop methods that diagnose system faults automatically, and to synthesize recommended actions: which components need to be debugged, which components need replacing, which components require more testing with what inputs? And when the system needs to be reconfigured, which components and subsystems should we reconfigure?

These questions, as well as current research on software analytics in general, were addressed at the BSR winter school, which brought together university researchers, including PhD students and postdoctoral-fellows as well as professionals from industry. The school program offered twelve tutorial sessions of one hour and half each. All tutorials were given by renowned researchers invited from Europe, US and Canada as well as senior professors involved in the 3TU.BSR project.

The program of the school also included poster sessions to which participants were asked to submit a poster and a short abstract. The participant of the best poster as judged by the BSR scientific committee was invited for an oral presentation of half an hour. The program further included three hands-on sessions in which three mini-challenges were organized. Three awards were granted to the winners of the challenges.

We are grateful to all lecturers and participants of the school for their enthusiasm and active participation. Our gratitude is also due to all members of the 3TU.BSR project for their hard work in organizing and preparing excellent material for the school.

October 2016

Nour Assy
Maikel Leemans

Table of Contents

Part I

Talks and Tutorials

Making Sense From Software Using Process Mining

Wil M.P. van der Aalst

Department of Mathematics and Computer Science
Eindhoven University of Technology, Eindhoven, the Netherlands
w.m.p.v.d.aalst@tue.nl

Abstract

Software-related problems have an incredible impact on society, organizations, and users that increasingly rely on information technology. Since software is evolving and operates in a changing environment, one cannot anticipate all problems at design-time. We propose to use process mining to analyze software in its natural habitat. Process mining aims to bridge the gap between model-based process analysis methods such as simulation and other business process management techniques on the one hand and data-centric analysis methods such as machine learning and data mining on the other. It provides tools and techniques for automated process model discovery, conformance checking, data-driven model repair and extension, bottleneck analysis, and prediction based on event log data. Process discovery techniques can be used to capture the real behavior of software. Conformance checking techniques can be used to spot deviations. The alignment of models and real software behavior can be used to predict problems related to performance or conformance. Recent developments in process mining and the instrumentation of software make this possible. This lecture provides pointers to the state-of-the-art in process mining and its application to software.

Introduction to Data Visualization

Jarke J. van Wijk

Department of Mathematics and Computer Science
Eindhoven University of Technology, Eindhoven, The Netherlands
j.j.v.wijk@tue.nl

Abstract

Data Visualization concerns the use of interactive computer graphics to obtain insight in large amounts of data. The aim is to exploit the unique capabilities of the human visual system to detect patterns, structures, and irregularities, and to enable experts to formulate new hypotheses, confirm the expected, and to discover the unexpected. In this lecture an overview of the field is given, illustrated with examples of work from Eindhoven, covering a variety of different data and application domains. The focus is on information visualization and visual analytics. We study how large amounts of abstract data, such as tables, hierarchies, and networks can be represented and interacted with. In many cases, combinations of such data have to be dealt with, and also, the data is often dynamic, which brings another big challenge. Typical use cases are how to understand large software systems, how to analyze thousands of medicine prescriptions, and how to see patterns in huge telecom datasets. In visual analytics, the aim is to integrate methods from statistics, machine learning, and data mining, as well as to support data types such as text and multimedia, and to support the full process from data acquisition to presentation.

Beyond Mixed Methods: Why Big Data Needs Thick Data

Margaret-Anne Storey

University of Victoria, Victoria, BC, Canada
mstorey@uvic.ca

Abstract

Software analytics and the use of computational methods on "big" data in software engineering is transforming the ways software is developed, used, improved and deployed. Software engineering researchers and practitioners are witnessing an increasing trend in the availability of diverse trace and operational data and the methods to analyze the data. This information is being used to paint a picture of how software is engineered and suggest ways it may be improved.

Although, software analytics shows great potential for improving software quality, user experience and developer productivity, it is important to remember that software engineering is inherently a socio-technical endeavour, with complex practices, activities and cultural aspects that cannot be externalized or captured by tools alone. Consequently, we need other methods to surface "thick data" that will provide rich explanations and narratives about the hidden aspects of software engineering.

In this tutorial, we will explore the following questions:

– What kinds of risks should be considered when using software analytics in automated software engineering?
– Are researchers and practitioners adequately considering the unanticipated impacts that software analytics can have on software engineering processes and stakeholders?
– Are there important questions that are not being asked because the answers do not lie in the data that are readily available?
– Can we improve the application of software analytics using other methods that collect insights directly from participants in software engineering (e.g., through observations)?
– How can we combine or develop new methods that combine "big data" with "thick data" to bring more meaningful and actionable insights?

We will discuss these questions through specific examples and case studies.

Active Learning of Automata

Frits Vaandrager

Institute for Computing and Information Sciences
Radboud University, Nijmegen, The Netherlands
F.Vaandrager@cs.ru.nl

Abstract

Active automata learning is emerging as a highly effective technique for obtaining state machine models of software components. In this talk, I will give a survey of recent progress in the field, highlight applications, and identify some remaining research challenges.

Supporting Data-Centered Software Development

Robert DeLine

Microsoft Research, Redmond, Washington 98052
rdeline@microsoft.com

Abstract

Modern software consists of both logic and data. Some use of the data is “back stage”, invisible to customers. For example, teams analyze service logs to make engineering decisions, like assigning bug and feature priorities, or use dashboards to monitor service performance. Other use of data is “on stage” (that is, part of the user experience), for example, the graph algorithms of the Facebook feed, the machine learning behind web search results, or the signal processing inside fitness bracelets. Today, working with data is typically assigned to the specialized role of the data scientist, a discipline with its own tools, skills and knowledge. In the first part of talk, I’ll describe some empirical studies of the emerging role of data scientists, to convey their current work practice.

Over time, working with data is likely to change from a role to a skill. As a precedent, software testing was originally the responsibility of the role of software testers. Eventually, the prevalence of test-driven development and unit tests turned testing into a skill that many developers practice. Will the same be true of the role of data scientists? Is it possible to create tools to allow a wide range of developers (or even end users) to analyze data and to create data-centered algorithms? The second part the talk will demo emerging tools that take initial steps toward democratizing data science.

Challenges in Automotive Software Development Running on Big Software

M.G.J. van den Brand

Eindhoven University of Technology, Groene Loper 5,
NL-5612 AZ Eindhoven, The Netherlands
m.g.j.v.d.brand@tue.nl

1 Abstract

The amount of software in vehicles has increased rapidly. The first lines of code in a vehicle were introduced in 1970s, nowadays over 100 million lines of code is no exception in a premium car. More and more functionality is realised in software and software is the main innovator at this moment in automotive domain. The amount of software will increase because of future innovations, think of adaptive cruise control, lane keeping, etc., which all leads to the ultimate goal of autonomous driving.

Automotive systems can be categorized into vehicle-centric functional domains (including powertrain control, chassis control, and active/passive safety systems) and passenger-centric functional domains (covering multimedia/ telematics, body/comfort, and Human Machine Interface). From these domains, powertrain, connectivity, active safety and assisted driving are considered major areas of potential innovation. The ever increasing amount of software to enable innovation in vehicle-centric functional domains requires even more attention to assessment and improvement of the quality of automotive software. This is because software-driven innovations can come with software defects, failures, and vulnerability for hackers attacks. This can be observed by the enormous amounts of recalls lately, quite a few are software related.

Functional safety is another important aspect of automotive software. Failures in the software may be costly, because of the recalls, but may even be life threatening. The failure or malfunctioning of an automotive system may result in serious injuries or death of people. A number of functional safety standards have been developed for safety-critical systems; the ISO26262 standard is the functional safety standard for the automotive domain, geared towards passenger cars. A new version of the ISO26262 standard will cover trucks, busses and motor cycles as well. The automotive industry start to apply these safety standards as guidelines

in their development projects. However, compliance with these standards are still very costly and time consuming due to huge amount of manual work.

In the last six years we have been doing research in the domain of automotive software development. We have investigated how to evaluate the quality of automotive software architectures. Software in the automotive domain is mainly developed in Matlab/Simulink and more recently SysML; from the developed models C code is generated. Of course, some functioanality is developed directly in C. We have used general software quality metrics frameworks to establish the quality of Matlab/Simulink and SysML models [1] and automotive software architectures in general [2, 3]. In the area of functional safety we have applied model driven techniques to support functional safety development process and safety assurance. We have done research on how to apply ISO26262 for functional safety improvement [4]. Furthermore, we have developed a meta-model for the ISO26262 standard; generated based on this meta-model tooling for safety case construction and assessment [5–8].

Recent research focuses on the integration of functional safety standards in the development process [4]. In the future we want to investigate how functional safety related requirements can be covered directly in the automotive architecture at an early stage. We will do this in relation to research in the field of autonomous driving.

References

1. Yanja Dajsuren, Mark G. J. van den Brand, Alexander Serebrenik, and Serguei A. Roubtsov. Simulink models are also software: modularity assessment. In *Proceedings of the 9th international ACM SIGSOFT conference on Quality of Software Architectures, QoSA 2013, part of Comparch '13 Federated Events on Component-Based Software Engineering and Software Architecture, Vancouver, BC, Canada, June 17-21, 2013*, pages 99–106, 2013.
2. Yanja Dajsuren, Mark van den Brand, Alexander Serebrenik, and Rudolf Huisman. Automotive adls: A study on enforcing consistency through multiple architectural levels. In *Proceedings of the 8th International ACM SIGSOFT Conference on Quality of Software Architectures*, QoSA '12, pages 71–80, New York, NY, USA, 2012. ACM.
3. Yanja Dajsuren, Christine M. Gerpheide, Alexander Serebrenik, Anton Wijs, Bogdan Vasilescu, and Mark G. J. van den Brand. Formalizing correspondence rules for automotive architecture views. In *QoSA'14, Proceedings of the 10th International ACM SIGSOFT Conference on Quality of Software Architectures (part of CompArch 2014), Marcq-en-Baroeul, Lille, France, June 30 - July 04, 2014*, pages 129–138, 2014.
4. Arash Khabbaz Saberi, Yaping Luo, Filip Pawel Cichosz, Mark van den Brand, and Sven Jansen. An Approach for Functional Safety Improvement of an Existing Auto-

motive System. In *Systems Conference (SysCon), 9th Annual IEEE International*, pages 277–282, 2015.
5. Yaping Luo, Mark van den Brand, Luc Engelen, and Martijn Klabbers. A modeling approach to support safety assurance in the automotive domain. In *Progress in Systems Engineering*, pages 339–345. Springer, 2015.
6. Yaping Luo, Mark van den Brand, Luc Engelen, John M. Favaro, Martijn Klabbers, and Giovanni Sartori. Extracting models from ISO 26262 for reusable safety assurance. In *Safe and Secure Software Reuse - 13th International Conference on Software Reuse, ICSR 2013, Pisa, Italy, June 18-20. Proceedings*, pages 192–207, 2013.
7. Yaping Luo, Mark van den Brand, Luc Engelen, and Martijn Klabbers. From conceptual models to safety assurance. In *Conceptual Modeling - 33rd International Conference, ER 2014, Atlanta, GA, USA, October 27-29, 2014. Proceedings*, pages 195–208, 2014.
8. Yaping Luo and Mark van den Brand. Metrics design for safety assessment. *Information & Software Technology*, 73:151–163, 2016.

Mining GitHub for Fun and Profit

Georgios Gousios

Delft University of Technology, Delft, the Netherlands
g.gousios@tudelft.nl

Abstract

Modern organizations use telemetry and process data to make software production more efficient. Consequently, software engineering is an increasingly data-centered scientific field. With over 30 million repositories and 10 million users, GitHub is currently the largest code hosting site in the world. Software engineering researchers have been drawn to GitHub due to this popularity, as well as its integrated social features and the metadata that can be accessed through its API. To make research with GitHub data approachable, we created the GHTorrent project, a scalable, off-line mirror of all data offered through the GitHub API. In our lecture, we will discuss the GHTorrent project in detail and present insights drawn from using this dataset in various research works.

Scalable Model Analysis

Jaco van de Pol

University of Twente, Enschede, The Netherlands
J.C.vandePol@utwente.nl

Abstract

Software is a complex product. This holds already for its static structure, but even more so for its dynamic behaviour. When considering Big Software on the Run, the role of models is changing fast: instead of using them as a blueprint (like engineers) we now use models to understand running software (like biologists). More extremely, we are now using machine learning to obtain models of complex software systems automatically. However, adapting a classic motto, we should “spend more time on the analysis of models, than on collecting logs, and learning and visualising models” [1].

We will discuss algorithms and tools for studying models of the dynamic behaviour of systems. Since their complex behaviour is essentially modeled as a giant graph, we will review various high performance graph algorithms. In particular, we will cover LTL as a logic to specify properties of system runs, and symbolic and multi-core model checking as the scalable means to analyse large models. We will illustrate this with Petri Nets modelling software systems, and timed automata modelling biological systems.

References

1. Variation on ”Spend more time working on code that analyzes the meaning of metrics, than code that collects, moves, stores and displays metrics - Adrian Cockcrof”, cited by H. Hartmann in Communications of the ACM 59(7), July 2016

Reliable Concurrent Software

Marieke Huisman

University of Twente, Enschede, the Netherlands
M.Huisman@utwente.nl

Abstract

Concurrent software is inherently error-prone, due to the possible interactions and subtle interplays between the parallel computations. As a result, error prediction and tracing the sources of errors often is difficult. In particular, rerunning an execution with exactly the same input might not lead to the same error. To improve this situation, we need techniques that can provide guarantees about the behaviour of a concurrent program. In this lecture, we discuss an approach based on program annotations. The program annotations describe locally what part of the memory are affected by a thread, and what the expected behaviour of a thread is. From the local program annotations, conclusions can be drawn about the global behaviour of a concurrent application. In this lecture, we discuss various techniques to verify such annotations. If a high-correctness guarantee is needed, static program verification techniques can be used. However, in many cases, checking at run-time that the annotations are not violated is sufficient. We discuss both approaches, and we show in particular what are the challenges to use them in a concurrent setting.

How NOT to Analyze Your Release Process

Bram Adams

Ecole Polytechnique de Montréal Montreal, Canada
bram.adams@polymtl.ca

Abstract

The release engineering process is the process that brings high quality code changes from a developer's workspace to the end user, encompassing code change integration, continuous integration, build system specifications, infrastructure-as-code, deployment and release. Recent practices of continuous delivery, which bring new content to the end user in days or hours rather than months or years, require companies to closely monitor the progress of their release engineering process by mining the repositories involved in each phase, such as their version control system, bug/reviewing repositories and deployment logs. This tutorial presents the six major phases of the release engineering pipeline, the main repositories that are available for analysis in each phase, and three families of mistakes that could invalidate empirical analysis of the release process. Even if you are not working on release engineering, the mistakes discussed in this tutorial can impact your research results as well!

