FAKULTÄT FÜR !NFORMATIK

Faculty of Informatics

Project Observation and Analysis in Heterogeneous Software & Systems Development Environments

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der technischen Wissenschaften

eingereicht von

Wikan Danar Sunindyo
Matrikelnummer 0828018

an der
Fakultät für Informatik der Technischen Universität Wien
Institut für Softwaretechnik und Interaktive Systeme

Betreuung
Betreuer : a.o. Univ. Prof. Dr. Stefan Biffl
Zweitbetreuer : o. Univ. Prof. Dr. A Min Tjoa

Diese Dissertation haben begutachtet:

a.o. Univ. Prof. Dr. Stefan Biffl   o. Univ. Prof. Dr. A Min Tjoa

Wien, 26.11.2012

Wikan Danar Sunindyo

Erklärung zur Verfassung der Arbeit

Wikan Danar Sunindyo
Dürergasse 7/12, 1060 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst have, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 26.11.2012
(Ort, Datum)

Wikan Danar Sunindyo

Acknowledgements

Praise to the God that I can finish my study in Vienna University of Technology.

I would like to thank Prof. Stefan Biffl and Prof. A Min Tjoa for guiding me through my PhD and for spending lots of time for thorough discussion and providing very constructive feedback for my research topics.

Furthermore, I want to thank my colleagues, Amin Anjomshoaa, Deepak Dhungana, Stefan Dösinger, Stefan Farfeleder, Christian Frühwirth, Erik Gotischa-Franta, Inah Omoronyia, Prof. Paul Grünbacher, Olga Kovalenko, Kristof Meixner, Martin Melik-Merkumians, Munir Merdan, Richard Mordinyi, Thomas Moser, Jürgen Musil, Andreas Pieber, Estefania Serral Asensio, Alexander Schatten, Natascha Surnic, Dindin Wahyudin, Florian Waltersdorfer and Dietmar Winkler, for interesting discussions, research collaborations and experience exchanges.

I also thank Reza Ferrydiansyah and Ida Siahaan for proofreading my thesis. Thank to my friends, Muhammad Asfand-e-yar, Muhammad Zuhri Catur Candra, Farman Ali Khan, Tasawwar Iqbal, and Iman Paryudi. Thank to Indonesian community in Vienna, Indonesian Student Association in Austria and other parties for supporting my life and my study during my time in Vienna.

Last but not least, I want to thank my wife Astri Indriyati, my children Serenada Cinta and Aninda Valerina, and my parents for supporting me during my studies.

Thank to the Directorate General for Higher Education, Department of Education of the Republic of Indonesia for sponsoring my study in Vienna.

Wikan Danar Sunindyo
Vienna, Austria
November 2012

Abstract

Software and systems development projects often depend on the expertise from multiple engineering domains, e.g., mechanical, electrical, and software engineering. Experts in these domains use tools from a heterogeneous software tool landscape, which poses the challenge of integrating the relevant data from several tools to support cooperation and to provide an overview on the project status. For example, in the power plant which consists of different stakeholders from mechanical, electrical, and software engineering fields who should exchange and share information about signals change which used by those different engineering fields, the project manager should observe the signal changes and analyze some metrics of signal changes, e.g., the number of signal changes for different stakeholders, the number of signal changes related to the project phases, the operation on signals, in order to support project progress and risk monitoring.

However, the heterogeneity of data formats and tools used by different stakeholders to manage the signal changes makes the project progress and risk monitoring is hard to do. In this thesis, a semantic integration approach is proposed to collect and integrate heterogeneous data to support the project progress and risk monitoring. By using the integrated signal change data, then the project manager can observe and analyze data to monitor the project progress and risk in broader view. The presentation of the analysis result can also be delivered to different types of stakeholders, e.g., the project manager and the engineer, make the project observation and analysis more dynamic.

The using and changing workflows in the project can be very helpful in managing process running on the project, but can also affect the progress of the project, e.g., delaying a project. The planned workflows and the event data captured on the real workflows during run time are main components for systematic process observation and analysis to support the project manager and other stakeholders.

Major challenges of engineering process observation and analysis in heterogeneous engineering environments are as follows. (1) Only loosely integrated data from heterogeneous data sources need human experts to integrate and are, therefore, not easily available for automated support for an overall analysis of the observed systems. For example, in the power plant, different stakeholders use different formats and tools to represent the signal change, hence make the observation and overall analysis hard. (2) Specific analysis methods often only work in a specific engineering field and do not support effective overview on multidisciplinary engineering projects. For example, in the power plant, the software engineers have a specific requirement to analyze only the signal change variables in database, which is actually connected to other engineering fields, but is not considered in software engineering fields. (3) The presentation of analysis results only supports a specific type of stakeholders. For example, in the power plant, the presentation of signal change analysis results for software engineer may differ to the presentation of analysis results for the project manager due to different analysis requirements.

This work introduces the *Project Observation and Analysis Framework* (POAF), a novel approach, which aims to support project managers and engineers in observing and analyzing engineering processes in heterogeneous engineering environments. The most important and novel contributions of POAF are (1) the semantic integration approach and integrated data model to support more efficient engineering process data collection and integration, (2) the using of combination of different analysis methods to strengthen the conclusion of the project status, and (3) the workflow validation cycle to support the conformance checking between the designed and the actual process model.

The POAF consists of data collection, data analysis, and data presentation steps, and builds on semantic web, statistical analysis, and process mining technologies to provide a range of methods, data sources, and tools that help project managers and engineers to conduct analysis and control tasks. For example, in the power plant, the POAF is very useful in making project progress and risk monitoring and checking the conformance between the designed and the actual process model.

The research results have been evaluated in two industrial application domains, namely open source software engineering projects and automated systems engineering projects, regarding feasibility, effectiveness and efficiency. Major results were that the framework was found useful, was at least as effective and supported more efficient workflow observation and analysis than traditional, mainly manual, approaches.

Kurzfassung

Software- und System-Entwicklungsprojekte brauchen oft die Expertise aus mehreren Engineering Disziplinen, etwa mechanischem, elektrischem und Software-Engineering. Experten aus diesen Disziplinen verwenden Werkzeuge aus einer heterogenen Software Werkzeug Landschaft. Daraus ergibt sich die Herausforderung der Integration der relevanten Daten aus mehreren Werkzeugen, um die Kooperation im Team zu unterstützen und Überblick über den Projektzustand bereit zu stellen. Projektteilnehmer aus unterschiedlichen Engineering Disziplinen arbeiten parallel und oftmals privat an Artefakten in ihren spezifischen Disziplinen, verfolgen spezifische Ziele und verfolgen unterschiedliche Arbeitsabläufe. An bestimmten Punkten im Projekt, etwa vor Meilensteinen, müssen die Experten kooperieren und ihr Wissen mit einander teilen, um gemeinsame Projektziele zu erreichen. Projektmanager haben die Aufgabe auf einer übergeordneten Ebene den Projektfortschritt in den verschiedenen Domänen zu überwachen, insbesondere auch die Beobachtung von Arbeitsabläufen, etwa Änderungskaskaden, über alle betroffenen Disziplinen hinweg.

Arbeitsabläufe im Projekt sind besonders wesentlich, um Prozesse zu steuern, können aber auch den Projektfortschritt beeinflussen, etwa ein Projekt verzögern. Die geplanten Arbeitsabläufe und die zu realen Arbeitsabläufen zur Laufzeit gesammelten Ereignisdaten sind wichtige Beiträge zur systematischen Prozessbeobachtung und -analyse um den Projektmanager und andere Projektteilnehmer zu unterstützen.

Wesentliche Herausforderungen bei der Verfolgung und Analyse von Engineering Prozessen in heterogenen Engineering Umgebungen sind daher: (1) nur lose integrierte Daten aus heterogenen Datenquellen brauchen Fachexperten zur Integration und sind daher nicht leicht für die automatisierte Analyse des Gesamtsystems verfügbar; (2) bestimmte Analysemethoden wirken oft nur in speziellen Engineering-Bereichen und unterstützen den Überblick in multidisziplinären Projekten nicht effektiv; und (3) die Darstellung von Analyseergebnissen ist oftmals auf eine bestimmte Gruppe von Projektteilnehmern zugeschnitten.

Diese Arbeit stellt das Projekt Beobachtungs- und Analyse Rahmenwerk vor, einen neuen Ansatz, der Projektmanager und Ingeneure dabei unterstützt unterschiedliche Engineering Prozesse in heterogenen Engineering Umgebungen systematisch zu beobachten und zu analysieren. Das POAF umfasst die Bereiche der Sammlung, Analyse und Präsentation von Daten aus heterogenen Software-Werkzeugumgebungen, und baut auf dem Semantic Web, statistischer Analyse und Process Mining auf, um verschiedene Methoden, Datenquellen sowie (semi-)automatisierte Werkzeuge bereitzustellen, die Projektmanager und Ingenieure bei Analyse- und Steuerungsaufgaben unterstützen.

Die Forschungsergebnisse wurden im Rahmen von zwei industriellen Anwendungsbereichen, Open Source Software Engineering Projekten und Projekten für das Engineering von Automatisierungssystemen, hinsichtlich Machbarkeit, Effektivität und Effizienz evaluiert. Wesentliche Ergebnisse der Arbeit zeigen, dass die das Rahmenwerk nützlich war, und zumindest so effektive war und die Beobachtung und Analyse von Arbeitsabläufen effizienter unterstützt hat als traditionelle, vor allem manuelle, Ansätze.

Contents

List of Figures

List of Tables

List of Listings

1 Introduction

Today’s software application development involves engineering systems and tools from several sources which have to cooperate for building agile process environments. In most cases, software cannot be seen as a stand-alone system and delivered as “shrink-wrapped package”, but embedded in larger context of systems, for example as service in some *software as a service* (SAAS)[1] context, as part of a network, or as part of some infrastructure where hardware and software components have to cooperate seamlessly (Biffl & Schatten, 2009). In heterogeneous software and systems development environments, capabilities for effective and efficient integration of engineering systems (Issarny, Caporuscio, & Georgantas, 2007) and the semantic integration of engineering knowledge (Aldred, van der Aalst, Dumas, & Hofstede, 2006) are key enablers for engineering process automation and advanced quality management.

A project is defined as a collaborative enterprise, frequently involving research or design, that is carefully planned to achieve a particular aim (Simpson & Weiner, 1989). Project management is the application of knowledge, skills, tools, and techniques to project activities to meet the project requirements (IEEE, 2011). To achieve these project requirements, project managers need to observe and analyze engineering process data from heterogeneous environments. The observation is done by collecting and integrating engineering process data across multidisciplinary engineering fields and the analysis is done by using different methods and metrics, e.g., the communication metrics, to link data to support project manager’s goal to monitor the heterogeneous systems.

Project stakeholders are individuals or groups that are actively in the project or whose interests may be positively or negatively affected as a result of project execution or project completion (IEEE, 2011). The key stakeholders on heterogeneous software and systems projects include *project managers*, *project investors*, *developers*, and *prospective users*. *Project managers* are individuals who are responsible for managing the project. *Project investors* are individuals or external groups that provide financial resources, in cash or in kind, for the project. Developers are the group that is developing products within the project. *Developers* are the group that is performing the work of the project. *Prospective users* are the individuals or organizations that will use the project’s product.

[1] http://www.saas.com

The heterogeneity of components and engineering fields involved in heterogeneous software and systems development environments make the observation and analysis of engineering processes more complex and difficult compared to of homogeneous systems. Hence, the goal of this thesis is to develop an approach to solve the observation and analysis problems in simpler environments like in Open Source Software projects before solving problems in more complex environments like in multidisciplinary automation systems. The evaluation criteria used for the proposed approach in this thesis are feasibility, efficiency and effectiveness.

Open-Source Software projects typically involve different stakeholders with different tasks and requirements, e.g., project investors, project hosts, project managers, project developers and users. Each stakeholder needs different information that could be extracted from the project data. Developers could provide the project data from their development data sources, e.g., source code management, developers' mailing list and bug reports. However, project managers should collect and integrate this data from different data sources for further analysis on the project, e.g., to support the decision for project investors or project hosts on the future of the project based on the current status of the project, whether it is sustainable or not. The challenges to observe and analyze the engineering process in OSS projects is in some extent similar to the challenges in automated systems environments, e.g., how to identify and collect the engineering process from different projects, since different processes can be applied in the project while we only focus on the engineering process, how to choose and decide analysis methods used to support different goals and requirements, and how to support results presentation to the different project stakeholders based on their analysis goals and requirements.

In multidisciplinary automation systems (Moser, Biffl, Sunindyo, & Winkler, 2010), different stakeholders from heterogeneous engineering domains, e.g., mechanical, electrical, and software engineering, often are required to collaborate to produce products, services or systems, such as power plants or production automation systems. In doing their jobs, these stakeholders use different engineering processes, methods, and tools with specific data models, addressing the individual needs of the involved engineers. One possible case is they write the processes of doing their jobs in different workflows (Lawrence, 1997) that represent the steps describing how processes are conducted, which inputs are needed and what output is produced. However, project managers need to have an integrated view on the heterogeneous workflows from different engineering fields in order to be able to manage the engineering processes and improve the process and product quality.

In general, the major challenges of engineering process observation and analysis in heterogeneous engineering environments for the two presented research application are as follows, (1) only loosely integrated data from heterogeneous data sources need human experts to integrate and are, therefore, not easily available for automated support for an overall analysis of the observed systems, (2) specific analysis methods often only work in a specific engineering field and do not support effective overview on multidisciplinary engineering projects, (3) the presentation of analysis results only supports a specific type of stakeholders. Some approaches for observing engineering processes in software engineering environments have been proposed, for example the Hackystat (Johnson, 2001) or Ginger2 (Torii, Kenichi, Kumiyo, Yoshihiro, Shingo & Kazuyuki, 1999) frameworks.

Hackystat (Johnson, 2001) is a technology initiative and research project that explores the strengths and weaknesses of a developer-centric, in-process, and non-disruptive approach to empirical software project data collection and analysis. It is designed to accelerate adoption of empirically guided software project measurement by providing a new approach to address the barriers of cost, quality, and utility. However, this research is more focusing on the observing engineering process in software project, which has limited scope comparing to the multidisciplinary engineering environments.

Ginger2 (Torii et al., 1999) is a project to create and develop a Computer-Aided Empirical Software Engineering (CAESE) framework as a substrate for supporting the empirical software engineering lifecycle. CAESE supports empirical software engineering in the same manner as a CASE environment serves as a substrate for supporting the software development lifecycle. This approach is using controlled experiments to develop and test hypothesis about particular aspect of human behaviour, especially in developing software products. Moreover this approach is focusing on collecting data of developers' action, e.g., key stroke, mouse movement, window operations, changes in window, rather on the artifacts of the project. This approach is good to measure the productivity of the developers and investigate the factors that can improve the productivity of the developers, however the quality measurement in multidisciplinary engineering environments need more aspects to observe than only the developers' activity.

Figure 1 shows the needs and motivation of engineering process observation and analysis in heterogeneous software and systems development environments that consist of different stakeholders, e.g., project managers, mechanical engineers, electrical engineers, software engineers, and bookkeepers. In design time, the project manager may have a structured process model that is expected to be run by other stakeholders in the run time. However, in the run time, the situation is more complex than expected. Each stakeholder may have his/her own workflow and run his/her own process. The project manager should collect and analyze engineering process from those different stakeholders in order to justify and match the expected process model with the process model which is discovered from the real-life actions. This justification is important to support the task of the project manager to monitor and manage the project in order to reach the goal of the project.

This thesis proposes a Project Observation and Analysis Framework (POAF) that aims to support project managers and engineers in observing and analyzing different engineering processes in heterogeneous engineering environments. The goal is to provide project managers and engineers the capabilities to observe and analyze the engineering processes more efficiently.

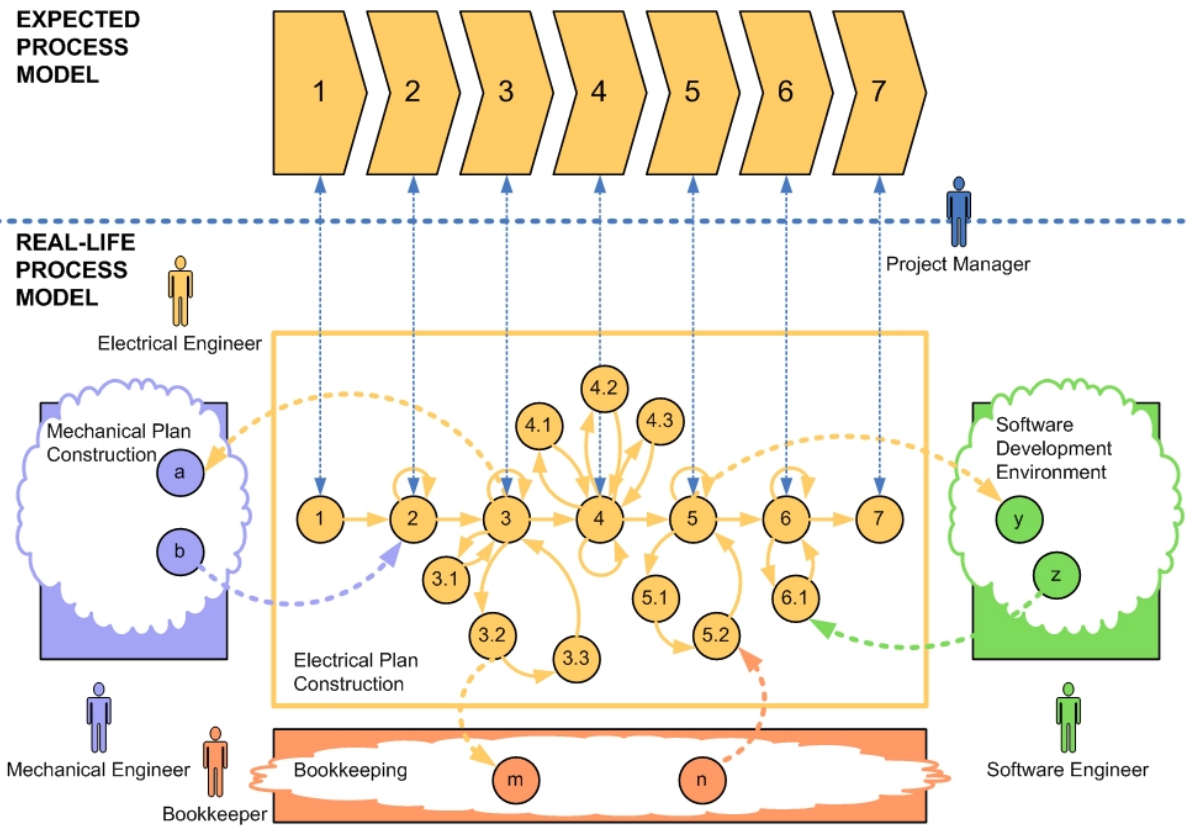


Figure 1. Motivation on engineering process observation and analysis

The POAF consists of data collection, data analysis, and data presentation steps, and involves different methods, data sources, and automated/semi-automated tools that can help project managers and engineers to do their jobs. The semantic web, statistical analysis, and process mining technologies are used in designing and developing this framework.

The aim of the POAF evaluation is to validate the feasibility of the framework to support more effective and more efficient engineering process observation and analysis in different application domains of the heterogeneous software and systems development environments. The feasibility of the framework has been tested by applying the steps and methods on two application domains, namely OSS and ASE projects to support the justification of generality of our approach. Some tools and pilot applications has been built and implemented to collect and integrate heterogeneous data from both application domains, for example Project Data Fetcher and Bug History Data Collector for OSS domain and Engineering Data Base for ASE domain.

Different methods have been introduced to analyze engineering process data for both application domains, namely health indicators, bug history, workflow validation, change management, project progress and risk monitoring, and process model validation methods. These methods are used to support different views and purposes from heterogeneous stakeholders. Some approaches also introduced to present the analysis results to different stakeholders, for example Project Monitoring Cockpit and Engineering Cockpit.

Major results show that the framework can support more efficient workflow observation and analysis compared to traditional mainly manual approaches in two application domains.

In OSS domain, a framework for OSS Data Analysis has been proposed as an instantiation of the POAF. By using this framework, we can reduce up to 30% of the data collection efforts required for the traditional manual approach. The integrated data model for OSS projects worked well to support process and project metrics for producing the health indicators. The analysis results of conformance checking of process models from OSS projects bug history can be used to improve the process quality. The findings on variations in the time needed to complete planned process steps and detected unexpected process paths can help the quality manager to plan focused and more detailed analyses and improve process control.

In ASE domain, a workflow validation cycle has been proposed as an instantiation of the POAF to observe and analyze engineering processes. By following the process steps of the framework, we can increase flexibility, improve collaboration capabilities, and the ability to measure process metrics across discipline borders, which could not easily be measured in common automation systems development processes. The project and risk monitoring in ASE domain following the process steps of the framework also can enhance the overall engineering project quality, thus enabling risk mitigation in ASE projects. The framework also can improve the efficiency of production process validation by providing information to support the project manager's decision on process improvement.

The benefits of the POAF for different stakeholders are as follows, (a) *project managers* can monitor the project progress more easily, (b) *project investors* can get the information on the project status faster, so can make immediate decision on the project continuity, (c) *developers* can collaborate with other parties from different engineering fields and track their contribution easier, (d) *prospective users* can follow the information on the project status faster, without waiting until the final product is released.

The remainder of this work is structured as follows. Chapter 2 summarizes related work on open source software development, automation systems engineering, and workflow validation. Chapter 3 describes the research approach by identifying the research issues, specifying the research methods and introducing the application scenarios. Chapter 4 introduces the Project Observation and Analysis Framework (POAF), describes usage scenarios and the generic framework architecture, as well as specifies the evaluation aspects. Chapter 5 describes the Open Source Software Projects monitoring context, integrated data model, health indicators analysis method, bug history analysis method, and workflow validation analysis method. Chapter 6 describes process analysis in automation systems engineering environments context, workflow validation cycle process, change management process, project progress and risk monitoring, and process model validation. Chapter 7 evaluates and discusses the results from two use cases. Finally, chapter 8 concludes this thesis and gives an outlook on future research perspectives.

2 Related Work

This section summarizes background information on OSS project monitoring, process analysis in ASE environments and workflow validation.

2.1 OSS Projects Monitoring

This section summarizes related work on monitoring OSS projects, methods to collect and analyze OSS project data, and research work on defining and implementing OSS project health indicators.

2.1.1 Overview

Von Krogh and von Hippel (von Krogh & von Hippel, 2003) investigated OSS development processes and found differences between monitoring commercial software development and OSS development. In commercial software development, the project manager can apply tight management of processes and take precautions, while in OSS development software architecture and functionality are governed by a community consisting of developers, who can commit code to the authorized version of the software. Therefore, OSS project development monitoring also has to be based on the community works and agreement rather than enforced regulation as in commercial software development.

Yamauchi et al. (Yamauchi, Yokozawa, Shinohara & Ishida, 2000) state that in a traditional perspective, managing and leading OSS development projects seems to be impossible, because no formal quality control program exists and no authoritative leaders monitor the development project. For them it is surprising that also OSS development can achieve smooth coordination, consistency in design and continuous innovation while relying heavily on electronic environments as face-to-face supplementary; however, project monitoring for OSS projects seems still quite fragile. In addition, they discuss how OSS development avoids limitations of dispersed collaboration and addresses the sources of innovation in OSS development. Further research is needed to reveal how typical project management methodologies can be adapted to the OSS domain in order to improve the software quality, e.g., by monitoring typical OSS project product and process data.

Wahyudin et al. (Wahyudin & Tjoa, 2007) discuss how project monitoring has traditionally been focused on human-based reporting, which is good for tightly coupled organizations to ensure the quality of project reporting. In loosely-coupled organizations like in OSS development projects, this approach does not work well, because the stakeholders typically work voluntarily and flexibly. One way to measure the performance of the project is by correlating and analyzing process event data (e.g., mailing list artifacts or bug reports) from the OSS community.

Sharma et al. (Sharma, Sugumaran & Rajagopalan, 2002) observe OSS development projects based on three aspects: structure, processes, and culture. The OSS communities can be structured along the dimensions of division of labour, coordination mechanisms, distribution of decision-making authority and organizational boundary. In OSS processes, stakeholders can have governance mechanisms, for example by applying membership management, rules and institutions, monitoring and sanctions, and reputation as one of the prime motivators for the OSS developers. Even though membership in OSS projects is open to anyone, the OSS communities manage membership effectively. They illustrate how OSS projects can be monitored via social interaction and sanctions from the communities. However, the relationships between the project data produced by the stakeholders, the activities of the stakeholders, and the quality measurement of OSS were not analyzed in their study.

To address semantic integration of data originating from heterogeneous OSS project data sources, a tool for the extraction of project data for Apache projects called Project Data Fetcher was initially developed and reported in (Moser, Biffl, Sunindyo, & Winkler, 2011). This tool allows gathering project artifacts from the mailing list, the Bugzilla[2] database, and the Subversion versioning system of Apache projects. The retrieved data allows evaluating so-called communication metrics. The Project Data Fetcher uses an ontology for storing extracted project data.

2.1.2 OSS Data Analysis Frameworks

There are several reports on tool support for a more comprehensive observation of OSS projects for data analysis. These reports involve different data sources and analysis methods as part of OSS data analysis frameworks, e.g., Alitheia Core[3] (Gousios & Spinellis, 2009a, 2009b) and Ohloh[4] (Hu & Zhao, 2008).

[2] http://www.bugzilla.org
[3] http://www.sqo-oss.org
[4] http://www.ohloh.net

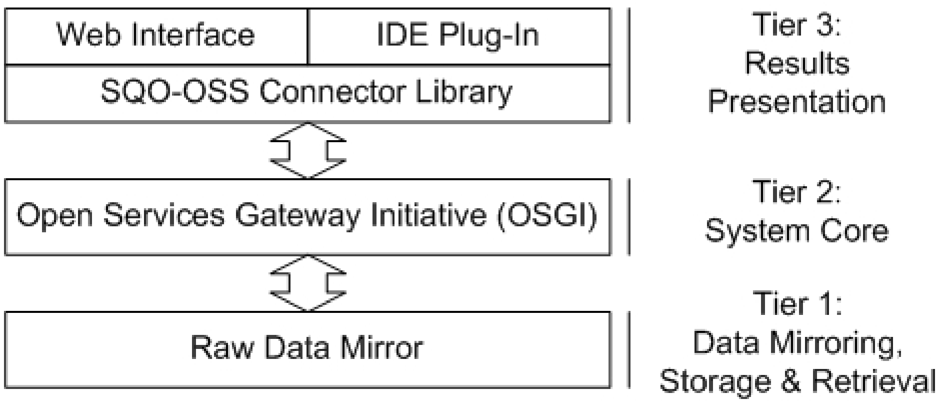

Web Interface
IDE Plug-In
SQO-OSS Connector Library
Tier 3: Results Presentation
Open Services Gateway Initiative (OSGI)
Tier 2: System Core
Raw Data Mirror
Tier 1: Data Mirroring, Storage & Retrieval

Figure 2. Simplified Alitheia Core System Architecture, see details in (Gousios & Spinellis, 2009a)

The Alitheia Core tool is an extensible platform for software quality analysis designed specifically to facilitate software engineering research on large and diverse data sources. Figure 2 shows the simplified Alitheia Core System Architecture, which consists of three tiers: (a) data mirroring, storage and retrieval (tier 1); (b) system core (tier 2); and (c) results presentation (tier 3). Tier 1 enables the collection of data from different data sources, e.g., subversion, mailing list, and bug reports. However, the mirroring of the data sources tends to make data management inefficient for large projects. Tier 2 provides a range of metric plug-ins for analyzing individual OSS data to support the interpretation of the OSS project status. However, the lack of interaction and combination between these different metrics makes the conclusions on the OSS project status less strong than analyzing integrated data sources. Tier 3 provides results presentation to web interface or IDE plug-in via SQO-OSS connector library. This platform allows importing data from OSS projects into a meta-database and provides an infrastructure to run metrics on clusters of processing nodes. Currently, this tool has been applied to analyze OSS data limited to a single OSS project community, namely the Gnome ecosystem (Gousios & Spinellis, 2009b).

Ohloh is an OSS directory that anyone can edit (Hu & Zhao, 2008). Ohloh retrieves data from revision control repositories (such as CVS[5], SVN[6], or Git[7]) and provides descriptive statistics about the longevity of projects, their license and metrics such as source lines of code and commit statistics. Currently, Ohloh provides information about 11,800 major OSS projects involving 94,330 people. However, the reports on introducing a framework for analyzing OSS project data do not report on health indicators, which allow detecting the OSS project status in a timely fashion.

[5] http://savannah.nongnu.org/projects/cvs
[6] http://tortoisesvn.net/
[7] http://git-scm.com/

2.1.3 OSS Health Indicators

The term OSS project “health indicators” was introduced by Wahyudin et al. (Wahyudin, Schatten, Mustofa, Biffl & Tjoa, 2006) to help OSS stakeholders to get an overview on a large portfolio of OSS projects. Using a health indicator can be seen as analogous to measuring the temperature of the human body with respect to indicating whether a person is likely to be sick or in healthy condition (Wahyudin, 2008). This work analyzed some project metrics, e.g., open issues, proportions, and communication metrics, in four OSS Apache projects, namely HTTPD[8], Tomcat [9], Xindice [10], and Slide[11], and discussed the results with OSS experts to investigate the external validity of the indicators. Major result was, that those important indicators such as developer activity or bug management performance are easy to measure but have to be augmented with other indicators, e.g., the probability of bug occurrence and/or experts’ opinion, which are concealed behind the development process to determine the project health comprehensively.

In the past decade, only a limited number of studies and publication addresses communication metrics, for example (Brügge & Dutoit, 1997), who reported empirical evidence that metrics based on communication artifacts generate better insight into the health of application development processes than code-based metrics. Many developers ignore the fact that software code is only available late in the development process, while communication artifacts, such as e-mails, mailing list entries, or memo notes are valuable information that is available early and can be used to investigate the health of development project. To draw valid conclusions on the communication behavior of the project members and measures for improvement, a new set of metrics has to be designed. Roche (Roche, 1994) showed that the results of these novel metrics may assist project managers and that their potential should not be ignored.

Early research towards OSS health indicators has been reported by Mockus et al. (Mockus, Fielding, & Herbsleb, 2000, 2002) who ran an experiment on two major OSS projects, Apache Web Server[12] and the Mozilla browser [13], to investigate aspects of developer participations to compare the strengths and weaknesses of OSS projects and commercial projects. However, the focus of this work is more on the comparison of

[8] http://httpd.apache.org/
[9] http://tomcat.apache.org/
[10] http://xml.apache.org/xindice/
[11] http://jakarta.apache.org/slide/
[12] http://httpd.apache.org/
[13] http://www.mozilla.org/

different aspects, e.g., size of the core team, productivity, and problem resolution intervals in the OSS projects than on project health. This research is a starting point to improve measurements that help the OSS developer and manager to obtain the project status faster by introducing the concept of project health indicators.

This work was continued by Wahyudin et al. (Wahyudin, Mustofa, Schatten, Biffl & Tjoa, 2007) by empirically evaluating development processes to get a status overview of OSS projects in a timely fashion and to predict project survivability based on the data available on project web repositories. However, the data collection for this approach was still done manually, by retrieving data from source code management, mailing lists, and bug reports websites. The high effort for manual data collection and for quality issues warrants the automation of data collection, integration, and quality assurance. The evaluation of the data was done separately for each data source, and did not yet discover further relationships between different sources, which could reveal further health indicators.

Bachmann and Bernstein (Bachmann & Bernstein, 2009) focus on using bug tracking databases and version control system log files to support a historical view analysis for improving software process data quality. The results show that a poor correlation between linked bug reports is a strong indicator for the missing traceability and justification of source code changes. The rate of linked bug reports can be observed by linking commit messages for valid bug report numbers to the numbers of all bug reports. A poor rate is obtained when the commit messages have few connections with the bug reports. We extend the using of bug reports as health indicators by combining with other metrics in OSS projects.

2.2 Process Analysis in ASE Environments

This section summarizes background information on Automation Systems Engineering, the Engineering Service Bus (EngSB), the Engineering Cockpit (EngCo) prototype, and on risk management.

2.2.1 Overview

Automation systems, e.g., complex industrial automation plants for manufacturing, steel mills, or power plants include a set of heterogeneous engineering environments, e.g., mechanical, electrical, and software engineering disciplines who should collaborate and interact for successfully completing ASE projects (Biffl, Sunindyo, & Moser, 2009). Expert knowledge is embodied in domain-specific standards, terminologies, people, processes, methods, models, and tools (Moser, Biffl, et al., 2011; Moser, Mordinyi, Mikula, & Biffl, 2009). Nevertheless, individual disciplines including discipline specific tools and data models and are isolated and/or with limited support for interaction and collaboration (Biffl, Schatten, & Zoitl, 2009). Thus, a major challenge is to synchronize specification data and plans from a wide range of engineering aspects in the overall engineering process, e.g., physical plant design, mechanical and electrical engineering artifacts, and process and project planning (Winkler & Biffl, 2012).

Figure 3 illustrates a basic engineering process, observed at our industry partner, including five phases in sequential order: initial, drawing started, customer approval, factory tests, and customer commissioning. Note that these phases correspond to the individual states of engineering objects. A more detailed view on the sequential steps, e.g., during the phase "drawing started", showed that engineers follow their own (isolated) engineering processes within their assigned discipline or domain. In addition, engineers from individual disciplines work in parallel on similar engineering objects from different perspectives (Winkler & Biffl, 2012). Thus, they have to synchronize and exchange data to keep the engineering project consistent. Note that similar processes apply for all engineering phases.

Changes from disciplines have to be passed to related engineers who might be affected by those changes. For instance changing a sensor from hardware perspective might have an impact on electrical engineers (how to connect the sensor) and to the software engineer (how to control and analyze sensor data). Observations in industry projects showed a less frequent and informal synchronization process, executed by experts manually. Because of a high effort of human experts, who are familiar with at least two engineering disciplines, this synchronization process is executed less frequently and, thus, include a high risk regarding the consistency of engineering objects and the impact of changes.

Based on our observation we found a set of risks in the ASE which can have a major impact on the individual engineers and on the project: (a) Domain specific risks focus on individual and isolated disciplines, where engineers apply well-established risk management approaches, e.g., RiskIt (Kontio, 1999) for the software engineering domain. As individual disciplines can apply appropriate countermeasures which have effects on these disciplines, related disciplines might be affected by these measures; (b) Collaboration risks focus on the need for frequent synchronization of individual artifacts and engineering objects coming from different disciplines. Because of a high manual effort for synchronization (if not automated) the frequency of data exchange is quite low; e.g., once per month. If done less frequently the number of changes might be very high leading to additional risks with respect to related disciplines in case of changes; (c) project management risks focus on project monitoring and control challenges, which usually depend on the capability to capture and analyze project data and draw appropriate solutions. Because of a lack of synchronization and limited access to comprehensive data additional risks arise, even if the data are available very late in the project. Thus, late changes, e.g., during the factory test or during the commissioning phase at the customers' site, result in inefficient, error-prone and risky engineering processes (Sadiq, Orlowska, Sadiq, & Foulger, 2004).

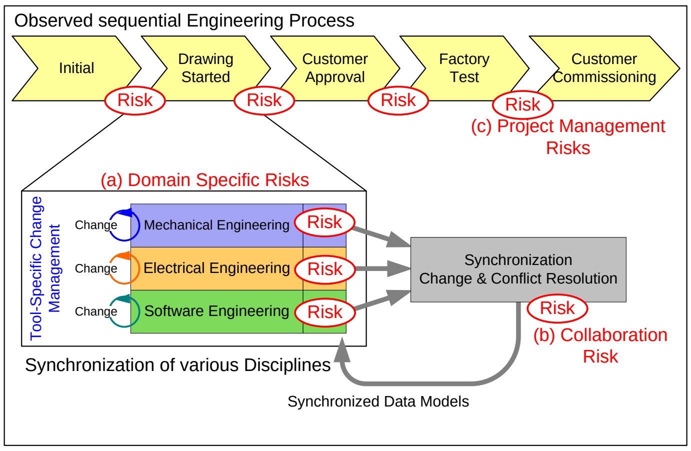


Figure 3. Management and Engineering Risks from Process Perspective (Winkler & Biffl, 2012).

To overcome risks on (a) management level, i.e., enabling project observation and control across disciplines and domain borders and (b) on engineering level, i.e., supporting efficient change management and frequent synchronization across disciplines, the Engineering Service Bus (Biffl, Schatten, et al., 2009) supports interaction of related stakeholder within a heterogeneous engineering environments with respect to improving (a) engineering processes and change management, (b) quality assurance activities, and (c) risk management in the ASE domain.