For more background: http://mcis.polymtl.ca/publications/2016/fose.pdf

Software Analysis: Anonymity and Cryptography for Privacy

Zekeriya Erkin

Cyber Security Group
Delft University of Technology, Delft, The Netherlands
z.erkin@tudelft.nl

Abstract

Validation in a big software system can be managed by dynamically analysing its behaviour. A software system in use occasionally reports information to developers about its status in the form of event logs. Developers use these information to detect flaws in the software and to improve its performance with the help of process mining techniques. Process mining generates process models from the collected events or checks the conformance of these events with an existing process model to identify flaws in the software. Algorithms in process mining to discover such process models rely on software behaviour through real event logs and indeed very useful for software validation. However, the existence of some sensitive information in the collected logs may become a threat for the privacy of users as seen in practice. In this talk, we present privacy enhancing technologies (PETs) for privacy-preserving algorithms for software modelling. We focus on different approaches, namely anonymization techniques and deploying advanced cryptographic tools such as homomorphic encryption for the protection of sensitive data in logs during software analysis. As a very new field of research, we introduce a number of challenges yet to be solved and discuss different aspects of the challenge in terms of level of privacy, utility and overhead introduced by deploying PETs.

Part II

Posters

A Cross-Organizational Process Mining Framework

Ü. Aksu, D.M.M. Schunselaar, H.A. Reijers

Vrije Universiteit Amsterdam, The Netherlands

Abstract

Software vendors offer various software products to large numbers of enterprises to support their organization, in particular Enterprise Resource Planning (ERP) software. Each of these enterprises use the same product for similar goals, albeit with different processes and configurations. Therefore, software vendors want to obtain insights into how the enterprises use the software product, what the differences are in usage between enterprises, and the reasons behind these differences. Cross-organizational process mining is a possible solution to address these needs, as it aims at comparing enterprises based on their behavior.

However, current cross-organizational process mining approaches use text matching techniques to relate terms between enterprises to find out differences in usage. This might cause inaccurate comparisons, because the same term can have different semantics across enterprises. For instance, a bank and a library can use the same term, loan, which carries different meanings for each of them. Therefore, in this research, we present a novel Cross-Organizational Process Mining Framework which takes as input, besides event logs, the semantics in an enterprise by means of a Business Semantics. The Business Semantic comprises of terms and their meaning in the enterprise. For a fair comparison between enterprises, our framework takes the information reflecting characteristics of the enterprise to determine if two enterprises are comparable. Within the framework, one is able to create a catalog of performance metrics to analyze the enterprises usage of a software product. In addition, the framework can monitor the (positive) effects of changes on processes using the catalog. An enterprise operating in a similar context might also benefit from the same changes. To accommodate these improvement suggestions, the framework creates an improvement catalog of observed changes.

Keywords: Cross-Organizational Process Mining

Taming the Herd: Statistical Analysis of Large Sets of Models

Önder Babur

Eindhoven University of Technology, The Netherlands
o.babur@tue.nl

Abstract. Many applications in Model-Driven Engineering involve processing multiple models, e.g. for comparing and merging of model variants into a common domain model. Despite many sophisticated techniques for model comparison, little attention has been given to the initial data analysis and filtering activities. These are hard to ignore especially in the case of a large dataset, possibly with outliers and sub-groupings. We would like to develop a generic approach for model comparison and analysis for large datasets; using techniques from information retrieval, natural language processing and machine learning. We are implementing our approach as an open framework and have so far evaluated it on public datasets involving domain analysis, repository management and model searching scenarios.

Statistical Analysis of Large Sets of Models

Model comparison is a fundamental operation in Model-Driven Engineering (MDE), used for tackling problems, such as model merging and versioning. Addressing the increasing size and complexity of models, many techniques have been developed based on pairwise and 'deep' comparison of models. However, another problem emerges when there is a large set (e.g., hundreds) of models to compare: the expensive pairwise techniques become inadequate and rather more scalable techniques are required.

This aspect of model comparison has been addressed in our previous work [1–3], noting the need for treating a large set of models holistically, e.g., for getting an overview of the dataset and potential relations, such as proximities, groups, and outliers. The proposed approach, i.e., model clustering, is inspired by document indexing and clustering in IR, notably Vector Space Model (VSM) with the major components of (1) a vector representation of term frequencies in a document, (2) *zones* (e.g., 'author' or 'title'), (3) weighting schemes, such as inverse document frequency (idf), and zone weights, (4) Natural Language Processing (NLP) techniques for handling compound terms and synonyms. The VSM allows transforming each document into an n-dimensional vector, thus resulting in an $m \times n$ matrix for m documents. Over the VSM, document similarity can be defined as the distance (e.g., Euclidean or Cosine) between vectors. These can be used for identifying similar groups of documents in the vector space.

We have applied this technique for models and developed a model clustering framework. As input to our framework, we obtain a set of models of same type, e.g. Ecore metamodels, UML class diagrams or feature models. The major workflow steps are:

– *Feature extraction:* The approach starts with the metamodel-based extraction of features from the models. The framework currently supports extracting typed identifiers of model elements (e.g., class or attribute names), metrics (e.g., number of attributes for a class) and an n-grams of those for capturing the structural relations between model elements (e.g., a bigram for $n = 2$, encoding two classes with an association in between) [4].
– *Feature comparison and NLP techniques:* The framework has several parameters on (1) using NLP techniques(tokenization, synonym checking, etc.) for comparing model identifiers, and (2) schemes for comparing features (e.g., whether to consider the types of model elements, or just ignoring them) and weighting features (e.g., whether to weight classes more than attributes).
– *Computing the VSM:* Here each model, consisting of a set of features, is compared against the maximal set of features collected from all models. The result is a matrix, similar to the term frequency matrix in IR, where each model is represented by a vector. Consequently, we reduce the model similarity problem into a distance measurement of the corresponding vectors.
– *Distance measurement and clustering:* The framework allows several parameters for (1) distance measures among vectors (e.g., Euclidean or Cosine), (2) different clustering algorithms, such as k-means and hierarchical agglomerative clustering (HAC), and (3) further clustering-specific parameters, such as linkage. The clusters in the dataset can be automatically computed to be used directly e.g., for the purpose of data selection and filtering. Also particularly for HAC, the resulting dendrogram can be visualized and inspected for an insight into the dataset.

References

1. Önder Babur, Loek Cleophas, Tom Verhoeff, and Mark van den Brand. Towards

statistical comparison and analysis of models. In *Proc. of the 4th Int. Conf. on Model-Driven Engineering and Software Development*, pages 361–367, 2016.
2. Önder Babur, Loek Cleophas, and Mark van den Brand. Hierarchical clustering of metamodels for comparative analysis and visualization. In *Proc. of the 12th European Conf. on Modelling Foundations and Applications, 2016*, pages 2–18, 2016.
3. Önder Babur. Statistical analysis of large sets of models. In *Proc. of the 31st IEEE/ACM Int. Conf. on Automated Software Engineering*, ASE 2016, pages 888–891, New York, NY, USA, 2016. ACM.
4. Önder Babur and Loek Cleophas. Using n-grams for the automated clustering of structural models. In *Proc. of the 43rd Int. Conf. on Current Trends in Theory and Practice of Computer Science*. Accepted for publication, 2017.

Attribution Required: Stack Overflow Code Snippets in GitHub Projects

Sebastian Baltes and Stephan Diehl

Software Engineering Group, University of Trier,Trier, Germany
research@sbaltes.com, diehl@uni-trier.de

Abstract

Stack Overflow (SO) is the largest Q&A website for developers, providing a huge amount of copyable code snippets. Using these snippets raises various maintenance and legal issues. The SO license requires attribution, i.e., referencing the original question or answer, and requires derived work to adopt the license. While there is a heated debate on SO's license model for code snippets and the required attribution, little is known about the extent to which snippets are copied from SO without proper attribution. This poster highlights issues resulting from careless usage of SO code snippets and presents first results from an empirical study analyzing the usage and attribution of such snippets.

Mining Software Process Lines

Fabian Rojas Blum

Computer Science Department
Universidad de Chile, Santiago, Chile
fblum@dcc.uchile.cl

1 Abstract

Software companies define their processes for planning and guiding projects. However, as these projects usually have different characteristics (e.g., small vs. medium team size), it is often not possible to define one process that fits all of them. For this reason they can define a *Software Process Line* [1]: a base process that represents the common elements, along with its potential variability. Specifying manually a SPrL is a time-consuming and error-prone task. However, even tough it is more expensive than specifying one process, the SPrL can be adapted to specific project context, minimizing the amount of extra work carried out by employees. As companies keep some kind of records about previous or current executions of their projects, this work proposes to use *Process Mining* [2] to discover a software process line.

There is a plethora of existing process discovery algorithms [2]. Each one of them has its own strengths and weaknesses. They produce processes in many different notations but they all share a common input: event log(s). Most of them are focused on discovering a single process. One particular exception is the *ETMc* [3] algorithm which is part of the Evolution Tree Miner framework. This framework works with the *process trees* representation as it has advantages over others and can be translated to other notations too. This algorithm takes as input a collection of event logs and discovers a *configurable process tree* which represents all the behavior with one specific configuration for each log. Thus, a configurable process tree contains different processes (similar to SPrL), but only one for each log.

The number of different configurations or processes that the ETMc discovers as a configurable process tree depends on the number of logs it has as input. So, it is not possible to find different configurations when only a single log is available, which could be the case of a software company trying to specify the processes they follow. Moreover, the ETMc algorithm is in turn based on the ETMd which is a genetic algorithm to

find single processes. Someone who is trying to use the ETM frameworks (as implemented in ProM the most widely used academic tool for process mining) might face some difficulties. First, the ETM algorithms have many different parameters which could confuse the user in case he or she is not familiar with it. Second, it will probably take longer than other approaches as it needs many generations to find an appropriate solution. Finally, it is unlikely to find the same results if the algorithm is executed more than once for the same log (even for small logs and with the default ProM parameters).

As part of this work, the v-algorithm [4, 5] was implemented as a ProM plugin. This algorithm is able to discover a SPrL from an event log. For this task, the behavior of the log is extracted from the log taking into account the number of relationships between each pair of activities. After that it uses two cut-off thresholds to split the behavior in three. The most frequent behavior is used to discover the common parts of the process (base process). The second part of the behavior is used to discover the variability. Finally, the least frequent part is discarded and for now considered as noise. The base process is discovered using the α-algorithm [2]. The variability is discovered using different rules to determine if an activity should be considered as optional or as a variation of other activity.

The v-algorithm as previously described can discover a SPrL from an event log that describes only the flow of activities. As it uses the α-algorithm it entails the problems it has (e.g., the base process is not always sound). Moreover, the defined rules are not general enough to cope with more complex situations than originally envisaged. So, the α-algorithm was replaced with several different approaches for the base process. Inspired by the results of the ETMd (e.g., as it is based on process trees it ensures to discover a sound model) we are working on an algorithm that is able to discover a process tree. The idea is to first create a very simple process (using the most frequent trace in a sequence tree) and adjusting only the parts of the tree that need to be changed to allow more of the seen behavior iteratively. This approach had interesting results, specially when the models tend to be structured (as is usually the case in software companies).

As the future work we are going to extend this "in construction algorithm" to be able to produce a SPrL that is also capable to use additional information that can be available in the log at event level (e.g., resource), or case level (e.g., size of the project) for the definition of more complete processes.

References

1. Tomás Martínez-Ruiz, Félix García, Mario Piattini, and J Munch. Modelling software process variability: an empirical study. *IET software*, 5(2):172–187, 2011.
2. W. M. P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes.* Springer Publishing Company, Incorporated, 1st edition, 2011.
3. Joos CAM Buijs, Boudewijn F van Dongen, and W. M. P. van der Aalst. Mining configurable process models from collections of event logs. In *Business Process Management*, pages 33–48. Springer, 2013.
4. Fabian Rojas Blum, Jocelyn Simmonds, and María Cecilia Bastarrica. The v-algorithm for discovering software process lines. *Journal of Software: Evolution and Process*, 2016.
5. Fabian Rojas Blum, Jocelyn Simmonds, and María Cecilia Bastarrica. Software process line discovery. In *Proceedings of the 2015 International Conference on Software and System Process*, pages 127–136. ACM, 2015.

The Smallest Software Particle

Robert Deckers, Patricia Lago, Wan Fokkink

Vrije Universiteit Amsterdam, the Netherlands

Introduction

The business success factors of software have changed from functionality to quality properties (e.g., availability, security, sustainability) and development qualities (e.g., time-to-market, development cost, project predictability). However, methods and tools for domain-specific languages and model-driven development are still focused on the delivery of software functionality instead of the guarantee of quality. We should treat qualities as first class elements of a human-oriented (rather than system-oriented) development methods and frameworks. The main subjects are specification of quality, integrating and managing multiple quality domains, and enabling automatic transformations between specifications in those domains.

This research investigates a method and framework for integrating domain-specific languages (DSL), building consistent specifications in those languages, and transformations between those specifications. The focus lies on quality properties (a.k.a. non-functional properties). Instead of aiming for a complete system specification in source code, we suggest that all involved persons can speak their own language and record decisions in that language. A development method and framework should support the transformation and synchronization between specifications in different languages. Software development should be seen as the integration of decisions and domain knowledge and not as the realization of executable source code.

People are more important than machines

Software development methods are focused too much on producing executable source code to feed a stupid machine. Instead, we should see software development as an accumulation of decisions made by humans with different knowledge, different opinions, and different stakes.

No support for your language

Software development has become a large-scale industry that employs people in many different areas of expertise and at multiple education levels. These people each speak their own language with their idiom and sometimes also their own grammar and syntax. The practice is however that we still use (development) languages that are an aggregation of machine primitives. These languages have dominated software development since the beginning. For example, BPMN and Java have still the same primitives as FORTRAN. The resulting disadvantage is that in many cases people have to transform the mental concepts of their own knowledge domain into machine-oriented concepts. A development process must enable people to communicate and reason in their own language.