2.2.2 Engineering Service Bus

Current developers of software systems use a wide range of tools from software vendors, open source communities, and in-house developers. Getting these tools to work together to support a development process in an engineering environment remains challenging as there is a wide variety of standards these tools follow (IEEE, 2007). Any integration approach has to address the levels of technical heterogeneity, i.e., how to connect systems that use different platforms, protocols, etc., so they can exchange messages (Chappell, 2004; Hohpe & Woolf, 2003; Rademakers & Dirksen, 2008) and semantic heterogeneity, i.e., how to translate the content of the messages between systems that use different local terminologies for common concepts in their domain of discourse, so these systems can understand each other and conduct a meaningful conversation (Aldred et al., 2006; Doan, Noy, & Halevy, 2004; Hohpe, 2006; Moser et al., 2009; Noy, Doan, & Halevy, 2005). Particularly in ASE, integration of engineering systems is a challenge as typically a broad range of engineering tools from different vendors are used to solve specific problems (Rangan, Rohde, Peak, Chadha, & Bliznakov, 2005).

Biffl and Schatten proposed a platform called Engineering Service Bus (EngSB), which integrates not only different tools and systems but also different steps in the software development lifecycle (Biffl, Schatten, et al., 2009). The platform aims at integrating software engineering disciplines e.g., mechanical, electrical or software engineering, rather than individual services (Chappell, 2004). The EngSB consists of the following main components: (1) engineering discipline specific tools to be integrated (2) so called connectors which enable communication between the bus and the specific engineering tool and which consist of a technical specific and a technical neutral interface. The technical specific interface is implemented within the engineering tool while the technical neutral interface (i.e. tool domain) represents a standardization of connectors of a specific engineering tool type. This seems possible since different tools, developed to solve the same problem have, more or less, similar interfaces. For example, the source code management (SCM) tools Subversion and CVS both provide similar functionality, which allows describing these tools as instances of the SCM tool domain. This concept allows the EngSB to interact with a tool domain without knowing which specific tool instances are actually present. Note that tool domains do not implement tool instances but provide the abstract description of events and services, which have to be provided by concrete connectors of tool instances to the EngSB. This implies that the EngSB not only facilitates data integration but more importantly functional integration as well (3) the Engineering Database (Moser, Waltersdorfer, Winkler, & Biffl, 2011) and the Engineering Knowledge Base (Moser, Biffl, et al., 2011) which enable versioning of common data used and an automated transformation of common concepts represented differently in the various engineering tools. (4) project relevant added-value applications like the Engineering Cockpit (Moser, Mordinyi, Winkler, & Biffl, 2011) for efficient project monitoring or the Engineering Object Editor (Mordinyi, Pacha, & Biffl,

2011) for quality assured integration of data sources. (5) a workflow engine executing engineering processes which describe a configurable sequence of process steps satisfying project integration requirements. The engine is responsible for the correct management of the workflow relevant rules and events while the configuration of it makes use of the modeled concepts of tool instances and tool domains in the Engineering Knowledge Base.

2.2.3 Engineering Cockpit

The Engineering Cockpit (EngCo) is a social-network-style collaboration platform for automation system engineering project managers and engineers, applying technical (Biffl, Schatten, et al., 2009) and semantic integration (Boehm, 1991; Jakoubi & Tjoa, 2009) (Sadiq et al., 2004) approaches for bridging gaps between heterogeneous ASE project data sources as foundation for comprehensive project monitoring and management, which was first introduced in (Moser, Mordinyi, et al., 2011). It builds on semantic web technology, the Engineering Knowledge Base (EKB) and semantic integration framework (Moser, Biffl, et al., 2011), to explicitly link the data model elements of several heterogeneous ASE project data sources based on their data semantic definitions.

The EngCo is generic framework for project reporting across tool and domain boundaries, and shows the prototypic implementation to demonstrate how to calculate a set of metrics for project managers and engineers. In (Moser, Mordinyi, et al., 2011) a general EngCo concept has been described and discussed by taking into account concrete evaluation data from industry. The feasibility of the EngCo prototype was evaluated by performing a set of project-specific queries across engineering discipline boundaries for information on current and historic project activities based on real-world ASE project data from our industry partner in the hydro power plant engineering domain.

Major results were that EngCo (a) enables the definition of project-specific queries across engineering discipline boundaries and therefore minimizes the effort for near-time analysis of the project progress, (b) automatically shows the current view on project progress as soon as the engineering groups send their local changes to planning data to the common data basis, and (c) enables early risk detection and analysis, e.g., an unexpectedly large number of changes to engineering objects late in the project.

2.3 Workflow Validation

This section summarizes related work on system integration technologies to integrate technologically and semantically heterogeneous systems to make them appear as one big system and process analysis and validation methods for engineering process analysis purposes.

2.3.1 System Integration Technologies

Technical integration. Several approaches have been reported for integrating technical and semantic aspects of SE environments. Mordinyi et al. propose a model-driven system configuration approach for integrating systems in the safety-critical Air Traffic Management domain (Mordinyi, Moser, Kühn, Biffl, & Mikula, 2009). This approach explicitly models the components of the heterogeneous network infrastructures to produce and deploy a technical solution model using an integration platform. However, this approach does not focus on process analysis that will be useful for analyzing system performance. Muller and Knoll propose an integrated approach for cross-platform automated software builds and the implementation of a test framework (T. Muller & Knoll, 2009). They use virtualization tools for automated software builds, tests and deployment with a large academic software library project as use case. By using this virtualization framework, the tasks for cross platform target operating systems can be performed efficiently and effectively. However, this framework has limitations in automating the interactive parts of an application.

The Enterprise Service Bus (ESB) concept originating from the business IT field offers a technical integration backbone for enterprise application integration (Chappell, 2004). The ESB separates the business logic from the integration logic and provides a distributed integration platform. MULE[14] is an example of a Java-based ESB framework that separates the business logic layer from the messaging layer. The application of the service-oriented performance modeling to the ESB is a good method that we can reproduce for different contexts in various engineering areas (Brebner, 2009). However, typical ESB systems cannot easily be bundled for deployment with individual solutions and do not support synchronization features for accommodating desktop applications that are usually not online permanently (Biffl & Schatten, 2009).

Semantic integration is defined as the solving of problems resulting from the intent to share data across disparate and semantically heterogeneous data (Halevy, 2005).

[14] http://www.mulesoft.org

These problems include the matching of ontologies or schemas, the detection of duplicate entries, the reconciliation of inconsistencies, and the modeling of complex relations in different data sources (Noy et al., 2005). One of the most important and most actively studied problems in semantic integration is establishing semantic correspondences (also called mappings) between vocabularies of different data sources (Doan et al., 2004). The application of ontologies as semantic web technologies for managing knowledge in specific domains is inevitable. Noy and Guinness (Noy & McGuinness, 2001) note five reasons to develop an ontology, namely (a) to share common understanding of the structure of information among people or software agents, (b) to enable reuse of domain knowledge, (c) to make domain assumptions explicit, (d) to separate domain knowledge from the operational knowledge, and (e) to analyze domain knowledge.

Moser et al. (Moser et al., 2010) introduced the Engineering Knowledge Base (EKB) framework as a semantic web technology approach for addressing challenges coming from data heterogeneity that can be applied for a range domains, e.g., in the production automation domain (Moser et al., 2010) and also SE. Further, Biffl et al. (Biffl, Sunindyo, & Moser, 2010a) used the approach for solving similar problems in the context of Open Source Software projects, in particular, frequent-release software projects.

Engineering Service Bus. Some approaches for managing tools in SE environments were done, for example by Heinonen (S. Heinonen, 2006; S. Heinonen, Kääriäinen, & Takalo, 2007) who introduced "tool chain", a framework supporting the efficient usage of resources and transparency between partners in collaborative software development. However, this approach primarily focuses on requirements management than on other steps of the software development lifecycle. Biffl and Schatten improved this situation by proposing a platform called Engineering Service Bus (EngSB) which integrates not only different tools and systems but also different steps in the software development lifecycle (Biffl & Schatten, 2009). The successful development of modern software-based systems, such as industrial automation systems, depends on the cooperation of several engineering disciplines, e.g., mechanical, electrical and software engineering, so-called (software+) engineering environments. The EngSB addresses requirements such as the capability to integrate a mix of user-centered tools and backend systems, mobile work stations that may go offline, and flexible and efficient configuration of new project environments and SE processes.

Figure 4 shows an overview on the elements of the EngSB platform. The technical integration of the components is based on the EngSB (1). The semantic integration between heterogeneous data models and tools is based on data models in the EKB (2). Project management includes tools to administrate, i.e., plan, monitor, and control, a software project and product requirements. Software development tools consist of the well-known types of SE tools, such as software development environments, source code management systems and build servers. Team communication tools consist of tools for synchronous and asynchronous communication and notification in the team regarding relevant events such as changes in/to the systems. The workflow engine (3) defines work steps beyond single process steps and provides functions to describe rules for integrating the communication between tools on the engineering team level. The event engine (4) stores the events on the EngSB for further process analysis and validation.

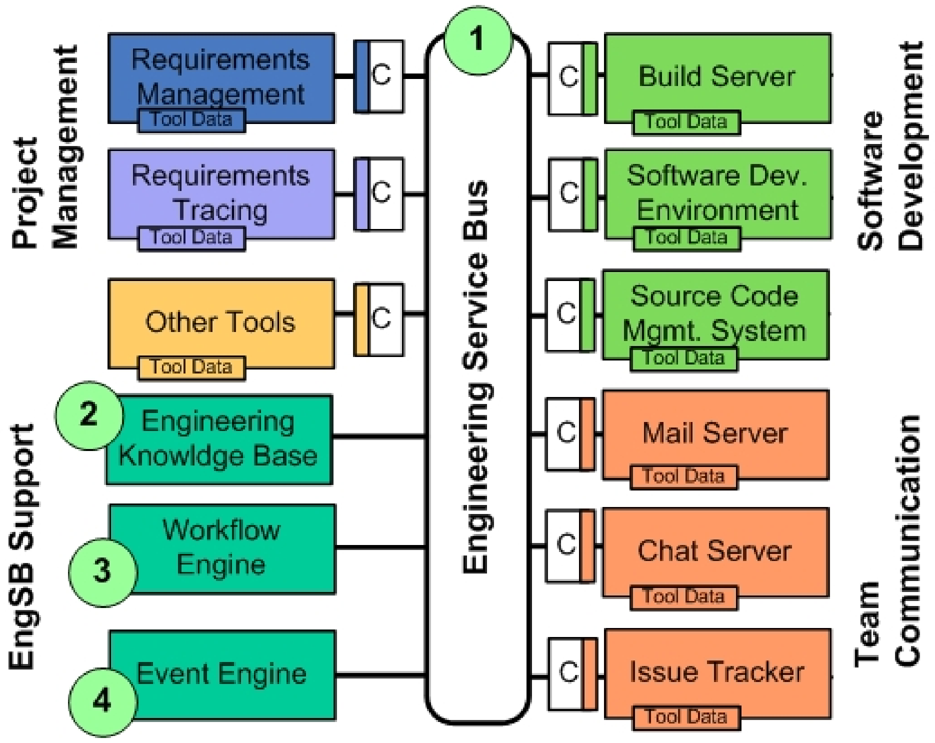


Figure 4. High-level view on tool connections with the EngSB (Biffl & Schatten, 2009)

2.3.2 Process Modeling, Analysis and Validation

Process analysis has been applied to complex systems, like workflow management systems, Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM) systems. Van der Aalst et al. use workflow technology to structure the processes running inside the systems. The workflow technology supports events provision that could be useful for process analysis in SE by enabling particular models that link basic tool events to process/workflow events (van der Aalst, Weijters, & Maruster., 2004).

Ferreira and Ferreira (Ferreira & Ferreira, 2004) proposed a reusable workflow engine based on Petri Net theory as basis for workflow management. They introduced the Workflow Kernel, a prototype implementation of common workflow functionality which can be abstracted and reused in systems or embedded in applications intended to become workflow-enabled. The workflow engine is based on common workflow functionality from several workflow engines, while the Petri net theory can be used as a process representation language for process analysis.

Another approach was proposed by van der Aalst et al. (van der Aalst et al., 2004). This approach uses stored events, which refer to tasks and process cases coming from people/tools/systems, to monitor and analyze real workflows with respect to designed workflows. This approach is called process mining, and can be used for process discovery, performance analysis, and conformance checking. The approach has been implemented in the open source tool ProM[15] and can be used to discover the process model based on the available event log, analyze the performance of the processes and suggest possible process improvement candidates.

Rembert and Ellis (A. J. Rembert & Ellis, 2009) extended process mining techniques, which focused on mining the control-flow of business processes, towards analyzing multiple perspectives of a business process. The extension of the process mining techniques includes explaining formal and general definitions of a business process perspective and presenting the approach to mine other business process perspectives using these definitions, i.e., the behavioral perspective and the role assignment perspective, that can be useful for analyzing processes in the SE context.

In order to allow more efficient and effective process monitoring, Ammon and Wolff (Ammon, Silberbauer, & Wolff, 2007) introduced complex event processing (CEP) (Luckham, 2002) for detecting event patterns in an event cloud or in event streams for Business Activity Monitoring (BAM). The reference models for event patterns can dramatically reduce time and costs as well as improve the quality of BAM projects. The challenge of the BAM domain is similar to a challenge in engineering systems, namely how to build the process model out of the event log. The events in the BAM domain are filtered from event clouds or from event streams for further process analysis, while the events in engineering domain are snapshots of running processes in certain periods. Therefore, workflow monitoring and event-analysis models and techniques can provide the theoretical foundations for event-based SE process analysis.

[15] http://www.processmining.org

3 Research Approach

This chapter describes the research approach by defining the research issues, identifying the research methods and their evaluation concept, and specifying the two application scenarios, namely Open Source Software (OSS) projects and Automation Systems Engineering (ASE) environments. The research issues consists of several issues on collecting and integrating engineering process data from different environments, how to analyze the integrated data, and how to present the analysis results to heterogeneous stakeholders.

The scope of this work is a development team consisting of project management and experts from one or several engineering discipline(s), who work on engineering process tasks with systems and role-specific tools that encapsulate project data and engineering models.

Different types of stakeholders work on the software/systems environments, e.g., the project/quality managers and the engineers from different engineering field who have different requirements and goals. The project/quality managers need to be able to monitor/observe the engineering process in order to enhance the quality of product, to ensure that the project is finished in time and the product is delivered as request.

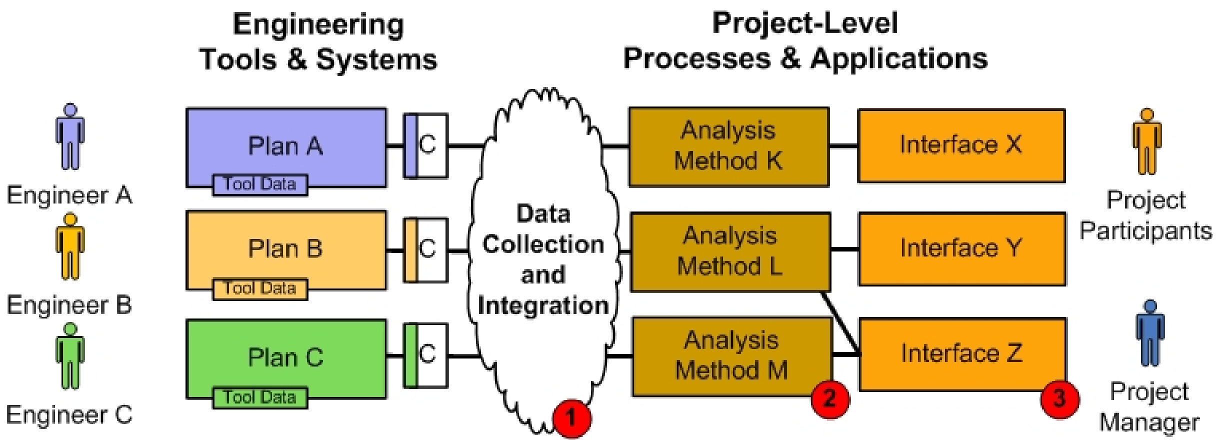


Figure 5. Overview of the research challenges

Figure 5 shows an overview of the research challenges of this thesis. The research challenges cover more general perspective of the problems, while the research issues cover more detail perspective of the engineering process observation and analysis in heterogeneous systems. In Figure 5, each engineering role (e.g., engineer A, engineer B, and engineer C) uses different engineering tools and systems (e.g., plan A, plan B, and plan C respectively) that have specific data models and formats. The example of engineering tools in the open source software projects are source code management (like SVN[16]), developers' mailing list, and bug reporting (like Bugzilla[17]) that support developers' tasks and communication between developers (Biffl et al., 2010a).

To support data exchange between different engineering tools and analysis on the engineering data, an additional component is needed. In a typical process step in the engineering process, translation tools are used to map the data model from one tool to other tools in different engineering fields. This allows data transfer and communication between the engineers. The major challenge here is that the number of translation tools across different engineering fields goes up exponentially based on the number of engineering fields and engineering tools involved in the system.

The research challenges for the heterogeneous software and systems development environments are represented as numbers in red circle shown in Figure 5, namely

(1) **Only loosely integrated data from heterogeneous data sources need human experts to integrate and are, therefore, not easily available for automated support for an overall analysis of the observed systems.** Different stakeholders used different representations of data/process model to illustrate their data/process requirements. For example, in the OSS projects, developers use SVN, developers' mailing list, and bug reports to develop the product (Biffl et al., 2010a), while in the automation systems engineering, the customers uses flowcharts to draw the informal process model and the designers use BPM notation to draw the formal process model (Sunindyo, 2011). These heterogeneous representations/notations should be managed (e.g., integrated) for further analysis by the higher levels of stakeholders, e.g., the project managers.

(2) **Specific analysis methods often only work in a specific engineering field and do not support effective overview on multidisciplinary engineering projects.** Heterogeneous data which is produced during project development of complex systems needs to be analyzed to support the project manager's decision on the project sustainability. Some specialized analysis methods have been applied to simple projects, for example using standard statistical method to measure the mean, median or mode of the projects' data. However, more complex systems demand more complex analysis methods as well. This typically involves multiple analysis methods rather focusing on single analysis method for a specific purpose (Sunindyo, Moser, Winkler, & Biffl, 2012) . We propose to use a combination of analysis methods to analyze heterogeneous data in a

[16] http://tortoisesvn.net/
[17] http://www.bugzilla.org/

complex system and compare the results with the results obtained from single analysis method.

(3) **The presentation of analysis results only supports a specific type of stakeholders.** Different stakeholders in complex systems have different concerns regarding the analysis of the project status. The presentation of analysis result should take into account the different layer/hierarchy of stakeholders, concerning their different roles and goals. Because complex, dynamic real-time data is used in complex systems, the results should be presented in a flexible way to support the usability and understandability of the analysis results. Failure to support the stakeholders' need will render the analysis results to be useless and obsolete. We propose to use dynamic visualization/presentation on reporting the analysis results for different layers of stakeholders. This visualization approach supports the analysis results by using different analysis methods for different stakeholders, for example, by giving different layouts and analysis results to the project managers and the engineers.

We propose a Project Observation and Analysis Framework ((POAF) to address these research challenges. The POAF provides tools and processes to observe and analyze engineering process from heterogeneous software and systems development environments. The major contributions that are offered by this framework are (1) **methods to collect and integrate data efficiently**, (2) **methods for analyzing engineering process data**, (3) **methods for presenting analysis results to heterogeneous stakeholders**.

3.1 Research Issues

This section identifies the research issues addressed in this thesis. The key research item of this thesis is the framework for engineering process analysis in heterogeneous software and systems development environments, which aims at enabling effective and efficient engineering process data observation and analysis to support the decision making of the project management. The common goal of these research issues is to gain a better quality assurance of the heterogeneous software and systems development environments, by monitoring the engineering processes of the projects.

The application area of the framework are engineering environments which use software engineering to manage other software or engineering fields, which range from single engineering field projects like OSS project to multidisciplinary engineering field projects like automation systems engineering projects. The heterogeneous engineering fields involved in the engineering environments make it hard for project managers to observe and analyze different engineering process data; hence it is difficult to make decisions on running projects based on dynamic/ever changing facts.

In the current approach project managers use their experience rather on real data from different engineering tools. The local perspective on local tools used by certain engineering field is quite a good fit for its own specific task. However it is not enough for larger perspectives, for instance for project managers who want to have overview on the situation across different engineering fields or different developers. Hence, a framework is proposed to help project managers to collect, integrate, and analyze heterogeneous engineering process data efficiently and effectively comparing to traditional approach which is based more on intuition and expertise.

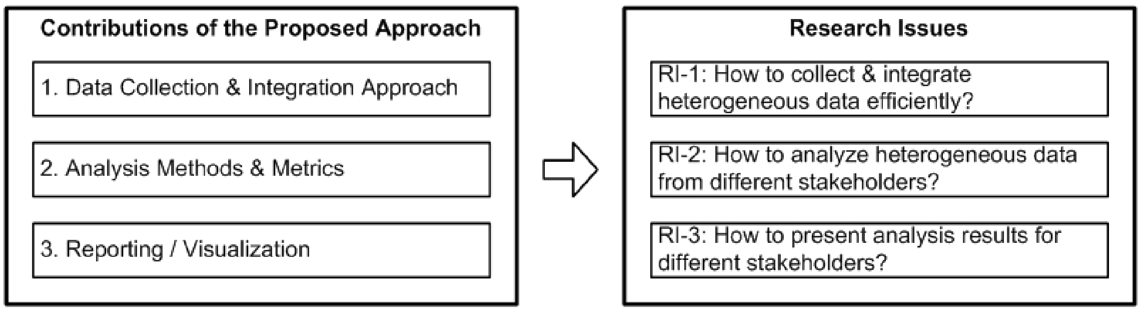

Contributions of the Proposed Approach
1. Data Collection & Integration Approach
2. Analysis Methods & Metrics
3. Reporting / Visualization
Research Issues
RI-1: How to collect & integrate heterogeneous data efficiently?
RI-2: How to analyze heterogeneous data from different stakeholders?
RI-3: How to present analysis results for different stakeholders?

Figure 6. Overview of the research issues

Figure 6 shows three major research issues, which were derived from three major contributions of the proposed approach. The first research issue deals with the question **how to collect and integrate heterogeneous data efficiently (RI-1)**, so that is useful for further analysis. Current data collection and integration is primarily manual, which is error-prone and takes a lot of time. We propose to develop and use automated tools to collect and integrate the data and compare it with the conventional approach to prove its efficiency.

The second research issue deals with the question **how to analyze heterogeneous data from different stakeholders (RI-2)**. The heterogeneity of data models, tools and formats used by different stakeholders makes it difficult to use uniform analysis methods. Some combination on analysis methods could be used to give different perspectives on the data to support stronger conclusion on status in the complex systems.

The third research issue deals with the question **how to present analysis results for different stakeholders (RI-3)**. The heterogeneity of analysis results from combined analysis methods used to analyze the data makes it difficult to support different goals of different types of stakeholders. The different types of stakeholders, e.g., developers, project managers, or project investors need specific analysis results in the form of reports that could be different from one type to another. That's why we come to a proposition to support different presentation types of analysis results to fulfill the different goals and requirements of each type of stakeholders. This kind of result presentation should be flexible, easy to maintain, easy to customize and understandable to support the stakeholders on monitoring the project and make a decision based on the project monitoring.

3.1.1 Heterogeneous Data Collection and Integration

In this thesis, we apply the Project Observation and Analysis Framework (POAF) to two application scenarios from two different application domains, namely Open Source Software Projects and Automation Systems Environments. The first research issue category deals with the heterogeneous data collection and integration for two application domains. As a precondition for this research issue, we needed to ensure that a) the application domain provides data in heterogeneous formats and data models, and b) the data can be accessed by tools, e.g., web-based tools or database.

RI-1.1. Feasibility of the proposed data collection and integration approach. In this research issue, we investigate the feasibility of the proposed approach, whether we can collect and integrate data from heterogeneous data sources by following the approach steps (collect, integrate, analyze, and present) by using supporting tools.

RI-1.2. Foundations for data collection and integration. In this research issue, we investigate the methods and tools as foundations for data collection and integration. For each application scenario, we propose different methods to collect and integrate heterogeneous data like using an Engineering Knowledge Base (EKB) (Moser, 2009) or Engineering Service Bus (EngSB) (Biffl & Schatten, 2009). Different tools can also be used to support the application of foundations to different application scenarios.

RI-1.3. Efficiency of data collection and integration approach. In this research issue, we compare the use of a traditional data collection process to a semantically-enabled data collection process to check the efficiency of proposed approach compared to the conventional one. Data collection is an important part in heterogeneous data collection and integration approach.

RI-1.4. Effectiveness of integration of additional data sources. In this research issue, we investigate the effectiveness of integration of additional data sources. In complex systems, adding new data sources to the current systems occurs often. The systems should be flexible enough to adapt with new data sources. The adaptation should be done quick and effectively to the whole systems.

3.1.2 Heterogeneous Data Analysis from Different Stakeholders

In this thesis, we apply several analysis methods to analyze data from different stakeholders in the heterogeneous software and systems environments. The use of different analysis methods in analyzing data is important to compare the strengths and the weaknesses of each analysis method, and whether further improvement to the current analysis methods could be applied to improve the quality of analysis process.

RI-2.1. Feasibility and validity of heterogeneous data analysis approaches. In this research issue, we investigate the feasibility of the different analysis approaches to heterogeneous data collected and integrated in complex systems. Our analysis approaches should not only be valid for a specific project but also valid to many projects in general. In the case of Open Source Software projects where the projects usually belong to certain project umbrella, we also check the validity of our approach to other project umbrellas.

RI-2.2. Integrated analysis approaches of data from heterogeneous sources. In this research issue, we investigate the strengths and weaknesses of integrated analysis approaches of data from heterogeneous sources compared to single analysis approaches. Integrated analysis approaches consist of several applications of different analysis methods that are applied to similar set of data. The results of integrated analysis approaches are expected to be more powerful than the result of the single analysis approach, because the integrated approaches offer different types of results overview that can be used by different types of stakeholders according to their goals and requirements.

RI-2.3. Validation of designed process model with actual engineering process data. This research issue involves the process model discovery of the engineering process data, validity/conformance checking between the designed process model and the actual process model generated from the actual data, and process performance/risk analysis including bottleneck analysis to the generated process model.

3.1.3 Analysis Result Presentation for Different Stakeholders

In this thesis, we apply several analysis result presentation approaches for different types of stakeholders in the heterogeneous software and systems environments. We define two types of stakeholders, namely project managers and engineers. Each type of stakeholder has different goals and requirements in getting the analysis results from previous approaches. For example, project managers focus on the results on the overall projects in the graph-mode, while engineers focus on the results on the specific parts of projects showing in the textual-mode.

RI-3.1. Feasibility of tool support for project monitoring and reporting. In this research issue, we investigate the feasibility of tool support for project monitoring and reporting. This approach is useful for different types of stakeholders, for example project managers and engineers, who want to monitor the project status in certain phase of project. In current complex systems, we cannot wait until the end of project to get the status of products being developed or services being delivered. The stakeholders should have abilities to monitor running projects to know the current status, not only at the end of the project. This approach could support the quality of the systems at run time.

RI-3.2. Usability and understandability of analysis results presentation. In this research issue, we investigate whether the analysis result presentations to different types of stakeholders are really useful, understandable and learnable. The project reporting/monitoring results that are not understandable by users will be useless and catastrophic. We want to check that the different types of stakeholders can use the project monitoring tools and result representation in an easy way to enhance their understanding on the process running, thus can improve the quality of project in advance.

3.2 Research Methods and Evaluation Concept

This section describes the research methods, the evaluation concept, and evaluation criteria used in this thesis.

3.2.1 Research Methods

The research on observing and analyzing heterogeneous software and systems development environments is done by following these steps.

Step 1: Systematic literature review. We conduct a systematic literature review (Brereton, Kitchenham, Budgen, Turner, & Khalil, 2007) for reviewing related literature on open source software project monitoring, engineering process observation, and workflow validation. In a systematic literature review, we focus on aggregating empirical evidence from widely differing contexts by using a variety of techniques to achieve certain goals, e.g., to find out the state-of-the-art of methods used in certain topics or to find out the open issues in certain topics.

A systematic literature review is defined as “a form of secondary study that uses a well-defined methodology to identify, analyze, and interpret all available evidence related to a specific research question in a way that is unbiased and (to a degree) repeatable" (Kitchenham, 2007).

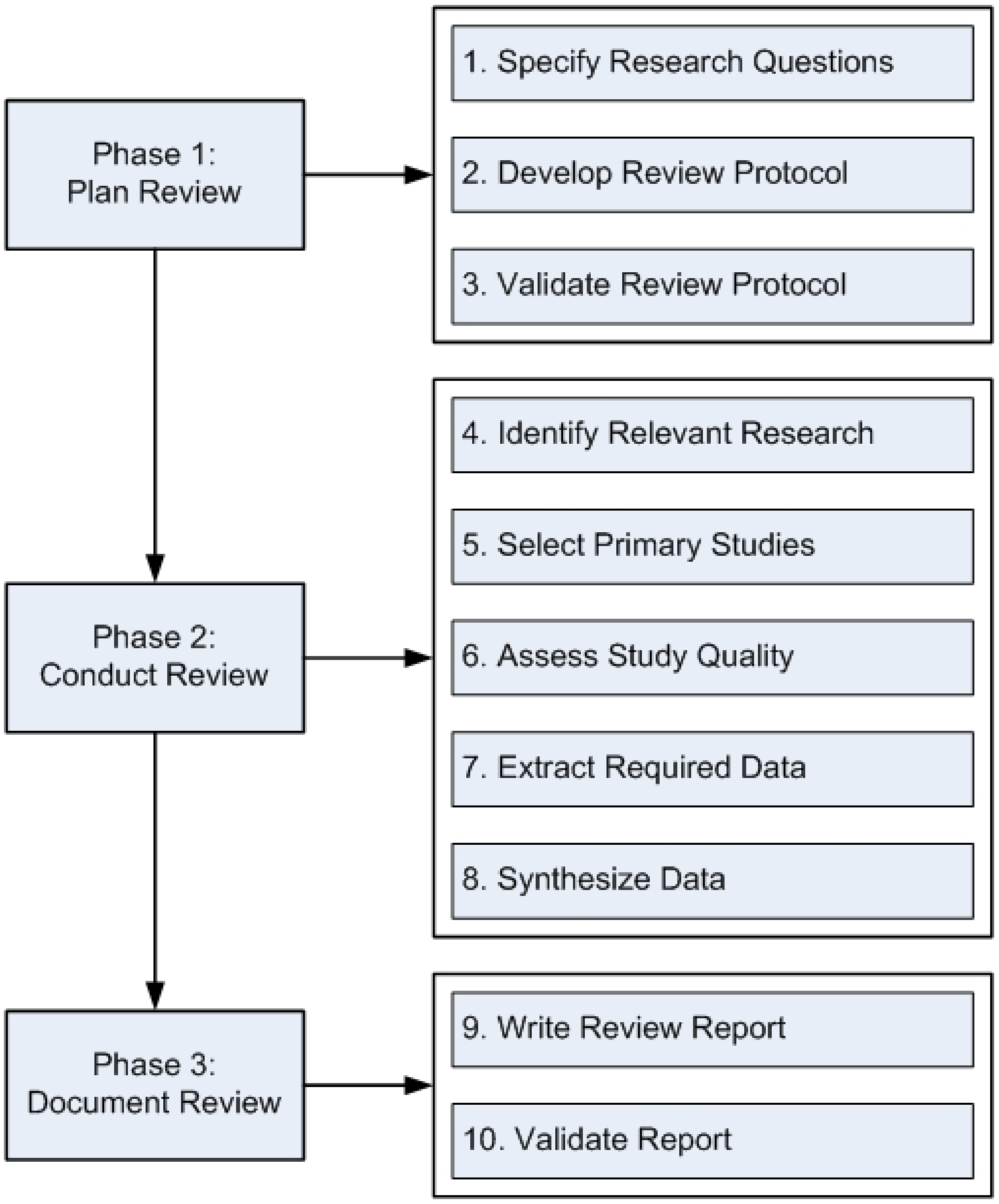


Figure 7. Systematic literature review process (Brereton et al., 2007)

The characteristics of systematic literature review can be summarized as follows: (a) a systematic review protocol defined in advance of conducting the review, (b) a documented search strategy, (c) explicit inclusion and exclusion criteria to select relevant studies from the search results, (d) quality assessment mechanisms to evaluate each study, (e) review and cross-checking processes to control researcher bias.

Conducting a systematic literature review involves three main phases of discrete activities, namely planning, conducting and documenting the review. Figure 7 shows the 10-stage review process in three main phases, namely (1) specify research questions, (2) develop review protocol, (3) validate review protocol, (4) identify relevant research, (5) select primary studies, (6) assess study quality, (7) extract required data, (8) synthesize data, (9) write review report, (10) validate report.

Step 2: Definition of data modelling. For modelling, we use the standard Unified Modeling Language (UML) for understandability reason. UML has been the industry standard for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. UML also facilitates communication and reduces confusion among project stakeholders as de facto standard modelling language (Booch, Rumbaugh, & Jacobson, 2005).

We use the entity-relationship (ER) model (Chen, 1976; Silberschatz, Korth, & Sudarshan, 2010) for semantic modelling (Hull & King, 1987). This model (Thalheim, 2000) incorporates some of the important semantic information about the real world. Semantic modelling provides mechanisms for representing structurally complex interrelations among data typically arising in commercial applications. In general terms, semantic modelling complements work on knowledge representation and on database models based on the object-oriented paradigm of programming languages.

Step 3: Prototypes building. For feasibility evaluation, we implement prototypes as proof-of-concept of our conceptual approaches (Floyd, 1984). The term prototype in relationship with software development indicates a primary interest in a process rather than in the prototype as a product. The goal of the prototyping process is the identification of processes which involve an early practical demonstration of relevant parts of the desired software, and which are able to be combined with other processes in system development with a view to improve the quality of the target systems. Many software developers are motivated to employ prototyping by important conclusions drawn from their working experience.

Step 4: Empirical Study. For performance evaluation, we follow the guidelines for empirical research in software engineering (Kitchenham et al., 2002). The guidelines are intended to assist researchers, reviewers, and meta-analyst in designing, conducting, and evaluating empirical studies.

Step 5: Statistical Analysis. For statistical evaluation, we use descriptive statistics as well a statistical tests. Descriptive statistics deal with the presentation and numerical processing of a data set. The goal of descriptive statistics is to get a feeling for how the data set is distributed. Statistical tests are available to test the experiment hypotheses. Hypothesis tests can be classified into parametric tests and non-parametric tests. Parametric tests care based on a statistical model that involves a specific distribution. Non-parametric tests do not make the same type of assumptions concerning the distribution of parameters as parametric test do (Freimut, Punter, Biffl, & Ciolkowski, 2001). In this research, we use parametric tests to test our hypotheses on application context.

3.2.2 Evaluation Concept

This section discussed the plan for validating our proposed framework to observe workflows in the multidisciplinary engineering environments. We classify the plan based on the validation criteria as follows.

(1) Feasibility. The first criterion is the feasibility of the proposed approach. We made a pilot application by following the framework, to collect, integrate, and analyze the workflow data. The input to this criterion is heterogeneous data representations from our industrial partner. The process steps to evaluate the feasibility criterion are as follows, (1) learning the workflow representation, (2) learning the workflow data collection and integration, (3) learning the proposed framework, (4) making a pilot application based on the proposed framework.

The output of these processes is a pilot application to collect and integrate the workflow data for managing several workflows from different engineering fields. An example of the pilot application is management of signal change workflows from mechanical, electrical, and software engineering fields for our industrial partner in a power plant system (Sunindyo, Moser, Winkler, Mordinyi, & Biffl, 2011).

(2) Effectiveness. Effectiveness means the capability to reach a defined goal. In this case, we aim at observing and validating the role of the workflow usage in project management. The measurement of the efforts to reach the goal is useful to justify the possible modification in the project progress monitoring.

The input to the measurement of this criterion is a workflow model and heterogeneous data from our industry partner. The processes are including (1) capturing the event data from different engineering fields, (2) measuring the process metrics, and (3) deriving the product metrics from process metrics. We expected product and process metrics as our output for this criterion. An example of effectiveness measurement is by using signal change management in the power plant to produce product and process metrics from a workflow model (Winkler, Moser, Mordinyi, Sunindyo, & Biffl, 2011).