An inconsistent view of a consistent world

Tools and methods, each with their own notations and artefacts, are already available and used for some of the domains in the development process. Although these tools overlap in the concepts they use and have interdependencies, they are mostly not integrated in a tool chain. For example, a requirements engineer uses a requirements management tool and a tester uses a test tool. Although a test tool might allow you to refer to requirements, there is typically no support to guard the consistency between the requirements and the tests. Development methods and tools must help guarding the consistency of all work products in the development process.

Quality matters, but is hardly covered

In the early days of software development, a computer with software offered functionality that wasnt possible before. Nowadays, functionality of software is unique only for a short time and soon offered by more suppliers. The business success factor of software has changed to quality attributes like ease of use, availability, security, and sustainability, and to development qualities like time-to-market, development cost, or project predictability. In spite of that, development methods and tools are still focused on the delivery of software functionality instead of the guarantee of quality. There are hardly any engineering methods for quality properties and quality is often not specified at all. A development method/framework should enable the engineering of quality.

Because needed quality is hardly specified and not clear to everyone involved, the quality properties of developed software are determined by personal beliefs and experience of the developers. This results in inconsistent quality throughout a system and its development. Because quality is not specified explicitly, it cannot be proven, guaranteed, or engineered. A development method must support the engineering of quality. This includes its specification, its realization, and the trade-offs between different qualities.

Waste of knowledge

A developed software system is the result of a series of development decisions. Making good decisions costs significant time and money. Therefore, the reuse of a good decision in another context increases the return on investment of that decision. The benefits are potentially the highest if the reuse is automated (when applicable of course). Automation of decisions can be divided in two categories. The first category is the reuse of elements that capture the result of the decision. Those elements can be part of the working system like a GUI library, a DBMS, or a rule engine. The second category is to automatically reuse decisions via transformations between development products, which reuse the decision each time the transformation is executed. Development methods must support the reuse and automation of any decision that can be made in a systems life-time: from idea, to design, to test, usage, and system deprecation.

Exploit and integrate knowledge

A formalism based on human thinking concepts instead of machine primitives should be the basis for a systems lifecycle that does not suffer from the issues stated in the previous section. Earlier literature research has shown that many partial elements already exist, but no integral solution seems to be available.

Integral specification of quality

There are standards like ISO25010 that define a limited set of quality attributes to address quality properties. ISO25010 however, does not help in making specifications for those properties, nor in building a complete and consistent specification of the design problem, the development, or the system itself. The research result should contain requirements and specifications for a method and framework that enable integration and transformation of specifications of quality via DSLs and inter-domain rules. The research result must also contain examples of specifications of quality and explain how quality attributes can be added to the framework. The result should ideally provide a method for specifying DSLs for any domain.

Extend MDD to anything formal

MDD approaches range from meta-modeling tools and meta-languages in which you build your own DSL and transformations (e.g., MetaEdit+, MPS), to more 4GL like platforms with a predefined modeling language and predefined application architecture (e.g., Mendix, Thinkwise). The models address mostly the application domain, data structure, and/or software functionality. Other system properties or stakeholder concerns are hardly modeled or even configurable. The result is that software systems built with such an MDD approach have fixed and implicit quality properties. This might be okay for some cases, but variable quality and trade-offs between quality attributes are desirable when an MDD approach aims at a large application domain. Also, quality should be predictable and explicitly known in many cases, for example when legislation demands it. The research result should contain requirements and specifications of how a model of a quality can be automatically transformed into software specifications and/or software development specifications. The result should also show examples of such a transformation.

A method based on multiple formalisms

Any specification, formal or informal, can only be (partially) understood if the language in which the specification is stated, is (partially) understood. If a specification must be processed by a machine, then that specification must be formal for at least the part that must be processed. Otherwise, a machine can simply not parse it. Some well-known formalisms exist that are used for specification languages, like Petri Nets, state machines, process algebra, set theory, and predicate logic. These mathematical mechanisms are applied in various specification languages that allow formal specification of software structures and software behavior. The idea is that a useful formalism for the specification of any quality is based on any of the above mentioned formalisms combined with a method to give clear semantics to the terms/words in a DSL. The research result should explain the basic formalism and show how it can be used for specifications of qualities that are made by a human and processed by a machine. It should also explain how a development language and method could be formalized gradually.

The targeted result

The core of this research is to investigate what are the ingredients for a method and framework that enables the guarantee of quality properties of a software system and its development. The targeted research result is threefold: a method (and/or requirements for a method), for defining DSLs and specifications for quality attribute and their integration, a framework for specifying and performing transformations between DSLs, and complete examples for a set of chosen quality attributes. The research result should also contain a complete example in which several qualities are specified via DSLs and integrated via MDD techniques.

Designing Optimal Runtime Architectures for Enterprise Applications on Multi-Tenant Heterogeneous Clouds

Siamak Farshidi, Slinger Jansen

Department of Information and Computing Sciences
Utrecht University, Utrecht, the Netherlands
{s.farshidi, slinger.jansen}@uu.nl

Abstract. Producing an enterprise application, which deploys in a multi-tenant heterogeneous cloud, faces with many architectural challenges as well as security, scalability, manageability, performance, maintainability, and etc. Therefore, choosing an appropriate runtime architecture that meets all requirements, is a key part of designing an enterprise application. This paper provides an overview of a set of research methods that focus on optimization of runtime architecture of enterprise applications that are deployed in multi-tenant heterogeneous cloud ecosystems. In other words, this research aims to create a set of decision support tools and strategies to find an optimum pattern for each architectural aspect of enterprise applications. Furthermore, the research is directed towards identifying the appropriate deployments scenario in multi-tenant heterogeneous clouds, based on their requirements and constraints.

Keywords: architectural patterns; multi-tenancy; heterogeneous cloud ecosystems; multi-criteria decision maker; deployment scenario

1 Introduction

Enterprise applications are complex, scalable, distributed, service-based, and mission-critical business applications. Nowadays, most of these applications are deployed in the cloud, a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [1].

Designing and developing such highly complex applications means satisfying segregate criteria just as constraints, functional requirements, and quality attributes. Moreover, every architectural and developmental decision to fulfill each of these criteria affects many other requirements and the failure to meet any of these requirements can mean the failure of the entire project.

An ongoing growing influence of cloud computing and Software-as-a-Service (SaaS) can be observed in the enterprise application domain [2]. One of the key features of SaaS is the ability to share computing resources in offering a software product to different customers. To benefit from this ability, the architecture of SaaS products should cater to the sharing of software instances and databases. A popular architectural style for achieving this is known as Multi-Tenancy, which is a property of a system where multiple customers, so-called tenants, transparently share the system's resources, such as services, applications, databases, or hardware, with the aim of lowering costs, while still being able to exclusively configure the system to the needs of the tenant [3].

The AMUSE research project is an academic collaboration between Utrecht University, Free University of Amsterdam, and AFAS Software to address software composition, configuration, deployment and monitoring challenges on heterogeneous cloud ecosystems through ontological enterprise modeling. In the project, the correlation between software models and resource requirements will be studied for finding optimal configuration and deployment on heterogeneous clouds. In other words, the main goal of the AMUSE project is improving scalability, maintainability, and performance of enterprise applications, that lead to reducing the cost of implementation, deployment, and maintenance of these type of applications.

2 Problem Statement and Proposed Contributions

Software Producing Organizations, such as open source and commercial organizations producing mass-market software, were increasingly engineering Very Large Software Systems. This presents many challenges like managing large service configurations, maintaining end-user variability, deployment on heterogeneous cloud ecosystems, managing system security, finding optimum way of storing and retrieving data, and handling complex errors.

The research methods that have to be addressed, when we talk about finding an optimum runtime architecture selection for enterprise applications on multi-tenant heterogeneous clouds, are related to different architectural aspects of enterprise applications. These characteristics including data store model, software architecture pattern, multi-tenant architecture, and deployment scenario which could be specified based on multiple criteria such as constraints, functional requirements, and quality attributes of each case studies.

2.1 Research Methods

From the problem and objective statements, the following research methods are derived.

Identify constraints, requirements, and metrics Various studies have argued that requirement analysis besides defining user and technological constraints are critical to succeed in producing a business application [3]. Moreover, one of the causes of project failures is the inconsistency of requirements, leading to unnecessary efforts, and can interfere with the performance of the final software product. Next, quality metrics need to be defined, because without any evaluation metrics we are facing with an uncertain level of quality in the finished product. Therefore, finding a set of functional requirements, quality attributes, and technological plus user constraints in different architectural aspects of enterprise applications, in addition to pre-check their consistency are essential tasks in the prior phase of this research.

Selection of suitable data storage model One important part of dealing with data is figuring out how and where to store it. A data storage model is a specification describing how a database is structured and used. Several data storage models have been suggested such as relational, object-oriented, graph-based, key-value, and so on. Thus, assigning an optimum set of storage models to the very large software system is not a straightforward decision, especially when realizing that the data storage requirements of enterprise applications significantly differ from other types of applications like Facebook or Etsy, for example. Ultimately, selecting the best-suited data store model has a huge impact on different classes of quality attributes like runtime, non-runtime, architecture, non-software, etc.

Specifying the optimum software architecture pattern Software Architecture comprises a set of decisions according to many factors in a wide range of software development, and each of these decisions has a considerable impact on the overall success of the enterprise application [4]. The optimum software architecture can decrease the business risks associated with building a software product, and make the system implementation and testing more traceable as well as increase the level of quality attributes. In contrast, inappropriate software architecture decisions create inefficient software in terms of cost and time, and it could be lead to real disaster for software-producing organizations. Software architecture styles, such as layered, object-oriented, message-based, CQRS, Microservices, and etc., are a proven reusable solution for well-known problems and contains a set of rules, constraints, and patterns of how to build an application based on its domain. Therefore, an appropriate architecture style can improve partitioning and promotes design reuse by providing solutions to frequently recurring problems [4].

Proposing appropriate multi-tenant architectures Multi-tenancy for enterprise applications reduces the overall cost of maintenance because multiple customers share application instances and data instances, so the total number of instances running will be much lower than in a single-tenant environment. Moreover, Multi-tenancy facilitates the development of enterprise applications, but practitioners will be faced with extra challenges in the implementation phase, like performance, scalability, security and re-engineering the current application. In addition, software architects struggle to choose adequate architectural styles for multi-tenant enterprise applications. Bad choices lead to poor performance, low scalability, limited flexibility, and obstruct software evolution [3]. Moreover, in the case of heterogeneous clouds which enable users by their variety of options, it is an essential task to find the optimum multi-tenant architecture for enterprise applications. Therefore, finding the most suitable multi-tenant architecture is crucial, because the architecture expresses a fundamental structural organization schema for a provider's software system. However, choosing the best solution is a complex task. Accounting for all the challenges and benefits complicates the decision process considerably.

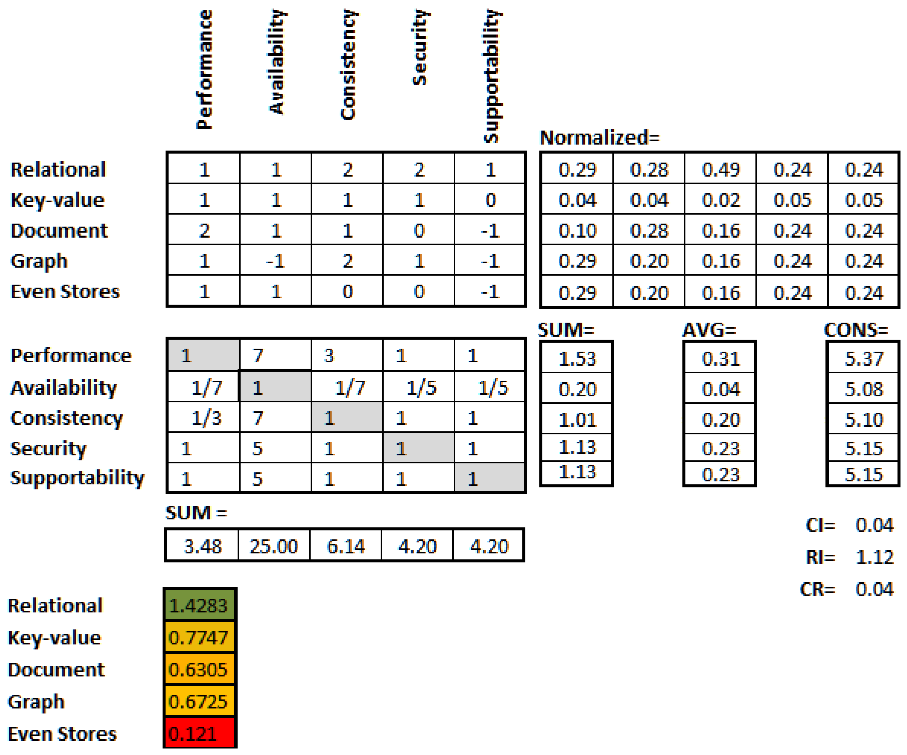
Choosing optimal deployment scenario Vertical scaling, or scaling the infrastructure, is an elementary method to increase the computational capacity. In the era of the legacy system, this task required software vendors to physically purchase more hardware resources. Nowadays cloud computing combines several existing technologies, as offering resources, to form a new computing service to facilitate the emergence of a central new characteristic of cloud computing as it allows to dynamically and rapidly scale resources based on the actual user demand. In addition, cloud providers offer a plethora of hosting products, and the impact of each product on the performance of an enterprise application is difficult to predict. Moreover, many cloud environments impose restrictions to deployed applications, such as disallowing modifying the filesystem or opening a range of network ports. Furthermore, the cloud ecosystem suffers from a lack of standardization and as a result, differences in cloud provider services, subsequently creating a heterogeneous environment [5]. Thus, specifying a cloud provider that covers all requirements, including functional and non-functional requirements, and satisfies all constraints is extremely challenging and of vital essence to the end product.

2.2 Potential contributions

The results of four first research methods have an impact on the early decisions of enterprise application companies because they can correctly choose the appropriate runtime architecture that satisfies their requirements and constraints. Moreover, by using evaluation metrics practitioners can measure the quality of their currents product. After choosing optimal runtime architecture, organizations can implement their enterprise application in a high level of trust based on their technological and user constraints, functional requirements, quality attributes, and own most important consideration. The final deliverable of fifth methods will be a method that allows for automated (simultaneous) multi-platform deployment without being dependent on a middleware that would result in vendor lock-in. To sum up, this research will make a huge contribution to Software Producing Organizations through finding optimum runtime architecture for enterprise applications, because an appropriate runtime architecture can reduce the business risks associated with building a technical solution, and make the system implementation and testing more traceable as well as achieve higher Quality Attributes. Furthermore, this research will be added empirically evaluated knowledge, and practitioners, from third parties and software product companies, can use the research methods as a starting point for their situation or as a guide throughout the entire process.

3 Proposed Methods and Current Progress

Decision analysis is a logical procedure for the balancing of factors that influence the decision. The procedure incorporates uncertainties, values, and preferences in a basic structure that models the decision. Typically, it includes technical, marketing, competitive, and environmental factors.



	Performance	Availability	Consistency	Security	Supportability
Relational	1	1	2	2	1
Key-value	1	1	1	1	0
Document	2	1	1	0	-1
Graph	1	-1	2	1	-1
Even Stores	1	1	0	0	-1

Normalized=

0.29	0.28	0.49	0.24	0.24
0.04	0.04	0.02	0.05	0.05
0.10	0.28	0.16	0.24	0.24
0.29	0.20	0.16	0.24	0.24
0.29	0.20	0.16	0.24	0.24

Performance	1	7	3	1	1
Availability	1/7	1	1/7	1/5	1/5
Consistency	1/3	7	1	1	1
Security	1	5	1	1	1
Supportability	1	5	1	1	1
SUM =	3.48	25.00	6.14	4.20	4.20

SUM=	AVG=	CONS=
1.53	0.31	5.37
0.20	0.04	5.08
1.01	0.20	5.10
1.13	0.23	5.15
1.13	0.23	5.15

CI=	0.04
RI=	1.12
CR=	0.04

Relational	1.4283
Key-value	0.7747
Document	0.6305
Graph	0.6725
Even Stores	0.121

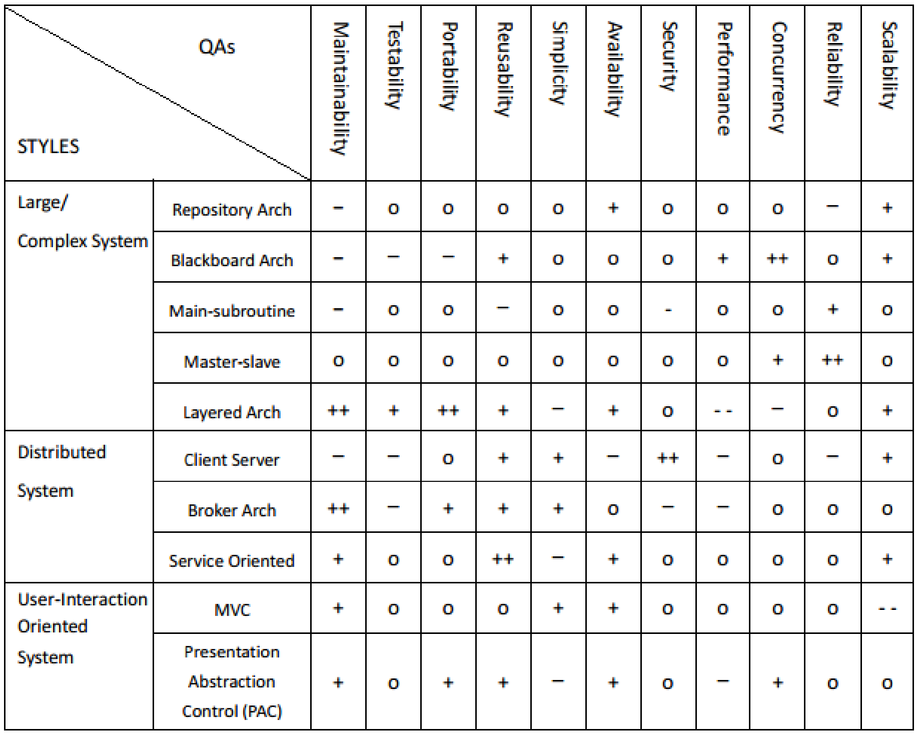
Fig. 1: an example of AHP

The essence of the procedure is the construction of a structural model of the decision in a form suitable for computation and manipulation [6]. Decision analysis addresses a decision problem that arises in many industries and business activities in which from a set of possible solutions one must be chosen. The same type of problem is stated in the problem statements of this research.

Identify constraints, requirements, and metrics Due to specify functional requirements, quality attributes, evaluation metrics, and scope of each proposed research method, a structured literature study based on guidelines of [7] besides variety interviews with domain experts will be carried out. In the AMUSE project we have unlimited access to technical documentation, resources, and experts at AFAS software company, which is a well-known ERP software company in the Netherlands, so we can also use their current product, which titled Next profit, as our immediate case study and source of primary data.

Selection of suitable data storage model A literature review is our major approach for data collection in this research. Our literature review aims at three specific aspects: commonly used data storage model at present, methods of selecting data storage model and research articles on specific data storage model. For this purpose, first, a deep comparison will be done between SQL and NoSQL databases. Multiple criteria are to be examined such as scalability, performance, consistency, security, analytical capabilities and fault-tolerance mechanisms. Second, the ten major types of database models through their twenty-eight cutting edge product candidates will be defined and compared. Multi-Criteria Decision Making Methods like Weighted Product Model, Analytic Hierarchy Process, or other available methods besides expert evaluations could be used in order to select the best-fitted data storage model based on functional and non-functional requirements of each case studies. AHP can handle this problem well, because weighing the relative importance of the decision criteria, like quality attributes, is an extensive process within AHP. This is an important aspect because the values of these weights could be assigned by experts in a case study for the decision criteria. These weights eventually result in the best data storage model for that case study. Fig. 1 illustrates an example of AHP method.

Specifying the optimum software architecture pattern Literature review in this research method aims at three specific perspectives:



STYLES	QAs	Maintainability	Testability	Portability	Reusability	Simplicity	Availability	Security	Performance	Concurrency	Reliability	Scalability
Large/ Complex System	Repository Arch	−	o	o	o	o	+	o	o	o	−	+
	Blackboard Arch	−	−	−	+	o	o	o	+	++	o	+
	Main-subroutine	−	o	o	−	o	o	-	o	o	+	o
	Master-slave	o	o	o	o	o	o	o	o	+	++	o
	Layered Arch	++	+	++	+	−	+	o	- -	−	o	+
Distributed System	Client Server	−	−	o	+	+	−	++	−	o	−	+
	Broker Arch	++	−	+	+	+	o	−	−	o	o	o
	Service Oriented	+	o	o	++	−	+	o	o	o	o	+
User-Interaction Oriented System	MVC	+	o	o	o	+	+	o	o	o	o	- -
	Presentation Abstraction Control (PAC)	+	o	+	+	−	+	o	−	+	o	o

Fig. 2: General architecture styles in category and evaluation [4]

frequently used software architecture styles at present, methods of comparing architecture patterns, and analysis articles on specific architecture patterns. In order to select architecture styles correctly and precisely all existing information related to the project should be considered; therefore, it is a multi-criteria decision-making problem. However, the collected information may interact with each other in some cases, which makes it difficult to select the best architecture style. In this way, making use of a decision support system which is able to consider not only different criteria but also the interaction among them is a solution to solve the problem in a more disciplined way which could provide software architects with most suitable results for the domain. Decision Support Systems such as Fuzzy Theory, Case-Based Reasoning (CBR), Weighted Product Model (WPM), or other available methods could be used to specifying the optimum software architecture pattern. In addition, multiple expert interviews will be conducted to find out practitioners experiences and evaluations. Fig. 2 shows a sample of expert evaluation which could be used to weight the proposed architectural patterns in AHP.

Finding appropriate multi-tenant architecture In order to find the currently most use multi-tenant architecture for enterprise applications, a structured literature study based on guidelines of [7] besides variety interviews with domain experts will be carried out. After that, based on prior found criteria, such as functional requirements, quality attributes, technical limitation, and user constraints, this problem will be converted a multi-criteria decision-making problem. Therefore, Multi-Criteria Decision Making Methods like Weighted Product Model, Analytic Hierarchy Process, Fuzzy Theory, or other available methods, besides interviewing with domain expert could be used in order to select the best fitted multi-tenant architecture for different case studies.

Choosing optimal deployment scenario The multi-cloud paradigm optimizes to decrease costs and optimizing quality by leveraging multiple cloud providers simultaneously. With cloud providers not standardizing anytime soon, one solution to decrease migration costs is by developing a multi-cloud broker that is able to deploy an application to multiple cloud providers. In addition, the mere fact that multiple cloud providers are now available for use adds the question of where to deploy a software application. This research, therefore, proposes a method that can automatically select, configure, and deploy an application within this highly heterogeneous cloud environment. Modeling both the application and the cloud environment could be achieved through combining the modeling language TOSCA [8] and feature models. Moreover, by adding user-defined constraints such as costs and hardware configurations, it becomes possible to select an optimal cloud provider for each application component. Finally, the generated deployment scenario is then automatically deployed to the appropriated cloud providers.

Acknowledgment

This research was supported by the NWO AMUSE project (628.006.001): a collaboration between Utrecht University, Free University of Amsterdam, and AFAS Software in the Netherlands.

References

1. P. Mell and T. Grance. Sp 800-145. the nist definition of cloud computing. technical report. nist. gaithersburg, md, united states, 2011.
2. R. Walters. The cloud challenge: Realising the benefits without increasing risk. *Computer Fraud & Security*, 2012:5–12, 2012.
3. S. Jansen J. Kabbedijk. Variability in multi-tenant environments: Architectural design patterns from industry. In *O. De Troyer, C. Bauzer Medeiros, R. Billen, P. Hallot, A. Simitsis, & H. Van Mingroot (Eds.), Advances in conceptual modeling. recent developments and new directions*, pages 151–160. Springer Berlin, Heidelberg, 2014.
4. Microsoft. Microsoft application architecture guide, 2nd edition. `http://msdn.microsoft.com/enus/library/ff650706.aspx`. [Online; accessed 06-Aug-2016].
5. P. Eads S. Crago, K. Dunn. Heterogeneous cloud computing. cluster computing (cluster). In *2011 IEEE International Conference on*, pages 378–385, 2011.
6. R. Howard. Decision analysis: Applied decision theory. In *In Proceedings of the fourth international conference on operations research*, pages 55–72, 1966.
7. S. Charters B.A. Kitchenham. Guidelines for performing systematic literature reviews in software engineering. In *Technical Report EBSE-2007-01, School of Computer Science and Mathematics, Keele University*, 2007.
8. OASIS. Topology and orchestration specification for cloud applications (tosca) version 1.0, committee specification 01. `http://docs.oasis-open.org/tosca/TOSCA/v1.0/cs01/TOSCA-v1.0-cs01.html`. [Online; accessed 06-Aug-2016].

Using Dynamic and Contextual Features to Predict Issue Lifetime in GitHub Projects

Riivo Kikas, Marlon Dumas, Dietmar Pfahl

University of Tartu, Estonia
`riivokik@ut.ee`

Abstract

Open source projects usually rely on publicly accessible issue tracking systems to manage unresolved bugs and development tasks. In contemporary open source code hosting sites, such as GitHub, the barrier for contributing issue reports is minimal. Everyone can create an issue, but only a limited set of stakeholders deal with resolving these issues. This tension between the ease of creating issues and the limited resources of the core project team leads to situations where issues receive sparse attention from the project team or do not even receive a preliminary screening upon their creation.

Knowing when an issue will be closed is important from two viewpoints. First, it has been found that timeliness is an important determinant of contributor engagement and community contribution acceptance in GitHub. If there is high uncertainty regarding the timeframe when the development team will address a given issue, the stakeholder who submitted it might be discouraged from making further contributions or even from using the software product. Having an estimate of issue closing time can help to reduce this uncertainty and provide greater transparency to all stakeholders. Second, an estimate of issue closing time provides core team members with a basis to prioritize their efforts and plan their contributions.

Methods for predicting issue lifetime can help software project managers to prioritize issues and allocate resources accordingly. In this setting, this paper addresses the problem of predicting, at a given time point during an issue's lifetime, whether or not the issue in question will close after a given time horizon, e.g. predicting if an issue that has been open for one week will remain open one month after its creation.

The general problem of an issue (or bug) lifetime prediction has received significant attention in the research literature. The focus of this study differs from previous work in four respects. First, the bulk of previous work has focused on analyzing a small number of hand-picked

projects. In contrast, this work studies this prediction problem based on a large sample of projects hosted on GitHub. Second, most previous work has focused on exploiting static features, i.e. characteristics extracted for a given snapshot of an issue – typically issue creation time. However, during its lifetime, an issue typically receives comments from various stakeholders, which may carry valuable insights into its perceived priority and difficulty and may thus be exploited to update lifetime predictions. The present study combines static features available at issue creation time, with dynamic features, i.e. features that evolve throughout an issue's lifetime. Third, previous approaches focus on predicting lifetime based on characteristics of the issue itself. In contrast, the present study combines characteristics of the issue itself with contextual information, such as the overall state of the project or recent development activity in the project. Finally, most previous studies do not employ temporal splits to construct prediction models. In other words, models are trained on future data and then evaluated on past data. In this study, we construct models predictively using strict temporal splits such that predictions are always made based only on past data, which reflects how such predictive models would be used in practice.

This work proposes an approach for training and evaluating issue lifetime prediction models at different points in time and highlights the importance of dynamic and contextual features in such predictive models. The results show that dynamic and contextual features complement the predictive power of static ones, particularly for long-term predictions. In particular, dynamic features observed for longer periods enhance precision up to 33% when estimating whether an issue will be closed in the upcoming year.

Reproducible and Reusable Research Assets Management

Armel Lefebvre, Marco Spruit, Sjaak Brinkkemper

Department of Information and Computing Sciences
Princetonplein 5, De Uithof, 3584 CC Utrecht
{a.e.j.lefebvre, m.r.spruit, s.brinkkemper}@uu.nl

Introduction

In the recent years the interest in reusing published research assets has increased tremendously. Building upon the view of freely available scholar publications (i.e. open access), Open science goes one step further by promoting research data reuse and sharing among academics, the industry and citizens.

At the same time, there is an inability to inspect in greater details papers relying on intensive data analysis. This led to what one might call a reproducibility crisis (Peng, 2015). The reproducible research movement asserted that there is frequently no code or data available to check the claims made by the authors or to reuse datasets for answering new research questions [6]. Hence, the demand for reproducible results and the availability of data and code is becoming more and more anchored in how research outputs and projects are assessed by journals, funding agencies and also universities, which are establishing research data management policies.

Meanwhile, researchers are facing greater challenges to get meaningful insights from large and complex datasets to answer their research questions. The requirement to properly trace and assess the interactions between software, data and hardware are presenting supplementary burdens for scientists [4]. For instance, important issues are faced to annotate datasets, identify which parameters were used during an analysis and which algorithms or software packages were involved. Furthermore, scientists might experience some difficulties in localizing relevant data after the departure of the data owner responsible for the experiment (e.g. a PhD student). These challenges lead to the following research question: “How can research assets be managed to ensure that computational experiments are reusable and reproducible?”