(3) Efficiency. Efficiency means unit of output per unit of input. We compare efficiencies between a primarily manual approach and the proposed automated approach following the proposed framework.

The inputs to this criterion measurement are the measured efforts using the manual approach and the automated approach. We expect to have the percentage of the efficiency from the comparison as our output. The processes are as follows, (1) measuring the efforts as input and the results as output, (2) comparing the efforts and the results between manual approach and automated approach, and (3) calculating the efficiency.

3.3 Application Scenarios

The project analysis and observation framework is applied to two different applications domain, namely the Open Source Software Projects domain and the automation systems engineering domain to show the ability of the framework to adapt to changed domain-specific requirements. We describe shortly the two application scenarios and their special characteristics in the following two subsections.

3.3.1 Open Source Software Projects

Open Source Software (OSS) development projects use different data management techniques for managing heterogeneous data from different data sources, e.g., Source Code Management (SCM), developer's mailing list, and bug repository, which produces a new insight on the software development (Biffl, Sunindyo, & Moser, 2010b). A set of different data management approaches, such as data warehousing (Biffl et al., 2010a) or data mining techniques (Gegick, Rotella, & Tao, 2010), have successfully been applied to observe the processes of OSS projects and to improve the quality of products and processes of these OSS projects (Biffl et al., 2010b). Data warehousing and data mining techniques enable the observation of OSS projects processes based on measuring differences of expected requirements and the actual implementation. Measurement results can be used for improving the techniques itself as well as the underlying methods for developing OSS.

OSS project managers need to collect, integrate and analyze heterogeneous data originating from different tools used in the OSS project, such as source code management, developer's mailing list, and bug reporting tools to observe the processes and determine the status of OSS projects (Biffl et al., 2010b). Typically, project management in OSS projects is based on dynamically changing conventions between developers or contributors - usually it is performed either by senior contributors or the project initiator - while project management in conventional software project is determined prior to the actual project life time.

Observations of OSS processes are needed as an initial way to improve the quality of OSS processes and products. By observing the OSS processes, we can measure the current state of certain processes, for example the times taken to report and resolve issues, and then find out how to reduce idle or non-productive time windows or to address bottlenecks. In OSS projects, project managers can not directly inspect the software development, since developers usually work geographically distributed including interactions through communication tools, such as SVN, mailing list, or Bugzilla (Biffl et al., 2010a). Another aspect of OSS project observation focuses on the structure and culture of OSS models. The structure of OSS projects typically is more democratic compared to conventional software development projects and consists of more flexible work structures as well as of global or multi-cultural communities (Sharma, Sugumaran, & Rajagopalan, 2002). A promising approach of project managers to observe OSS engineering processes is by analyzing data generated during the development phases, e.g., bug data, developers' communication data, and source code management data. However, the analysis process itself is not straightforward. Some preparatory steps are required to get the data ready for analysis, for example data collection, data integration, and data validation. Since the data typically originates from more than one, often heterogeneous, tool, project managers need to identify the relationships between heterogeneous data sources to get meaningful patterns out of the collected data for further decisions regarding the OSS project.

Status determination with respect to the project timeline and prediction of project survivability based on "health indicators" (Wahyudin, Schatten, Mustofa, Biffl, & Tjoa, 2006) of OSS projects is a key activity of project managers (Wahyudin, Mustofa, Schatten, Biffl, & Tjoa, 2007; Wahyudin et al., 2006). By knowing the status of OSS projects early in the development project and phases, project managers and other stakeholders, e.g., project hosts and project contributors, can decide whether the project is healthy, sustainable and worth-supported. Among key indicators to determine the OSS project health status is a proportional number of code contributions per developer email and the number of bug status report per developer email. In "unhealthy" OSS projects, the ratio between code contributions per developer email and bug status report per developer email is imbalanced (Wahyudin et al., 2007).

A more specific part of OSS projects in this thesis is the Continuous Integration and Test (CI&T) use case (Biffl & Schatten, 2009). The CI&T use case illustrates a key part in an iterative software development process: if part of a system or engineering model gets changed, the system has to be re-built and tested in order to identify defects early and to provide fast feedback on implementation progress to the project manager and the owners of the changed system parts.

The implementation of CI&T use case in OSS context is done in Open Continuous Integration and Test (OpenCIT[18]) server. The OpenCIT server can be used to automatically build, test and deploy projects. It makes it easier for developers to integrate changes and easier for users to obtain fresh builds of a project. The Open CIT tries to provide additional benefits compared to other CI&T solutions by building upon the open and highly customizable OpenEngSB[19] platform. Furthermore the OpenCIT itself is also a FOSS (Free and Open Source Software) project, which provides its users with the possibility to adapt all aspects of the CI&T process. The domain and connector concept of the OpenEngSB make it possible to define the OpenCIT workflow completely independent of specific tools. Therefore any build, test and deploy tool can be used together with the OpenCIT.

3.3.2 Automation Systems Engineering

Automation Systems Engineering (ASE), e.g., the engineering of production automation systems or power plants, includes a wide range of heterogeneous disciplines, such as mechanical engineering (e.g., physical layout), electrical engineering (e.g., circuit diagrams), and software engineering (e.g., UML diagrams, function plans, and software code) (Biffl & Schatten, 2009). Normally, different disciplines apply individual engineering processes, methods, and tools with specific data models, addressing individual needs of involved engineers.

Observations at our industry partner – a hydro power plant systems integrator – showed that traditional ASE processes follow a basic sequential process structure with distributed parallel activities in specific phases and suffer from a lack of systematic feedback to earlier steps, inefficient change management and synchronization mechanisms of disciplines, and low engineering process automation across domain boundaries (Moser & Biffl, 2010). Specific tools and data models typically address only the needs of one individual discipline and hinder efficient collaboration and interaction between disciplines. Because of this lack of collaboration, change management becomes even more difficult, leading to development delays and risks for system operation.

[18] http://opencit.openengsb.org/about.html
[19] http://www.openengsb.org/

For instance, changing hardware components (e.g., hardware sensors) might require changes in software components (e.g., caused by changed value ranges, data types (e.g., analogue / digital sensor), or number of connection points). Other cases of automation systems engineering are production automation systems. The industrial production automation systems typically involve manufacturing systems, such as assembly workshops that combine smaller parts into more complex products, e.g., cars or furniture. Several domains have to cooperate for manufacturing: (1) business order processing and work order scheduling, (2) technical processes for workshop and systems coordination, and (3) technical designs of a set of machines in a defined workshop layout (Lüder, Peschke, & Reinelt, 2006).

In typical engineering disciplines, models (e.g., model-based design and testing (Baker, Zhen, Gabrowksi, & Oystein, 2008)) help to construct new systems products and to verify and validate the solutions regarding the requirements, specification, and design models. Traditional systems engineering processes follow a water-fall like engineering process with late testing approaches. Unfortunately, insufficient attention is paid in the field of automated systems engineering to capabilities for Quality Assurance (QA) of software-relevant artifacts and change management across engineering domains, possibly due to technical and semantic gaps in the engineering team. Thus, there is considerably higher effort for testing and repair, if defects get identified late in the engineering process.

4 Project Observation and Analysis Framework

This chapter summarizes the proposed Project Observation and Analysis Framework (POAF). In the first section, an overview of the framework is given, as well as an explanation of preconditions and the used technologies for the application of the POAF. The first section also summarizes the challenges of the POAF and tries to classify the approach regarding related approaches. The second section presents some scenarios that could benefit from using the POAF. The third section explains the generic architecture of the POAF as well as the two major phases of the process of using the POAF.

4.1 Overview

Heterogeneous software and systems development environments introduce different kind of tools to produce engineering process data that is useful for project monitoring and reporting. The result of the project monitoring and reporting then can be analyzed to improve the quality of the products.

However, a major challenge in current project monitoring and reporting approaches is insufficient observation and analysis across different stakeholders (Moser, Mordinyi, et al., 2011)(Sunindyo, Moser, Winkler, et al., 2012). Different and partly overlapping observation and analysis requirements from different stakeholders make the project monitoring and reporting inefficient. Consequently, the weak tool support for project monitoring for project managers and engineers hinders quality management and flexible engineering process automation, leading to late project status acquisition and risks for system operation.

The strategic goal of making project monitoring process more flexible without delivering significantly more risky end products translates into the capability to efficiently observe and analyze the engineering process of a project environment. While there are approaches based on personal software processes (Humphrey, 1996, 2000a) and team software processes (Humphrey, 2000b; Humphrey, Chick, Nichols, & Pomeroy-Huff, 2010), experience has shown that such engineering process tends to be different compared to the designed model (Sunindyo, Moser, Dhungana, Winkler, & Biffl, 2012) and that different stakeholders need more flexibility to define their own workflows to do their tasks (Sunindyo et al., 2011). Thus a key goal is to allow the stakeholders to analyze engineering process data using different methods and tools based on their requirements and to provide mechanisms to integrate data sources for those different data analysis. In the past, several approaches for providing observation and analysis framework for engineering process data have been investigated (Gousios & Spinellis, 2009a, 2009b; Hu & Zhao, 2008). However, these approaches focus primarily on analyzing engineering process data for specific stakeholders rather than providing support for observing and analyzing engineering process data from various data sources for heterogeneous stakeholders, which is the main focus of this thesis.

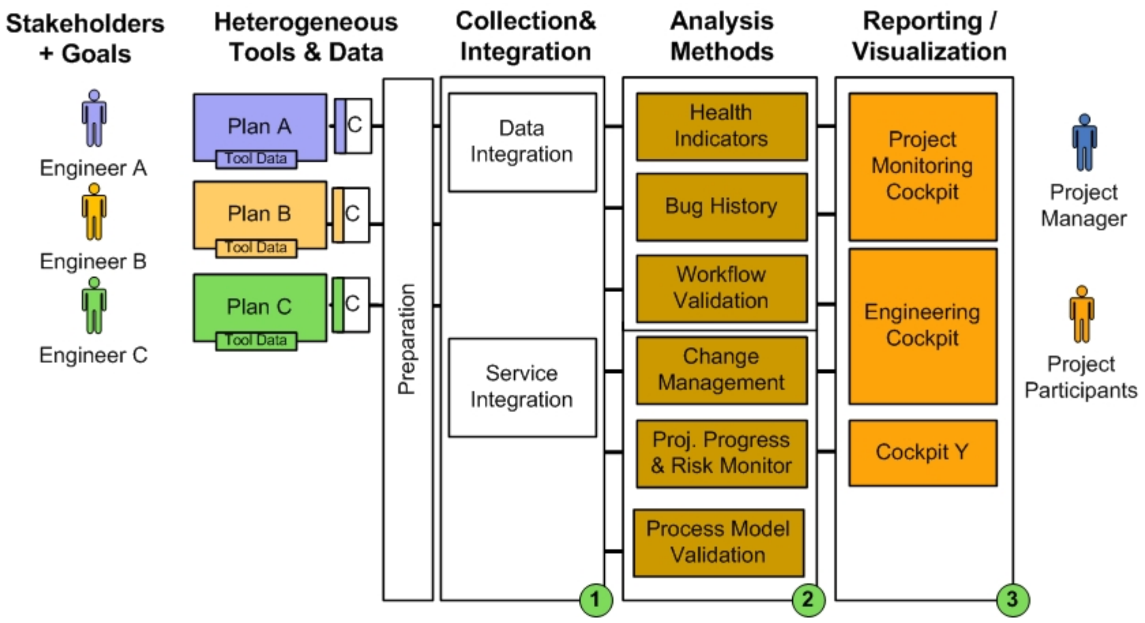


Figure 8. Overview of Project Observation and Analysis Approach (Sunindyo, 2011)

In this thesis, we introduce a generic approach for project monitoring and reporting in software and systems engineering (see Figure 8) with a focus on providing support between heterogeneous tools and data sources to support the analysis and visualization for different stakeholders and thus making project reporting more efficient and flexible. Our approach is the so called Project Observation and Analysis (POA) framework, a project monitoring and reporting approach which supports heterogeneous engineering process data collection and integration for observation and analysis in the heterogeneous software and systems development environments. Therefore, we can automate the data collection and integration processes that build the integrated view of engineering process data for observation and analysis needs. The POAF collects, integrates and stores the engineering process data in ontologies and provides integrated data for flexible analysis methods and reporting/visualization approaches to different stakeholders. The POAF aims at making data collection, integration, analysis and reporting more efficient.

4.2 Architecture of the Project Observation and Analysis Framework

This section introduces the Project Observation and Analysis Framework (POAF), describes the POAF architecture and pictures the process for establishing and using the POAF. For understandability reasons, an example from the open source software project domain is chosen. We describe the architecture of POAF, as shown in the Figure 8. The general mechanism of the POAF uses heterogeneous goals and analysis requirements from stakeholders as basis for engineering process data collection, integration, analysis and presentation for the project manager and participants (Sunindyo, 2011). In the following, the detail architecture of the POAF is described; the numbers directly refer to the numbered tags in Figure 8.

4.2.1 Preparation

The preparation part consists of intermediary steps should be taken before data collected and integrated. The content of the preparation part depends on the engineering domain and treatment to adapt the engineering process data for further steps.

4.2.2 Collection and Integration (1)

As first step, heterogeneous engineering process data produced by different tools are collected and then integrated by using some integration mechanisms, e.g., data integration or service integration. The data integration mechanism, like Engineering Knowledge Base (EKB) (Moser, 2009; Moser et al., 2010) or ontology approach (Doan et al., 2004) integrate heterogeneous data semantically. It means that interrelated information from the different data models and tools are integrated based on their meaning. The service integration mechanisms, like Enterprise Service Bus (ESB) (Chappell, 2004)(Yin, Chen, Deng, Wu, & Pu, 2009) or Engineering Service Bus (EngSB) (Biffl & Schatten, 2009; Biffl, Schatten, et al., 2009) provide a platform for integrating heterogeneous services from different systems and software engineering environments.

4.2.3 Analysis Methods (2)

The collected and integrated engineering process data are processed and analyzed using different analysis methods, e.g., health indicators (Wahyudin et al., 2007, 2006), bug history (Sunindyo, Moser, Dhungana, et al., 2012), workflow validation (Sunindyo et al., 2011), change management (Winkler et al., 2011), project progress and risk monitoring (Sunindyo, Moser, Winkler, & Mordinyi, 2013), and process model validation (Sunindyo, Moser, & Winkler, 2012). The selection of the methods used to analyze engineering process data is based on (a) the context of the domain, (b) the goals from the stakeholders, and (c) the requirements from the stakeholders. Further explanation for each analysis method will be in the chapter 5 and 6.

4.2.4 Reporting / Visualization (3)

The reporting/visualization part consists of several approaches and tools to present the analysis results that can be useful for the project manager or other project participants. The reporting/visualization could be in the form of project monitoring cockpit (Biffl et al., 2010b), engineering cockpit (Moser, Mordinyi, et al., 2011), or other interfaces to connect to other application like Decision Support Systems or Executive Information Systems.

4.3 Preparation of the Engineering Process Observation and Analysis

This section describes the process for the preparation of the engineering process observation and analysis using the POAF in more details. Figure 9 shows the overview of those preparation processes. The following sections describe each process step in details.

4.3.1 Define process model

As a first step for the preparation of the engineering process observation and analysis, a process model is defined to identify intended running process in the heterogeneous engineering environments. The input of this process comes from the requirements for process definition that can be easily obtained from the stakeholders.

4.3.2 Transform to formal process model

The process model obtained in the first step then is formalized using some formal notation, for example BPMN[20]. The formalization of process model is useful to make the model machine-readable by the rule engine and event generator.

[20] http://www.bpmn.org/

4.3.3 Evaluate formal process model

The formal process model obtained in the previous step is evaluated with the requirements from the stakeholders to check whether all items have been fulfilled in the formal process model. The evaluation is also useful for checking the formal process model with the notation standard.

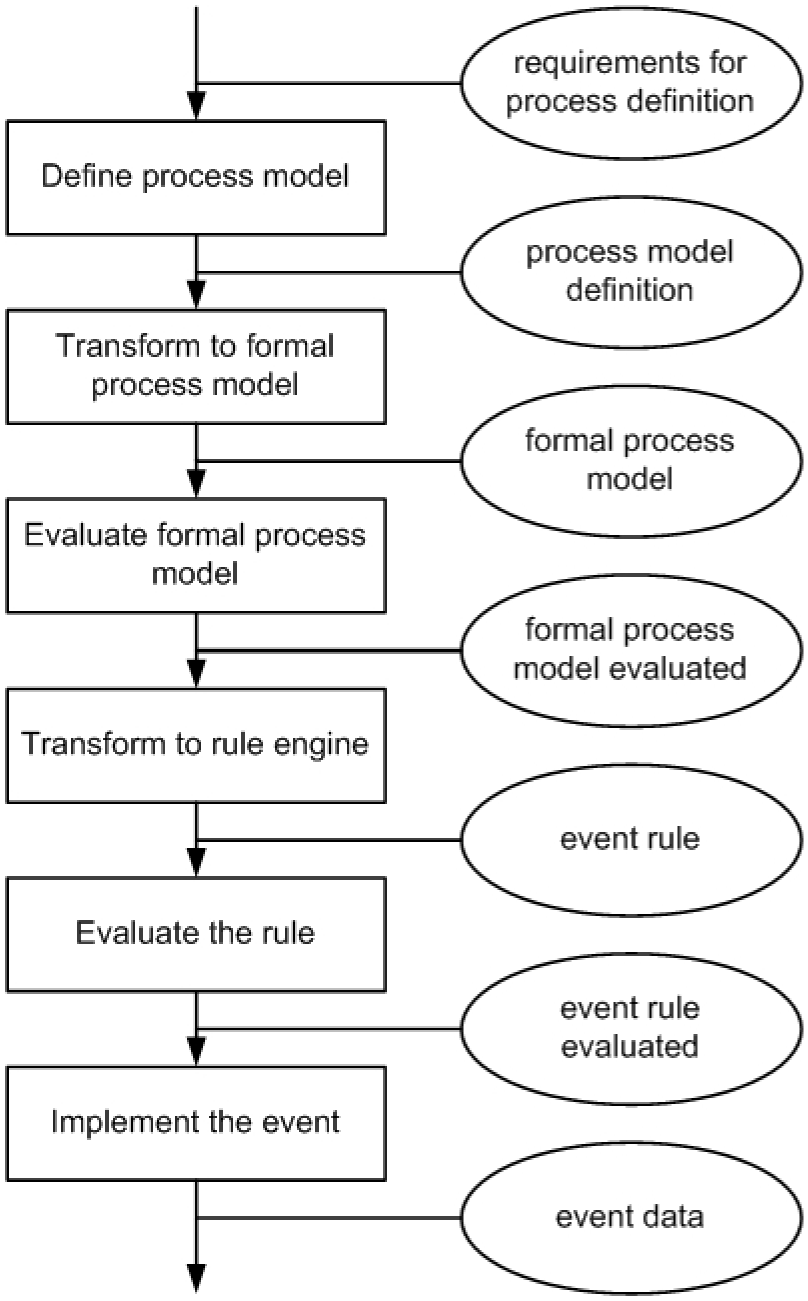


Figure 9. Preparation of the Engineering Process Observation and Analysis

4.3.4 Transform to rule engine

The rule is useful to set what kind of event we want to produce. The transformation of the process to the rule engine is following the formal process model obtained in previous step.

4.3.5 Evaluate the rule

The rule generated from the rule engine then is evaluated to check whether the rule is really following the formal process model, before the event is generated using event generator.

4.3.6 Implement the event

The event then is generated using an event generator, following the rule defined before. We can observe and analyze the generated event for checking the conformance between the designed process model and the actual process in the running systems.

4.4 Use of the Engineering Process Observation and Analysis

This section describes the process for the use of the engineering process observation and analysis using the POAF in more details. Figure 10 gives an overview of the process for the use of engineering process observation analysis. The following sections describe each process step in details.

4.4.1 Collect Events

The event data generated from heterogeneous systems and software development environments are collected for further steps, e.g., observation, integration, and analysis. The information in the event data show the type of event, the time when the event takes place, the source of event, and other related information that could be useful for analysis.

4.4.2 Observe Events

The event observation is useful to filter the completed event sequences from the uncompleted event sequences. Only completed event sequences are useful for analysis purpose, because they can show the behaviour of the events from the beginning until the end.

4.4.3 Integrate Events

The observed events then are integrated to support engineering process data analysis. By using integrated event data, it is much easier for the project managers to analyze the data because they don't have to deal with heterogeneous data models and tools that are usually appear prior to data analysis process.

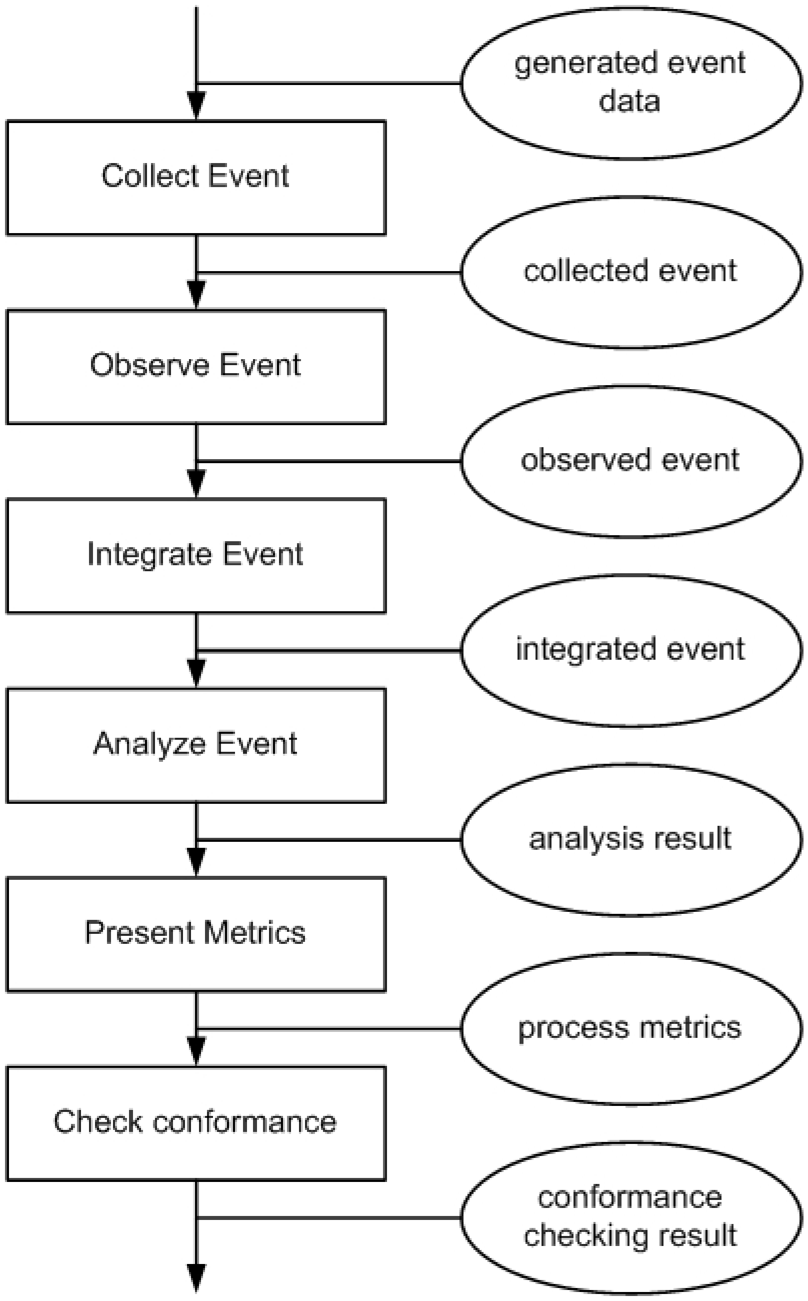


Figure 10. Use of the Engineering Process Observation and Analysis

4.4.4 Analyze Events Data

The next process is to analyze integrated events data. In this phase, the project managers or other stakeholders can analyze integrated events data for their purposes. The analysis methods used are based on their needs, for example by using descriptive statistics method to show the tendency of data (mean, mode, median) (Wohlin et al., 2000) or by using statistical tests to test the experiment hypotheses (Juristo & Moreno, 2001).

4.4.5 Present Metrics

The results of events data analysis are shown in the form of metrics. The usage of metrics to show the results of event data analysis makes it easier for the project managers and other stakeholder to read the analysis results. One of methods to produce metrics from analysis process is Goal Question Metrics (GQM) method (Basili, Caldiera, & Rombach, 1994; Van Solingen & Berghout, 1999).

4.4.6 Check Conformance

Conformance checking means comparison between existing process model and an event log of the same process (A Rozinat & van der Aalst, 2008; Anne Rozinat & van der Aalst, 2006). The existing process model we have designed in prior steps, while the event log is generated during the project runs. By performing conformance checks, we want to find out the similarities and deviation between the reality and the designed process model. By using conformance checking, we can detect the problems of the project, e.g., a bottleneck of the project, so we can suggest for process improvement.

5 Open Source Software Projects Monitoring

This chapter summarizes the results of applying the POAF to an application scenario from the Open Source Software projects domain, as described in section 3.3.1.

5.1 Overview

Current Open Source Software (OSS) projects involve a range of stakeholders, from core developers and co-developers to potential users and project investors. Typically, stakeholders, such as potential users or project investors need to know the status and the likely future performance of the project to determine whether the project is likely to sustain for a reasonable period of time in order to justify their investments into the project.

Recent research on using project data to support OSS health monitoring to provide immediate OSS project status, e.g., *Sourcerer* (Linstead, Bajracharya, Ngo, Rigor, Lopes & Baldi, 2009), focus on analyzing author-topic relationships in different OSS artifacts to increase understanding of the project and to raise the awareness on the health status of a project. Gall *et al.* (Gall, Fluri & Pinzger, 2009) introduced the *Evolizer* approach to analyze the software evolution of OSS projects within Eclipse. This analysis is useful to investigate the current stage of OSS to be adapted continuously to changing environments, business reorientation, or modernization. Recent research on OSS project status monitoring includes participation aspects (Choi, Chengalur-Smith & Whitmore, 2010), productivity aspects (Wahyudin, Dindin & Tjoa, 2007), communication aspects (Biffl, Sunindyo & Moser, 2010a), and community aspects (Kaltenecker, 2010). The research presented in this thesis is based on the concept of project health indicators, which has been introduced by Wahyudin *et al.* for monitoring the health status of OSS projects during development (Wahyudin, D., Schatten, Mustofa, Biffl & Tjoa, 2006). Example indicators that can be used by experts to assess an OSS project are: (a) service delays on open issues – the time it takes to fix bugs and issues listed in the project bug reporting system; (b) proportions of activity metrics in the community, e.g., the volume of mailing list postings, bug status changes per times slot, and updates in the SVN to learn the health of relationships between relevant activities, e.g., activities on the same bug; and (c) communication and use intensity. In a healthy project community, a reasonable relationship can be expected between measures such as the number of downloads, mailing list postings, and developer interactions in mailing lists (Wahyudin, D., Mustofa, Schatten, Biffl & Tjoa, 2007).

However, challenges for monitoring the health status of OSS projects easily and frequently are: (a) manual data collection and integration from heterogeneous data sources, i.e., data sources, which represent common project-level concepts in various data formats that are non-trivial to reconcile, tend to be prone to errors and take considerable effort to integrate (Conklin, 2006); (b) the need to correlate data on different activities requires data integration; (c) manual data validation of the integrated data is hard due to different representation of common concepts, e.g., different names for one person in the data models involved; (d) data analyses of individual data sources, e.g., mailing lists, bug database (Mockus, Fielding & Herbsleb, 2002), SVN/CVS (German 2004), and change logs (Chen, Schach, Yu, Offutt & Heller, 2004) have been shown to be weak to detect the health status of OSS project accurately; and (e) the large amount of data to maintain for analysis in an OSS project over time takes significant resources for storing.

5.2 Framework for OSS Data Analysis

The framework for OSS data analysis (FOSSDA) is an instantiation of Project Observation and Analysis Framework (POAF) in chapter 4 for OSS project context. The proposed FOSSDA consists of four layers: (a) Layer 1: data sources, (b) Layer 2: core framework, (c) Layer 3: key performance indicators, and (d) Layer 4: presentation layer. This framework is built based on an ontology-based knowledge representation, mapping, and reasoning and uses the Project Data Fetcher tool for data collection. Major novel contributions of this framework are (a) an ontology to store the structure of integrated data collection results, (b) improvement of tools for collecting data, and (c) improvement of tools to analyze the collected and integrated data. Figure 11 illustrates the framework for analyzing OSS engineering process data using a combination of existing analysis approaches. The description of each layer is as follows.

Layer 1: Data Sources. OSS projects generate a wide range of data during development and as product application. These data can be classified as process metrics or product metrics. Process metrics can be derived from development tools that are used by the developers or other stakeholders during the development process, e.g., source code management systems, bug reporting systems, or mailing list. Product metrics can be derived from the final product, e.g., by counting the number of lines of code, the number of modules, the coupling and cohesion metrics between modules. In this layer, we provide the basis for collecting data from several types of OSS projects artifacts for further use in the next steps. Examples of common data sources for OSS process analysis and improvement are: source code management systems, bug reporting systems, and mailing lists.

Layer 2: Core Framework. The OSS data sources have heterogeneous data formats and models that impede the analysis process. Therefore, the data have to be integrated before storing and analyzing them. We applied an ontology as promising approach for data storage. The ontology consists of three heterogeneous data models (see Figure 14) derived from the tools used during software development process (i.e., versioning system, mailing list, and issue tracker). The data models of these tools contain elements which are only used in the context of the specific tool, as well as elements which are also used in context of other tools. In order to integrate the data models respectively tools, so called "common concepts" (see Figure 14 for project/process level) has to be introduced to link heterogeneous data models and tools. As a next step, the concepts of local tool data models has to be mapped to the common data model concept (mapping from data source level to project/process level in Figure 14) (Biffl, Sunindyo & Moser, 2010a; Moser, 2009).

We apply an ontology-based tool called *Project Data Fetcher* (see Figure 11) (Biffl, Sunindyo & Moser, 2010b) to collect data from heterogeneous tools and store the results in an ontology. The application of an ontology allows (a) checking the data and (b) linking connected data (based on identified common concepts) from different sources, e.g., identifying that several author names in mailing list postings actually belong to the same persons who committed code to the SCM. After collecting and integrating the data, we check the validity of the data and their connections by reasoning to the ontology. The integrated and validated data are the foundation for good analysis results in the next layer. Note that the Project Data Fetcher allows data validation by using an ontology for data consistency checking.

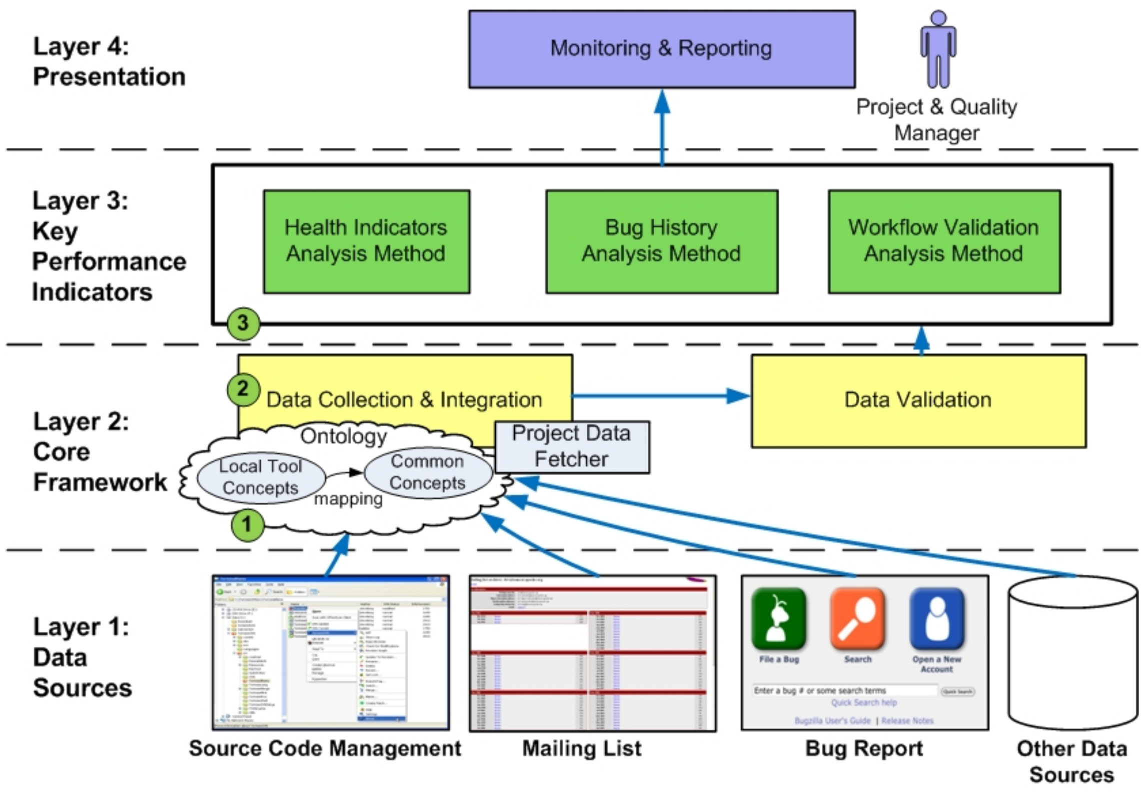


Figure 11. An Analysis Framework to Support OSS Data Analysis (Sunindyo, Moser, Winkler, et al., 2012)

Layer 3: Key Performance Indicators. This layer provides approaches to derive key performance indicators based on the integrated and validated data. The key performance indicators are defined and applied in FOSSDA by analyzing the OSS project ontology as a knowledge representation of heterogeneous data sources. The related data sources are identified and then calculated to produce process metrics for key performance indicators. We use several OSS key performance indicators, namely health indicators (Wahyudin et al., 2007), bug history (Sunindyo, Moser, Dhungana, et al., 2012) and workflow validation (Sunindyo, Moser, Winkler, & Biffl, 2010a).

Layer 4: Presentation Layer. This layer contains monitoring and reporting tools to provide OSS health indicators to project and quality managers. The information is provided in a textual or graphical format that is most suitable for project and quality managers to take actions based on the health indicators. One example of such tools is the Project Monitoring Cockpit (ProMonCo) tool (Biffl, Sunindyo & Moser, 2010a) which can be connected to the Project Data Fetcher and provide analysis results as sample implementation of the FOSSDA presentation layer.

5.3 Integrated Data Model

This section explains the simple and advanced examples of the integrated data model to answer the heterogeneous data collection and integration research issues (see section 3.1.1) in OSS domain.

5.3.1 Simple Example

In this section, we want to show the simple example of integrated data model from two heterogeneous data sources, namely SVN and mailing list. SVN[21] is a software versioning and revision control system distributed under an open source license. Developers use Subversion (SVN) to maintain current and historical versions of files such as source code, web pages, and documentation. A mailing list is a collection of names and addresses used by an individual or an organization to send material to multiple recipients. In OSS projects, the developers use mailing lists for communication and distribution of the materials related with the projects.

Figure 12 left-hand side shows the data model of SVN (upper side) and the data model of mailing list (lower side). From this figure, we can see the heterogeneities of both data models. We can identify four types of heterogeneities, namely (a) different data model, (b) different data entities, e.g., person, developer, (c) different data types, e.g., string, number, and (d) different data representation, e.g., time representation, with date or without date, with timezone or without timezone.