Method

We follow a Design Science Research method [3] to build socio-technical artefacts to help researchers manage ongoing and published research data. Two outcomes are expected. First, a multidimensional metadata model. Second, a middleware implementing the structures, operations and constraints set by the model. We collect evidence and requirements through literature reviews, interviews, surveys and a field study in bioinformatics. We plan to collect further data through the middleware to adapt the model and its implementation based on the analysis of log data.

Design Proposition

In metadata management, types of metadata are traditionally divided into two or more categories such as the pair system metadata and user-defined metadata. These taxonomies hold in a closed environment where business information can be extracted consistently. Here, research asserts are generated and/or processed in an open world with no single point of truth. Hence, there is a need to support a wider range of, sometimes unexpected, scenarios of computational experimentation, data complexity and researchers' goals (e.g. the whole pipeline to transfer data from a sequencing facility to a laboratory then between a laboratory and a high performance computing cluster).

This calls for a redefinition of how metadata is structured, extracted and filtered for managing scientific data. New ways to integrate and annotate data depending on a wide diversity of data sources started with systems such as dataspaces, which were advocated by Franklin, Halevy, & Maier [2]. To achieve the design and deployment of a scientific data space, we build upon a multidimensional theory of information called the Descriptive Theory of Information suggested by Boell & Cecez-Kecmanovic [1]. The main challenges are the repurposing, extension of the descriptive theory with relevant dimensions and facet of information for research (meta)data management and representation of this model as a multigraph supporting "on the fly" conversions to ontologies. This including the constraints of reusability, reproducibility and even privacy in case of sensitive data.

To conclude, we suggest a multidimensional metadata model and middleware to be able to dynamically structure research assets as a dataspace. The metadata model will assist researchers in their data steward role, i.e. the management of research data by scientists themselves. The middleware is a collection of artifacts that are implementing this metadata model. Both will be developed and evaluated in research labs.

References

1. Boell, S. K., & Cecez-Kecmanovic, D. (2015). What is Information Beyond a Definition? In ICIS 2015 Proceedings (p. Paper 1363). Retrieved from http://aisel.aisnet.org/icis2015/proceedings/ConferenceTheme/4/
2. Franklin, M., Halevy, A., & Maier, D. (2005). From databases to dataspaces: a new abstraction for information management. ACM Sigmod Record, 34(4), 2733. http://doi.org/10.1145/1107499.1107502
3. Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. MIS Quarterly, 28(1), 75105.
4. Lefebvre, A., Spruit, M., & Omta, W. (2015). Towards reusability of computational experiments Capturing and sharing Research Objects from knowledge discovery processes. In Proceedings of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2015) (Vol. 1, pp. 456462).
5. Peng, R. D. (2011). Reproducible research in computational science. Science (New York, N.Y.), 334(6060), 12267. http://doi.org/10.1126/science.1213847
6. Peng, R. D. (2015). The reproducibility crisis in science: A statistical counterattack. Significance, 12(3), 3032. http://doi.org/10.1111/j.1740-9713.2015.00827.x

Resource Prediction for Applications Generated From Enterprise Models

Gururaj Maddodi, Slinger Jansen, Rolf de Jong

Utrecht University, AFAS Software

Abstract

Performance is increasingly becoming important aspect of non-functional requirements of a software application. This has become further important as the software becomes more and more complex and they start to be hosted on cloud environments instead of traditional on-premise servers. On cloud platforms the application has to share resources with other running applications also. Many research works have focused on predicting performance from representative modeling of software application. Arcelli et al. [1] proposes a framework for automated generation of software models subject to continuous performance analysis and refactoring. Authors in [2] make a survey of different approaches in performance prediction using model-based approach. Becker et al. [3] use Palladio Component Model (PCM) approach to parameterize resource usage of software.

Resource requirement prediction of a software application developed using MDD approach requires gaining deep understanding of the domain-specific language (DSL) that is used to describe the model of the software to be generated. For each the elements of the DSL resource consumption need to discovered, which can then lead to building resource profile of the generated application from bottom-up. This along with the associated parameters and relations between elements make it complicated to predict resource requirement.

The research is conducted in an industrial setup. The case company is AFAS Software who are developing an Ontological Enterprise Modeling (OEM) language to describe operations of real-world enterprises. In this context, the modeling DSL is the language that is going to be used for describing models for software generation for an individual enterprise. Though the OEM DSL is still under construction, the basic elements and the relations are mostly defined.

One of the key design drivers in the development of NEXT is the focus on Command Query Responsibility Separation (CQRS) [4] distributed

architecture. CQRS is an architectural pattern developed to support scalability, and provides a practical solution to Brewers theorem [5] in a distributed system architecture. This pattern prescribes separation in client-server communication between commands and queries, commands being actions to modify the data while queries are the requests to access the said data. The CQRS architecture can be combined with event sourcing [4] concept, then we have an additional component called events which propagate changes from command side to the query side. In AFAS context the DSL is tightly linked to the CQRS architecture as domain-driven design (DDD) principles are used in generating the application. Using DDD, meta-model elements of the NEXT DSL are grouped into so called aggregates. Aggregates determine how database schema will be defined. Hence it affects how commands, queries, and events are going to look like from a given model.

In order to discover the resource requirements of NEXT DSL models, we start from a bottom-up approach, i.e. use the meta-model elements from which the models are built and associate resource metrics with them. Here attributes associated with the meta-model elements and the relationships that can be formed between then also needs to be taken into account when associating resource metrics. Several models with varying complexity can be created. Then by subjecting the models to workload and performance are traced. Since the workload of the NEXT consists of commands and queries (due to CQRS architecture), by tracing the application generation phase from the meta-model elements formation of aggregates, commands, queries, and events can be known. Performance results will be in terms of resource utilized for each commands and queries. By correlating the resource metrics for commands and queries to the meta-model elements and their relations, resource requirements per meta-model elements, attributes, and relations can be discovered.

References

1. D. Arcelli and V. Cortellessa. Assisting software designers to identify and solve performance problems. In *Proc. of the 1st International Workshop on Future of Software Architecture Design Assistants*, pages 1–6, 2015.
2. S. Balsamo, A. Di Marco, P. Inverardi, and M. Simeoni. Model-based performance prediction in software development: A survey. *IEEE Transactions on Software Engineering*, 30(5):295–310, 2004.
3. S. Becker, H. Koziolek, and R. Reussner. Model-based performance prediction with the palladio component model. In *Proc. of the 6th international workshop on Software and performance*, pages 54–65, 2007.
4. G. Young. Cqrs and event sourcing. feb. 2010.

5. S. Gilbert and N. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. 33(2):51–59, 2002.

Flu-Now : Nowcasting Flu based on Product Sales Time Series

Ioanna Miliou[1], Salvatore Rinzivillo[2], Giulio Rossetti[2], Dino Pedreschi[1], Fosca Giannotti[2]

[1] KDDLab, University of Pisa, {miliou, pedre}@di.unipi.it
[2] KDDLab, ISTI-CNR, {name.surname}@isti.cnr.it

Abstract

Big Data offer nowadays the capability of creating a digital nervous system of our society, enabling the measurement, monitoring and prediction of various phenomena in quasi real time. But with that, comes the need of more timely forecast, in other words *nowcast* of changes and events in nearly real-time as well. The goal of *nowcasting* is to estimate up-to-date values for a time series whose actual observations are available only with a delay. Choi and Varian [1] introduced the term nowcasting to advocate the tendency of web searches to correlate with various indicators, which may reveal helpful for short term prediction. In the field of epidemiology, it was showed in various works that search data from Google Flu Trends, could help predict the incidence of influenza-like illnesses (ILI). But as Lazer and al. notice [2], in February 2013, Google Flu Trends predicted more than double the proportion of doctor visits for ILI than the Center for Disease Control.

In this work we are studying the influenza time series, of cases from 2004/05 to 2014/2015 influenza season, from physicians and pediatricians from all over Italy. We are interested to examine whether is possible to use retail market data as a proxy for influenza prediction. Our dataset consists of economic transactions collected by COOP, a system of Italian consumers' cooperatives which operates the largest supermarket chain in Italy. The whole dataset contains retail market data in a time window that goes from January 1st, 2007 to April, 27th 2014. First, we identified the products that have adoption trend similar to the influenza trend with the help of an 1-nearest neighbor classifier that uses dynamic time warping as the distance measure between time series. Based on these products, we identified the customers that buy them during the influenza-peak, since those individuals would have higher possibility to be either infected or close to an infected individual. We extracted their most frequent baskets during the peak using the Apriori algorithm, an algorithm

for frequent item set mining and association rule learning over transactional databases, and we use those baskets-sentinels as control set for the following year influenza peak. Monitoring the behavior of these baskets-sentinels we are able to detect patterns similar to the ones of the previous year's influenza peak, and as a result obtain an alarm for the appearance of the influenza.

Many lines of research remain open for future work, such as studying whether the retail market data can manage to predict the influenza peak even in particular cases such as the year 2009 non-seasonal H1N1 influenza (flu) pandemic that peaked in October and then declined quickly to below baseline levels by January.

References

1. Hyunyoung Choi and Hal Varian. Predicting the present with google trends. *Technical Report*, 2009.
2. David Lazer, Ryan Kennedy, Gary King, and Alessandro Vespignani. The parable of google flu: Traps in big data analysis. *Science Magazine*, 343(6176):1203–1205, 2014.

Power-efficient quality-assuring decision framework

Fahimeh Alizadeh Moghaddam[1], Paola Grosso[2], Patricia Lago[3]

[1] S2 and SNE groups
VUA and UvA
Amsterdam, The Netherlands
f.alizadehmoghaddam@vu.nl

[2] System and Network Engineering group (SNE)
University of Amsterdam (UvA)
Amsterdam, The Netherlands
p.grosso@uva.nl

[3] Software and Services group (S2)
VU University Amsterdam (VUA)
Amsterdam, The Netherlands
p.lago@vu.nl

Keywords: power efficiency, quality assurance, decision framework

1 Abstract

Running software in cloud is very expensive in terms of energy costs and environmental impacts. As [1] estimates, Google had up to 67M$ for electricity costs in 2008. Software as the initiator of power consumption, involves different components of data centers, which are operational implementations of cloud-based environments. Therefore, the achievable quality of software, like performance and power efficiency, strongly depends on the quality of the contributing components. This effect is even clearer in case of distributed applications, when heavily relying on the network infrastructure.

We see room for improvement in power efficiency of the software, while still satisfying non-energy related quality requirements. This is because from the data center providers perspective power efficiency needs to be weighted against other quality requirements such as performance, reliability and availability. This results in the deployment of large number of devices in data centers, connected via multiple (including back-up) links, whose average utilization rate is only around 30% [2].

Satisfying quality requirements of the software is not an easy and a common task to perform in large-scale complex environments. The mapping between the application quality requirements and the available resources is often done inefficiently, which results in low performance and

high power consumption. We believe that by adjusting the network component based on power efficiency purposes, we can achieve higher power efficiencies in the application layer. In this work, we propose a decision framework that lies between the application layer and the network infrastructure. It translates all pre-defined requirements to load scheduling setups in the infrastructure in an efficient manner.

We make the software more intelligent by providing a clear image of the underlying utilized resources and adapting the resources configurations according to the quality requirements in realtime. We adopt a combination of three power saving approaches in the network layer to provide power savings in the software layer: 1) Putting idle network devices into sleeping mode, 2) Increasing number of idle network devices by shaping the incoming traffic, and 3) Prioritizing the existing active devices over the sleeping ones to carry the load.

Depending on the application quality requirements, our decision framework is able to perform the best fitting scheduling task. It deploys one or more of the aforementioned approaches to classify the applications in a range from power sensitive to performance sensitive. We used linear programming (LP) scheduling to formulate the application requirements and power efficiency metrics in one objective function. We distinguish between the application requirements by their reflection on power efficiency priorities:

– ***LP-v1****: Semi power efficient* Applications classified in this category require maximization on the achievable performance of the application while still providing power saving. They allow our decision framework to make use of the three power saving approaches.
– ***LP-v2****: Semi performance guaranteed* Similar to LP-v1, applications in this category prioritize power efficiency as much as non-energy related quality requirements. However, they are less greedy for utilizing the resources. Therefore, our decision framework can schedule the upcoming load on idle sleeping devices, while there is still capacity available in already active devices.
– ***LP-v3****: Only performance guaranteed* We take performance as an example of non-energy related quality requirement of the application. In this version, the application is performance-sensitive and can not accept degradation in performance metric due to power-efficiency improvements.
– ***LP-v4****: Only power efficient* In this version the application focuses the most on maximizing the power saving as such it is the most power

sensitive version. Non-energy related quality requirements of the application do not influence the resource utilization in the network.

Fig. 1 presents the architecture of our decision framework consisting of three main modules: 1) *Load scheduler* at the center, which receives the application quality requirements and profiled statistical data and provides the resources scheduling setup accordingly, 2) *Modification System*, which performs modifications to the resources configurations, and 3) *Profiling System* that measures the statistical information periodically from the underlying resources.

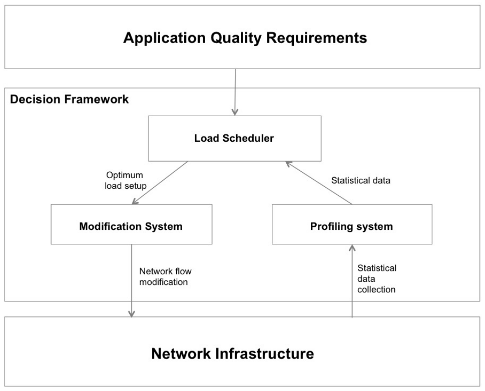


Fig. 1: Our decision framework consisting of the load scheduler, the modification system and the profiling system components

We implemented a number of experiments in the Mininet[4] simulation environment. We use the open-source POX control software[5] and the Gurobi Python optimizer[6] as our controller and scheduler components. To compare the behavior of our decision framework in presence of different application requirements, we focus on two performance metrics: *Power consumption* of the total data center infrastructure and *Time to complete (TTC)*, which is the time it takes for the application to send predefined number of bytes to another host(in our case 38GB of data). To evaluate our smart decision framework, we compare it against Smart SP, an example decision framework. Smart SP does not follow any power efficient approaches and no intelligence is provided to the application layer. We run a one-one scenario in the infrastructure but we define different quality requirements for the same application in each round of simulation.