[21] http://subversion.apache.org/

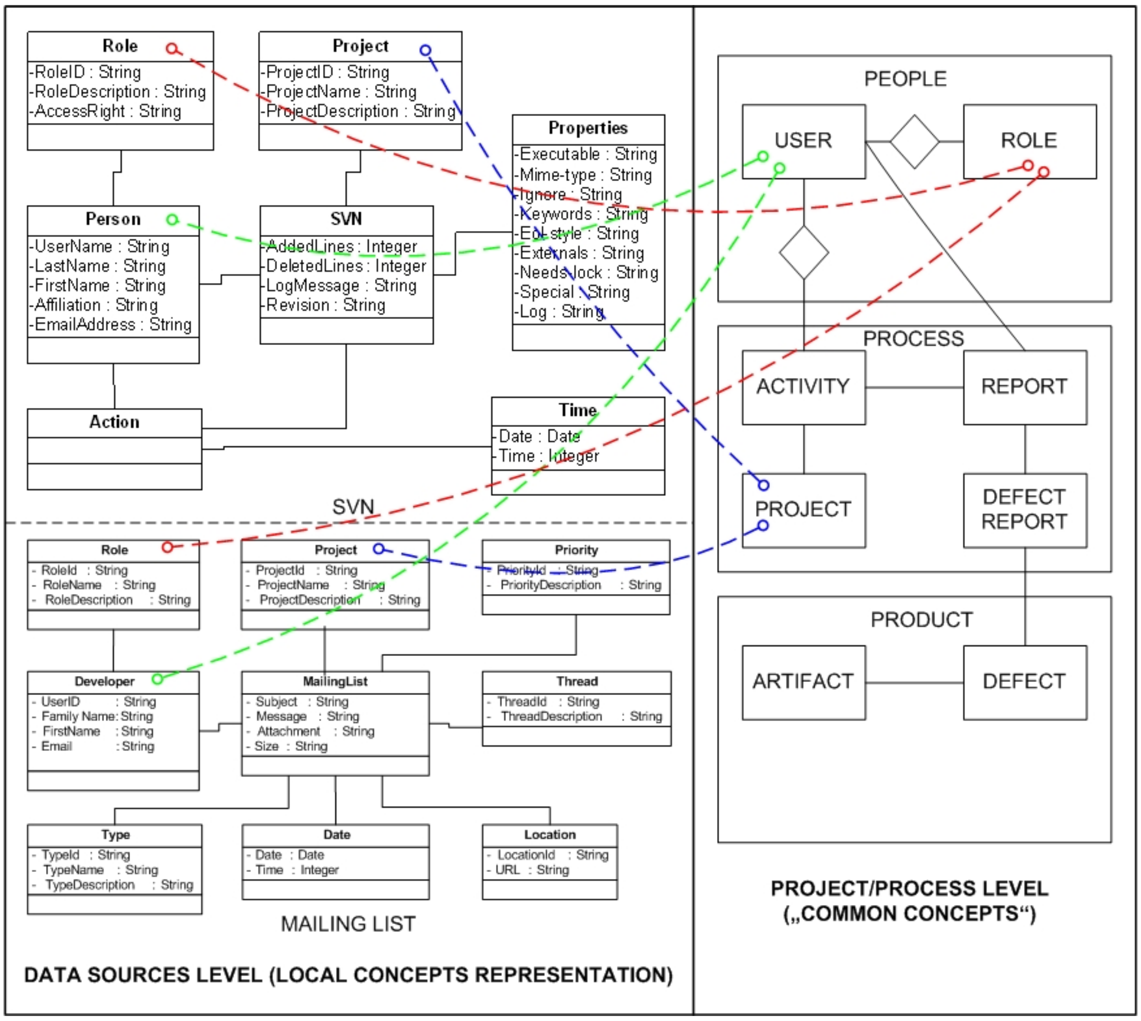


Figure 12. Integrated Data Model for SVN and Mailing List (Sunindyo, Moser, Winkler, et al., 2012)

Figure 13 shows how the ontology area of Engineering Knowledge Base (Biffl, Sunindyo, et al., 2009; Moser, 2009) can provide integration approach for bridging heterogeneity from SVN and mailing list data model. The SVN has the local terminology "Person", while the mailing list has the local terminology "Developer". Both share the common concept "User" in the Ontology Areas. Then, both terminologies will be mapped to the class "User" as mentioned in Listing 1.

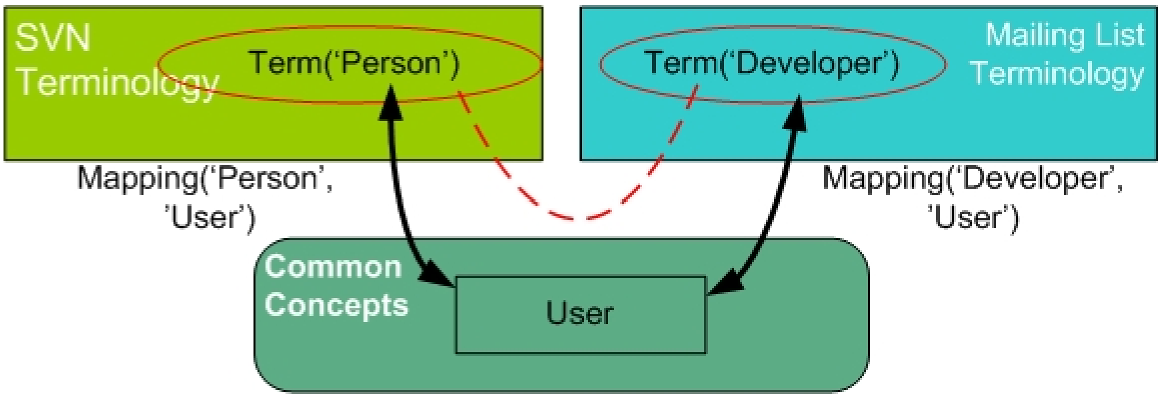


Figure 13. Translation from different data sources to the common concepts (adaptation from (Biffl, Sunindyo, et al., 2009))

Listing 1. Mapping terminologies to common concepts (Biffl, Sunindyo, et al., 2009)

```
mapping('Person','User').
mapping('Developer','User').
```

From the mappings above, we can have a translation between two local terminologies by using a rule, e.g., the rule described in Listing 2. The query and result can be seen in Listing 3.

Listing 2. Simple translation rules (Biffl, Sunindyo, et al., 2009)

```
translate(Term1,Term2) :-
 mapping(Term1,CommonConcept),
 mapping(Term2,CommonConcept),
 not(Term1 = Term2).
```

Listing 3. Simple translation rules (Biffl, Sunindyo, et al., 2009)

```
translate(X,Y).
X = 'User'
Y = 'Developer'
```

This translation is a simple example of translations in general. Ontology Areas for this use case would consider the parts of the ontologies for the stakeholders involved (e.g., stakeholder concepts, their local terminologies and mappings), which can more easily be added to and removed from an ontology as stakeholders change in a particular context (Biffl, Sunindyo, et al., 2009).

5.3.2 Advanced Example

In this section we describe an advanced example of the using of EKB approach (Moser, Biffl, et al., 2011) for integrating heterogeneous OSS data sources.

To support the finding process of OSS health indicators efficiently, we propose an integrated data model based on different data model based on different OSS data sources, which is illustrated in Figure 14. The left-hand side of Figure 14 shows how to identify the local data models from different data sources, e.g., SVN, mailing list and bug tracker and relevant relationships of the local data sources. The process and product level concepts ("common concepts") are obtained from these local data model as illustrated by the dashed colored linked to the common concepts in the top right hand side of Figure 14. The health indicators we derived from querying the ontology in the analysis level are shown in the bottom right hand side of Figure 14.

Typical efforts for collecting data from different OSS data sources have been summarized by Robles *et al.* (Robles, González-Barahona, Izquierdo-Cortazar & Herraiz, 2009), who proposed to collect data from source code management, mailing list archives, and bug tracking systems. This work reports experiences on obtaining and analyzing information from rich set of development-related information in OSS projects. It gives advice for the problem that can be found when retrieving and preparing the data sources that is useful for our analysis. We follow their suggestion by capturing data from SVN, developers' mailing list, and bug report. Those data sources have similarities, e.g., the names of people involved, the time stamps when some action occurred, and names of artifacts involved that could be useful to derive the relationships between those different data sources (see Figure 14).

By identifying different data sources on the data source level we can create the integrated data model on the project/process level. We classify the data model into three parts, namely people, process, and product, based on their support for calculating the health indicators. The benefit of the integrated data model is the flexibility of the data model regarding the data sources in the data source level, i.e., we can add/introduce a new data source into the existing data sources, as long as the new data source can provide the information for the integrated data model on the project/process level. The common concepts in the integrated data model for calculating e.g., the health indicators, remain stable, even if we experiment with varying data sources to run tests or variants of empirical studies.

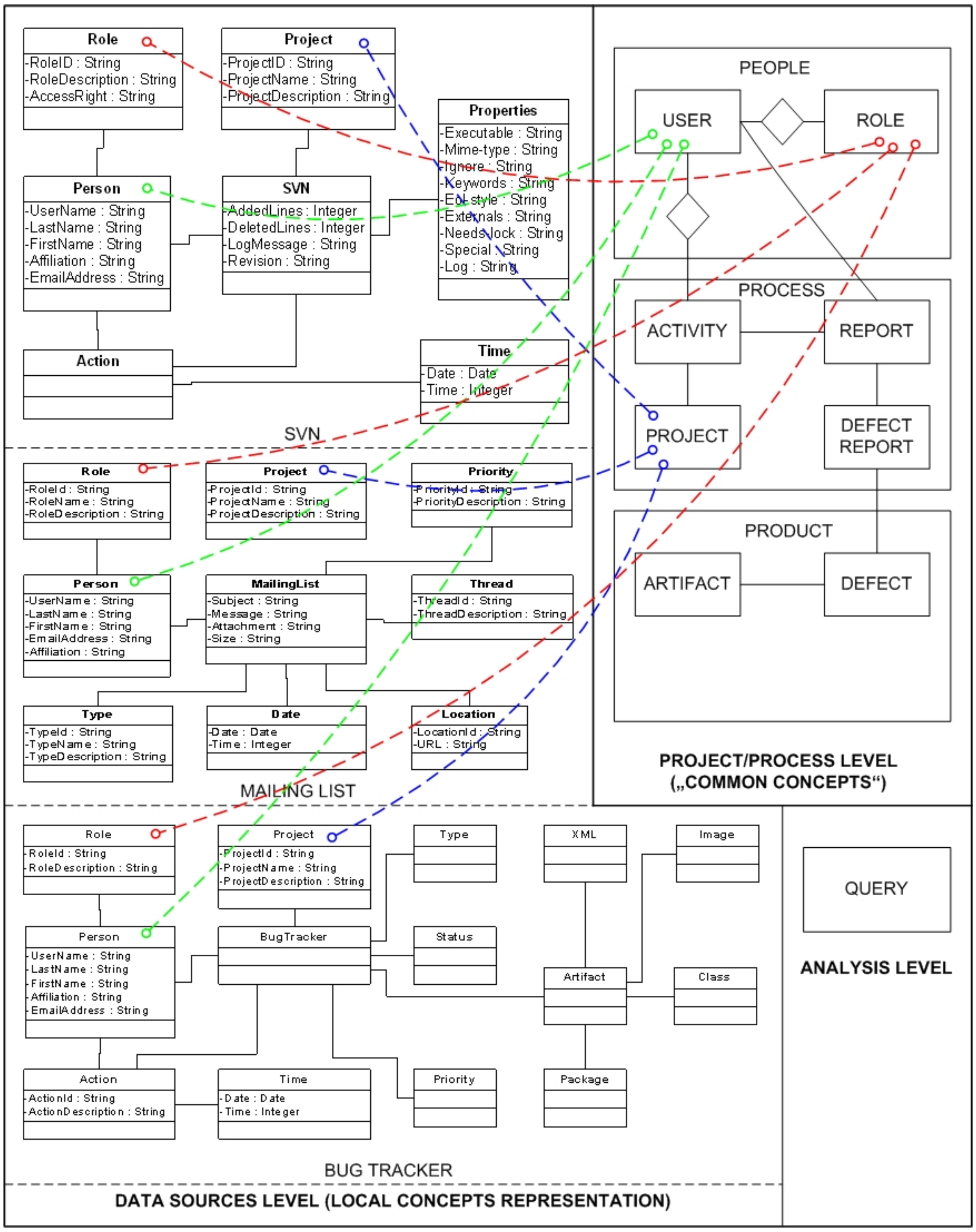


Figure 14. Integrated Data Model to Support OSS Health Indicators (Sunindyo, Moser, Winkler, et al., 2012)

This integrated data model shown in Figure 14 is formulated in UML. Listing 4 shows an example of the formal representation of software developer concepts in *OWL* (Ontology Web Language, http://www.w3.org/TR/owl-features/). This example shows the project ontology including developers, deadline, name and role as representation of the data model. This example and other representation models were built with the *Pro-tégé* (http://protege.stanford.edu) editor that provides useful tools to generate and manage ontologies. The *Project Data Fetcher* tool (Huber, 2010) supports the storage of project data modeled in the data model using an ontological representation. The general ontology architecture consists of a set of so-called tool ontologies (one for each data source to be integrated), and a single so-called project ontology representing the common concepts as well as the mappings and required transformation between tool specific concepts and common concepts.

Listing 4. Excerpt of the ontology represented in OWL (Sunindyo, Moser, Winkler, et al., 2012)

```
<SubClassOf>
   <Class URI="&Ontology1265201409169 ; Project" />
   <ObjectMinCardinality cardinality="1">
        <ObjectProperty URI="&Ontology1265201409169; developers"/>
        <Class URI="&Ontology1265201409169; Developer"/>
   </ObjectMinCardinality>
</SubClassOf>
<SubClassOf>
   <Class URI="&Ontology1265201409169; Project“/>
   <DataMinCardinality cardinality="1">
        <DataProperty URI="&Ontology1265201409169 ; deadline"/>
        <Datatype URI="&xsd ; dateTime"/>
   </DataMinCardinality>
</SubClassOf>
<SubClassOf>
   <Class URI="&Ontology1265201409169; Project“/>
   <DataMinCardinality cardinality="1">
        <DataProperty URI="&Ontology1265201409169 ; name"/>
        <Datatype URI="&xsd ; string"/>
   </DataMinCardinality>
</SubClassOf>
<Declaration>
   <Class URI="&Ontology1265201409169; Project“/>
</Declaration>
<SubClassOf>
   <Class URI="&Ontology1265201409169 ; Role"/>
   <DataExactCardinality  cardinality="1">
        <DataProperty URI="&Ontology1265201409169 ; name"/>
        <Datatype URI="&xsd ; string"/>
   </DataExactCardinality>
</SubClassOf>
<Declaration>
    <Class URI="&Ontology1265201409169 ; Role "/>
</Declaration>
```

Listing 5. Example query to find related SVN entries from a mailing list issue (Sunindyo, Moser, Winkler, et al., 2012)

```
SELECT count(?a) WHERE {domain:bug_id_17034 domain:resolvedBy ?a}

domain:bug_id_17034 owl:equalTo bugtracker:bug_id_17034

SELECT (?b) WHERE
{bugtracker:bug_id_17034 bugtracker:hasAffectedArtifact ?b}
Result: b = bugtracker:dist.xml

bugtracker:dist.xml owl:equalTo domain:dist.xml
domain:dist.xml owl:equalTo SVN:dist.xml

SELECT (?c) WHERE
{?c SVN:hasAffectedArtifact SVN:dist.xml}
Result: c = SVN:SVN_891529_dist.xml
        c = SVN:SVN_891533_dist.xml

SVN:SVN_891529_dist.xml owl:equalTo domain:SVN_891529_dist.xml
SVN:SVN_891533_dist.xml owl:equalTo domain:SVN_891533_dist.xml

Result: count(?a) = 2
```

The *Project Data Fetcher* tool has been developed using the *Java* programming language. For accessing the ontologies, we use the ontology processing features of the *Jena* (http://jena.sourceforge.net) framework. The *Jena* framework provides an *OWL* API for programmatic access to *OWL* ontologies using *Java*. *Jena* also provides the tool "*schemagen*", which creates a *Java* class file containing an instance of the ontology model as well as the elements of the input ontology as static fields. The *Jena* Framework also contains a basic element, the *OntModel* class. The *OntModel* provides features to modify the model and persist the model into a file. Once the ontology model is accessible, the next step is to provide a way to configure the tool using "*Common Configuration*". This configuration enables *Java* applications to read configuration data from a variety of sources, e.g., the URL of SVN, mailing list, and bug tracker data sources. For accessing the subversion repository, we used the "*SVNKit*" (http://svnkit.com) code library. This library is an OSS toolkit for Java and provides an API to access and manipulate subversion repositories online as well as local working copies. For accessing the data from the mailing list archives, we used "*mstor*" (http://mstor.sourceforge.net) library. This library provides access to email messages in *mbox* format, which is stored file-based.

Listing 5 shows an example of ontology-based querying for project monitoring to find related SVN entry from the mailing list issue.

5.4 Health Indicators Analysis Method

The term "health indicators" was introduced by Wahyudin et al. (Wahyudin et al., 2006) to measure the healthy status of OSS project, as analogous to measuring the temperature of the human boy with respect to indicating whether a person is likely to be sick or in healthy condition (Wahyudin, 2008). The aim of the using of health indicators analysis method is to help the OSS stakeholders to get an overview on a large portfolio of OSS projects.

In this section, we provide an overview of a pilot application for empirically investigate the feasibility of health indicators analysis method in four OSS Apache projects (Lenya, Log4J, Excalibur, and OJB), its study objects, and its threats to validity.

5.4.1 Pilot Application

As explained in section 5.2, the FOSSDA is an instantiation of the POAF for OSS project context. To investigate the feasibility of FOSSDA, we implement a pilot application based on a set of tools, e.g., the *Project Data Fetcher* (Biffl, Sunindyo & Moser, 2010b) and the *Project Monitoring Cockpit* (Biffl, Sunindyo & Moser, 2010a). The implementation of each FOSSDA layer is as follows.

Layer 1: Data Sources. In the data sources layer (FOSSDA layer 1 in Figure 11) of our pilot application, we used SVN[22], developers' mailing lists, and *Bugzilla*[23] tools. SVN is a source code management tool that is widely used to control the revision of source code during the development of OSS products. We used SVN as a data source for analysis purposes because of its popularity and ease to provide data. Developer's mailing lists allow collecting information about the activities of developers during development phase. *Bugzilla* is a bug reporting system, which can be used to monitor information on bugs and for tracking the status of the bugs.

Layer 2: Core Framework. The core framework includes the *Project Data Fetcher* (Biffl, Sunindyo & Moser, 2010b) to collect and integrate data from the different data sources listed in layer 1. We can collect the data by defining the tool configuration, e.g., the starting number of SVN revision, starting date of mailing list posts, and the starting date of bug report collection, and then run the application. By using the ontology, we

[22] http://subversion.tigris.org
[23] http://www.bugzilla.org

can also validate the data by using reasoning to the collected and integrated data, e.g., to identify missing or incomplete entries.

Layer 3: Key Performance Indicators. In this pilot application, we implemented the calculation of two health indicators derived from (Wahyudin, D., Mustofa, Schatten, Biffl & Tjoa, 2007).

Indicator 1: Bug delays. We measured the service delays on open issues by subtracting the closing date of issues in the bug report from the opening date of the issues. We called this service delay the "bug closure duration". We classify the bug closure duration of a project into five categories: closure duration of (a) less than 7 days, (b) between 7 and 30 days, (c) between 30 and 100 days, (d) between 100 and 365 days, and (e) more than 365 days. We use bar graphs to show the number of bugs for each category as percentage values (see Figure 28). A healthy project should provide shorter bug resolution durations with most bugs fixed in less than 7 days. We use this threshold value to see the response of developers in addressing a new bug status change within one week. We consider the developers to be fast enough to react to the bug status change within one week; otherwise they are not aware of that change.

Indicator 2: Proportion of activities. We measured the proportions by comparing the number of bug status changes per times slot and the volume of mailing list postings in the same time slots. We used a line graph to show the proportions between the bug status changes and the volume of mailing list postings per month (see Figure 29). A healthy project shows a stable proportion of activities (neither many mails nor few mails per bug). The fluctuations of activities show the imbalance between developer email submissions and bug status reports.

Layer 4: Presentation. For presenting the results of OSS health indicators analysis, we used a tool, called *Project Monitoring Cockpit* (*ProMonCo*) (Biffl, Sunindyo & Moser, 2010a). The *ProMonCo* takes the analysis results as inputs and displays the health indicators in graphical format for the project/quality managers.

5.4.2 Study Objects

We studied four projects from the Apache Software Foundation[24], namely Apache Lenya[25], Apache Log4J[26], Apache Excalibur[27] and Apache OJB[28]. The reasons of

[24] http://www.apache.org/
[25] http://lenya.apache.org/
[26] http://logging.apache.org/log4j/1.2/
[27] http://excalibur.apache.org/
[28] http://db.apache.org/ojb/

choosing these study objects were (a) the completeness of data sources to collect, i.e., at least SVN, mailing list, and bug reports, (b) the ease to collect the data and obtain information about the projects, (c) the activities of developers during development phase, (d) the maturity and lifetime of the projects are quite long for investigation, and (e) access to OSS experts who can provide expert opinions on the actual health of these projects. According to OSS experts' opinion, Lenya and Log4J were active projects. We used Excalibur and OJB, two inactive projects that were moved to Apache Attic[29] () as counter examples for the comparison of health indicator analysis results on different project conditions.

Table 1 shows the information of four projects used as study objects.

Table 1. Different Projects used in Project Monitoring Researches

Project Name	Stable Release	Date Release	Status	Type
Lenya	2.0.3	20.01.2010	Active	Content management system
Log4J	1.2.16	06.04.2010	Active	Logging tool
Excalibur	2.2.3	05.07.2007	Inactive	Inversion of control framework
OJB	1.0.4	31.12.2005	Inactive	Object-relational mapping

We collected SVN entries, mailing list, and *Bugzilla* data from four projects starting from January 1st, 2007 until December 31st, 2010 (36 months), so we have enough data for comparison of developers' activity in long period. The data set retrieved from *Lenya* consists of total 8,464 e-mail conversations (mean per month = 176.33) and 810 bug status changes (mean per month = 16.87). The data set retrieved from *Log4J* consists of total 4,605 e-mail conversations (mean per month = 95.94) and 580 bug status changes (mean per month = 12.08). The data set retrieved from *Excalibur* consists of total 886 e-mail conversations (mean per month = 18.46) and 20 bug status changes (mean per month = 0.42). The data set retrieved from *OJB* consists of total 369 e-mail conversations (mean per month = 7.68) and 18 bug status changes (mean per month = 0.38).

5.4.3 Threats to Validity

In this section, we discuss four types of threats to the validity of an empirical study and how we addressed these threats.

[29] http://attic.apache.org/

Conclusion validity. Threats to conclusion validity are the reliability of treatment implementation and random heterogeneity of subjects. To deal with these threats, we use an automated tool to collect data to avoid human error during data collection. We also used limited data sources, i.e., SVN, mailing list, and *Bugzilla* instead of a bigger number of data sources to reduce the heterogeneity of our subjects.

Internal validity. Threats to internal validity are the risk that the history affects the experimental results and the subjects respond differently at different time, if the test is repeated. To deal with these threats, we used a specific range of date for data collection, e.g., from January 1st, 2007 to December 31st, 2010. This date range provides stable results each time the experiment is repeated. The use of an automated tool to collect the data also makes the subject, i.e., project data sources; respond similarly each time the test is conducted.

Construct validity. Threats to construct validity are the inadequate preoperational explication of constructs and mono-method bias. To deal with these threats, we conducted a literature survey on related topics and conducted prior experiments with different methods to get experience with the OSS data analysis topic. We also used several methods for health indicator analysis.

External validity. Threats to external validity are the limited number of projects we analyzed and the use of single project management standard in our experiment. Therefore, the study results have reasonable validity for OSS Apache projects but should be applied to other kinds of projects with care. To strengthen external validity in future work, we will add more projects from other project management standards, e.g., *SourceForge* and *RedHat*.

5.5 Bug History Analysis Method

Bug resolution process could be one of project observation sources. By analyzing bug report data, we can provide engineering process observation for improving the process quality (Sunindyo, Moser, Dhungana, et al., 2012). In this section, we provide an overview of a pilot application for empirically investigate the feasibility of the bug history analysis method, its study objects, and its threats to validity.

5.5.1 Pilot Application

The pilot application for bug history analysis method is based on FOSSDA and a set of tools, e.g., the Bug History Data Collector (Sunindyo, Moser, Dhungana, et al., 2012) and the Process Mining (ProM) (van der Aalst, 2011). The implementation of each FOSSDA layer is as follows.

Layer 1: Data Sources. In the data source layer, we have bug database which contains all bugs information that are used in software development. However, in this case, we don't need all of those data and focus on bug history data, which can be extracted from the bug data.

Layer 2: Core Framework. In the core framework layer, we extract and collect bug history data from filtered bugs. Filtering on the bug database for example focusing on project, version, status (open or closed bugs), times duration of bugs, priority, severity, or bug reporter. We collect the bug history data from the bug database by using a bug history collector.

A bug history is a set of state transitions of one bug id. We collect bug history data from some bug ids, integrate and validate them in the bug event data log by using data integrator and validator.

Layer 3: Key Performance Indicators. The even data log from previous level is analyzed by using Process Mining tool. As a key performance indicator here, we investigate the frequency of states for different versions of OSS project.

Layer 4: Presentation. The analysis results are presented to the project managers. The results of Bug History Analysis Method are obtained by comparing the designed process model and the process model generated from actual bug history data.

5.5.2 Study Objects

In this section, we use Bugzilla on RHEL[30] as case of our bug database on OSS projects. RHEL is a Linux distribution produced by Red Hat Company and is targeted toward the commercial market, including mainframes. RHEL is a popular, quite stable and mature OSS development project that is well-supported by the company and community.

Currently, in the Red Hat Bugzilla browser, there are in total 21.640 bugs reported for RHEL version 4 (2.300 open bugs and 19.340 closed bugs), 41.187 bugs reported for RHEL version 5 (6.441 open bugs and 34.746 closed bugs) and 23.768 bug reported for RHEL version 6 (7.465 open bugs and 16.303 closed bugs).

We focus on the use of closed bugs data from RHEL 4, RHEL 5, and RHEL 6. The usage of closed bugs only allows for an easier analysis of the process model based on the historical data, especially on the status changes of each bug.

[30] https://bugzilla.redhat.com

The selection of using closed bugs data in our research is based on the assumption that closed bug data contains all necessary steps which are required from introducing bugs till closing the bugs in a complete cycle. Open bugs may contain a lot of introductionary bug states and unnecessary intermediate states that may hinder an effective and efficient generation of valid process models. We also have conducted preliminary experiments on previous versions of RHEL (version 2.1 and 3), but the resulting data contains more unnecessary or duplicate bug states which were getting reduced in the next versions. The usage of data from version 4, 5 and 6 is representative enough to show the trend of states reducing between versions.

5.5.3 Threats to Validity

We identified and addressed threats to internal and external validity of our evaluation results as follows.

Threats to internal validity - Numbers of states. As we have conducted previous experiments using fewer data, there is a tendency of increasing of the numbers of states as new data is added. So we put more focus on the frequency of states taken during development, rather than only the number of states in the process model, since the number of states can be unnecessary increasing, while the top states remain stable.

Threat to external validity. In this study we focus on three versions of RHEL projects with similar size and characteristics. The selection of these homogeneous OSS projects may raise concerns whether the resulting process models are also valid for other project contexts. While we assume our approach to hold for projects similar to our study objects (i.e., under Red Hat or similar managements with active and large developer community), further research work is necessary to investigate projects with strongly differing characteristics.

5.6 Workflow Validation Analysis Method

In heterogeneous software and systems development environments, different stakeholders from different engineering fields, typically work separately in their workplaces, defining and using their own workflows to solve some specific tasks. The interactions between different stakeholders are coordinated and monitored by the project managers, who have the responsibility to monitor the progress of project and take actions or decisions based on the current status of the project, e.g., to add more personnel or change/improve the overall engineering process

To be able to monitor the progress of projects, the project managers require an integrated overview across different workflows used by the heterogeneous project stakeholders, such as that project managers can monitor the interactions between engineers and that they can validate the designed workflows compared to the actual engineering processes (Sunindyo et al., 2011).

The workflow validation analysis method provides the project manager the way to integrate and validate different workflows used by the different stakeholders. In this section, we provide an overview of a pilot application for empirically investigate the feasibility of workflow validation analysis method in Continues Integration and Test (CI&T) context and its study objects.

5.6.1 Pilot Application

The pilot application for workflow validation analysis method is based on FOSSDA. The implementation of each FOSSDA layer for the workflows validation analysis method is as follows (Sunindyo et al., 2010a).

Layer 1: Data Sources. The data source layer consists of event data from different tools, e.g., build servers, software development environment, and source code management systems.

Layer 2: Core Framework. The core framework layer collects and integrates heterogeneous tool event data for further process analysis and improvement. The capability to observe the tool-based engineering process enables model-based analysis of the engineering process in order to compare expected and actual process variants as foundations for planning and tracking the improvement of the engineering process.

Layer 3: Key Performance Indicators. We perform three types of analysis (van der Aalst, 2005): (a) process conformance of actual to designed processes, i.e., the analysis of processes and occurring unexpected exceptions as foundation for process improvement); (b) performance analysis based on the process models to identify process bottleneck for process improvement; and (c) decision point analysis, i.e., to measure relative frequency of process execution paths to identify normal and exceptional paths.

Layer 4: Presentation. The analysis results are presented to the project managers. The results of Workflow Validation Analysis Method are obtained by comparing the designed process model and the process model generated from actual event log data.

5.6.2 Study Objects

In this section, we describe the design and results of the workflow validation in the context of the standard CI&T process. We selected the CI&T process approach because of the involvement of a set of various tools (build, automated tests, and deployment) as a representative best-practice approach from the agile software engineering. The evaluation fulfils requirements for SE process analysis and improvement, namely (a) design events as system outputs and collect all events from the system for further usage, (b) filter and transform collected events to the format of the process analysis tool, so that they can be used for further analysis, (c) define what kind of process analysis methods we use for analyzing event logs, (d) use the process analysis tool for analyzing event logs and use the results for improving processes.

The evaluation also follows process analysis and mining guidelines (van Dongen & van der Aalst, 2005) to organize, plan, and execute the evaluation study. Goal of the evaluation is (a) to show the benefits of integrated event capturing across disciplines, (b) to illustrate the capabilities of the proposed event analysis process, and (c) to analyze the designed process and the real-life process. First, we define the expected process model based on the CI&T use case and then present a set of analysis results gathered using the process analysis tool ProM.

The expected SE process model for the CI&T use case implemented in EngSB systems in our lab as shown in Figure 15 is represented using Business Process Modeling Notation (BPMN) notation. The model consists of a set of activities for the CI&T process implementation: building the system, running tests, deploy activities, and finally reporting test and deployment results. The CI&T use case shows a key feature of an iterative software development process: if part of a system or engineering model gets changed, the system has to be rebuilt and tested in order to identify defects early and to provide fast feedback on the implementation progress to the project manager and the owners of the changed system parts. In modern SE environments this part is done by Continuous Integration (CI) servers like Continuum (http://continuum.apache.org/) or Hudson (http://hudson-ci.org/). For a typical Java project a Maven (http://maven.apache.org/) or Ant (http://ant.apache.org/) script will guide the CI process (Biffl & Schatten, 2009) . In a large system, the process model involves more processes for more components. Some parts could be on testing level, while other parts are already on deployment level.

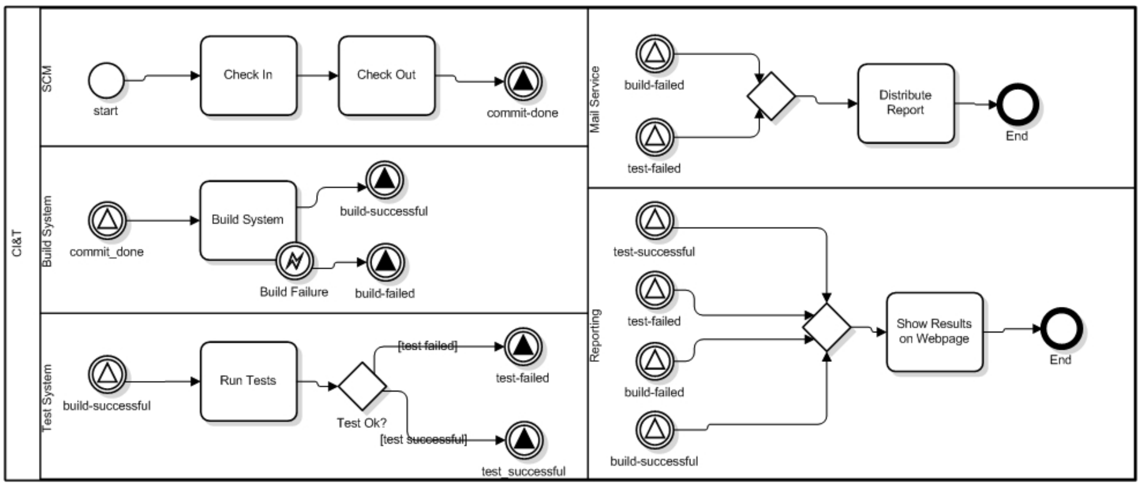
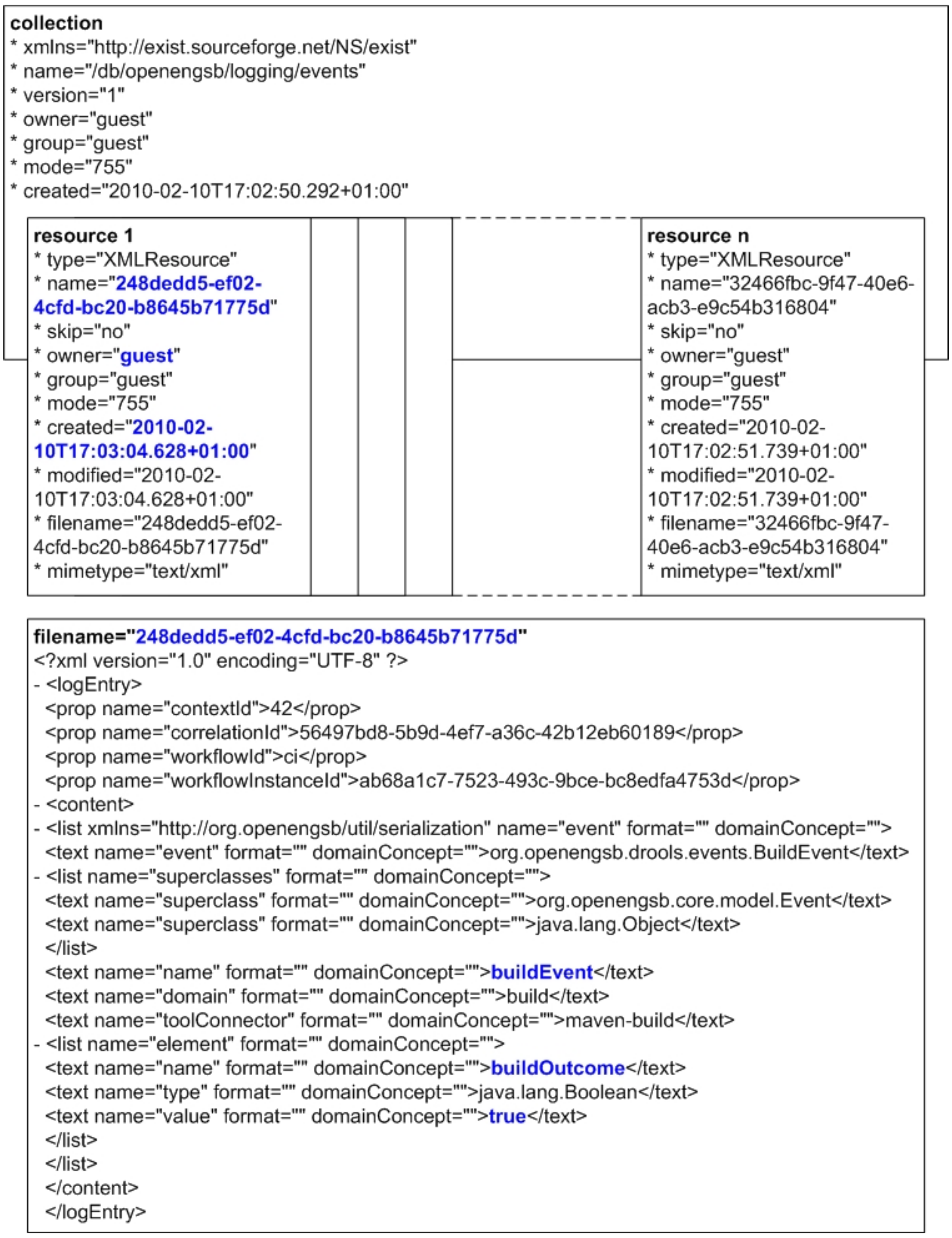


Figure 15. Model of the Expected CI&T process (Sunindyo et al., 2010a)

With respect to tool level events, we record events between every activity, from the start event to the end event. After the check-in event, we set the build-start event before the Build System that produces either build-failed event or build-successful event. If the build is successful, then the test-start event is set before the tests run (Test Run). The test activity result is either the test-failed event or the test-successful event. If the test is successful, then the deploy-start event is set before the deployment activity (Deploy). The deployment activity result is either the deploy-failed event or the deploy-successful event. The failed events or the deploy-successful event leads to the end event, which triggers report generation and notification of the relevant SE roles. Note that a report will be generated and sent to related stakeholders in case of failed activities (Send Report). For evaluation, we identified and captured event logs from running EngSB systems. There are seven event types: check in, build-start, build-complete, test-start, test-complete, deploy-start and deploy-complete. The tool-level events in Figure 15 are the model of expected CI&T process that is compared with the actual model derived from SE process-level events in Figure 30.

We collected 360 event log files from running a CI&T configuration of the EngSB system with the structure as shown in Figure 16.



```
collection
* xmlns="http://exist.sourceforge.net/NS/exist"
* name="/db/openengsb/logging/events"
* version="1"
* owner="guest"
* group="guest"
* mode="755"
* created="2010-02-10T17:02:50.292+01:00"
```

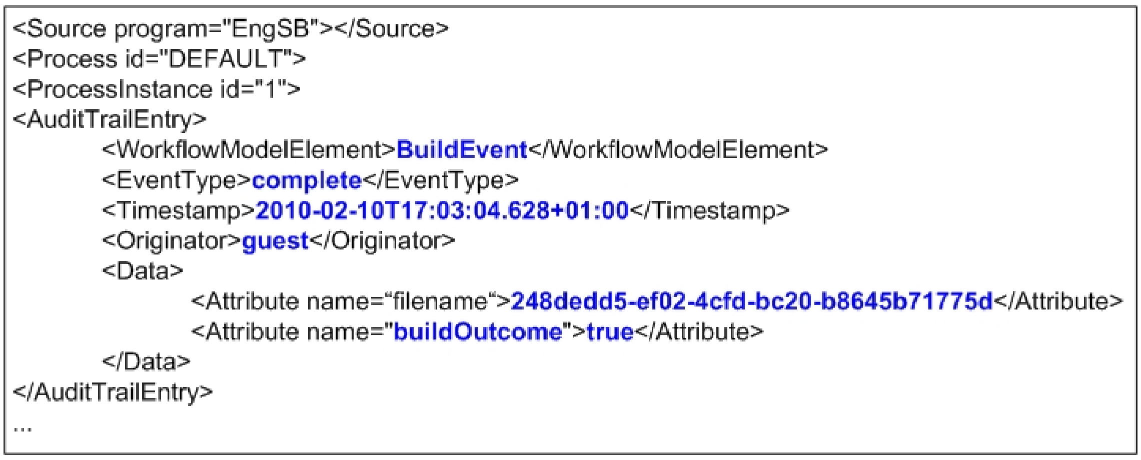
```
resource 1
* type="XMLResource"
* name="248dedd5-ef02-4cfd-bc20-b8645b71775d"
* skip="no"
* owner="guest"
* group="guest"
* mode="755"
* created="2010-02-10T17:03:04.628+01:00"
* modified="2010-02-10T17:03:04.628+01:00"
* filename="248dedd5-ef02-4cfd-bc20-b8645b71775d"
* mimetype="text/xml"
```

```
resource n
* type="XMLResource"
* name="32466fbc-9f47-40e6-acb3-e9c54b316804"
* skip="no"
* owner="guest"
* group="guest"
* mode="755"
* created="2010-02-10T17:02:51.739+01:00"
* modified="2010-02-10T17:02:51.739+01:00"
* filename="32466fbc-9f47-40e6-acb3-e9c54b316804"
* mimetype="text/xml"
```

```
filename="248dedd5-ef02-4cfd-bc20-b8645b71775d"
<?xml version="1.0" encoding="UTF-8" ?>
- <logEntry>
  <prop name="contextId">42</prop>
  <prop name="correlationId">56497bd8-5b9d-4ef7-a36c-42b12eb60189</prop>
  <prop name="workflowId">ci</prop>
  <prop name="workflowInstanceId">ab68a1c7-7523-493c-9bce-bc8edfa4753d</prop>
- <content>
- <list xmlns="http://org.openengsb/util/serialization" name="event" format="" domainConcept="">
  <text name="event" format="" domainConcept="">org.openengsb.drools.events.BuildEvent</text>
- <list name="superclasses" format="" domainConcept="">
  <text name="superclass" format="" domainConcept="">org.openengsb.core.model.Event</text>
  <text name="superclass" format="" domainConcept="">java.lang.Object</text>
  </list>
  <text name="name" format="" domainConcept="">buildEvent</text>
  <text name="domain" format="" domainConcept="">build</text>
  <text name="toolConnector" format="" domainConcept="">maven-build</text>
- <list name="element" format="" domainConcept="">
  <text name="name" format="" domainConcept="">buildOutcome</text>
  <text name="type" format="" domainConcept="">java.lang.Boolean</text>
  <text name="value" format="" domainConcept="">true</text>
  </list>
  </list>
  </content>
  </logEntry>
```

Figure 16. Structure of EngSB event logs (Sunindyo et al., 2010a)

The collected event logs were transformed into MXML format as shown in Figure 17 for further analysis. From the transformed event logs, we performed process model verification and process performance analysis by using the ProM tool.



```
<Source program="EngSB"></Source>
<Process id="DEFAULT">
<ProcessInstance id="1">
<AuditTrailEntry>
        <WorkflowModelElement>BuildEvent</WorkflowModelElement>
        <EventType>complete</EventType>
        <Timestamp>2010-02-10T17:03:04.628+01:00</Timestamp>
        <Originator>guest</Originator>
        <Data>
                <Attribute name="filename">248dedd5-ef02-4cfd-bc20-b8645b71775d</Attribute>
                <Attribute name="buildOutcome">true</Attribute>
        </Data>
</AuditTrailEntry>
...
```

Figure 17. Transformation of event logs for further process analyses (Sunindyo et al., 2010a)

5.7 *Summary*

In this chapter we proposed and evaluated the FOSSDA (an instantiation of POAF) to observe and analyze engineering process in OSS domain to provide an easier way for the project management to monitor and manage the OSS projects based on the immediate status of OSS projects.

In contrast to current project management technologies like SVN or bug reporting, the FOSSDA-based approached integrate different methods to monitor and manage the projects, make the project managers have greater confidence on the project status that is supported by the hard data on the OSS project.

The following sub-sections describe the findings and results for the FOSSDA description, for integrated data model, for the health indicators analysis method, for the bug history method and for the workflow validation analysis method.

5.7.1 Framework for OSS Data Analysis

In this section, we introduced and evaluated the framework for OSS data analysis (FOSSDA) as an instantiation of the POAF for OSS Project context. The FOSSDA consists of four layers, namely data sources, core framework, key performance indicators, and presentation. Major contributions of FOSSDA are ontology to store the structure of integrated data collection results, improvement of tools for collecting data, and improvement of tools to analyze the collected and integrated data. (Sunindyo, Moser, Winkler, et al., 2012)

5.7.2 Integrated Data Model

In this section, we explained the simple and advanced examples of the integrated data model to support the collection and integration of heterogeneous data sources, for example SVN, mailing list, and bug report. By following the example of integrated data model, there is possibility to add new data sources to existing integrated data model (Sunindyo, Moser, Winkler, et al., 2012).

5.7.3 Health Indicators Analysis Method

In this section, we explained the use of health indicators analysis method to measure the healthy status of OSS project. A pilot application is implemented to investigate the feasibility of the approach in four OSS Apache projects, namely Lenya, Log4J, Excalibur and OJB. In this section we also discuss four types of threats to the validity of the empirical study, namely conclusion validity, internal validity, construct validity, and external validity, and how we addressed these threats (Sunindyo, Moser, Winkler, et al., 2012).

5.7.4 Bug History Analysis Method

In this section, we explained the bug history analysis method to observe and analyze bug report data to improve the process quality. We use Bugzilla on RHEL as case of our bug database on OSS projects. The number of states and the using of homogeneous OSS projects are among threats to our experiment validity for this method (Sunindyo, Moser, Dhungana, et al., 2012).

5.7.5 Workflow Validation Analysis Method

In this section, we explained the workflow validation analysis method to observe and analyze engineering process data from CI&T domain. The workflow validation analysis method provides the project manager the way to integrate and validate different workflows used by different stakeholders. By comparing the designed process model with the process model generated from actual event log data we could get the information of process deviation and suggest for process improvement (Sunindyo et al., 2010a).

6 Process Analysis in Automation Systems Engineering Environments

This chapter summarizes the results of applying the Project Observation and Analysis Framework (POAF) to application scenarios from automation systems domain. The process directly relates to the three general steps of the POAF, namely data collection and integration, data analysis, and data presentation.

The first section gives an overview of the application scenarios, the common challenges and a detailed description of the specific requirements of these application scenarios. The second section gives explanation on the workflow validation cycle process as instantiation of the POAF applied to support signal change management process observation and analysis in ASE environments. The third section gives an overview of the signal change management as an example of application domains of workflow validation in automation systems. The fourth section gives an overview of the project progress and risk monitoring method. The fifth section gives an overview of the process model validation method. Finally, the sixth section concludes the chapter.

6.1 Overview

In multidisciplinary engineering environments, different stakeholders from heterogeneous engineering domains, e.g., mechanical, electrical, and software engineering, often are required to collaborate to produce products or services, like power plants or production automation systems. In doing their jobs, these stakeholders use different engineering processes, methods, and tools with specific data models, addressing the individual needs of the involved engineers.

The processes of doing their jobs are written in different workflows that represent the steps describing how processes are conducted, which inputs are needed and what output is produced. However, project managers need to have an integrated view on the heterogeneous workflows from different engineering fields in order to be able to manage the engineering processes and improve the process and product quality.

Major challenges for the project managers in the multidisciplinary engineering environments are as follows, (1) the heterogeneity of data models, tools, and semantic to represent workflows is hard to manage, (2) manual data collection for workflows analysis is time-consuming and error-prone, (3) the relationship analysis on different workflows is hard to do due to the lack of linkage information between different workflows that use and share similar information.

Some approaches for observing engineering processes in software engineering environments have been proposed, for example Hackystat (Johnson, 2001) and Ginger2 (Torii et al., 1999) frameworks. However, the research on engineering processes observation in multidisciplinary engineering environments is surprisingly limited.

In this research, we propose applications of the workflow validation cycle process as an instantiation of common observation and analysis framework to four application scenarios of the automation systems, namely continuous integration and test, cascading continuous integration and test, signal change management, and production automation systems.

6.2 Workflow Validation Cycle Process

For achieving the research objective and solving the research questions in section 3, we propose a framework for supporting workflow observation and validation for project progress monitoring. This work is a part of larger project in the CD Lab[31] with intention to monitor the impact of introducing/using different workflows to the project progress in general.

Figure 18 illustrates the framework for observing and validating the engineering workflows in the multidisciplinary engineering environments. The framework includes some preparation steps and core steps. The preparation steps consist of some preliminary steps to prepare and process workflow data from different stakeholders, including their transformation and evaluation to different forms. The core steps consist of important steps in observing and validating the engineering workflows which become the main part of our research contributions. The explanation for each step is as follows.

(1) Informal model description. The users, e.g., the customers use informal model, e.g., flowchart to represent their intended process. They design their intended workflow by using representation of model and free tool, e.g., flowchart or sketch to make a higher flexibility to the customers to express their requirements and not restricted to the rules of the notation. The customers then send the informal model to the workflow model for formalization of the model in machine-readable notation.

(2) Formal model description. The workflow designers get the informal model description of the workflow from the customers. They transform the informal model into

[31] http://cdl.ifs.tuwien.ac.at

the formal model, e.g., by using BPM notation. The output of this transformation is a formal workflow model, including tools, events and process steps needed to implement the workflow. The workflow designers then check with the customers about the validity of the model to fulfill the customers' requirements, e.g., by interview or survey.

(3) Rule engine transformation. The transformation of the formal workflow model to the rule engine is aiming at deriving some rules from the workflow model that must be obeyed during the workflow implementation and the running of the system. The workflow designers set the rules for the model, e.g., what kind of rules for events produced. The rules contain condition and action, criteria and objects to be arranged. The workflow designers also set the different cases for the events produced, e.g., normal case, exception, abort, start and end. The rules are also evaluated to check whether the rules in the rule engine are really suitable with the formal workflow model in the BPM notation.

(4) EngSB transformation. After generating rules from the rule engine, the rules structures are sent to EngSB for generating workflow events. Even though the EngSB has many functions, in this case we see the EngSB as events generator for workflow analysis.

(5) Workflow results. The EngSB send the workflow to the Engineering Data Base (EDB) and the Workflow Results. The EDB will store the workflow data for semantic analysis purpose, while the workflow results will be useful for quality analysis purpose.

(6) Event logs. The results of workflow for quality analysis purpose are kept in the event log with certain format. The contents of event log are evolving following the contents of the workflow results.

(7) Process analysis. The event data in the log are sent to be analyzed in the next step. The analysis is including validating the event log data with the workflow model in the BPM Notation or other representation.

(8) Evaluation of process metrics. The analysis step takes event log data as inputs and produce some metrics, e.g., process and product metrics, as results of analysis.

(9) Conformance checking. The process and product metrics produced from analysis step, later are sent for conformance checking by using process mining tool (ProM). The results of this step are displayed to the project manager for further actions/decisions.

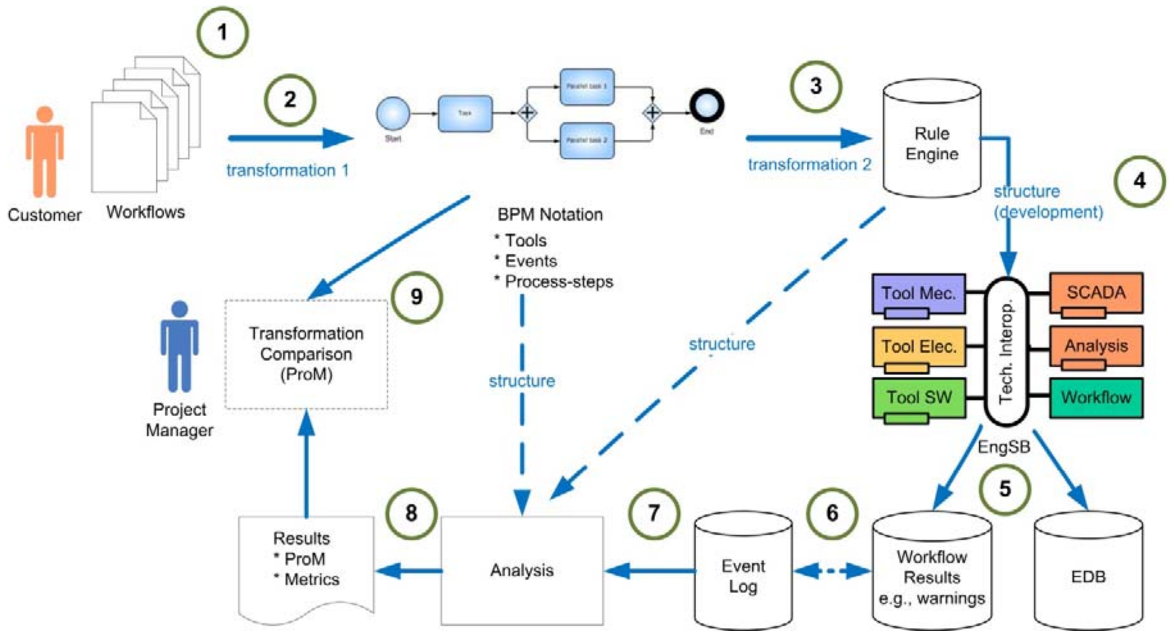


Figure 18. A Framework for Workflows Observation and Analysis (Sunindyo, 2011)

6.3 Change Management Process Observation and Analysis

This section provides an overview, pilot application, and study object of the change management process observation and analysis method as implementation of the workflow validation cycle framework to the automation systems engineering.

6.3.1 Overview

Automation Systems Engineering (ASE), e.g., the engineering of production automation systems or power plants, includes a wide range of heterogeneous disciplines, such as mechanical engineering (e.g., physical layout), electrical engineering (e.g., circuit diagrams), and software engineering (e.g., UML diagrams, function plans, and software code) (Biffl & Schatten, 2009). Normally, different disciplines apply individual engineering processes, methods, and tools with specific data models, addressing individual needs of involved engineers.

Observations at our industry partner – a power plant systems integrator – showed that traditional ASE processes follow a basic sequential process structure with distributed parallel activities in specific phases and suffer from a lack of systematic feedback to earlier steps, inefficient change management and synchronization mechanisms of disciplines, and low engineering process automation across domain boundaries (Moser & Biffl, 2010). Specific tools and data models typically address only the needs of one individual discipline and hinder efficient collaboration and interaction between disciplines. Because of this lack of collaboration, change management becomes even more difficult, leading to development delays and risks for system operation.

For instance, changing hardware components (e.g., hardware sensors) might require changes in software components (e.g., caused by changed value ranges, data types (e.g., analogue / digital sensor), or number of connection points). Thus, key questions in context of change management are (a) how changes can be handled more efficient and (b) how relevant change requests can be passed to involved engineers. From project management perspective, loosely coupled processes hinder a comprehensive observation of current project states and make project control more difficult (Biffl & Schatten, 2009). Thus, there is a need for flexible and comprehensive engineering process support across disciplines to enable collaboration and interaction between disciplines, tools, and data models. In context of process observation, key questions include (a) how to model and evaluate engineering processes in heterogeneous environments and (b) how to derive project metrics based on these observations.

The Engineering Service Bus (EngSB) – a middleware platform for supporting collaboration across disciplines and domain borders – bridges the gap between heterogeneous disciplines by providing semantic integration of data models (Biffl, Sunindyo, et al., 2009; Moser, 2009; Moser & Biffl, 2010) based on the technical integration of domain-specific tools (Biffl & Schatten, 2009; Biffl, Schatten, et al., 2009). An integrated view on heterogeneous engineering environments enables (a) comprehensive process support across disciplines, (b) efficient change management, and (c) process and project observation. Defined processes intertwined with tools and data models can enable process automation, observation, and improvement based on process measurement.

6.3.2 Pilot Application

This section presents (a) "Signal Engineering" as a common concept in the automation systems domain, (b) a proposed change management process for signal changes at a power plant systems integrator, and (c) concepts for process evaluation (verification and validation) and analysis (as basis for project observation).

6.3.2.1 Signal Engineering

Collaboration between heterogeneous disciplines and tools requires common concepts for mapping individual models and activities. Our observation at the power plant systems integrator showed that signals (Sunindyo, Moser, Winkler, & Biffl, 2010b) are common concepts in this domain that link information across different engineering disciplines. Signals include process interfaces (e.g., wiring and piping), electrical signals (e.g., voltage levels), and software I/O variables. Consequently, we use signals as a vehicle to link domain-specific data between different engineering disciplines and define the application field “Signal engineering” with focus on signal management facing the following important challenges: (a) make signal handling consistent, (b) integrate signals from heterogeneous data models/tools, and (c) manage versions of signal changes across engineering disciplines. Note that the identification of common concepts (signals in this thesis) depend on the application domain and may differ. More general, the common concept can be described as “engineering object” and may include other building blocks of the automation systems, e.g., hardware components or software control units.

6.3.2.2 Change Management Process Design

System integrators have to synchronize several engineering data, i.e., signals, derived from heterogeneous sources, e.g., electrical, mechanical and software engineering, to a virtual common data model (VCDM) used by the EDB. Note that changes are defined within modified signal lists derived from individual tools and have to be synchronized with the current overall signal list in the EDB. Thus, change management refers to the merging process of signal lists with EDB data during synchronization. Figure 19 presents a basic signal check-in workflow at the synchronization step.

In addition to unchanged signal, changes can include (a) new signals, (b) removed signals, and (c) modified signals. Signal modifications result in a notification of involved stakeholders based on the project environment, e.g., involved stakeholders, related roles, and engineering process phase. Based on signal synchronization, individual engineers gain updated signals (prepared for specified tools and data models) for application within their individual tools. We introduced events (marked by circles in Figure 19) to (a) evaluate the change management process and (b) to measure project metrics based on changes. Table 2 summarizes the events used to observe and evaluate the proposed change management process.

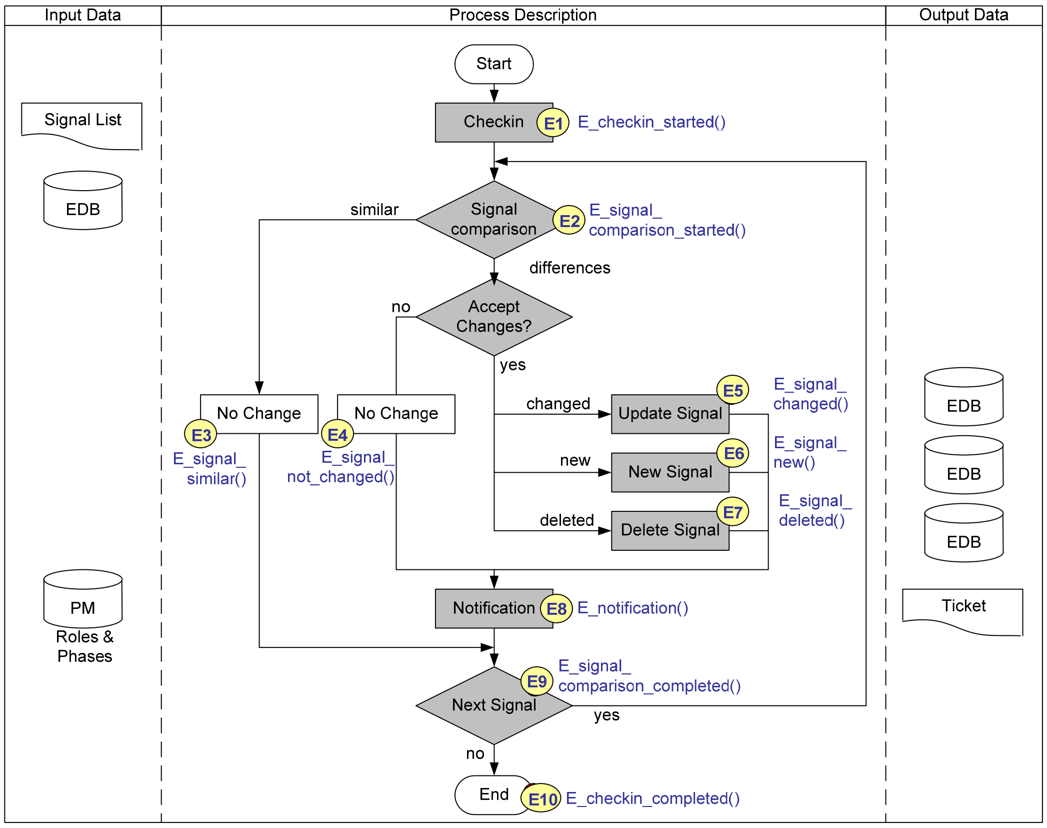


Figure 19. Signal Change Management Workflow (Winkler et al., 2011)

Table 2. Change management process: table of events (Winkler et al., 2011)

Abbr.	Event Name & Description
E1 E10	*E_checkin_started (E1) and E_checkin_completed (E10)* represent one completed check-in sequence, i.e., a sequence of individual signals derived from a defined source.
E2 E9	*E_signal_comparison_started (E2) and E_signal_comparison_completed (E9)* focus on one signal within a check-in process in one signal list.
E3	Unchanged signals are reported by using the event *E_signal_similar (E3)*. This event is necessary to see whether the change management process works as expected.
E4 E5 E6 E7	Signal changes (i.e., deviations of EDB signal attributes and new signal list attributes) can be rejected *(E_signal_not_changed (E4))* or accepted. In case of accepting signal changes, three different events are required: (a) signal modified *(E_signal_changed (E5))*, (b) a new signal introduced *(E_signal_new (E6))*, or an existing signal should be removed from the EDB, i.e., missing entry in the signal list *(E_signal_deleted (E7))*.
E8	After change handling (accepted or rejected) a summarized notification *(E_notification (E8))* will be sent to all related stakeholder.

6.3.2.3 Process Evaluation and Project Metrics

Events are the foundation for process evaluation and project measurement. Derived event data from the change management process implementation show whether the process behaves like expected (process verification and validation) and what the bottle-necks of individual process steps are in terms of number of executed transitions and duration. In addition, process event data is the foundation for analyzing data for project planning and control. We applied ProM[32], a process mining workbench (van der Aalst, 2005) for process evaluation, observation and analysis, i.e., defining the expected process model and evaluating the implemented process workflow based on captured events. Table 2 presents the event description of the implemented change management workflow presented in Figure 19.

Evaluating captured project event data enables project observation and monitoring from project management perspective. Based on discussions with our industry partners, important metrics focus on signal changes over project phases and time. A key assumption of our industry partner was that approximately 20% of signals are changed along the project progress. These assumptions are based on expert estimations. Missing process observation data and loosely coupled non-transparent processes of individual disciplines hinders measurement of the amount of changes in detail. Thus, we evaluated the captured events and derived a set of metrics based on the defined change management process.

6.3.3 Study Objects

This section presents the study objects of the prototype implementation of the change management process based on real world-data from a project at our industry partner, a power plant system integrator.

We used three different signal lists from a real world project at our industry partner and captured occurring events, defined by the change management process. Table 3 presents an overview on used signals per signal list and captured events by the EngSB application. Note that the project is in a very early stage of development, i.e., in the systems design phase, where changes come up frequently. Also note that the signal change handling process does not include the overall project data but a subset of selected and defined components.

Table 3. Source Signal Data from our Industry Partner (Winkler et al., 2011)

	Phase 1.1	Phase 1.2	Phase 1.3
Number of signals	708	720	592
No of captured events	2,834	5,113	2,450

[32] http://prom.win.tue.nl/tools/prom6/

6.4 Project Progress and Risk Monitoring

This section provides an overview, pilot application, and study object of the project progress and risk monitoring method as implementation of the POAF to the automation systems engineering environments.

6.4.1 Overview

Automation Systems Engineering (ASE) project management typically involves heterogeneous engineering fields, e.g., mechanical, electrical, or software engineering, which should work together to reach common goals such as delivering high quality automation systems using an efficient and effective automation systems development process (Schafer & Wehrheim, 2007). However, engineers from different engineering fields typically use their own tools and data models for performing specific tasks within their specific engineering fields. The heterogeneity of data models hinders efficient project progress monitoring and risk management. Hence project managers need an integrated view coping with the semantic heterogeneities of the different involved heterogeneous engineering fields.

For communicating, disseminating, and managing objects across the borders of different engineering fields, engineers typically create and use their own specific engineering workflows (Becker, Lew, & Olsina, 2011; Biffl, Sunindyo, et al., 2009) with limited interaction capabilities between the different fields. An engineering workflow (usually the part of more generic company-wide engineering processes) has its main focus on observation and monitoring of a set of individual engineering steps (within one discipline), instead on business processes that usually provide a comprehensive view on the entire project. The goal of using engineering workflows is to support engineers and project managers with suitable information on the implementation and enactment of processes running in the system. Unlike typical business workflows, engineering workflows not necessary are connected directly to customers (Sunindyo et al., 2011). In addition, current approaches for managing engineering workflows still do not satisfactorily address risk awareness during process analysis. This leads to analysis results which are hard to justify from a business management perspective, e.g., the costs of changes are typically higher if performed at a later stage of an engineering process (Winkler et al., 2011). Current solutions only provide limited capabilities to analyze and present change management process data across disciplines.

With focus on raising the risk awareness of object change management workflows, the key questions for project management and engineers are (a) how changes can be handled more efficient and (b) how relevant change requests can be passed to the involved engineers. Thus, there exists the need for flexible and comprehensive engineering process support across disciplines to enable collaboration and interaction between disciplines, tools, and data models. In the context of addressing risk awareness for engineering workflow validation, the major challenges are (a) different stakeholders of ASE projects, who need to be able to identify and mitigate risks efficiently, and (b) different stakeholders of ASE projects, who need to classify their specific risk factors based on their requirements.

6.4.2 Pilot Application

This section presents the pilot application for project progress and risk monitoring in automation systems engineering. The project progress monitoring is part of project management processes (IEEE, 2011) which can be classified into five sub-processes as shown in Figure 20, namely initiating, planning, executing, controlling, and closing processes. Initiating processes means to authorize a project or phase. Planning processes means to define and refine objectives and to select the best of the alternative courses of action to attain the objectives that the project was undertaken to address. Executing processes means to coordinate people and other resources to carry out the plan. Controlling processes means to ensure that project objectives are met by monitoring and measuring progress regularly to identify variances from plan so that corrective action can be taken when necessary. Closing processes means to formalize the acceptance of a project or phase and to bring it to an orderly end.

Risk monitoring (IEEE, 2011) is the process of keeping track of the identified risks, of monitoring of residual risks and of identifying new risks, of ensuring the execution of risk plans, and of evaluating their effectiveness in risk-reduction. Risk monitoring collects risk metrics that are associated with contingency plans. Risk monitoring is an ongoing process for the life of the project. Risks change as new risks develop, anticipated risks disappear, or the project is getting more mature.

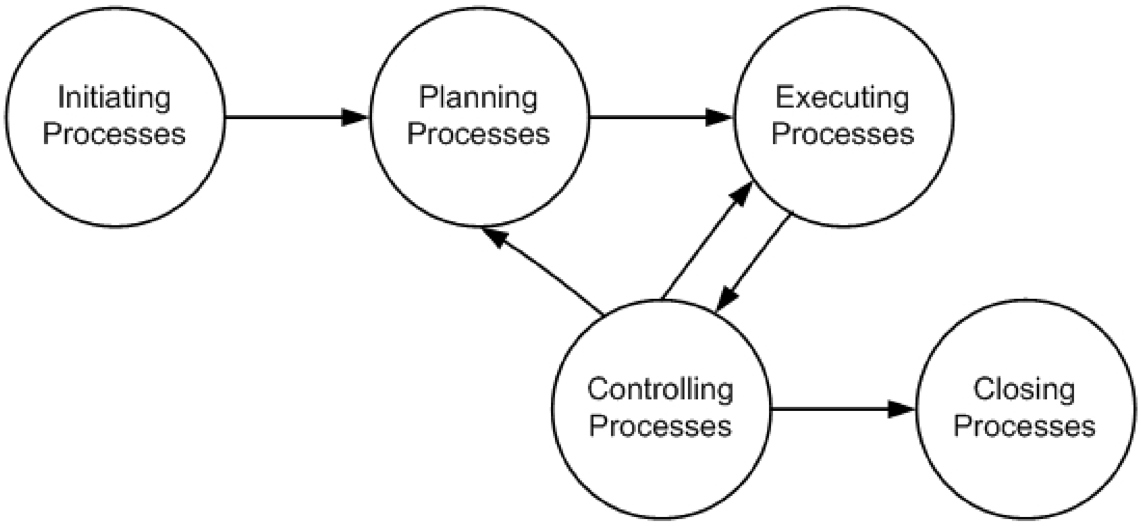


Figure 20. Links Among Process Groups in a Phase (IEEE, 2011)

6.4.2.1 Risk Factors Analysis

The scope of the risk factor analysis framework is to support multidisciplinary engineering teams that add, update and delete signals as well as project managers to support monitoring and decision making process. Each discipline has specific engineering models and tools. These engineering models work well for the specific discipline or expert, but are not well designed for interdisciplinary cooperation. The goal of this framework is to support risk factor analysis across different types of stakeholders, e.g., engineers and project managers, and to fulfill different requirements of different types of stakeholders.

The target audiences of this risk factor analysis framework are two types of stakeholders, namely engineers and project managers. Engineers, e.g., mechanical engineers, electrical engineers, or software engineers, want to effectively and efficiently analyze risk factors of their engineering process in signal change management, e.g., incorrectness or incompleteness of the signal change process. However, often problems of integrity appear due to heterogeneous data models and formats used in those different engineering fields.

Knowledge beneficiaries, such as project managers, want to monitor, control and improve engineering processes such that the processes do not violate risk factors like over budgeting or late project deliveries. This intention is often complicated by the required high effort for performing cross-domain risk factors analyses, e.g., to know which parties should be responsible for over budgeting or project delays.

The major precondition for using the risk factor analysis framework is a working communication link between the engineering tools to be integrated, such as Engineering

Service Bus (Biffl, Schatten, et al., 2009), Enterprise Service Bus (Chappell, 2004), or point-to-point integration.

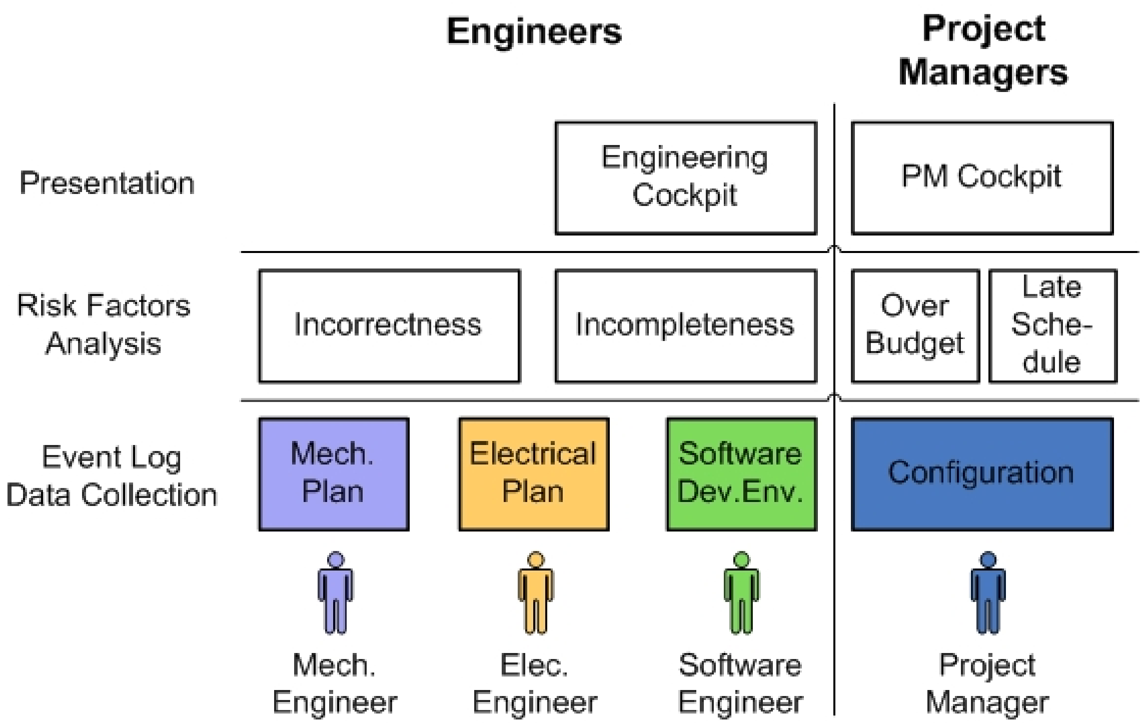


Figure 21. Risk Factor Analysis Framework (Sunindyo et al., 2013)

Figure 21 shows the framework for risk factor analysis which consists of two types of stakeholders, namely engineers and project managers. Engineers, e.g., mechanical engineer, electrical engineer, and software engineer give inputs in the form of event log data which are based on their own development environments (e.g., mechanical plan, electrical plan, and software development environment). The configuration of the event log is set up by project managers.

This event log is useful for further risk factors analysis in the next layer, which is distributed into two parts, namely the engineers' part and the project managers' part. Engineers are more concerned about the correctness and completeness of the signal changes between different engineering fields, and consider the incorrectness and in-completeness of the changed signals as risk factors between the engineers. In contrast, project managers are more concerned about budget and project schedule, such that the risk factors for the project managers are related to over budgeting and project delays. The results of this risk factor analysis are presented in the engineering cockpit to show the risks that should be mitigated by each type of stakeholder.

6.4.2.2 Risk Factors Classification

This section further classifies risk factors based on different stakeholder types. The types of risks classified here are related to the data of specific projects and specific level which may be defined beforehand. The risk analysis can be based on the phases (e.g., timeline of the project) or on the related tools (e.g., EPlan[33], logi.DOC[34]). Source of changes in the project could be an option for risk analysis, e.g., 5 % of changes may originate from external partners, if the number of changes exceeds 5 %, then a new contract needs to be negotiated and thus the budget is affected. Any change by project related tool is considered as an internal change, while any change using the Engineering Object Editor (EOE[35]) (Mordinyi et al., 2011) is considered as external change. The EOE is an Excel Add-on to support efficient quality assurance activities for contributions from external project partners.