[4] http://mininet.org
[5] http://www.noxrepo.org/pox/about-pox/
[6] http://www.gurobi.com/

Scalability of the decision framework is influenced by the quality requirements the running application defines towards power efficiency. We investigated the power saving provided by our decision framework in presence of four different power efficiency schemes in different network sizes. The values derived from LP-v4 and Smart SP are considered as our baseline, as they specify the minimum and the maximum power consumption of the infrastructure. As Fig. 2 shows, in all three network sizes, we observed that LP-v1 is the most power efficient variation that shows around 95% improvement, which means only 5% degradation from the optimum power saving. LP-v2 with achieving 50% of maximum power saving outperforms LP-v3 with achieving 45% of maximum power saving, which provides higher performance for the running applications. It is interesting to see that all the variations remain in the same range of power saving for different network sizes.

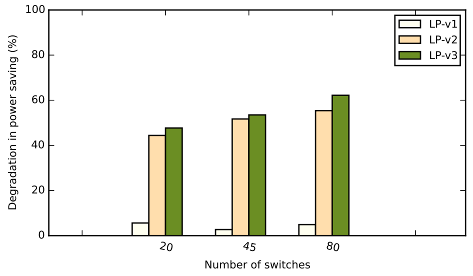


Fig. 2: Degradation in power savings of three power saving schemes namely, LP-v1, LP-v2 and LP-v3 in the three simulated network sizes. LP-v4 and Smart SP are considered as the baselines for calculation of power saving.

Fig. 3 shows the TTC measured from the applications running in the hosts as function of the four application requirement schemes when running with maximum incoming load. LP-v1 and LP-v4 show a considerable increase in TTC when the network size grows. Differently, LP-v2 and LP-v3 appear to be more performance-focused and stable for different sizes of the network.

Our decision framework empowers the cloud-based applications to map their quality requirements to the optimum resources subset. Our power efficiency schemes are designed for data center networks, in which realtime and scalable scheduling is crucial. Our results show that two of the variations (LP-v2 and LP-v3) remain stable in terms of power saving and the TTC metrics, as the underlying infrastructure grows in size. Two

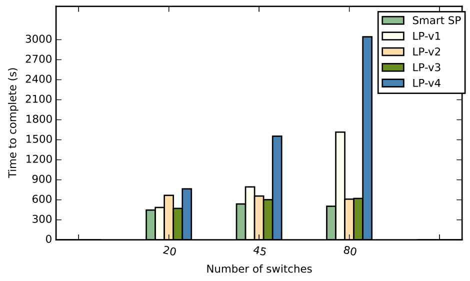


Fig. 3: Time to complete for the four different power saving schemes in the three simulated network sizes (38GB of data)

other (LP-v1 and LP-v4) provide the highest power savings and they are suitable for delay-insensitive applications.

Acknowledgment

This work has been sponsored by the RAAK/MKB-project "Greening the Cloud" and by the Dutch national program COMMIT. This work has been partially submitted to the Network Softwarization (NetSoft2016) conference.

References

1. Asfandyar Qureshi. *Power-demand routing in massive geo-distributed systems*. PhD thesis, Massachusetts Institute of Technology, 2010.
2. Jie Liu, Feng Zhao, Xue Liu, and Wenbo He. Challenges towards elastic power management in internet data centers. In *Distributed Computing Systems Workshops, 2009. ICDCS Workshops' 09. 29th IEEE International Conference on*, pages 65–72. IEEE, 2009.

Discovering Software Engineering Related Web Resources and Technical Terms

Gao Sa

Nanyang Technological University

Abstract

A sheer number of techniques and web resources are available for software engineering practices and that number continues to grow. Discovering semantically similar or related technical terms and web resources offer the opportunity to design appealing services to facilitate information retrieval and information discovery.

In our work, we first extract technical terms and web resources from community Q&A discussions, and propose a neural-language model based approach to learn semantic representations of technical terms and web resources in a joint low-dimensional vector space. Our approach maps technical terms and web resources to a semantic vector space based on only the surrounding technical terms and web resources of a technical term (or web resource) in a discussion thread, without the need of mining text content of the discussion. We apply our approach to Stack Overflow data dump of March 2016.Through both quantitative and qualitative analysis in three exploratory tasks: clustering task, search tasks, and semantic reasoning tasks, we show that the learned technical-term and web-resource vector representations can capture the semantic relatedness of technical terms and web resources, and they can be exploited to support various search tasks and semantic reasoning tasks, by means of simple K-nearest neighbor search and simple algebraic operations on the learned vector representations in the embedding space. Based on the semantic embedding space of tag and URLs, we further propose an accurate URL auto link service to automated recommend web resources to users. Web resources recommendation can help developers reduce the burden to manually search the web resources for SE related terms. In this work, we extract anchor texts and corresponding web resources from community Q&A discussions, then we incorporate the popularity feature and semantic similarity of context tags and candidate web resources to automatically help user link the software related technical terms to web resources.

Application of Genetic Algorithms for Automated Crash Reproduction

Mozhan Soltani[1] , Annibale Panichella[2], and Arie van Deursen[1]

[1] Delft University of Technology
m.soltani@tudelft.net,
Arie.vanDeursen@tudelft.net
[2] University of Luxembourg
annibale.panichella@uni.lu

Abstract. Often the crash data that is available to software developers is insufficient for debugging purposes. This issue negatively affects the productivity of the developers when it comes to debugging. Therefore, various automated techniques have been proposed which strive for using the available crash data to reproduce the target crashes, and thereby, derive useful hints regarding the location of the existing defects in the source code. Despite the advances made in this area, there are still many crash cases from well-known open-source projects that have not been covered by the existing approaches. We propose to tackle this problem by applying a novel fitness function which could be used to guide genetic algorithms in search for an ideal test case that can trigger the target crash. Our primary results show that application of this approach leads to increased coverage of crash cases for the same open-source projects.

Keywords: Genetic Algorithms, Automated Crash Reproduction, Software Testing, and Software Debugging

1 Introduction

Typically, the minimality of crash data that is reported to software developers leads to longer periods of time that it takes to produce fixes for the reported crashes. Therefore, several automated techniques for crash reproduction have been proposed that take the minimal data and derive hints that can guide the developers for identifying the location of the existing defects. However, the existing techniques [1, 2, 4–8, 10] impose limitations which reduce their capabilities in covering a wider range of crashes [9].

We propose to address automated crash reproduction by applying an evolutionary approach that uses the crash stack data to search for an ideal test case that can mimic a target crash. In this approach, we defined a novel fitness function that guides the search process towards finding the ideal test case. We implemented the fitness function as an extension of EvoSuite [3], an evolutionary test generation framework, and evaluated it on well-known crashes from the Apache Commons libraries. In this paper we outline the definition of the fitness function and present the primary results of the evaluations of our approach.

2 The Evolutionary Approach and Primary Results

We built our approach on top of EvoSuite [3], which is a search-based test generation framework that can be used to maximize code coverage. Thus, we defined a novel fitness function to guide the evolution of test cases towards finding the ideal test case. Such test case must crash at the same location as the original crash and produce a stack trace similar to the original one. A standard stack trace in Java contains (i) the type of the exception thrown, and (ii) the list of method frames being called at the time of the crash. Therefore, our fitness function, shown in Equation 1, must consider the following conditions to guide the search: (i) the line where the exception is thrown must be covered, (ii) the target exception must be thrown, and (iii) the generated stack trace must be as similar to the original one as possible.

$$f(t) = 3 \times d_s(t) + 2 \times d_{except}(t) + d_{trace}(t) \tag{1}$$

In Equation 1, $d_s(t)$ denotes how far t is to execute the target statement; $d_{except}(t) \in \{0, 1\}$ indicates whether the target exception is thrown or not; and $d_{trace}(t)$ measures the distance between the generated stack trace and the expected trace.

For the line distance $d_s(t)$, we use the *approach level* and the *branch distance*. The *approach level* measures the distance between the path of the code executed by t and the target statement. The *branch distance* scores how close t is to satisfy the branch condition for the branch on which the target statement is directly control dependent. For the trace distance $d_{trace}(t)$, we define a new distance function as follows. Let $S^* = \{e^*_1, \ \ldots, e^*_n\}$ be the target stack trace to replicate, where $e^*_i = (C^*_1, m^*_1, l^*_1)$ is the i-th element in the trace composed by class name C^*_i, method name m^*_i, and line number l^*_i. Let $S = \{e_1, \ldots, e_k\}$ be the stack trace generated when executing the test t. We define the distance between the expected trace S^* and the actual trace S as follows:

$$D(S^*, S) = \sum_{i=1}^{\min\{k,n\}} \varphi\left(\mathrm{diff}(e^*_i, e_i)\right) + \mid n - k \mid \tag{2}$$

Table 1 in Appendix 2 compares our results with two state-of-the-art methods, namely (i) STAR [2], and (ii) MuCrash [10]. The former uses symbolic execution while the latter is based on mutation analysis. Our results indicate that our approach can reproduce 8 out of 10 crashes. Based on our manual check, all reproduced crashes are useful to fix the bug. For six bugs, our prototype constantly replicates the crash in all 30 independent runs. For ACC-53, there are only two out of 30 runs where a replication is not achieved. Comparing our results with those achieved by STAR [2] and MuCrash [10], we observe that there are bugs that can be reproduced by our technique and not by the alternative ones. The results of our pilot study show the strength of evolutionary testing techniques, and evolutionary test case generation tools in particular, with respect to symbolic execution based on precondition analysis and mutation analysis.

References

1. S. Artzi, S. Kim, and M. D. Ernst. Recrash: Making software failures reproducible by preserving object states. In *ECOOP 2008–Object-Oriented Programming*, pages 542–565. Springer, 2008.
2. N. Chen and S. Kim. Star: Stack trace based automatic crash reproduction via symbolic execution. *IEEE Tr. on Sw. Eng.*, 41(2):198–220, 2015.
3. G. Fraser and A. Arcuri. Whole test suite generation. *IEEE Transactions on Software Engineering*, 39(2):276–291, Feb. 2013.
4. W. Jin and A. Orso. Bugredux: reproducing field failures for in-house debugging. In *Proceedings of the 34th International Conference on Software Engineering*, pages 474–484. IEEE Press, 2012.
5. A. Leitner, A. Pretschner, S. Mori, B. Meyer, and M. Oriol. On the effectiveness of test extraction without overhead. In *International Conference on Software Testing Verification and Validation (ICST)*, pages 416–425. IEEE, 2009.
6. S. Narayanasamy, G. Pokam, and B. Calder. Bugnet: Continuously recording program execution for deterministic replay debugging. In *ACM SIGARCH Computer Architecture News*, volume 33, pages 284–295. IEEE Computer Society, 2005.
7. J. Rossler, A. Zeller, G. Fraser, C. Zamfir, and G. Candea. Reconstructing core dumps. In *2013 IEEE Sixth Int. Conf. on Software Testing, Verification and Validation*, pages 114–123. IEEE, 2013.
8. D. Saff, S. Artzi, J. H. Perkins, and M. D. Ernst. Automatic test factoring for java. In *Proceedings of the 20th IEEE/ACM international Conference on Automated Software Engineering*, pages 114–123. ACM, 2005.
9. M. Soltani, A. Panichella, and A. van Deursen. Evolutionary testing for crash reproduction. In *Proceedings of the 9th International Workshop on Search-Based Software Testing*, pages 1–4. ACM, 2016.
10. J. Xuan, X. Xie, and M. Monperrus. Crash reproduction via test case mutation: Let existing test cases help. In *ESEC/FSE*, pages 910–913. ACM, 2015.

Appendix: Primary Evaluation Results

Table 1 shows the primary results of evaluating our approach on crash cases, which were also used to assess two other recent techniques for crash reproduction, STAR, and MuCrash.

Table 1: Detailed crash reproduction results

Bug ID	**% Successful Replication**	**STAR** [2]	**MuCrash** [10]
ACC-4	30/30	Yes	Yes
ACC-28	30/30	Yes	Yes
ACC-35	30/30	Yes	Yes
ACC-48	30/30	Yes	Yes
ACC-53	**28/30**	Yes	**No**
ACC-70	**30/30**	**No**	**No**
ACC-77	**30/30**	Yes	**No**
ACC-104	0/30	Yes	Yes
ACC-331	**10/30**	**No**	Yes
ACC-377	0/30	No	No

Variability Mining for Extractive Software Product Line Engineering of Block-Based Modeling Languages

David Wille

Institute of Software Engineering and Automotive Informatics
Technische Universität Braunschweig, Germany

Motivation

Nowadays, industry faces an increasing number of challenges regarding the functionality, efficiency, and reliability of developed software. Companies attempt to overcome these challenges by abstracting from the concrete problems using a multitude of programming and modeling languages. Depending on the development domain, these languages and the resulting degree of abstraction differs significantly. For instance, procedural programming languages or object-oriented programming languages, such as C, C++, or Java, only allow to abstract from the underlying execution hardware (i.e., the concrete memory position of variables), while, in other domains, this abstraction is not sufficient to efficiently handle the problems [1]. Especially in domains developing embedded software systems for complex hardware, such as the automotive domain or the aviation domain, companies attempt to overcome challenges during development by applying model-based languages. These model-based languages, such as *The Mathworks MATLAB/Simulink*[1] and *IBM Rational Rhapsody*[2], can be used to abstract from concrete problems on a much higher level in order to allow developers to solve problems on a more understandable level [1]. Using the developed software models, up to 80% of the code used on embedded control units can be generated [2].

While all these programming languages help to alleviate the problems linked with the complexity inherent in the different target domains, developing *variants* with largely similar yet different functionality, compared to existing products, is still a time consuming and costly task. For example, such variants need to satisfy the growing demand of customized car variants in the automotive domain. Thus, copying existing software variants and modifying them to changed requirements is common practice,

[1] `http://www.mathworks.com/products/simulink/`
[2] `http://www.ibm.com/software/awdtools/rhapsody/`

especially for domains with large numbers of product variants (e.g., the automotive domain) [3]. These so-called *clone-and-own* approaches are an efficient means to create new variants from existing products as the new software does not have to be developed from scratch, but reuses a large set of existing functionality.

Although, these clone-and-own approaches help companies to reduce the initial overhead when creating new software variants, they involve risks for the development process in the long run and are considered to be harmful for maintaining a set of related variants as no managed reuse strategy is applied [4, 5]. Over time, the set of related variants and the associated maintenance effort grows as, in most cases, the relation between created variants is not documented and only is implicitly known by domain experts. As a result, fixing an identified error in all variants of the growing *software family* becomes a complex task because each variant has to be manually analyzed and multiple solutions might be needed to fix the bug in all variants. Overall maintaining a large set of related software variants, which evolved using clone-and-own techniques, becomes a costly and tedious task [3].

Literature recommends to apply *product line engineering (PLE)* methods in order to realize more sophisticated reuse techniques and systematic reuse of artifacts amongst related products [6, 7]. Nevertheless, most companies refrain from considering these *software product lines (SPL)* as adopting them involves migration processes, which expose them to high risks (e.g., the development is interrupted during the transition to an SPL) [5].

Approach

To provide a possible alternative solution to these high-risk SPL adaption strategies, we introduced *family mining*, a reverse engineering technique allowing to (semi-)automatically analyze an existing basis of related block-based model variants and to generate an annotated *150% model* containing the variability information for the corresponding software family [8–11]. Such a 150% model stores all implementation artifacts of the analyzed software family and annotates whether the artifacts are contained in all variants (i.e., common artifacts) or only in certain variants (i.e., varying artifacts). The resulting 150% model identified by our extractive variability mining approach provides a basis for systematic reuse of the existing software artifacts in an SPL and analysis of the underlying software system.

Based on the generated 150% model, we realized an approach to automatically transition from clone-and-own approaches to an SPL. For our approach, we utilize *delta-modeling* [12] as a transformational SPL technique to transform an existing variant to any other variant by using transformational *delta operations* to *add*, *remove*, or *modify* elements. Using the identified variability from the 150% model, we are able to automatically generate *delta languages* providing corresponding delta operations specifically tailored for modifying model elements from the used programming language [13]. In addition, we use the variability information from the 150% model to generate a set of *delta modules* storing the delta operation calls for the newly generated delta language that are needed to transform between the variants analyzed during family mining [13]. These delta modules allow to generate all variants contained in the analyzed product family by applying them to an existing variant. In addition, developers can easily extend the generated SPL by manually developing new delta modules using the delta language generated with our approach. Furthermore, it is possible to automatically integrate complete new variants by generating corresponding delta modules using our automatic generation technique. As a result, our approach allows to minimize the initial risks of adopting PLE methods as the variants from the existing software family can be transfered step-by-step to a *fully-integrated platform* [14] relying completely on SPL techniques. Thus, our approach allows to make PLE techniques available for a set of related software variants that were previously maintained in isolation and can now benefit from development in a single repository of efficiently reusable functionality, automatic variant generation, and improved maintenance (e.g., bug fixes are applied to a single repository and solve problems in all variants by simply regenerating all affected variants).

References

1. Florian Deissenboeck, Benjamin Hummel, Elmar Jürgens, Bernhard Schätz, Stefan Wagner, Jean-François Girard, and Stefan Teuchert. Clone Detection in Automotive Model-based Development. In *Proc. of the Intl. Conference on Software Engineering (ICSE)*, ICSE '08, pages 603–612. ACM, 2008.
2. Michael Beine, Rainer Otterbach, and Michael Jungmann. Development of Safety-Critical Software Using Automatic Code Generation. Technical Report 2004-01-0708, SAE International, 03 2004.
3. Nam H. Pham, Hoan Anh Nguyen, Tung Thanh Nguyen, Jafar M. Al-Kofahi, and Tien N. Nguyen. Complete and Accurate Clone Detection in Graph-based Models. In *Proc. of the Intl. Conference on Software Engineering (ICSE)*, ICSE '09, pages 276–286. IEEE, 2009.

4. C. Kapser and M. W. Godfrey. "Cloning Considered Harmful" Considered Harmful. In *Proc. of the Working Conference on Reverse Engineering (WCRE)*, WCRE '06, pages 19–28. IEEE, 2006.
5. Yael Dubinsky, Julia Rubin, Thorsten Berger, Slawomir Duszynski, Martin Becker, and Krzysztof Czarnecki. An Exploratory Study of Cloning in Industrial Software Product Lines. In *Proc. of the European Conference on Software Maintenance and Reengineering (CSMR)*, CSMR '13, pages 25–34. IEEE, 2013.
6. Paul C. Clements and Linda M. Northrop. *Software Product Lines: Practices and Patterns.* 2001.
7. Klaus Pohl, Günter Böckle, and Frank J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques.* Springer, 2005.
8. Sönke Holthusen, David Wille, Christoph Legat, Simon Beddig, Ina Schaefer, and Birgit Vogel-Heuser. Family Model Mining for Function Block Diagrams in Automation Software. In *Proc. of the Intl. Workshop on Reverse Variability Engineering (REVE)*, SPLC '14, pages 36–43. ACM, 2014.
9. David Wille, Sönke Holthusen, Sandro Schulze, and Ina Schaefer. Interface Variability in Family Model Mining. In *Proc. of the Intl. Workshop on Model-Driven Approaches in Software Product Line Engineering (MAPLE)*, SPLC '13, pages 44–51. ACM, 2013.
10. D. Wille. Managing Lots of Models: The FaMine Approach. In *Proc. of the Intl. Symposium on the Foundations of Software Engineering (FSE*, FSE '14, pages 817–819. ACM, 2014.
11. D. Wille, S. Schulze, C. Seidl, and I. Schaefer. Custom-Tailored Variability Mining for Block-Based Languages. In *Proc. of the Intl. Conference on Software Analysis, Evolution, and Reengineering (SANER)*, volume 1 of *SANER '16*, pages 271–282. IEEE, 2016.
12. Ina Schaefer, Lorenzo Bettini, Viviana Bono, Ferruccio Damiani, and Nico Tanzarella. Delta-Oriented Programming of Software Product Lines. In *Software Product Lines: Going Beyond*, volume 6287, pages 77–91. Springer, 2010.
13. D. Wille, T. Runge, C. Seidl, and S. Schulze. Model-Based Delta Generation Using Extractive Software Product Line Engineering. In *Proc. of the Intl. Conference on Generative Programming and Component Engineering (GPCE)*, GPCE '16, 2016. submitted for review.
14. Michał Antkiewicz, Wenbin Ji, Thorsten Berger, Krzysztof Czarnecki, Thomas Schmorleiz, Ralf Lämmel, Stefan Stănciulescu, Andrzej Wasowski, and Ina Schaefer. Flexible Product Line Engineering with a Virtual Platform. In *Proc. of the Intl. Conference on Software Engineering (ICSE)*, ICSE '14, pages 532–535. ACM, 2014.

Decision-Making Support for Software Adaptation at Runtime

Edith Zavala, Xavier Franch, Jordi Marco

Software Service Engineering research group (GESSI)
Universitat Politècnica de Catalunya (UPC)
Barcelona, Catalunya, Spain

Abstract

In the last years, self-adaptive systems have become a crucial topic in the research and industry areas. Many initiatives to explore solutions for this kind of systems have emerged from both communities. One of the most popular approaches is the use of feedback loops. The autonomic element introduced by Kephart and Chess [1] and popularized with the IBMs architectural blueprint [2] for autonomic computing is one of the most accepted feedback loop solutions (see Figure 1). For instance, in [3] a feedback loop is utilized for detecting and adapting requirements that depend on context, affected by uncertainty. This approach uses machine learning techniques for identifying patterns on top of sensed data and determining the best adaptation of requirements, at runtime. According to the validation performed in [4] this approach is a very promising solution for executing self-adaptation at runtime.

Currently, more and more approaches use data analytics (e.g. data mining) for exploiting great amounts of runtime data collected through a monitoring infrastructure (e.g. through sensors and online data sources) in order to support the system runtime adaptation, as in [3]. The quality of the data (i.e. completeness, correctness, timely, etc.) provided by the monitoring infrastructures (e.g. Monitor element in Fig. 1) affects directly the performance of the self-adaptive systems. Several approaches have been proposed to support monitoring of software systems; however, according to [5] most of them assume that the monitors are static components. This implies that the providers must know everything to be monitored at design time. This vision is too rigid to be usable in realistic settings. Nowadays, monitoring infrastructures supporting self-adaptive systems need to be reconfigured at runtime as well in order to respond to context changes, e.g. new measures to collect, at a different sampling rate, with a different protocol, etc.

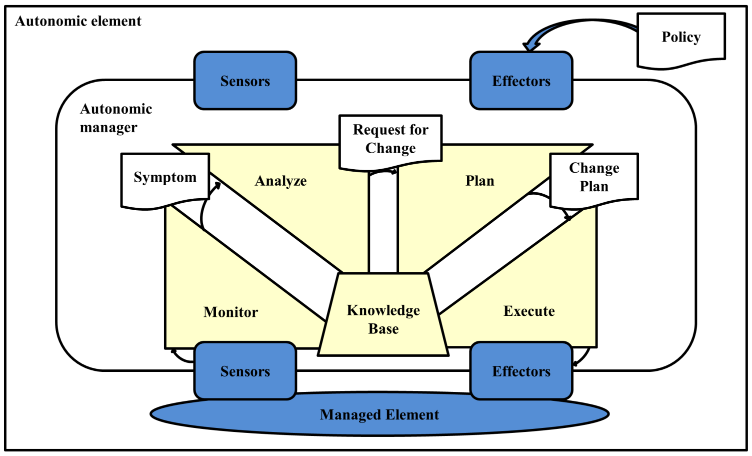


Fig. 1: IBMs autonomic element (from [1–3])

Motivated by the need of new and innovative methods and techniques for effectively and efficiently analyzing, planning and applying monitoring infrastructures reconfiguration decisions at runtime, we have explored new and existing solutions, and we have proposed an ongoing Monitoring Reconfiguration approach (see Fig. 2). Our approach, based on the MAPE-K loop, intends to support Monitor element's reconfiguration for self-adaptive systems (managed by a feedback loop). Through the MAPE-K feedback loop our approach identifies conditions in which the Monitor could require a reconfiguration aligning it with the reconfiguration process of the Managed Elements and coordinately analyses, plans, and executes the reconfiguration and adaptation of both the Monitor (using the Knowledge Base as communication channel) and the Managed Elements.

In order to do so, we introduced a set of new elements to the MAPE-K loop presented in [3] and validated in [4]: a Mine Data element which is in charge of intelligently apply one or more data analytic techniques over the sensed data; a Reconcile Reconfigurations element which coordinates the reconfigurations of the Monitor and the Managed Elements in order to prevent e.g. contradictory reconfigurations, undesired states, etc.; and

Policies which in our solution, unlike [3], are utilized by all the MAPE-K elements, in order to abstract specific application (i.e. Managed Elements) details, and distributed by the Knowledge Base. Finally, our approach allows the Domain Experts to modify Policies and supervise and request reconfigurations at runtime.

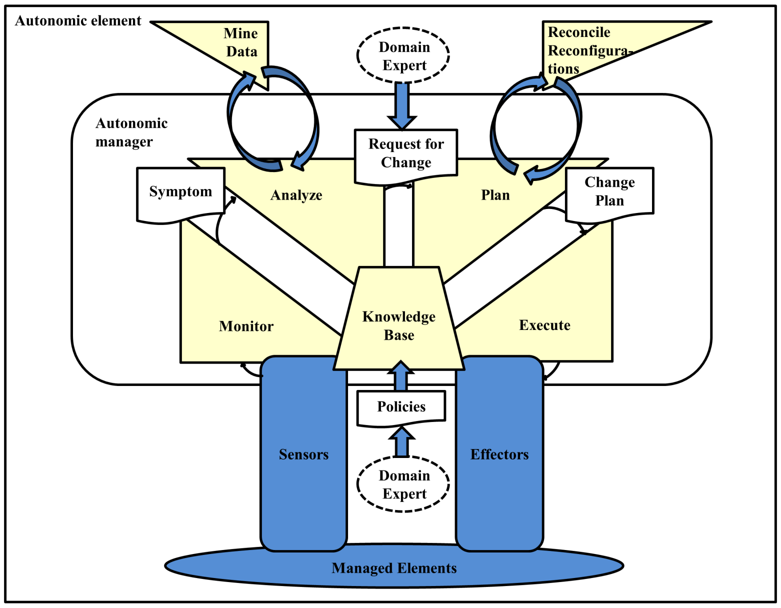


Fig. 2: Monitoring Reconfiguration approach high-level global view

Thanks to the abstraction of the Managed Elements' details (e.g. domain variables, technologies, etc.) our solution can support and be reused for an increasing number of applications at both architectural and implementation level and at design and runtime. Currently, we are defining a set of use cases in order to validate our approach.

References

1. J.O. Kephart and D.M. Chess. The vision of autonomic computing. *IEEE Computer*, 36(1):4150, 2003.
2. IBM Corporation. An architectural blueprint for autonomic computing, 2006.

3. A. Knauss et al. Acon: A learning-based approach to deal with uncertainty in contextual requirements at runtime. *Information and Software Technology*, 70:85–99, 2016.
4. E. Zavala et al. Sacre: A tool for dealing with uncertainty in contextual requirements at runtime. In *23rd IEEE International Requirements Engineering Conference (RE)*, pages 278–279, 2015.
5. R. Contreras and A. Zisman. A pattern-based approach for monitor adaptation. In *IEEE International Software Science, Technology & Engineering Conference (SW-STE)*, pages 30–37, 2010.

HDSO: Harvest Domain Specific Non-taxonomic Relations of Ontology from Internet by Deep Neural Networks (DNN)

Xuejiao Zhao[*]

Nanyang Technological University
xjzhao@ntu.edu.sg

Abstract. In ontology learning domain, non-taxonomical relations extraction is less studied. In general, there are two research tasks to get the non-taxonomical relations: detect concepts and extract their relations. Traditional methods get the concepts by shollow NLP tools like part-of-speech (POS) and analyze on morphologic or tf/idf measurement, then extract the non-taxonomic relations of the concepts by rule-based method. However, all of the traditional methods ignore the dependency of terms in sentences, and their statistic refinement process also lose many important relations which appear infrequently in the data resource. In this research, a automatic method named *HDSO* is proposed to discover domain specific concepts and their non-taxonomic relations of ontology from domain specific website. We leverage the *openIE* to get the non-taxonomic relations candidates and extract novel features of those candidate sentences, then we rank the importance of the relations by self-training DNN(deep neural networks) classifier. The accuracy of HDSO is 0.74 higher than 0.68 which get by SVM, the result shows that with the self-training DNN classifier, non-taxonomic relations can be extracted more accurately. For evaluation of our method, we utilize *HDSO* to Stack Overflow – a Q&A website about computer programming, and the tagWiki, questions and answers contain huge number of sentences with affluent relations descriptions of software engineering domain. As a result we get 15790 concepts, 9660 non-taxonomic relations.