Risk analysis could also be done to analyze project values based on the experience of different stakeholders, e.g., to analyze the data of multiple comparable projects or to analyze the number of changes per component. Other types of risk analyses can be based on the number of internal changes per user, or the occurrence of signal changes in late project phases.

From the discussion with our industry partner, we classified a set of risk factors, which often lead to a high effort for analysis and rework during development, commissioning, and deployment. Besides, modifications (e.g., change of a sensor) are critical issues during maintenance because changed sensor attributes at the mechanical site may have an impact on electrical requirements (e.g., wiring) and software requirements (e.g., modified value ranges as data input).

Based on the three important risk groups, i.e., (a) Domain Specific Risks, (b) Collaboration Risks, and (c) Project Management Risks, we focus on collaboration risks and project management risks as they are the most critical aspects in ASE projects to (a) enable efficient collaboration between disciplines and (b) enable a comprehensive view on the project from project management perspective. Thus, we identified a set of risk factors based on (a) the number of signals as project progress indicator, (b) the number of changes, and (c) the periods of engineering object changes (i.e., signals). Figure 22 illustrates the context of the investigated risks.

[33] EPlan Electric: http://www.eplan.de
[34] Logi.DOC: http://www.logicals.com
[35] http://cdl.ifs.tuwien.ac.at/files/CDL-Flex_ASB_UC31_EOE_en.pdf

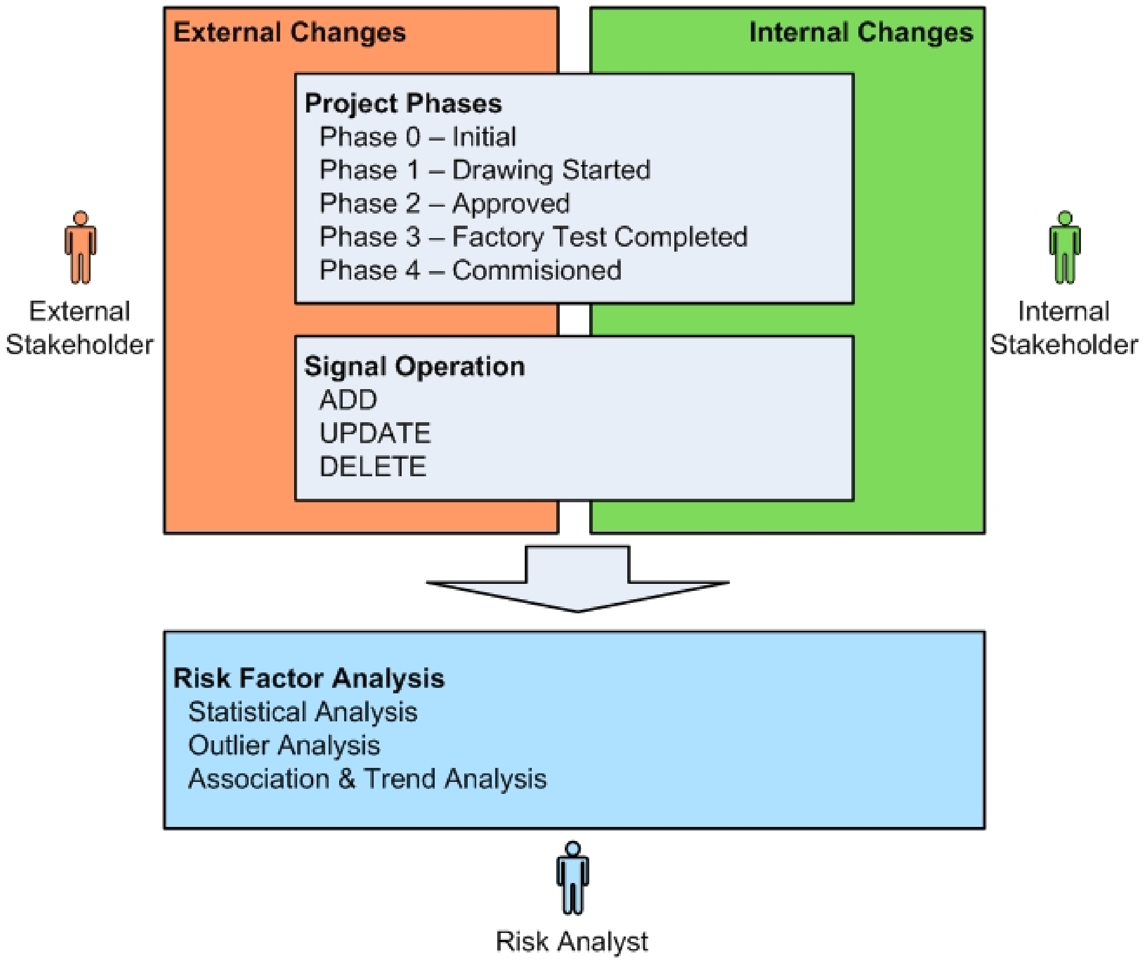


Figure 22. Risk Factors Classification (Sunindyo et al., 2013)

Project Progress Overview. The project progress overview presents the overall number of the signals grouped by engineering phase over time. It illustrates the fluctuation of the number of signals available in a certain phase based on operations applied to (a subset of) signals. If signals are added, it means that the number of signals in a certain phase is increasing. If some signals are deleted, it means that the number of signals in certain phase is decreasing. Updates include two different types of changes: (a) modifications of signal content (non-status updates) and (b) status updates (i.e., upgrading individual signals or groups of signals to the next sequential engineering phase). Signal status updates do not have any impact on the numbers of signals; if a signal has been modified (content change) its status is reset to initial. Based on this setting, we can observe how the signal change operations affect the number of signals available in certain phases, and how the signal updates change signals from an initial phase to a commissioned/final phase. In a healthy project, we could expect a continuous increase of the number of signals (i.e., added signals) and increasing signal status information (i.e., the signals are passing individual phases) over time. On the opposite a decreasing number of signals and the reset of signals from advanced states to initial might indicate risks.

Impact of Stakeholders. Changes might be initiated from different sources, e.g., by internal engineers or externally by the customer. A high number of external changes (even late in the project) might lead to high risks; a high number of internal changes (especially removed and updated signals) might indicate issues in the engineering process of a related discipline. Thus the source of change is an important measure for risk identification. We identify the sources of signal changes to analyze the potential risks, e.g., what's the most frequent signal changes source? What's the trend of signal changes across different types of stakeholders? Or how signal changes can be displayed over time?

Impact on Project Phases. Projects can be divided into several phases, namely initial, drawing started, approved, factory test completed, and commissioned. Risks arise if signals are changed very often, especially late in the engineering project, e.g., a sensor has to be changed during the commissioning phase. Thus the related signals have to be changes as well. In addition, related disciplines might be affected by this change as well. Modification of signals results in resetting the signal state to initial (i.e., the starting phase) and all other phases have to be processed again. Thus, signals assigned to a project phase might be an indicator for risk assessment. As risk, we identify the number of signal changes for each phase across the period of time. From this analysis, we can observe the fluctuation of signal changes across time, depending on the project phase. Some signals can be changed from a phase to next phase, and it is expected that at the end of a project all signals will be in the final phase (commissioned).

Impact of Signals Operations. Operations on signals increase (add new signals) or decrease (deletion of signals) the number of signals available in the project. Signal updates will not change the number of signals but either the signal content or the assignment to a project phase. The update operation itself can be divided further into updates of a signal status or updates without signal status changes, i.e. signal content updates. In this type of risk factor analysis, we observe the relationship between the types of signal change operations over time. The results of this analysis can be used to measure possible risks that could happen during signal changes, e.g., the number of deletion should always less than or equal to the number of available signals.

6.4.3 Study Object

This section presents a multi-disciplinary engineering study objects from an industrial partner developing, creating, and maintaining hydro power plants, and demonstrates a typical process related to the management of signal changes during the life cycle of the power plant. Depending on the size of the commissioned power plant there are about 40 to 80 thousand signals to be managed and administrated in different tools of different engineering disciplines. Signals consist of structured key value pairs created by different hardware components and represent one of the base artifacts in the course of developing power plants. Signals include process interfaces (e.g., wiring and piping), electrical signals (e.g., voltage levels), and software I/O variables. Today's integrated tool suites often consist of a pre-defined set of tools and a homogeneous common data model, which work well in their narrow scope but do not easily extend to other tools in the project outside the tool's scope. Therefore, system integrators in multi-disciplinary engineering projects want to be able to conduct automated change management across all tools that contribute project-level data elements regardless of the origin of the tool and data model.

The current life cycle of a power plant is divided into several phases, each of them reflecting the progress in building the system and the states of the signals. Highly simplified, the following steps are retrieved from the experiences of the industrial partner: (1) First of all engineers start with the requirement & specification phase. In this phase the required data is gathered, such as signals for turbines and generators. It results in the number of sensors, signals and types of sensors. (2) From this data the typology of the system can be created. The output of this step is a number of I/O cards and a network typology. (3) In the next step the circuit diagram is designed. It produces the allocation plan for mechanical resources. (4) Finally the hardware design is finished to be assembled. (5) After this step the Programmable Logic Controller (PLC) software is created to map hardware pin to software pin addresses. (6) Finally the system can be rolled out. In overall these phases are mapped on one of the following signal status: Initial (1), Drawing Started (2, 3), Approved (4) Factory Test Completed (5) and Commissioned (6).

At least there are two different types of stakeholders of the system who are responsible for changes in the power plant system, namely external and internal stakeholders. The external stakeholders, e.g., the customers or the business managers may introduce new requirements or new rules/regulations that affect to the signal changes. The internal stakeholders, e.g., the internal engineers or the project managers also have their own requirements to change the signals in the systems. The previously described process refers to a perfect scenario, whereas in general 25% of all signals change due to changing customer requirements at any point in the life cycle of the development of the power plant. However, the later signals are changed, the more effort has to be invested in coordination with other disciplines and thus the more costs are created. Project managers would welcome monitoring tools allowing them to identify risks and hotspots in the different phases of development. The combination of data sources from different disciplines may provide information about e.g., customer behavior due to the number of change request per project phase, difficult and complex areas in construction due to high number of explicit and implicit changes. A specific type of risk may be related to the source of changes. For example if there are more than 5% of changes from external stakeholders, it triggers an alarm to revise the budget. Hence the project manager should measure the sources of the signals change and calculate the percentage of overall change for risk mitigation.

Figure 23 presents a basic change management process, a signal check-in workflow. The process refers to the fact that collaboration between heterogeneous disciplines and tools requires common concepts for mapping individual models and activities and that system integrators have to synchronize engineering data from those tools.

Signal as a common concept link information across different engineering disciplines. Consequently, management of signals face important challenges like: (a) make signal handling consistent, (b) integrate signals from heterogeneous data models/tools, and (c) manage versions of signal changes across engineering disciplines. The check in workflow supports handling of such challenges by tracking changes on signals and notifying particular engineers.

Note that changes are defined within modified signal lists derived from individual engineering tools to be synchronized with the current overall signal list. Thus, change management refers to the merging process of signal lists provided by engineering tools with signal data known to the Engineering Service Bus. In addition to unchanged signal, changes can include (a) new signals, (b) removed signals, and (c) modified signals regarding its content or status. Signal changes result in a notification of involved stakeholders based on the project environment, e.g., involved stakeholders, related roles, and engineering process phase.

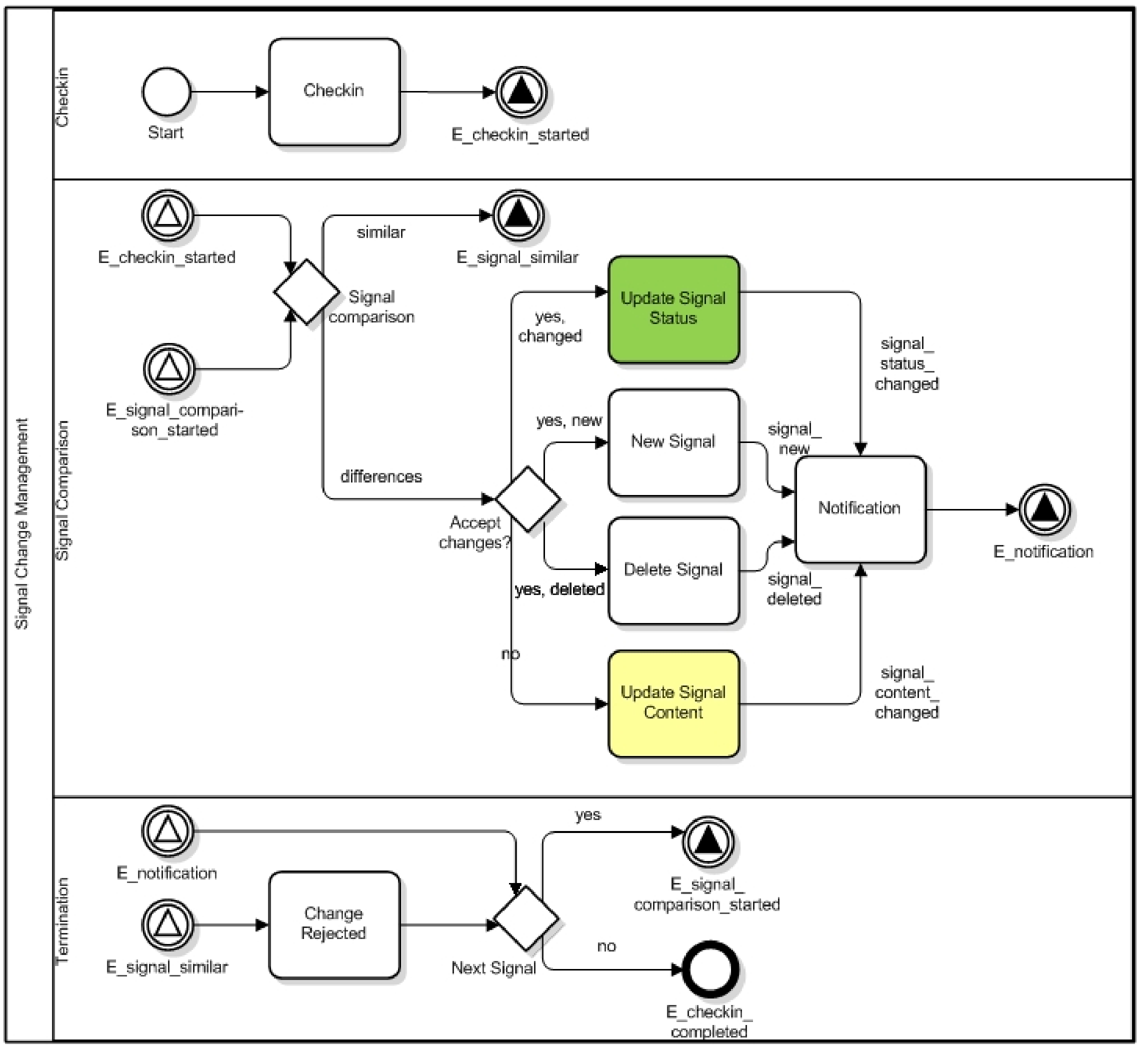


Figure 23. Workflow Model for Signal Change Management (Sunindyo et al., 2013)

6.5 Process Model Validation

This section provides an overview of application of workflow observation and analysis framework to the process model validation of the automation systems engineering environments.

6.5.1 Overview

In heterogeneous engineering environments, like production automation systems, cooperation between different engineering fields, e.g., mechanical, electrical, and software engineering is required to obtain a common goal, e.g., to produce a good quality of product or to deliver a good service to the customers (Biffl & Schatten, 2009). In these systems, the software engineering provides an increasing share of added value to the resulting software-intensive systems and also depends on the seamless collaboration with all other engineering disciplines.

The quality of the processes and products of the production automation systems will always be a concern for the project managers. However, measuring and validating the processes and products of the production automation systems is not an easy task, due to the heterogeneity of engineering fields involved in those systems. The current challenges to measure and validate the processes and products of the production automation systems are as follows. (1) Manual process and product data collection and integration is tedious and error-prone, (2) Testing heterogeneous process components from different engineering fields is hard due to the complexity of tools and data models used by each engineering field, (3) Validating the process models from different levels is difficult, e.g., from business level to production level, due to different interfaces used by those levels.

In this section, we propose to improve a simulation-based process validation (SbPV) (Biffl & Schatten, 2009) with conformance checking analysis to validate production processes on the system level. Currently process validation on system level is done by e.g. checking manually whether the processes are correctly done to produce products required by the business manager. SbPV is focusing more on validating processes rather than testing of products. SbPV is used to simulate the main behavior of the system and test different kinds of parameters to validate the system, e.g., by measuring the number of finished products. Major benefits of SbPV were that (1) simulation allows (automated) process validation on various levels, such as between the business level and the process level and between the machine level and the process level; (2) simulation is the foundation for capturing data on process level with respect to run-time data (Terzic, Merdan, Zoitl, & Hegny, 2008); and (3) when comparing runtime data with simulation data, this can improve the simulation and support runtime diagnosis.

6.5.2 Pilot Application

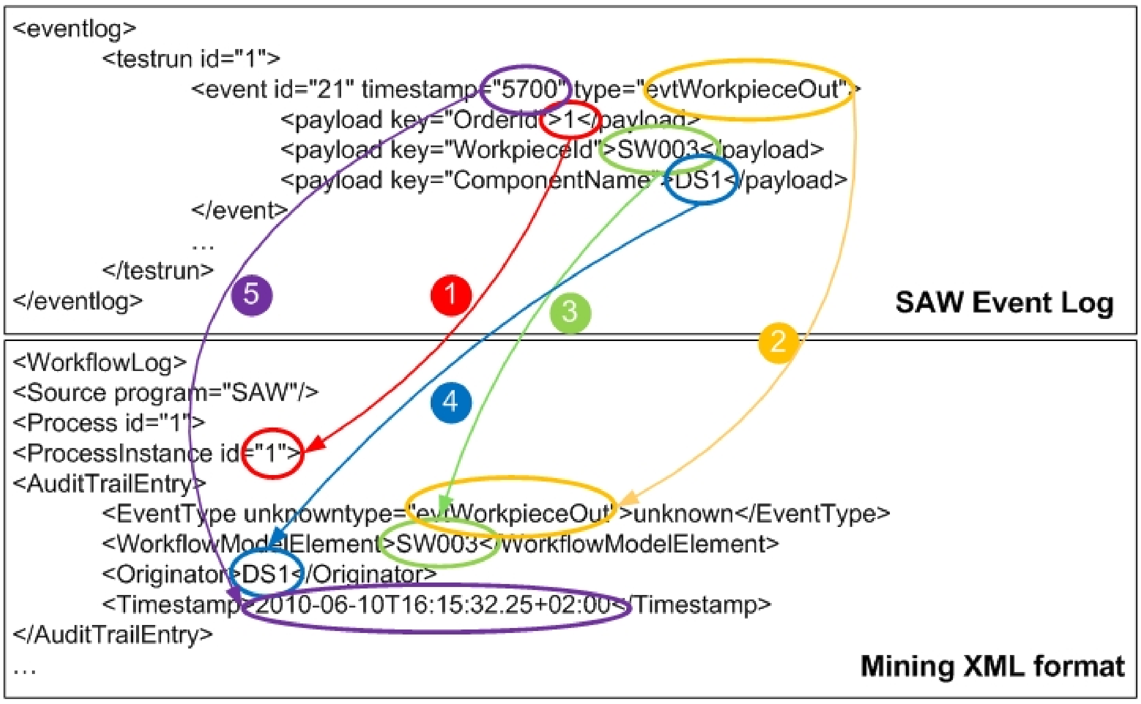
The validation between the business processes and the production processes in the production automation systems is done by collecting and analyzing data from both processes. The business processes data are obtained from the product trees, while the production processes data are obtained from the process event log (see Figure 26).

The process event logs contain information of conveyors and machines activities in the form of XML files, during the running of experiment on the SAW simulator. The files consist of following attributes, i.e., identifier for test run, identifier of event, timestamp, type of event, identifier of order, identifier of work piece, and component name. This information will be validated with the product trees from the business layer by using Process Mining (ProM) tool.

ProM is an open-source tool for implementing process and organizational mining techniques in a standard environment, which allows the extraction of information from event logs. The using of ProM is based on the minimal amount of information that should be present in the general cooperative information systems, e.g., event type, name of event, originator, and timestamp.

The event log should follow these requirements i.e., (1) each logged event should be a single event that occurred at a defined point in time, (2) each logged event should refer to one single activity only, (3) each logged event should contain a description of the event that happened with respect to the activity, (4) each logged event should refer to a specific process instance (case), and (5) each process instance should belong to a specific process. The originator of the event is optional information for the event. This information is useful for advanced analysis, i.e., organizational mining.

The analysis process in ProM needs a special format of input file which is called mining XML. Hence, we have to transform our SAW event logs format into mining XML format. Figure 24 shows the SAW event logs (top) and its transformation (bottom). The transformation is as follows. (1) The OrderId becomes ProcessInstance Id. (2) The type of event becomes EventType. (3) The Workpiece Id becomes Workflow Model Element. (4) The Component Name becomes Originator. (5) The Timestamp is calculated to get the date and time format. The transformed file becomes an input for ProM tool and produces a process model for business process validation.



```
<eventlog>
        <testrun id="1">
                <event id="21" timestamp="5700" type="evtWorkpieceOut">
                        <payload key="OrderId">1</payload>
                        <payload key="WorkpieceId">SW003</payload>
                        <payload key="ComponentName">DS1</payload>
                </event>
                ...
        </testrun>
</eventlog>
```

```
<WorkflowLog>
<Source program="SAW"/>
<Process id="1">
<ProcessInstance id="1">
<AuditTrailEntry>
        <EventType unknowntype="evtWorkpieceOut">unknown</EventType>
        <WorkflowModelElement>SW003</WorkflowModelElement>
        <Originator>DS1</Originator>
        <Timestamp>2010-06-10T16:15:32.25+02:00</Timestamp>
</AuditTrailEntry>
...
```

Figure 24. Structure and Transformation of SAW event log (Sunindyo, Moser, & Winkler, 2012)

6.5.3 Study Objects

This section presents the study objects of the prototype implementation of the process model validation. For evaluating the business goal achievement in the production automation systems, we use the SAW simulator (Merdan, Moser, Wahyudin, & Biffl, 2008) as our use case of the production automation systems.

The SAW simulator consists of heterogeneous agent-controlled components that simulate the components and behaviors of the real assembly workshop layout. It consists of 40 conveyors, 17 junctions, more than 15 pallets, 3 product parts and storage areas, 6 machines, and 4 robots. By using the SAW simulator, we can set the relevant parameters of simulation, e.g., failure classes, scheduling strategy, and number of pallets, to accommodate the different risks and situations that could be faced in the real assembly workshop.

The input of the SAW simulator is a set of test cases. During production process simulation, the SAW simulator produces process events, e.g., starting events, finishing events, and other relevant events which are stored in the process event log.

For test cases on the business goal validation, we design experiments on the simulator by setting several parameters on the business orders fed into the simulator. The business orders consist of 1,500 products with two fictional types of products, named Billy Medium and Billy Complex. The illustration of Billy Medium and Billy Complex product trees can be seen in Figure 25.

Billy Medium consists of one simple part and one intermediate part. Medium int part 1 and medium int part 2 are combined by machine function M2 and become medium int 1. By using machine function M4, the medium int 1 together with medium part 1 will build the Billy Medium product.

Billy Complex consists of two intermediate parts. SW003 and DP003 together will build K003 via machine function M3. F002 and F003 together will build P003 by using machine function M3 as well. By using machine function M5, K003 and P003 will build the Billy Complex. We evaluate 40 test cases of business goals, by comparing the results of simulating 4 different classes of failures.

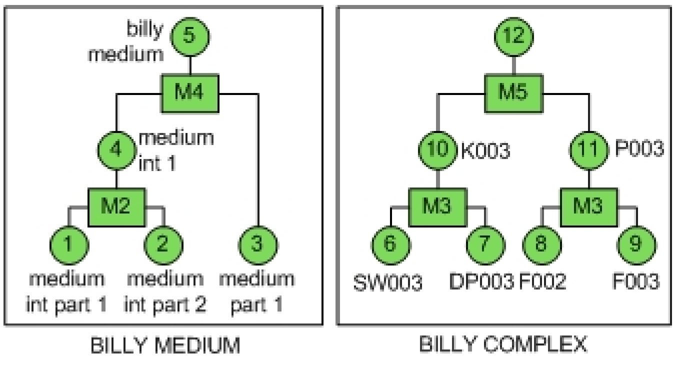


Figure 25. Product Trees of Billy Medium and Billy Complex (Sunindyo, Moser, & Winkler, 2012)

The classification of classes of failures is based on the risk of machine failures and/or conveyor failures in the simulation workshop, according to the position and the importance of the machine and conveyor for the overall system (refer to Table 4). For effective comparison of the robustness of workflow scheduling strategies regarding their exposure to failures in the transportation system, we used First Come First Served (FCFS) strategy, which execute the first allocated task first.

The explanation of test cases with different class of failures is as follows. (1) C0 consists of test cases with no failure. (2) C1 consists of test cases with 5 conveyor failures in each test case. (3) C2 consists of test cases with 2 machine failures in each test case. (4) C3 consists of test cases with combination of 5 conveyor failures and 2 machine failures in each test case. The machine failures and the conveyor failures occur randomly in the test cases.

Table 4. Failure Classes and Risk Analysis (Sunindyo, Moser, & Winkler, 2012)

Classes of Failure	Failure Impact
C0	No Failure
C1	Conveyor Failures
C2	Machine Failures
C3	Combined Conveyor and Machine Failures

We propose a framework to evaluate the business goal achievement in the SAW simulator. The framework consists of three layers, namely business layer, process layer and machine layer, which is illustrated in Figure 26.

In the business layer, the business manager dispatches and schedules business orders, which contain specification of products in the form of product trees and other required information, e.g., the number of products and due date of production process.

In the process layer, the workshop configurator configures the layout of simulation of assembly workshop (SAW) (Merdan et al., 2008) that supports the business manager's requirements. The workshop operator runs the simulation based on the business orders. Each process event during the production process is stored in the process event log for further purposes, e.g., process analysis by quality manager. In the middle of process layer, there is a schematic view of an assembly workshop.

In the machine layer, the system engineer provides the real workshop systems that interact with real machines which are provided by the machine vendor. In this thesis, we don't discuss further about the machine layer and put focus on the interaction between the business layer and the process layer.

The results of the simulation can be used as input for real assembly workshop in the machine layer. The usage of a software simulator here is needed to accommodate the reconfiguration of the production automation system in order to get a better performance. Validation on hardware test bed is expensive, hence we build software simulator with agent-controlled components that imitate behaviors of real components in the real system.

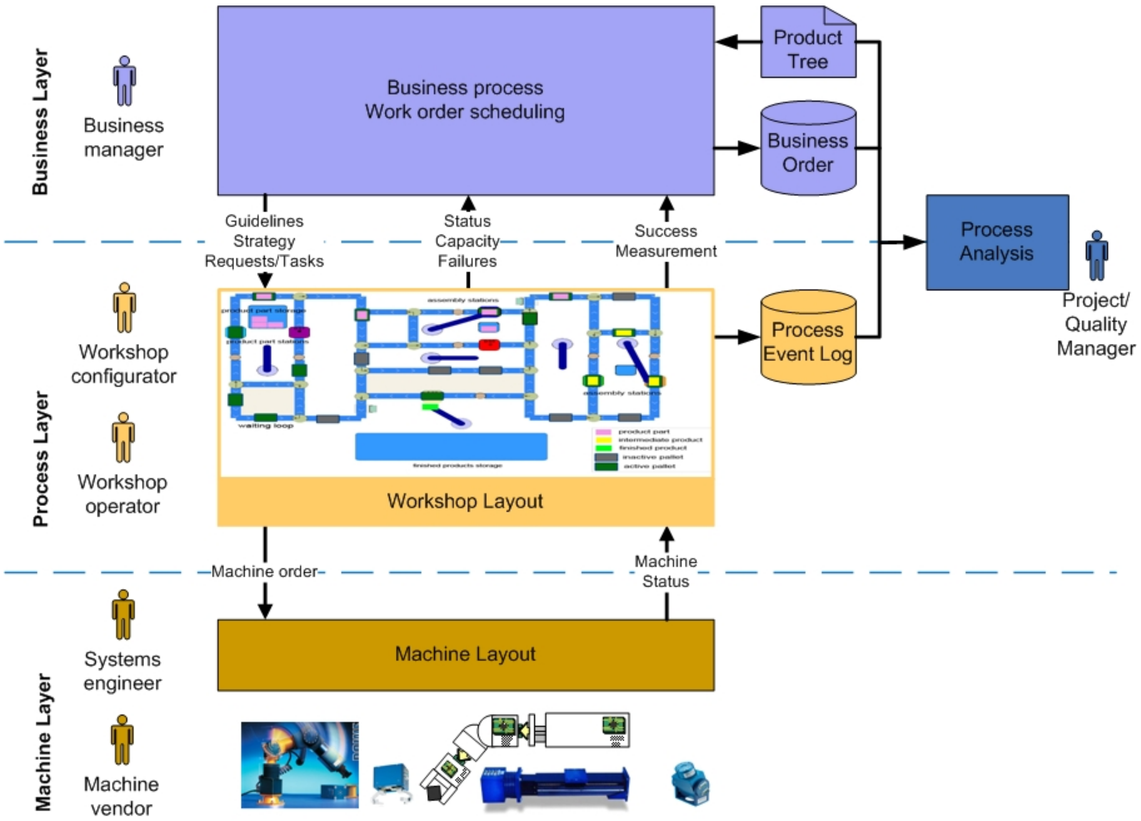


Figure 26. Business Goal Evaluation Framework for Production Automation Systems (Sunindyo, Moser, & Winkler, 2012)

6.6 Summary

In this chapter we proposed and evaluated the POAF to observe and analyze engineering process in Automation Systems Engineering environments, to provide an easier way for the project management to monitor and manage the ASE projects based on the immediate status of OSS projects.

We also provide a workflow validation cycle as an instantiation of POAF to support signal change management process observation and analysis in ASE environments, especially in the power plant as our industrial partner.

The current project management technologies focus on the using of single and specific tool to analyze and mine the process data, like ProM. However, the POAF provides more integrated approach and view on managing heterogeneous data from different stakeholders in ASE project, before submitting data for analysis. The POAF also provides the possibility of different stakeholders to get different kinds of analysis result presentations based on their requirements and goals.

The following sub-sections describe the findings and results for the POAF process description, for change management process observation and analysis method, for project progress and risk monitoring method and for process model validation method.

6.6.1 Workflow Validation Cycle Process

In this section, we introduced and evaluated the workflow validation cycle as an instantiation of the POAF for ASE project context.

The workflow validation cycle consists of nine process steps, namely informal model description, formal model description, rule engine transformation, EngSB transformation, workflow results, event logs, process analysis, evaluation of process metrics, and conformance checking.

Major results show that the framework can support the project manager in observing and validating workflows more efficiently compared to the traditional mainly manual approaches (Sunindyo, 2011).

6.6.2 Change Management Process Observation and Analysis

In this section, we explained the change management process observation and analysis method as implementation of the workflow validation cycle to the automation systems engineering domain.

A pilot application is proposed to introduce “signal engineering” as a common concept in the automation systems domain, to explain a change management process for signal changes at a power plant systems integrator, and to explain concepts for process evaluation and analysis.

As study objects, we used three different signal lists from a real world project at our industry partner (Winkler et al., 2011).

6.6.3 Project Progress and Risk Monitoring

In this section, we explained the project progress and risk monitoring in the power plant as implementation of POAF to the automation systems engineering domain.

A pilot application is proposed to monitor project progress and risk in the ASE domain. The project progress monitoring is part of project management processes which can be classified into five sub-processes, namely initiating, planning, executing, controlling, and closing processes.

We classified the risk based on the project progress overview, the impact of the stakeholders, the impact on the project phases, and the impact of signals operations. As study object, we used signals changes during the life cycle of the power plants. Signals here could represent process interfaces (e.g., wiring and piping), electrical signals (e.g., voltage levels), or software I/O variables (Sunindyo et al., 2013).

6.6.4 Process Model Validation

In this section, we explained the process model validation in the production automation systems as implementation of POAF to the automation systems engineering domain.

A pilot application is proposed to validate between the business processes and the production processes of the production automation systems by collecting and analyzing data from both processes.

As study object, we use the business orders of 1,500 products of two types of products, namely Billy Medium and Billy Complex to simulate 4 different classes of failures, namely no failure, conveyor failures, machine failures, and combined conveyor and machine failures (Sunindyo, Moser, & Winkler, 2012).

7 Evaluation and Discussion

This chapter explains the evaluation results and their discussion based on the research issues and the application scenarios in previous chapter (see section 3.1). The first section describes the empirical evaluation on two application scenarios, namely OSS and ASE application scenarios. Evaluation results from different analysis methods are explained, namely from health indicators analysis method, bug history analysis method, workflow validation analysis method, change management process observation and analysis, project progress and risk monitoring, and process model validation. In the second section, the evaluation results are discussed according to the specified research issues and the analysis methods used in different application scenarios.

7.1 Evaluation

This section describes the evaluation results for different analysis methods used to analyze OSS and ASE application scenarios. The explanation of each analysis method can be found on chapter 5 and 6.

7.1.1 OSS Project Monitoring

The evaluation on OSS application scenarios is based on three different analysis methods, namely health indicators analysis method, bug history analysis method, and workflow validation analysis method.

Health Indicators Analysis Method

The criteria of analyzing OSS projects using health indicators analysis method are the efficiency of data collection and the accuracy of health indicators. The efficiency of OSS projects data collection is measured by comparing between the manual and the automated approach to collect and process heterogeneous data for deriving OSS health indicators. The accuracy of health indicators is measured by comparing the results from different indicators with the experts' opinion.

Efficiency of Data Collection

The efficiency of data collection is measured by comparing the efforts to collect and process heterogeneous data for deriving OSS health indicators (see Table 5). Figure 27 illustrates the comparison of efforts between two approaches, namely the traditional manual approach and the proposed (semi-)automated approach with FOSSDA.

The traditional manual approach follows three steps for data collection and analysis (see Figure 27 upper part).

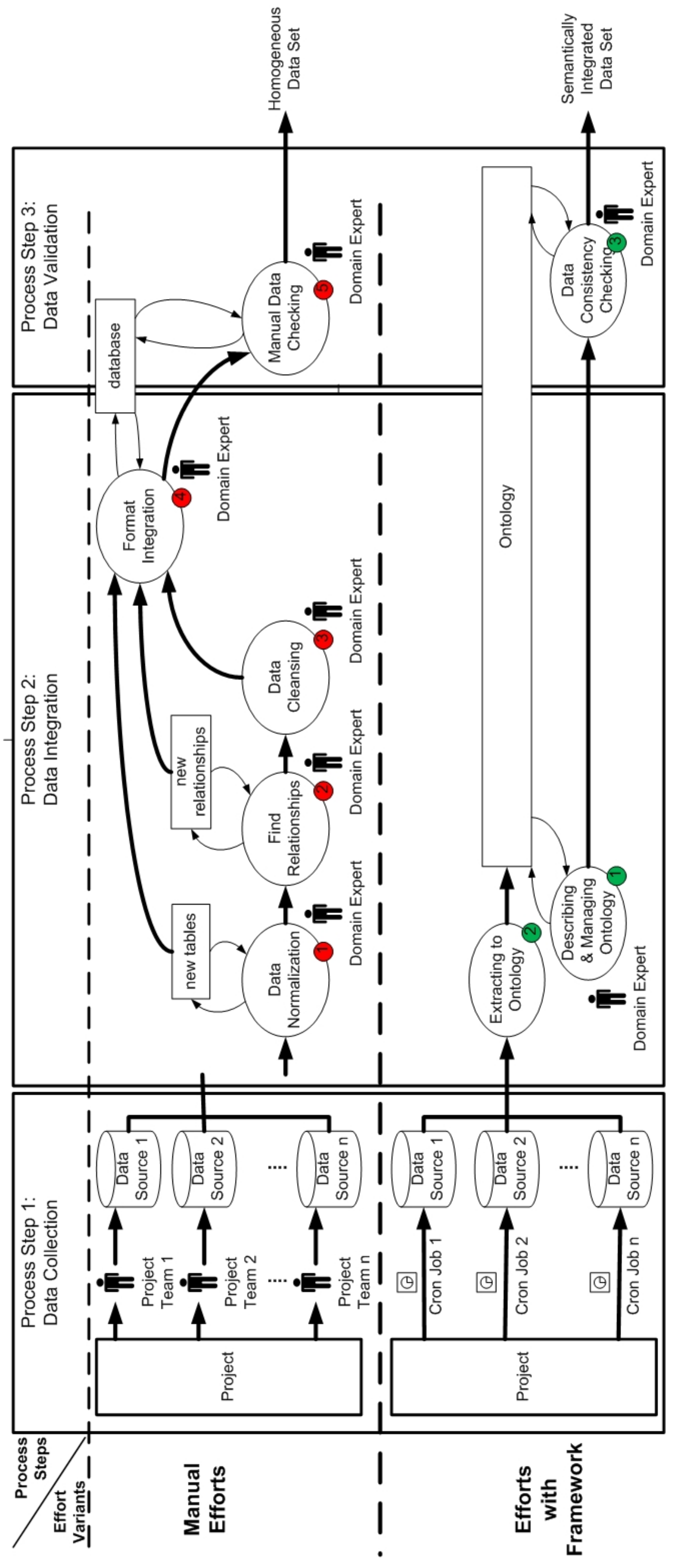


Figure 27. Comparison of efficiency for manual and FOSSDA data collection process variants (Sunindyo, Moser, Winkler, et al., 2012)

(a) Manual data collection. The researcher chooses the projects to analyze, and then collects data directly from the project tools, e.g., by downloading the repository content from SVN, postings from developers' mailing list website, and bug reports from the *Bugzilla* website.

(b) Manual data integration. The data from different data sources are integrated manually. The researcher has to analyze each data source and find out the structure of each data source and similarities between different data sources. This work is hard to do manually, because there are hundreds or even thousands types of data points, which may have possible relationships. The possibility to find relationships manually between different data sources is easier to be based on time aggregation (e.g., weekly, monthly, or annually), rather based on other detail information (e.g., name of authors, email address).

(c) Manual data validation. Manual data validation is hard to do. One possible effort is to manually query the database to check the validation of data, for example, make query on the name of developers and the name of mailing list authors and find out their relationships.

Table 5. Comparison of Effort for Manual and Automated Process Variants (in work-hours) (Sunindyo, Moser, Winkler, et al., 2012)

Effort Variants	Process Steps	Process Items	Observed Projects			
			Lenya	Log4J	Excalibur	OJB
Traditional Manual Process Effort	Data Collection	Data coll. SVN	1.4	0.7	0.4	0.2
		Data coll. mailing	1.0	0.5	0.3	0.2
		Data coll. bug	0.7	0.4	0.2	0.1
	Data Integration	Normalize data	1.0	0.6	0.3	0.2
		Identify	4.0	3.0	3.0	3.0
		Clean data	4.0	3.0	3.0	3.0
		Integrate format	3.0	3.0	3.0	3.0
	Data	Manual data	6.0	6.0	6.0	6.0
	TOTAL		**21.1**	**17.2**	**16.2**	**15.7**
Tool-Supported Process Effort	Data	Run data	0.1	0.1	0.1	0.1
	Data Integration	Describe	2.0	2.0	2.0	2.0
		Manage ontology	1.0	1.0	1.0	1.0
		Extract data to	0.8	0.8	0.5	0.5
	Data	Data Consistency	3.0	3.0	3.0	3.0
	TOTAL		**6.9**	**6.9**	**6.6**	**6.6**

The proposed (semi-)automated approach with FOSSDA uses specific tools to automate project data collection and analysis (see Figure 27 lower part).

(a) Tool-supported data collection. The *Project Data Fetcher* tool supports the automated collection of data from different data sources, i.e., SVN, mailing list, and bug report. The input of this tool is the configuration setting of each data source tool (i.e., SVN, mailing list, bug report) to specify scope and range of project data we want to analyze. The example configuration setting includes information of repository URL for SVN, start revision for SVN, archive URL for mailing list, starting data for mailing list, Bug List URL for Bug Report, and Starting Date for Bug Report. The configuration setting does not involve ontology setting, since the ontology setting is done automatically in the tool. This approach is suitable for normal developers and not necessarily a job for an ontology expert. The processes for data collection are defined as follows: (a) choose a project to analyze; (b) set the configurations of SVN, mailing list, and bug report, e.g., the revision number (for SVN) and the starting date of the data (mailing list and bug report); (c) run the *project data fetcher* to collect the required data automatically.

(b) Tool-supported data integration. The steps of data integration are as follows: (a) inspect the structure of collected data, (b) find similarities of data attributes between different data models, e.g., name of committers in SVN and name of authors in mailing list, (c) map each data source to the ontology as data storage, (d) set the relationships between different data sources. The input to these steps is the collected data, including their structure. The output is the integrated data in the ontology.

(c) Tool-supported data validation. The steps of the data validation are as follows: (a) take integrated data in the ontology as input, (b) use rules and restrictions to check the relationships and constraints of the data, (c) use queries to the ontology to check the consistency and integrity of the data, (d) check the data with intended data model to support the data analysis. The input to these steps is the integrated data as an ontology, including rules and relationships. The output is validated data ready for analysis.

For analyzing the data and presenting the results, we used the Project Monitoring Cockpit (ProMonCo) (Biffl et al., 2010b) to display the health indicators to project and quality managers. This automated tool can be combined with the Project Data Fetcher and provides synergies of the data collection and analysis to support efficient health indicator analysis with FOSSDA for project supervisors. The analysis steps are as follows (a) handling project ontology as an input for ProMonCo, (b) processing this ontology and analyze into different purposes and criteria, e.g., total communication metrics, user coupling metrics, bug history metrics, (c) presenting the analysis results to the project managers. Table 5 reports the effort needed to complete the process steps using manual and tool-supported process approaches. Using the automated approach allows reducing % of the effort needed for operating traditional approach by up to 30%.

Accuracy of Health Indicators

This section provides results from the health indicator analysis study with two indicators, namely bug delays and proportion of activities for four OSS projects: Lenya, Log4J, Excalibur and OJB.

Indicator 1: Bug Delays. Figure 28 shows the percentage of bug delays between active projects (*Lenya, Log4J*) and inactive projects (*Excalibur, OJB*).

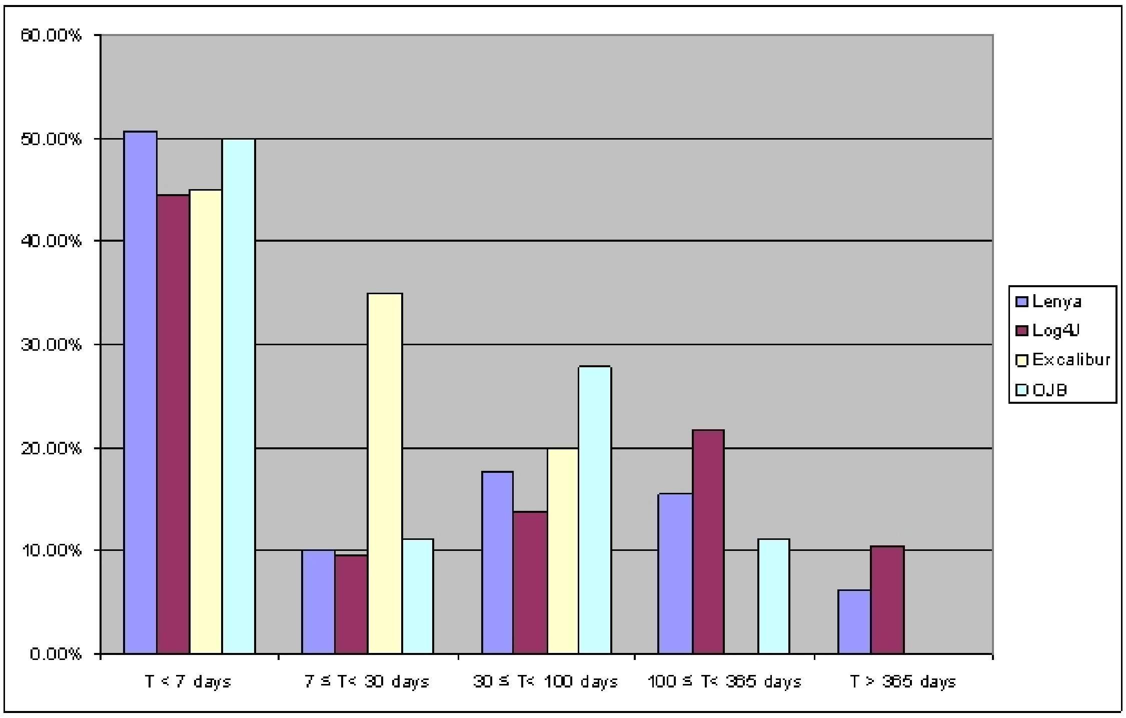


Figure 28. Indicator 1: Bug fixing delays in four OSS projects (Sunindyo, Moser, Winkler, et al., 2012)

Most bugs get solved in less than 7 days (*Lenya*: 51 %, *Log4J*: 45 %, *Excalibur*: 45 % and *OJB*: 50 %). Further, 55% of bugs in *Excalibur* and 39 % of bugs in *OJB* can be solved in less than 100 days. The conclusion of OSS health status is only based on these indicators and thus limited. The results cannot identify the current status of the OSS project.

Indicator 2: Proportion of Activities. Figure 29 shows the proportion of activities between the numbers of bug reports per number of mailing list postings each month from January 1st, 2007 until December 31st, 2010 for Lenya, Log4J, Excalibur and OJB projects. It is assumed that in a healthy community a project should exhibit more uniform ratio among process metrics, i.e., every bug report ideally should be followed up by the developer discussion in the mailing list. We found that the inactive projects (Excalibur and OJB) show more fluctuation and higher ratios between the number of mailing list postings and bug reports as illustrated in Figure 29. It means, the developer community retrieved more notification related to bug reports but responded less in the mailing list.

This situation may indicate illness symptoms, e.g., developers pay less attention to the project status changes and the project employs a small proportion of active developers which also signifies discouragement of developers, which need to be investigated further with OSS experts.

On the other side, the Lenya and Log4J projects show more reasonable proportions in the ratio of developer contribution. This can be interpreted as the fact that most of the changes of bug status may trigger some responses from the developers.

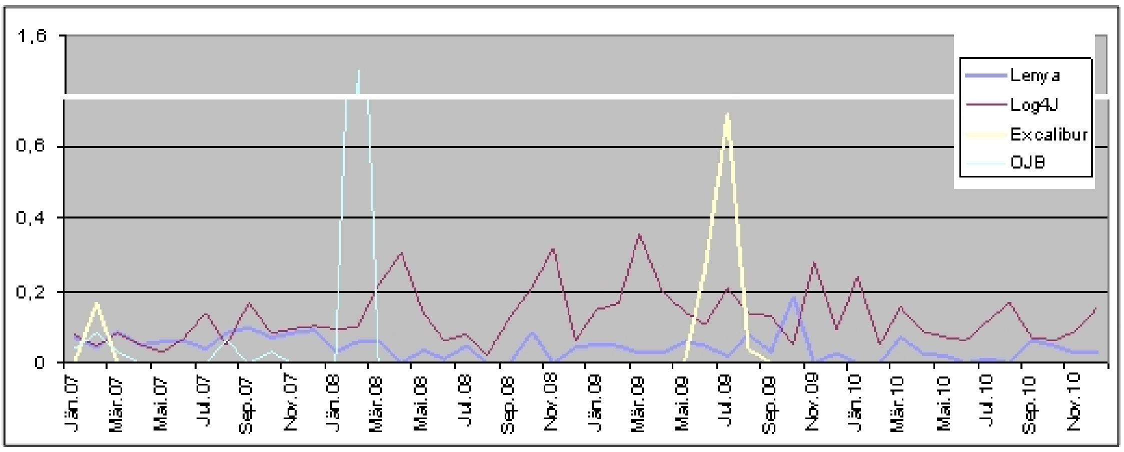


Figure 29. Indicator 2: Proportion between the number of Mailing List Postings and Bug Reports (Sunindyo, Moser, Winkler, et al., 2012)

The accuracy of health indicator analysis depends on many factors, e.g., the number of data sources and the type of methods used. To measure the accuracy of health indicators across different projects, we compare the results of different analysis methods and data sources with the opinion of OSS experts on assessing the OSS project health status. In the context of this study we handle the OSS expert opinion as “truth” to allow assessing the quality of the results of the proposed on the health indicator approach. The experts were selected from Apache Project Management Committee (PMC) or the Apache

Software Foundation (ASF). There are 10 experts selected from the projects and the foundation (4 from Lenya, 2 from Log4J, 1 from Excalibur, 1 from OJB and 2 from ASF). Their opinion was taken from observations on the OSS project development activities which are lead by the PMC. The experts decide their judgment on the status of the project based on their experiences and comparison of different project activities and their opinion was consistent during this study.

Table 6 shows the comparison between the results of the health indicator analysis and OSS expert opinion to assess the status of OSS project health. The results of using combined project metrics as health indicators (bug delays and proportion of activities) shows that the use of a single project metric (namely bug delays) provided in the empirical study context overoptimistic assessments of the OSS project health status compared to the expert opinion, which is assumed to be correct as the expert opinion uses a much wider range of input for their status assessment.

Table 6. Comparison between Health Indicators Analysis Results and OSS Expert Opinion (Sunindyo, Moser, Winkler, et al., 2012)

Indicators Item	Project	Indicators Result	Expert's Opinion
Bug Delays (single project metric)	Lenya	Healthy	Healthy
	Log4J	Healthy	Healthy
	Excalibur	**Healthy**	**Unhealthy**
	OJB	**Healthy**	**Unhealthy**
Proportion of Activities (combined project metrics)	Lenya	Healthy	Healthy
	Log4J	Healthy	Healthy
	Excalibur	Unhealthy	Unhealthy
	OJB	Unhealthy	Unhealthy

The use of multi-data sources and combination of different health indicators makes matched the expert opinion for all investigated projects and can, therefore, be considered as more accurate than using a single data source and a single health indicator analysis method in the context of this study.

Bug History Analysis Method

The evaluation is done by analyzing the bug history data sets from three different RHEL versions (4, 5 and 6) by using Heuristics Mining algorithm from Process Mining (ProM) tool. We analyzed the number of states in the process models generated by ProM and counted the frequency of each state for each RHEL version. We compare the results with the designed process model from Bugzilla life cycle.

The RHEL developers are following the naming and the ordering of the bug states from Bugzilla life cycle. Table 7 shows the comparison of bug states used in the

Bugzilla life cycle, RHEL 4, RHEL 5, and RHEL 6. From Table 7 we can see different bug state names are used during addressing bug in different RHEL versions.

This result shows us that the developers don't really follow the process model in the Bugzilla Life Cycle. The Bugzilla Life Cycle is build to give guidance for the developers in handling the bug issues in the project. However, in the implementation, the developers have capability to introduce and modify new bug states as long as this is mutually agreed among the developers.

Table 7. Name of States used in different RHEL versions and Bugzilla Life Cycles (Sunindyo, Moser, Dhungana, et al., 2012)

States	Bugzilla LC	RHEL 4	RHEL 5	RHEL 6
UNCONFIRMED	✓	×	×	×
NEW	✓	✓	✓	✓
ASSIGNED	✓	✓	✓	✓
RESOLVED	✓	✓	×	×
VERIFIED	✓	✓	✓	✓
REOPENED	✓	✓	✓	✓
CLOSED	✓	✓	✓	✓
NEEDINFO	×	✓	✓	✓
MODIFIED	×	✓	✓	✓
ON_QA	×	✓	✓	✓
RELEASE_PENDING	×	✓	✓	×
QA_READY	×	✓	×	×
NEEDINFO_REPORTER	×	✓	✓	✓
INVESTIGATE	×	✓	✓	✓
NEEDINFO_PM	×	✓	×	×
PROD_READY	×	✓	×	×
FAILS_QA	×	✓	✓	×
PASSES_QA	×	✓	×	×
NEEDINFO_ENG	×	✓	✓	×
ASSIGN_TO_PM	×	✓	✓	✓
ON_DEV	×	✓	×	✓
SPEC	×	✓	×	×
POST	×	×	✓	✓
# States	**7**	**21**	**15**	**13**

The using of too many different states in the Bugzilla sometimes is confusing and makes a lot of confusion among the developers. Some intervention should be taken to make common understanding about the meaning of the states and when it should be used to prevent ambiguity and duplication of similar states. From this result, we can see how the using of bug states evolves in different RHEL versions that bring more common understanding between the developers about the using of states in handling the bug issues.

From Table 7, we can see that there are differences between the names and orders of bug states from RHEL different versions and those from Bugzilla Life Cycle. The names and orders of bug states in the Bugzilla Life Cycle are focusing on main states of the bugs and minimal requirements of the bugs with assumption that the bug information and the bug states are self-explained. However, in the reality, not all bug information and bug states are understandable by other developers. Thus, some states are created to ask for further explanation, e.g., needinfo, needinfo_reporter, needinfo_PM, and needinfo_eng. Thus four new states represent a need for further information from other parties, e.g., reporter, project manager, or engineer.

Other new states also related with QA (quality assurance), e.g., on_QA, QA_ready, fails_QA, and passes_QA, which mean that the quality assurance become a part of improvement in dealing with the bug. On_QA means that the bugs resolving is still on quality assurance. QA_ready means the bugs are ready to enter the quality assurance phase. Fails_QA means that the bugs are failed in quality assurance testing. Passes_QA means that the bugs have passed the quality assurance testing. However, these QA-related states are introduced in RHEL version 4 and not continued in version 5 and 6, means that the QA-related states are not really useful in dealing with the bugs.

Other new states are including modified, investigate, release_pending, prod_ready, on_dev, spec, and post. These states are more specific to some conditions, e.g., modified means that the bugs are still modified, investigate means that the developers need more investigation on the bugs, release_pending asks for pending of the product release, prod_ready means that the product is ready, on_dev means that the product is on development, spec asks for specification, and post means that the product has been posted.

From this result, we can see that by analyzing the different states using bug historical data, we can learn how the developers using the bug states to communicate the idea how to deal with the bugs. The change of using bug state names in different versions also show the importance of the bug state names in handling the bugs, some names are remain, but some others are not used anymore.

The RHEL developers are using all bugs states for each bug history in the same number of frequency. Table 8 shows the frequencies of the using of each bug state in the bug history. From Table 8, we can see that the frequencies for different bugs in one

RHEL version are not similar. The usage of some states is more frequent then of other states.

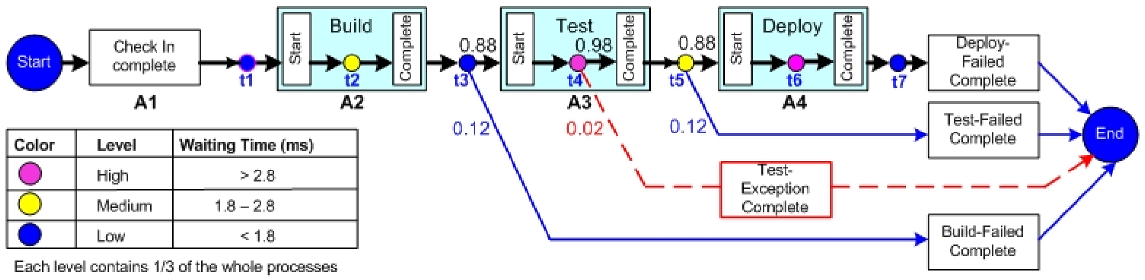
By seeing this result, we can learn how the developers deal with the bug issues by using Bugzilla. The developers seem to uses some state rather than the other states. The "closed" state is on the top of each version means that (1) whatever state as starting state, all states tend to go to closed states, (2) there are some possibility to reopen the closed bug and close it again (especially in RHEL version 4). From this result, we can see the priority of the developers, in managing bug states and suggest for improvement, e.g., reducing the number of bug states to make the development/bug handling more efficient.

Table 8. Frequency of States for Different Versions of RHEL (Sunindyo, Moser, Dhungana, et al., 2012)

States	Version 4		Version 5		Version 6	
	Occ. (abs)	Occ. (rel)	Occ. (abs)	Occ. (rel)	Occ. (abs)	Occ. (rel)
CLOSED	**557**	**41.0 %**	**578**	**30.3 %**	**546**	**33.6 %**
ASSIGNED	**282**	**20.8 %**	**407**	**21.3 %**	**255**	**15.7 %**
NEEDINFO	**150**	**11.1 %**	**204**	**10.7 %**	6	0.37 %
MODIFIED	**126**	**9.3 %**	**345**	**18.1 %**	**306**	**18.9 %**
REOPENED	**52**	**3.8 %**	8	0.4 %	1	0.1 %
RESOLVED	47	3.5 %				
ON_QA	29	2.1 %	67	3.5 %	**259**	**16.0 %**
RELEASE_PENDING	25	1.8 %	61	3.2 %		
QA_READY	25	1.8 %				
NEW	16	1.2 %	44	2.3 %	7	0.4 %
NEEDINFO_REPORTER	11	0.8 %	14	0.7 %	2	0.1 %
INVESTIGATE	10	0.7 %	2	0.1 %	2	0.1 %
VERIFIED	9	0.7 %	**122**	**6.4 %**	**199**	**12.3 %**
NEEDINFO_PM	3	0.2 %				
PROD_READY	3	0.2 %				
FAILS_QA	3	0.2 %	10	0.5 %		
PASSES_QA	2	0.1 %				
NEEDINFO_ENG	2	0.1 %	1	0.1 %		
ASSIGN_TO_PM	2	0.1 %	2	0.1 %	1	0.1 %
ON_DEV	2	0.1 %			8	0.5 %
SPEC	1	0.1 %				
POST			43	2.3 %	31	1.9 %

Workflow Validation Analysis Method

The process model verification is done by comparing the real process with respect to the expected designed process. Figure 30 shows the real process model based on the event logs that were fed into the ProM tool and processed by using the process modeling plug-in. By using the process performance analysis plug-in of ProM, we can assess the Key Performance Indicators (KPIs) in an intuitive way. In this thesis, we focus on the relevant performance information of the ProM plug-in: the place time metrics and activity metrics.



Color	Level	Waiting Time (ms)
●	High	> 2.8
●	Medium	1.8 – 2.8
●	Low	< 1.8

Each level contains 1/3 of the whole processes

Figure 30. Performance Analysis of 300+ CI&T process runs (Sunindyo et al., 2010a)

The transition time metric, based on the Petri Net model, is illustrated in Table 9. It consists of the statistical values of the waiting time (average, minimum, and maximum in milliseconds) that passes from the (full) enabling of a transition between processes until its firing, i.e., time that tokens spend in the place waiting for a transition (to which the place is an input place) to fire and consume the tokens. Based on this metric, we can identify bottlenecks of the system. A bottleneck in a process is a transition with a high average waiting time. In our analysis three levels of waiting time are distinguished: low, medium and high waiting time (in our case low, medium, and high thirds of waiting time observed over several instances of the process). After calculation, the used Petri net will be visualized with colors according to their waiting time level, pink for high, yellow for medium and blue for low. The use of auto bottleneck settings will estimate values for upper bounds, each level will contain approximately one third of the state transitions. If project manager does not have prior knowledge about threshold for bottleneck, this auto setting can help him to know the situation and then reapply performance analysis with better bottleneck estimation setting.

Table 9. Transition Time Metric for CI&T process steps and overall process (Sunindyo et al., 2010a)

No	Transition	Avg (ms)	Min (ms)	Max (ms)
t1	Check In	1,806	1,442	4,154
t2	Build Start	2,808	2,176	13,524
t3	Build Complete	1,785	1,425	2,941
t4	Test Start	3,473	2,778	13,022
t5	Test Complete	2,243	1,393	7,536
t6	Deploy Start	3,131	2,211	18,380
t7	Deploy Complete	0	0	0
	CI&T Process	**15,246**	**11,425**	**59,557**

Figure 30 shows the high waiting time (above 2.808 ms) activities are during the test (between start and complete events) and during the deployment (between start and complete events). This bottleneck information can be used as input for the project manager for further process improvement. The activity metric is illustrated in Table 10. It consists of waiting time, execution time, and sojourn time. Waiting time is defined as the time between the moment the activity is scheduled and the moment at which execution of the activity is started. Execution time is the time in which the activity is actually executed. Sojourn time is the time between the scheduling the activity and finishing its execution, the sum of waiting and execution time. In most cases the actual CI&T process conformed well to the designed process. Note that the event logs showed some unexpected path in the process: in rare cases the test process step timed out due to an exception. This new path is shown (red) in Figure 30.

Table 10. Activity Metric for the CI&T process steps (Sunindyo et al., 2010a)

No	Activity	Time	Avg (ms)	Min (ms)	Max (ms)
A1	Check In	Waiting time	0	0	0
		Exec. time	0	0	0
		Sojourn time	0	0	0
A2	Build	Waiting time	1.806	1.442	4.154
		Exec. time	2.808	2.176	13.524
		Sojourn time	4.614	3.756	17.678
A3	Test	Waiting time	1.785	1.425	2.941
		Exec. time	3.473	2.778	13.022
		Sojourn time	5.258	4.397	15.963
A4	Deploy	Waiting time	2.243	1.393	7.536
		Exec. time	3.131	2.211	18.380
		Sojourn time	5.374	3.878	25.177

7.1.2 Process Analysis in ASE Environments

The evaluation on ASE environments is based on three different analysis methods, namely signal change management, project progress and risk monitoring, and process model validation.

Signal Change Management

Process evaluation focuses on verification and validation of the implemented change management process. The basic process evaluation includes (a) inspection of event traces with ProM (van der Aalst, 2005), (b) analysis of captured events per event type (see Table 11), and (c) consistency checks. In a first step, we analyzed the captured events after every phase, i.e., phase 1.1, 1.2, and 1.3, and compared the event traces with the expected traces, defined in the change management workflow (see Figure 19). The results showed that the implemented process behaves like expected, i.e., the process and the event capturing approach was implemented correctly. The second step includes a basic evaluation of captured events based on the ProM evaluation results. Table 11 presents the details of this analysis.

Table 11. Occurrences of Events based on ProM Data Analysis (Winkler et al., 2011)

Occurrence of events	Phase 1.1		Phase 1.2		Phase 1.3		Total	
	Abs.	Rel.	Abs.	Rel.	Abs.	Rel.	Abs.	Rel.
E1: E_checkin_started	1	<0.1%	1	<0.1%	1	<0.1%	3	<0.1%
E2: E_signal_comparison_ started	708	25.0%	1,300	25.4%	720	29.4%	2,728	26.3%
E3: E_signal_similar			89	1.7%	432	17.6%	521	5.0%
E4: E_signal_not_changed			20	0.4%	12	0.5%	32	0.3%
E5: E_signal_changed			19	0.4%	148	6.0%	167	1.6%
E6: E_signal_new	708	25.0%	592	11.6%			1,300	12.5%
E7: E_signal_deleted			580	11.4%	128	5.2%	708	6.8%
E8: E_notification	708	25.0%	1,211	23.7%	288	11.9%	2,207	21.2%
E9: E_signal_comparison_ completed	708	25.0%	1,300	25.4%	720	29.4%	2,728	26.3%
E10: E_checkin_completed	1	<0.1%	1	<0.1%	1	<0.1%	3	<0.1
Total	2,834	100%	5,113	100%	2,450	100%	10,397	100%

The third step includes a set of consistency checks to verify that the process was completely executed and that all signals have been processed. The consistency checks include the following metrics:

(a) *Number of signals*: According to the defined workflow, the number of signals (input data and EDB data) must be equal to the number of signal comparison events (E2 = E9).

(b) *Signals compared*: The number of compared signals summarizes similar signals, accepted and rejected changes.

(c) *Notification*: Signal changes (rejected or accepted signal changes) result in notifications to related stakeholders. Note that notification objects (e.g., engineering tickets) are summarized on component level to keep the number of notifications as small and focused as possible.

Based on ProM process analysis and analyzing the individual occurrences of events within the defined workflow, we reason on a well-designed and correctly implemented workflow.

Project Progress and Risk Monitoring

This section presents the empirical results of the risk factor analysis based on the data from our industrial partner in the area of hydro power plant engineering. Special emphasis is put on monitoring project progress overview, changes from different type of stakeholders, changes in different project phases, and changes in different signals operation.

Project Progress Overview

The first analysis shows the overall number of signals grouped by phase per commit which is done weekly. Note that we focus on one snapshot per week for analysis purposes. Thus several commits could have been executed during the previous week. The result illustrates the project progress overview, where we can see the progress of numbers of signal changes across project phases. The number of signals is increasing when new signals are added, and is decreasing when old signals are deleted. Updating the content of signals will not change the number of changes. Updates of the signal status will move signals to the next phase, illustrated by the colors given in Figure 31. Figure 31 presents the number of signals per week and project phase based on different operations on signals, namely add, update, and delete.

The results showed an overall number of 3000 signals (after week 44) when the project is completed and strong variations of the number of signals along the project course.

The number of signals is increasing from week 1 to 7, and then the signals are upgraded to the next phase (drawing started). From week 9 to 12, new signals are added and then upgraded to the next phase (drawing started) in week 13. From week 14 to 18, new signals are introduced and then deleted in week 19. From week 19 to 21, new signals are added and then upgraded to the next phase in week 22. Some new signals are added in week 23 and upgraded to the drawing started phase in week 24, and then deleted in week 25. From week 25 to 28, new signals are introduced, upgraded to the next phase in week 29, and deleted in week 30. From week 30 to 32, new signals are introduced and upgraded to the next phase (drawing started) in week 33. Some new signals are still added, until week 35, and then upgraded to the phase approved in week 36. The signals are changed to the phase factory test completed in week 40, and in week 44 all signals available are moved to the phase commissioned. Note that – starting from week 36, a high share of signals passed all sequential phases to the final phase commissioning completed.

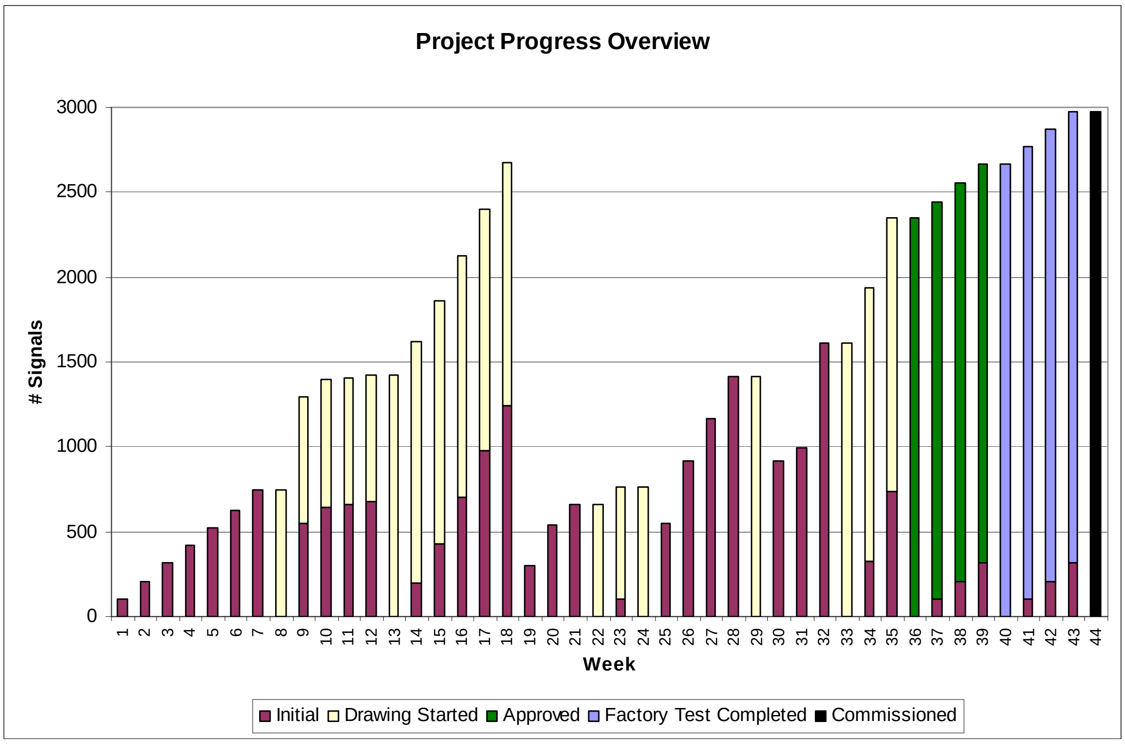


Figure 31. Project Progress Overview (Sunindyo et al., 2013)

Nevertheless it is notable that in week 19 a very high number of signals (about 80%) has been removed. A more detailed investigation of the results showed that the engineers used templates of components and also reused components from other projects without adjusting them to the current project. During a project review the components have been moved from the integrated solution to the local representations to adjust their project data. A smaller but similar effect happens in week 25 and week 29. Thus this analysis supports project managers and engineers in better assessing the current project state over time.

Number of Signal Changes by Stakeholder Group

The second analysis focuses on the impact of changes by different stakeholders, i.e., internal (engineers) and external stakeholders (customer). Signal changes originate from external stakeholders, if the customers ask for signal changes based on their requirements. Signal changes are coming from internal stakeholders, if engineers add new signals, update signals, or delete signals used in the project. Signal changes between both stakeholder groups are communicated across the project.

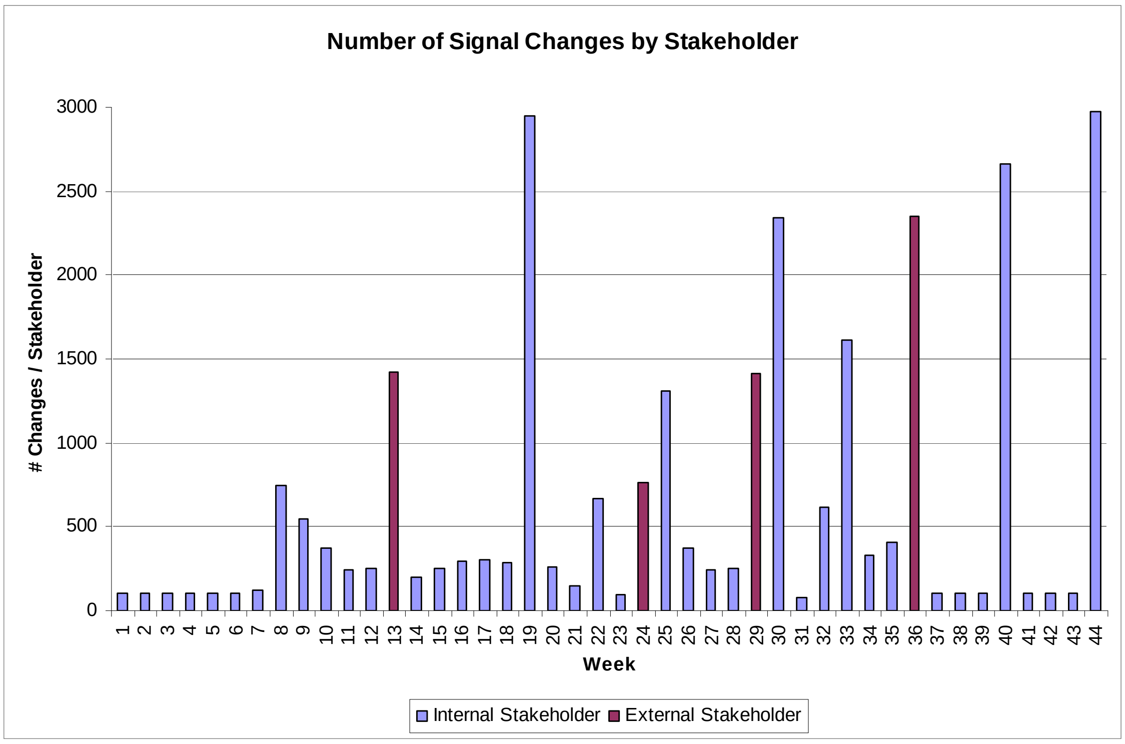


Figure 32. Number of Signal Changes by Stakeholders (Sunindyo et al., 2013)

Figure 32 illustrates the bar graph of the number of signal changes by stakeholder. Most of the changes were introduced by the engineering company and their engineers, typically add, update, and signal changes. Infrequent changes were introduced by the customer e.g., caused by reviewing processes or status meetings. The external stakeholder passes signal changes to the project during a certain period, for example in week 13, week 24, week 29, and week 36, while the other regular changes originate from internal stakeholders. A more detailed investigation of the external changes showed that typical changes focuses on signal description changes rather than on critical changes by the customer. Nevertheless, it is notable that in week 36 - very late in the project – a high number of external changes happens. As late changes make projects more critical, error-prone and risky, this analysis results supports project managers in better negotiating changes with the customer.