Introduction

Ontology is a formal, explicit specification of a shared conceptualization [1]. The define of ontology is $O = (C, R, A, Top)$, which C is the concepts sets, R is the relations set of the assertions among the concepts, A is the axioms and Top is the highest concepts of current domain. There are 2 subsets of R — H (taxonomic relations) and N (non-taxonomic relations) [2]. Ontology represents knowledge with high degree of organization in structural or semi-structural semantics, it's very useful in information

[*] Advisor:Zhenchang Xing, Nanyang Technological University,ZCXing@ntu.edu.sg

retrieval, information maintenance and information intelligence, etc. [3]. But manual building of ontologies by domain sophisticates is labour intensive, time-consuming, biased towards sophisticates, and specific to the construction motivation [4].

In this research, we propose a novel system called *HDSO*(Harvest Domain Specific Ontology) to extract all the concepts and their non-taxonomic relations from domain specific website automatically. In which we leverage the advanced Open information extraction (*open IE*) of Stanford to the extraction of structured relation triples. Then we extract the text features, relations features, concepts features and resources features from the candidate relations by leveraging NLP parsing techniques, incorporating statistical, lexico-syntactic, enrichment of Google trend, etc. With the features extracted above, a self-training DNN(deep neural networks) classifier is adopted to rank all the candidate relations. We conduct a case study which extracts ontological knowledge of software engineering domain by *HDSO*, specifically key concepts and semantic relations from *tagWiki*. The experiment results show that the concepts and corresponding non-taxonomic relations extracted by our system is more concise and contains a richer semantics compared with alternative systems.

More specifically, our major contributions include:

– Propose a new method called *HDSO* to extract domain specific concept and non-taxonomic relations of ontology from domain related website.
– Leverage *HDSO* to Stack Overflow tagWiki, and get 15790 concepts, 9660 non-taxonomic relations.
– Deliver 3 corpus to public: software engineering domain specific concept set, non-taxonomic software engineering domain specific relations set with category, software engineering domain specific concept relation network.

The Approach

Architecture of HDSO

In this paper, we propose a novel system called *HDSO*(Harvest Domain Specific Ontology) to extract all non-taxonomic relations from domain specific website automatically. The *HDSO* comprises 5 main components as shown in Fig. 1, namely data pre-processing, extract candidate relations, add NLP makeup, extract features of candidate relations, rank candidate relations. The input of *HDSO* is the text materials from do-

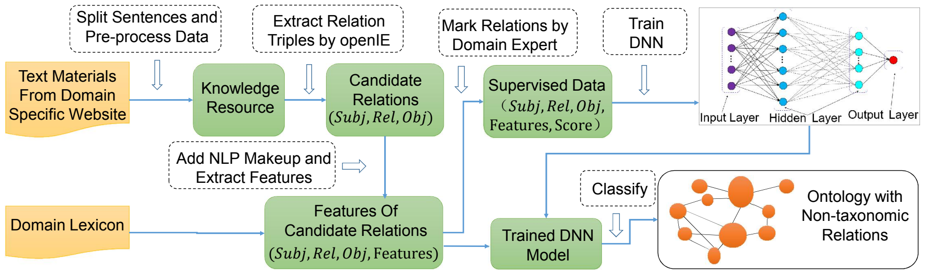

Split Sentences and Pre-process Data
Extract Relation Triples by openIE
Mark Relations by Domain Expert
Train DNN
Text Materials From Domain Specific Website
Knowledge Resource
Candidate Relations (Subj, Rel, Obj)
Supervised Data (Subj, Rel, Obj, Features, Score)
Input Layer
Hidden Layer
Output Layer
Add NLP Makeup and Extract Features
Classify
Ontology with Non-taxonomic Relations
Domain Lexicon
Features Of Candidate Relations (Subj, Rel, Obj, Features)
Trained DNN Model

Fig. 1: The Structure of *HDSO*

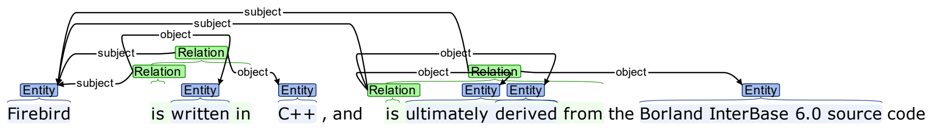

subject
object
Relation
Entity
Firebird
is written in
C++ , and
is ultimately derived from the Borland InterBase 6.0 source code

Fig. 2: *open IE* results of the definition sentence of the Firebird

main specific websites and the domain lexicon. e.g. Healthcare website, cooking website, computer programming website. We extract text materials from the domain specific websites, after data pre-processing, we use open IE to split the input sentences into a set of clauses. *open IE* is a system referring to discover possible structured relations of interest from plain sentence, leverage the dependency parse in *open IE*, we generate candidate relation triples like $O = (subj, rel, obj)$ as shown in Fig. 2.

Then we incorporate coreNLP of Stanford for POS tagging, and utilize domain lexicon to generate features(e.g. sum of tf-idf, POS fractions) for candidate relation triples. We mark 1000 candidate relation triples as supervised data and train a self-training DNN classifier. Then we use the trained model to classify the rest candidate relation triples.

After the above 4 steps, the non-taxonomic relations of domain ontology are extracted from the raw materials.

Extract Features of Candidate Relations and Self-Training DNN Classifier

After getting the candidate relations from *open IE* and add the NLP makeup to all of them, we need to rank them by how reasonable the extracted concepts and how well the relations describe the concepts. There are many properties beyond simple term statistics which can be extract from the candidate relation then use to evaluate the qualification of the candidate relations. We extract the features of every candidate relations and leverage a self-training DNN classifier to find the important relations. For the complex features we provide a brief description bellow. **cr**

Feature These features evaluate the quality of the whole candidate relation. TF-IDF is a numerical statistic which can evaluate the importance of a word to a document in a corpus [5]. The feature *sum_tfidf* and *average_tfidf* reflect the importance of the words in candidate relation. If the words in the candidate relation is very general like it, language, the *sum_tfidf* and *average_tfidf* value will be very low. Other wise if the words in the candidate relation is unique and distinctive like "c++ library", "MySQL relational database", the *sum_tfidf* and *average_tfidf* value will be high. For the *#mention_cr*, high value meant this relation extract from the raw materials many times, so the candidate relation is probably a reasonable relations. *#domain_keyword_cr*, obviously the candidate relation which contains more domain keywords show higher reasonability. The pos fraction use following definition:

- Noun : pos tag equal 'NN', 'NNS', 'NNP', 'NNPS'.
- Verb : pos tag equal 'VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ'
- Adj : pos tag equal 'JJ', 'JJR', 'JJS'

Relation Phrase Features These features evaluate the quality of the relation phrase. The *span_rel* is the start token and the end token which the rel extract from the original sentence.

Concept Features These features evaluate the quality of the concept. Google Trends is a public web facility of Google Inc., based on Google Search, that shows how often a particular search-term is entered relative to the total search-volume across various regions of the world, and in various languages. The horizontal axis of the main graph represents time (starting from 2004), and the vertical is how often a term is searched for relative to the total number of searches, globally.

Evaluate Candidate Relations by Deep Neural Networks (DNN)

From those mentions, we manual mark 1000 candidate relations with 0(unimportant relations) and 1(important relation)

Fig.3 is the framework of using deep neural networks, x stands for input, the features passed forward from the networks previous layer. Many xs will be fed into each node of the last hidden layer, and each x will be multiplied by a corresponding weight w. The sum of those products is added to a bias and fed into an activation function. In this case the activation function is a rectified linear unit (ReLU), commonly used and highly useful because it doesnt saturate on shallow gradients as sigmoid activation functions do. For each hidden node, ReLU outputs an activation, a, and the activations are summed going into the output node, which simply passes the activations sum through. That is, a neural network performing binary classification will have one output node, and that node will just multiply the sum of the previous layers activations by 1. The result will be 0 or 1 according to the sum.

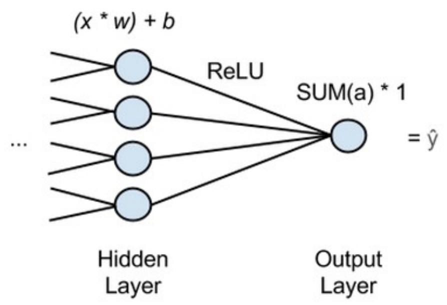


Fig. 3: Using Deep Neural Networks Classifier

Result

Fig.4 is non-taxonomic relation network of ontology which extracted from the tagWiki of Firebird. The blue lines indicate taxonomic relations and the other lines indicate non-taxonomic relations. With such a ontology we can develop the direct search engine which answer the query with entity name.

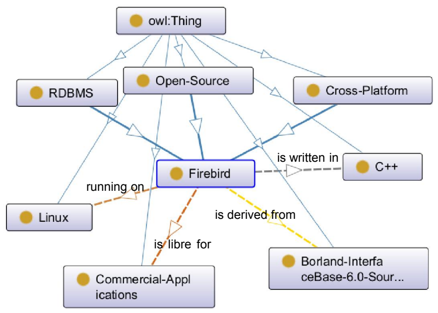


Fig. 4: An Example of the Result of HDSO

Fig.5 is the accuracy in each iteration. According to the results, the accuracy increases consistently when we perform more iterations. In particular, after the seventh iteration, our approach achieves the best accuracy of 74.05%.

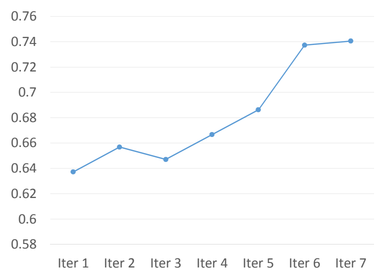


Fig. 5: Accuracy in Each Iteration

Acknowledgements

This work was conducted within the Rolls-Royce@NTU Corporate Lab with support from the National Research Foundation (NRF) Singapore under the Corp Lab@University Scheme.

References

1. Thomas R Gruber et al. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.
2. Mehrnoush Shamsfard and Ahmad Abdollahzadeh Barforoush. Learning ontologies from natural language texts. *International journal of human-computer studies*, 60(1):17–63, 2004.
3. Yan Yalan and Zhang Jinlong. Building customer complaint ontology: Using owl to express semantic relations. 2006.
4. Harith Alani, Sanghee Kim, David E Millard, Mark J Weal, Wendy Hall, Paul H Lewis, and Nigel R Shadbolt. Automatic ontology-based knowledge extraction from web documents. *IEEE Intelligent Systems*, 18(1):14–21, 2003.
5. Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2014.

Lessons Learned from a Literature Survey on the Application of Mutation Testing

Qianqian Zhu*, Annibale Panichella, and Andy Zaidman

Software Engineering Research Group,
Delft University of Technology
{qianqian.zhu, a.panichella, a.e.zaidman}@tudelft.nl
`http://swerl.tudelft.nl/bin/view/Main/WebHome`

Abstract. We conducted a systematic literature review (SLR) on mutation testing, with a particular focus on the understanding of how people *actually* apply mutation testing and how they cope with the well-known limitations of mutation testing, e.g., the cost to generate mutants. Our systematic literature review is based on a collection of 159 papers from 17 venues. In this article, we will mainly present the lessons we learned from the literature survey. In particular, we are going to highlight three primary findings from our systematic literature review and illustrate our research plan.

Keywords: mutation testing; systematic literature review; application; lessons

1 Introduction

Mutation testing has been very actively investigated by researchers since the 1970s and remarkable advances have been achieved in its concepts, theory, technology and empirical evidence. While the latest realisations have been summarised by existing literature review (e.g. Jia and Harman [1] surveyed more than 390 papers on mutation testing), we still lack insight into how mutation testing is *actually* applied. Therefore, we conducted a systematic literature review on the application perspective of mutation testing [2] based on a collection of 159 papers published between 1981 and 2015. We identified and classified the main applications of mutation testing, and also analysed the level of replicability of empirical studies related to mutation testing. Particularly, we characterised these studies on the basis of seven facets, including in which testing activities mutation testing is used, which mutation tools and which mutation operators are employed, and how the core inherent problems of mutation

* Software Engineering Research Group, Delft University of Technology, Mekelweg 4, 2628CD Delft, The Netherlands. Email address: qianqian.zhu@tudelft.nl

testing, i.e. the equivalent mutant problem and the high computational cost, are addressed during the actual usage. In the following sections, we will discuss our key findings related to SLR and our derived research plan.

2 Important takeaways from the SLR

Due to the sheer size of our SLR, we will highlight three key points in this paper (the readers can refer to [2] for more details):

- Many of the supporting techniques for making mutation testing applicable are still under-developed. Also, existing mutation tools are not complete concerning the mutation operators they offer. The two inherent problems of mutation testing, especially the equivalent mutant detection problem, are not well-solved in the context of our research body. For the equivalent mutant problem, manual analysis ranked the top 1 among the others.
- A deeper understanding of mutation testing is required, such as what particular kinds of faults mutation testing are good at finding, and how a certain type of mutant work when testing real software. This would help the community to develop new mutation operators as well as overcome some of the inherent challenges.
- The awareness of appropriately reporting mutation testing in testing experiments should be raised among the researchers. We analysed 159 papers where around half percent did not provide their mutation tool source and subject program source. We considered the following five elements to be essential: mutation tool source, mutation operators used in experiments, how to deal with the equivalent mutant problem, how to cope with high computational cost and subject program source.

3 Research plan

From the previous section, we remember that the key inherent problems related to mutation testing are not well-solved during the application. Our interest is to reduce the high computational cost of mutation testing caused by the generation and execution of a vast number of mutants. Our method is inspired by Fraser and Arcuri [3]: they defined *infection distance* for their mutation-based test data generation to represent weak mutation condition. We then apply this idea to pre-analyse whether the mutants are "infected" against the test cases, thus to filter these "uninfected" mutants. By employing such a mutant infection analysis, we can speed up the process of mutation testing.

References

1. Yue Jia and Mark Harman. An analysis and survey of the development of mutation testing. *Software Engineering, IEEE Transactions on*, 37(5):649–678, 2011.
2. Qianqian Zhu, Panichella Annibale, and Andy Zaidman. A systematic literature review of how mutation testing supports test activities. *PeerJ Preprints*, 2016.
3. Gordon Fraser and Andrea Arcuri. Achieving scalable mutation-based generation of whole test suites. *Empirical Software Engineering*, 20(3):783–812, 2015.