Number of Signal Changes Related to Project Phases

The impact of signal changes on the project state is another critical issue, as all signal changes (i.e., content changes) require a reset to the initial state and all phases/reviews (i.e. signal status upgrades) have to be repeated. As this process requires some effort and might delay the project, these analysis results help project managers in better understanding possible delays of the project.

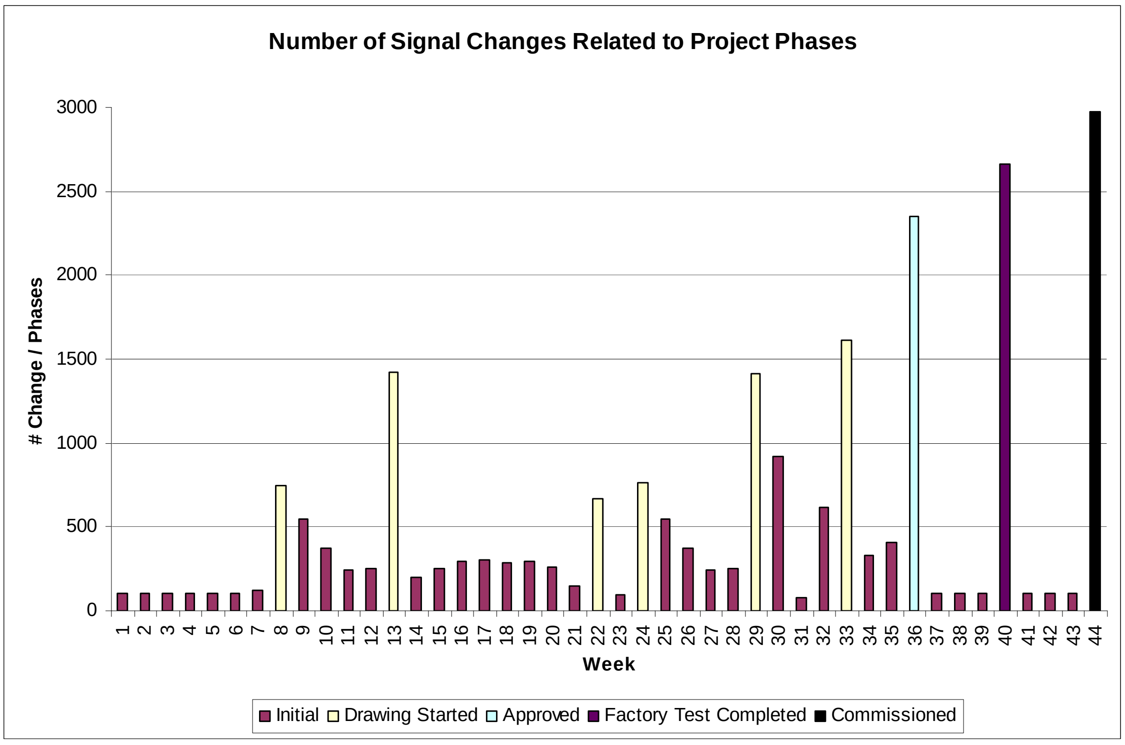


Figure 33. Number of Signal Changes Related to Project Phases (Sunindyo et al., 2013)

Figure 33 illustrates the bar graph of the number of signal changes related to the project phases. Most of the signal changes are done during the initial phases, while some are also done during drawing started, approved, and factory test completed. At the end of our observation (week 44), all signals are upgraded to the final phase (commissioned). It is notable that until week 35 almost all signals and changes are in a very early stage, i.e., in the initial and the drawing started phase. This indicates that the requirements are not well-defined and/or the customer initiated a set of changes. In week 36 we observed a high number of external changes and also the approval by the customer. This indicates that these minor changes were implemented and approved within a very short time interval, i.e., one week. As the project proceeds, short iterations and interaction with the customer could be observed, i.e., adding a small set of new signals by the engineering company and signal status updates in week 40 of about 2600 signals.

Finally a similar effect could be observed in weeks 41-44, in which the project has been completed.

Operations on Signals

Another interesting aspect focuses on the impact of operations, i.e., the amount of operations applied to the signals (i.e., signals added, signals updated, and signals removed). Two different types of operations are introduced, i.e., content updates of signals and signal status updates. Content updates do not change the status/phase of signals and focuses on changing the content of signals, e.g., range of devices, device description, tool names, hardware addresses, and path used. Status updates refer to the upgrade of signal stati/phases from one phase to the next phase. All signal content changes result in a reset of the signal status to initial.

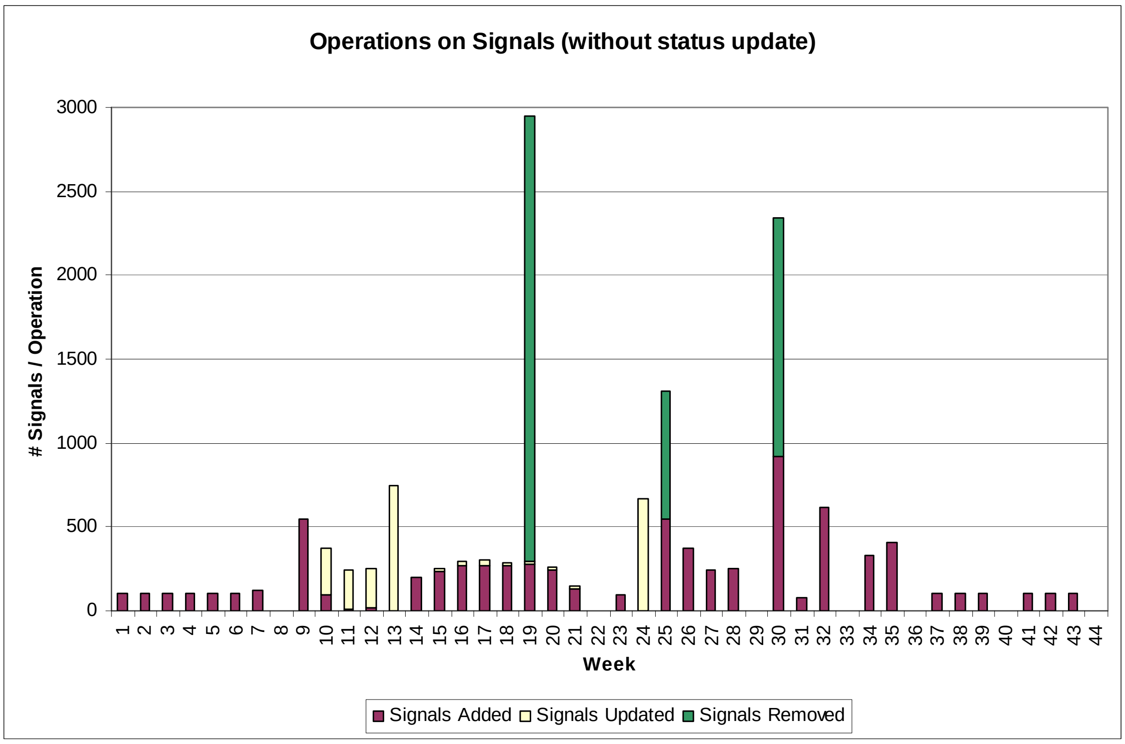


Figure 34. Operation on Signals (without status update) (Sunindyo et al., 2013)

Figure 34 illustrates the stack-bar chart of the number of signals grouped by operations (i.e., add, delete, and update content). Similar to the previous analysis results, we observed added signals along the project duration until the very end of the project, i.e. in week 43. In addition we observed two other issues: (a) a relatively low number of signal content updates, mainly between week 10 to 13 (early in the project) and in week 24. An explanation for this process could be that engineers applied templates and reused components from previous projects; in addition they modified them according to new project requirements. Anyway, as the amount of changes is rather high, a large number of components and signals have been removed in week 19, 25 and 30. After this clarification and cleanup steps the new (and correct) signal have been introduced. There are only few changes on the already available signals.

Finally the last evaluation focuses on the impact of signal status updates (see Figure 35 for details). It is notable that signal status updates are typically following a systematic approach, e.g., once a month during project progress meetings. Figure 35 shows that the number of signal status updates is increasing in the project, especially during week 24 to week 44. In general, signal status updates are performed monthly.

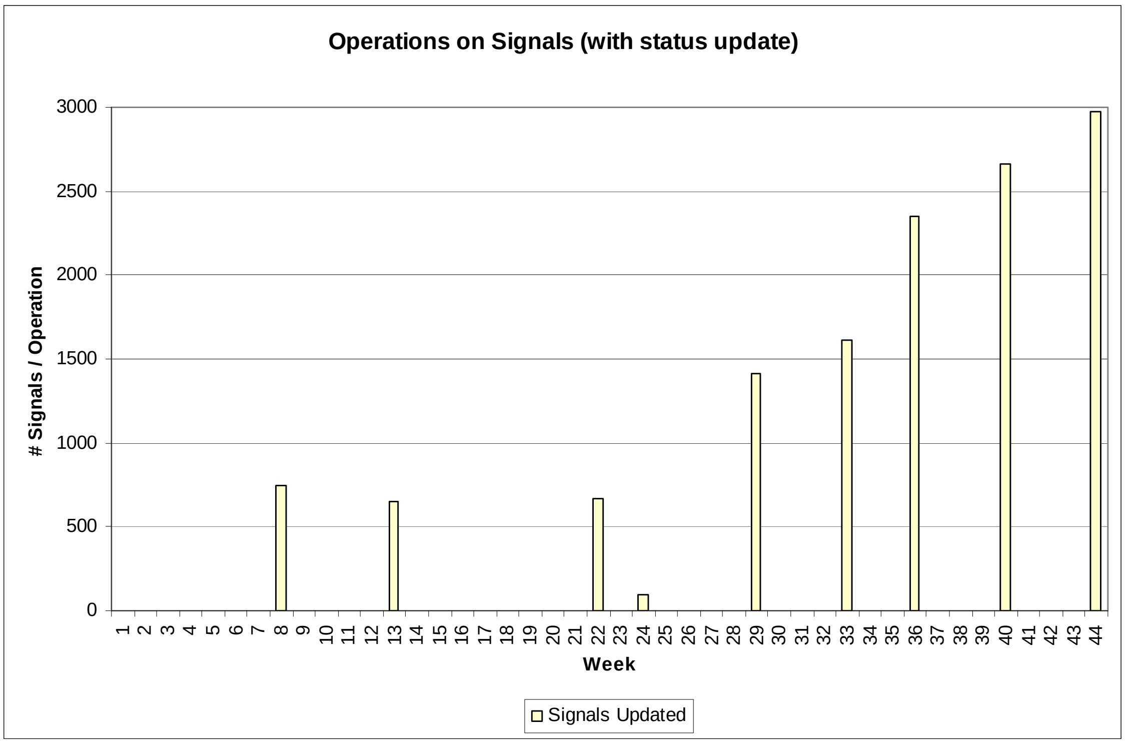


Figure 35. Operation on Signals (with status update) (Sunindyo et al., 2013)

Process Model Validation

This section describes the results of our solution approaches which are written in section 6.5. The results show that the business goal evaluation framework can be used to provide illustration of different outputs achieved by different parameters setting for inputs, e.g., the product types, the number of orders, and the classes of failures. The validation between the business processes and production processes can be achieved via conformance checking between the business process models (product trees) and the production process model generated from process event logs.

Business Goal Achievement Evaluation

The evaluation of business goal achievement is done by running several experiments on the SAW simulator by setting different parameters, e.g., classes of failures and the product types, and showing the results in the graphical mode. Figure 37 shows the relationships between the different parameters and the number of finished products.

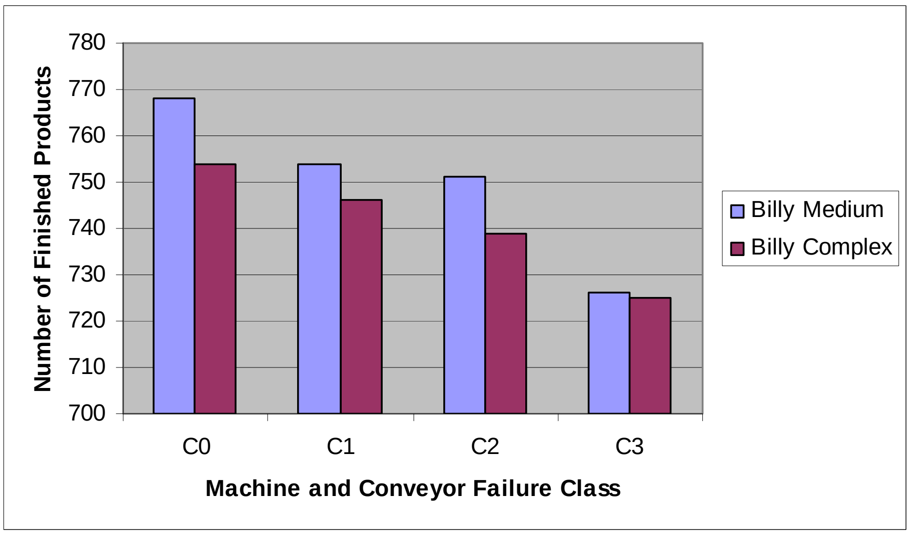


Figure 36. Relationships between failure classes and number of finished products (Sunindyo, Moser, & Winkler, 2012)

There are 1,500 business orders that distributed into of two types of products, namely Billy Medium and Billy Complex. The SAW simulator introduces four failure classes, C0 (no failure), C1 (conveyor failures), C2 (machine failures), and C3 (combined conveyor and machine failures) as simulation to the possible failures in the real situation.

From Figure 36, we can see that the number of finished products depends on the type of products produced and the failure classes occurred during the running experiment in the SAW simulator. The number of finished products is decreasing following the complexity of the product trees and the failure classes. The machine failures have a greater impact to the number of finished product rather than the conveyor failures. This is because when some machines fail, it is harder to overcome this problem. The operator should change the machine, or reroute the conveyors that lead to failing machine to other substitute machine. The machine failure could lead to other conveyor failure as well. While in case of conveyors fail, the operator can just reroute the direction of other conveyors to substitute the failing conveyor.

Business Processes Validation

The validation of business processes in the production automation systems is done by conformance checking between the designed process model in the business layer and the generated process model from actual process event data in the process layer. The top of Figure 37 shows the process model from actual process event data, which is generated by using Alpha Algorithm plugin of ProM. This process model conforms to the product trees (see bottom of Figure 37) of two product types, namely Billy Medium and

Billy Complex. A product tree consists of description of products, the parts to build the product and machine function that build the product from its parts. The product tree is written in XML notation and can be illustrated as a tree with the product as a root and its parts as nodes and a machine connects between the product and its parts.

The validation between the business processes and the production processes in the SAW simulator is done by using Alpha Algorithm from ProM tool. By analyzing the process event log obtained from running experiments in the SAW simulator using ProM, we obtain a process model that shows the orders how the products are built.

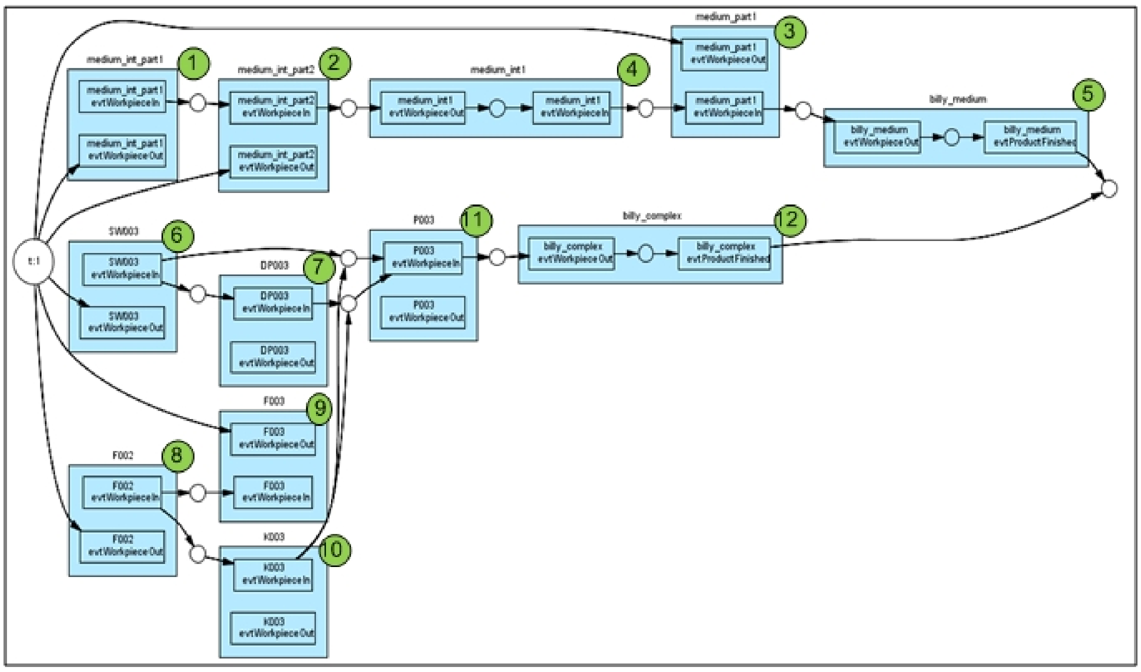


Figure 37. Overview Process Model and Product Tree Conformance (Sunindyo, Moser, & Winkler, 2012)

Figure 37 shows the analysis results by using Alpha Algorithm plugin from ProM, and illustrates orders and structure on how the products building from their parts. Billy Medium product (5) is built from medium_part1 (3) and medium_int1 (4). The medium_int1 (4) is built from two raw materials, namely medium_int_part1 (1) and medium_int_part2 (2). Billy Complex product (12) is built from two intermediate materials, namely K003 (10) and P003 (11). P003 (11) is built from F002 (8) and F003 (9), while K003 (10) is built from SW003 (6) and DP003 (7). This result validates the product trees in Figure 25.

7.2 Discussion

This section discusses the results evaluation of analysis methods on two application scenarios, namely OSS and ASE application scenarios, based on research issues on section 3.

7.2.1 OSS Project Monitoring

The discussion on OSS project monitoring analysis methods focus on efficiency, accuracy, and how the analysis methods can support and improve the engineering process quality.

Health Indicators Analysis Method

In this section, we discuss the results from our research issues, namely (a) efficiency of FOSSDA tool-supported collection, integration and validation of heterogeneous data and (b) accuracy of OSS health indicators by using multiple data sources and combined project metrics.

Data collection and validation efficiency

For the empirical study, we measured the effort for collecting, integrating and validating heterogeneous data from Apache projects (Lenya, Log4J, Excalibur, OJB) with the proposed FOSSDA framework. We compared the results with the traditional manual effort to collect, integrate and validate data from heterogeneous data sources. The key size metrics for data collection and integration from these projects were especially based on the derivation of communication metrics and bug reporting metrics.

The design and implementation of FOSSDA is based on the needs to analyze OSS data more efficiently, based on our experience on collecting the data manually. Hence we have experience regarding measuring the effort of manual data collection and furthermore regarding the tool-supported data collection process variants. The results in Initial Empirical Results Section show that the automated approach can reduce up to 30% of the efforts required for the traditional manual approach.

The advantage of the manual approach is that there is no effort needed to create a new tool to collect and analyze data, i.e., the approach uses just existing tools and knowledge on the project to collect and analyze the data. Limitations of this approach are: (a) the error-proneness of the results and tediousness to repeatedly measure data from a project at different points in time, (b) need to follow similar steps each time we want to analyze a new project, (c) no use of links with similar data that could support data analysis, e.g., the name of committers and the name of mailing list post's authors.

The benefits of using the tool-supported FOSSDA can be described as follows: (a) by using an ontology as project data storage, it is easy to maintain and query the data (see the example listings 1 and 2 in Research Issues and Approach Section); it is easy to validate data, and it is easy to establish links between related data (e.g., the name of author and contributor of different data sources) using heuristic algorithm. For example, it is assumed, that the username of a person is the part of the email address before the '@' character. Further, it is assumed that every person uses the same username in all of the tools associated with the project, hence our data sources. By applying these assumptions, it is possible to identify individuals already included in the ontology as the identical person. Even if there are variations to this rule, new rules or exceptions can be easily added to the ontology. The project manager does not have to store all project data but can focus on (the limited amount of) data needed for analysis. (b) The similarity of the Apache foundation management practices for all of their projects (e.g., all projects have a SVN, mailing list, and bug report to manage their project development) makes the configuration and the automation of data collection easy. (c) If a researcher/project and quality manager wants to analyze another project, they just have to change the configuration setting and run the tool to collect the new project data. They do not have to rebuild the tool fundamentally.

Initial extra costs of FOSSDA was the effort to use the Project Data Fetcher tool, the need to learn the semantic web technology to build the tool, and to learn the project tool setting before it was possible to operate and run the tool to collect data. Now, operating the Project Data Fetcher only needs the knowledge about running a Java application and how to set the tool and configurations, i.e., the structure of project to investigate, e.g., the location of artifacts and posting date of artifacts.

This work improved the manual data collection and validation which was proposed by Wahyudin et al. (Wahyudin, Mustofa, Schatten, Biffl & Tjoa, 2007). The usage of ontologies for data model representation and data storage is also one of FOSSDA framework benefits compared to the Alitheia (Gousios & Spinellis, 2009a) or the Ohloh (Hu & Zhao, 2008) framework.

Health Indicator Accuracy

Current using of individual health indicator methods to observe the OSS project status may raise inconsistencies on the conclusions given by those different methods. Therefore, the conclusion on the OSS project status could be different depending on the chosen indicator, e.g., one indicator may label an OSS project status healthy, while another indicator labels the same OSS project status unhealthy, which may confuse a decision maker.

In the study context, health indicators based on several project data sources have proven consistent with expert opinion to provide an overview on the status of an OSS project, i.e., whether the project is "unhealthy" or "healthy". Therefore, this information can be useful for the project manager, project investor, and other stakeholders to support decision on the OSS project, e.g., whether to add new programmers or allocate some programmers to develop some critical modules.

The combination of different indicators can strengthen the conclusion of the OSS project status compare to using the indicators separately. Adding new types of data sources is easily possible using FOSSDA and can further increase the accuracy of health indicators in varying OSS contexts.

In this research, we have compared the usage of single indicators (bug delays) and combined indicators (proportion of activities). While this study provided a promising research result, more empirical studies are needed to strengthen the external validity for a wider range of OSS projects.

We improved the health indicators notion from Wahyudin et al. (Wahyudin, Mustofa, Schatten, Biffl & Tjoa, 2007) and involving expert's opinion as one of aspects to assess the project status. The combination of different metrics, e.g., communication metrics and bug report metrics provide a new variant of data source combination, for example related to the research of Bachmann and Bernstein (Bachmann & Bernstein, 2009) which focuses more on the usage of bug tracking databases and version control system log files.

Bug History Analysis Method

The evaluation of actual engineering process model is done by checking its conformance to the designed process model. The actual engineering process model can be generated by using Heuristic Mining algorithm, e.g., process model which is generated from RHEL 6 bug history data shown in Figure 39. This process model can be conformed to the designed process model (Bugzilla Life Cycle) shown in Figure 38 to see the similarities and differences of both process models.

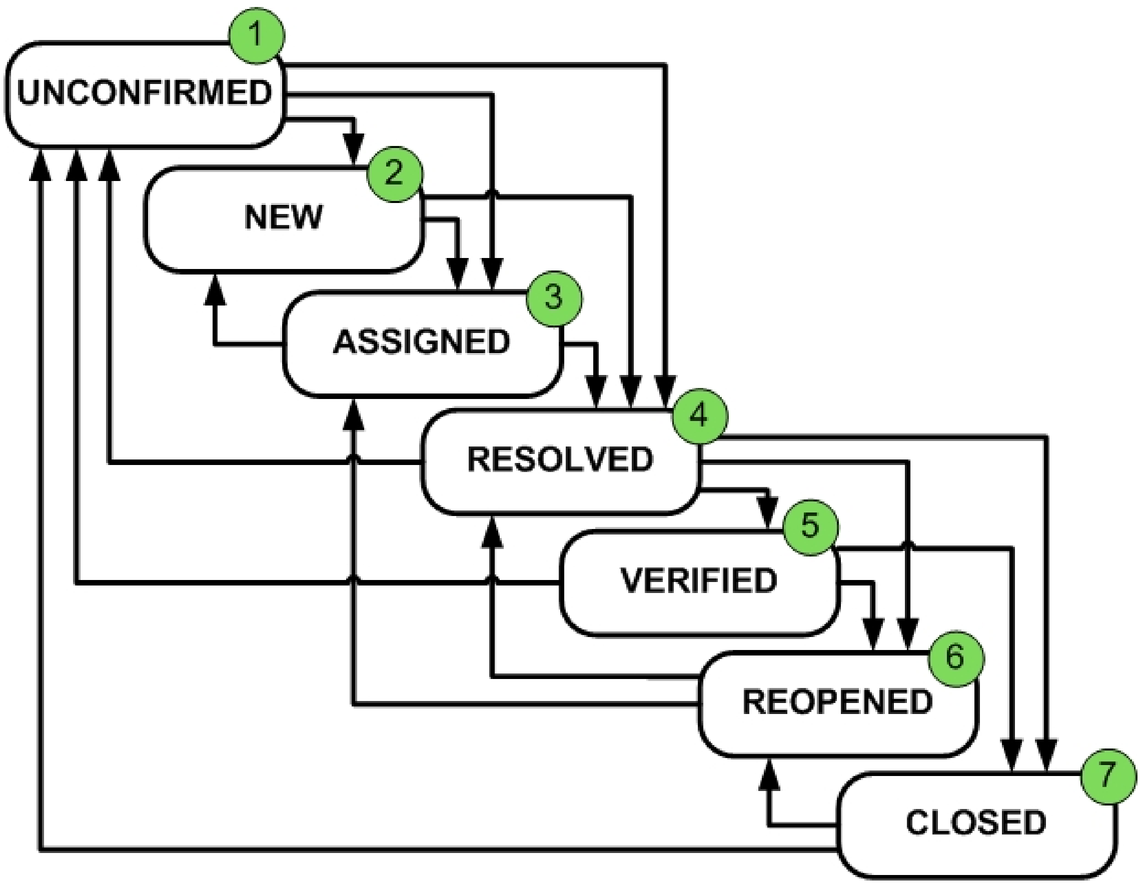


Figure 38. Bugzilla Life Cycle[36]

From the process model generation on 3 RHEL versions, we answer the two research hypotheses as follows.

The RHEL developers are following the naming and the ordering of the bug states from Bugzilla life cycle. As shown in Table 7, we can see the differences of number of states used in the designed process model (Bugzilla life cycle) and states used in the generated process models from RHEL 4, RHEL and RHEL 6.Therefore $\{\exists \psi \in (V4,V5,V6) \mid P(\psi) \neq P(\phi)\}$ thus we can reject our null hypothesis **H01**.

[36] http://www.bugzilla.org/docs/3.0/html/lifecycle.html

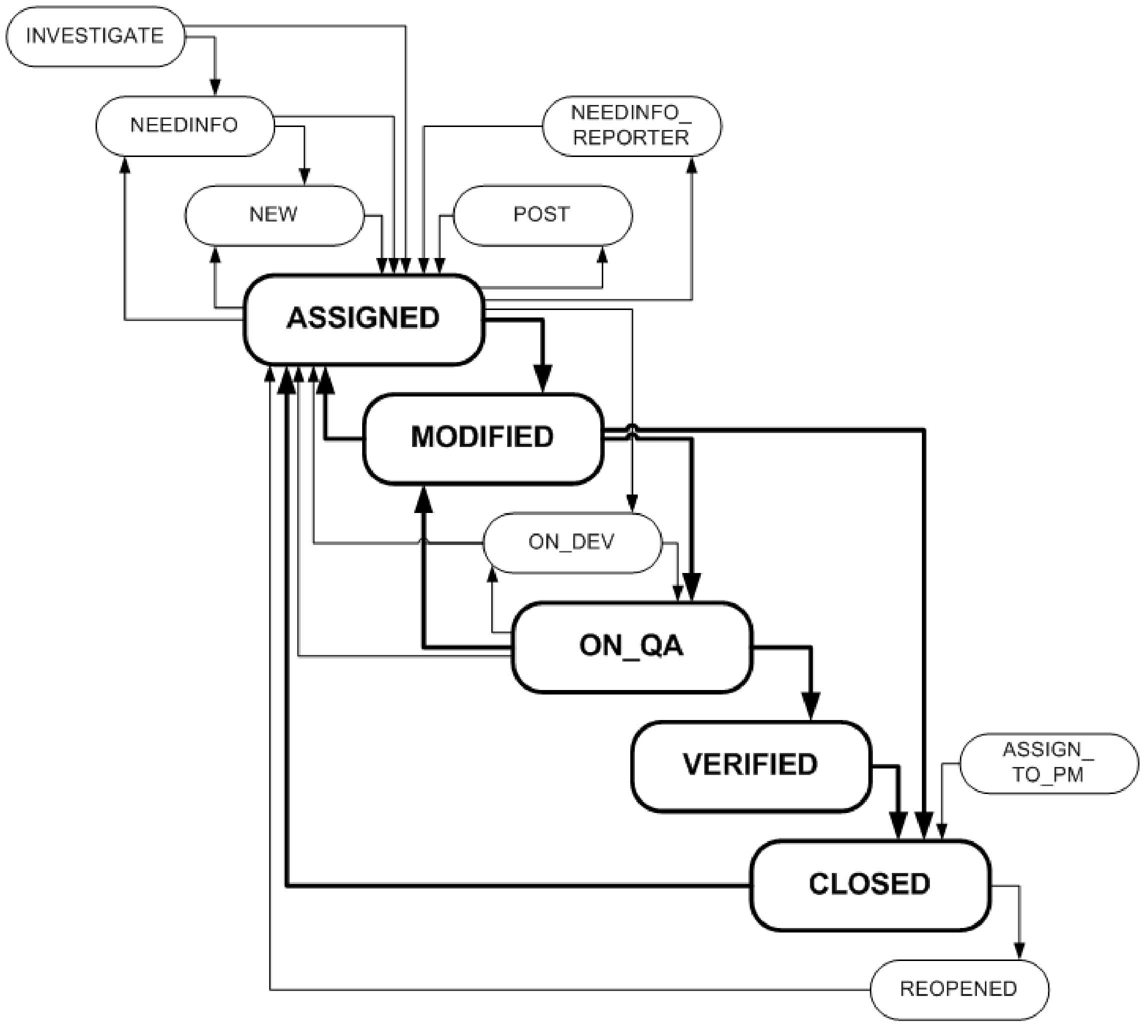


Figure 39. Process Model from RHEL version 6

An interpretation of these results can be the fact that the number of bug states available and used in the different RHEL versions decreases with the version number, meaning that states which were not used at all or only very infrequently are removed for the next RHEL version.

The using of extra bug states of the RHEL versions comparing to the original Bugzilla Life Cycle represents the needs of developers to enhance their understanding on handling the bug issues. The decreasing of the bug states used between different RHEL versions means the developers have come to some convergences of understanding, such that they don't use some bug states, due to better way to explain and deal with the bugs, e.g., some bug states which means need for more information from different parties, means that the explanation about the bugs is getting better than previous version. The better explanation of the bug to other developers can increase faster bug handling, thus increasing the productivity and process quality.

The RHEL developers are using all bug states for each bug history in the same number of frequency. As shown in Table 8, each bug state is used in different frequency by the developers. Some bug states are used more often than the others. Therefore $\{\forall s_i, s_{i+1} \mid N(s_i) \neq N(s_{i+1})\}$ thus we can reject our null hypothesis ***H02***.

An interpretation of these results can be the fact the typically OSS projects do not follow a strict waterfall-like software engineering process, but rather a sometimes mixed dynamic software engineering process.

Some bug states are more frequent than the others, for example closed, assigned, modified, verified, and On_QA.

Closed state is on the top of all versions justifies this state as the goal of other states. All other states are tending to finish in the closed state, even though there are some options to reopen the closed bugs.

Assigned state represents the beginning of the bug state which should be assigned among the developers. When the bug issue is introduced for the first time, there is an opportunity whether to offer the bug handling to a specific person or to ask other developers publicly to voluntarily taking the chance for handling the bug issue. At some points, the bug reporter should ensure that each bug issue has been assigned to another developer such that the bug issue can be solved immediately.

Modified state represents the condition where the bug has been modified. The result shows that the numbers of modified bugs are increasing across different versions, means that more bugs are identified as modified rather than other states that are less frequent and not used in later version (e.g., resolved, QA_ready).

Verified state becomes more common across different versions. More bug issues are needed to be verified during their handling, and in other side, the resolved state becomes extinct in later version.

On_QA state is also increasing across different versions, while other states related to QA (QA_ready, Fails_QA, Passess_QA) become extinct, means the QA-related states converge to On_QA state.

Other states are used not so frequently. From this result, we can see how the developers focus on some important states rather than the others. The understanding of the developers in dealing with the bug states is also increasing the bug resolving period, hence improving the process quality.

Workflow Validation Analysis Method

In this thesis, we have proposed a process analysis approach based on the observation by using the EngSB platform. The process analysis on the EngSB has been performed on event logs produced in the research lab regarding a research prototype, i.e., a CI&T use case. In this section, we discuss the benefits and limitations of the proposed approach.

Observation of tool-based engineering processes. The major issue is how to integrate the events from a range of heterogeneous tools into a consolidated event framework for further analysis of SE processes. The evaluation of the CI&T standard SE process showed the capability of the EngSB platform to enable the observation of this kind of processes: SE processes that are automated and readily provide the required events. In comparison to the standard process running on dedicated CI servers there no difference in performance observed from implementing the process in a more flexible and better observable way that enables automated process analysis.

In the evaluation use case, which is based on back-end SE tools, the EngSB had the capability to provide technical integration for the SE tools involved, which are representative of modern SE environments. Based on the successful integration of backend tool events, a next step would be to extend the scope of processes and also include events from SE tools that focus on the interaction with humans, such as IDEs and making events from human actions observable, e.g., with a ticketing system. Semantic integration was successfully used with knowledge-based components in the EngSB context to consolidate the data models of the tools involved and transform the tool events into an SE process event model, which can be understood by process analysis tools. In SE environments data models in tools that contribute to SE processes vary considerable in their complexity. Therefore, the investigation of more complex and heterogeneous data models in SE processes is a natural next step, e.g., change management of data models across engineering tools.

Process analysis based on the integrated data. For process analysis we focused in this thesis on conformance analysis, performance analysis, and decision point analysis. In the CI&T use case the process could be observed in sufficient detail to support these kinds of analysis. We found that the implemented process worked most times very well but led in some cases to unexpected exceptions that need further investigation. Also we found considerable variation in the waiting and execution times of the process considering that the process instances were run under comparable circumstances. The empirical data analysis can help to set controls in the process, e.g., time out levels, to balance process risks and effort of operators.

The input to compare the expected and actual process variants in engineering environment is the model of the expected process based on the expertise of local domain experts. The actual process model can be obtained from analyzing the event logs from the running system. A challenge in practice may arise for processes that use events that are not easily obtained from the tools that regularly support the process. In these cases the EngSB platform allows integrating tools that support human processes. Most cases could be addressed with a ticketing system, where the ticket data model can be extended to hold relevant information on the process activities and the tools involved. Events that contain the information on these tickets can then provide the relevant event log attributes for process analysis and improvement. In this context we see the need for further empirical studies on process observation requirements in engineering processes, in particular, for projects, which bring together engineers and domain experts from several disciplines.

Lessons learned. We proposed the foundations for event-based engineering process analysis that show several strong points. With this approach, we can build data integration models that link available tool events to relevant SE processes and allow their automated observation and analysis. While the effort to build and validate the platform was considerable, the effort to model the tool events, their transformation and analysis seems moderate compared to the manual analysis of SE processes, which is likely to focus on a few process instances. Similar to the benefits of test automation, automated process analysis can be repeated often and with little additional cost. In our case we were able to identify rare cases of deviating process behavior and variations in process performance that would be unlikely to observe with a manual approach. We started with the rather simple CI&T process to show the capability of performing process analyses, which can be applied to more complex processes. As success factors we see the availability of domain knowledge, sufficiently well-defined SE processes, and access to the tool events. Our research focuses on making tool events in heterogeneous engineering environments available to SE process analysts. This approach can be generalized in other systems, especially in process aware information systems, where the information of processes is stored in the form of event log. Investments should be done by the project manager are on producing event log from non-event systems and finding right process analysis methods and building tool supporting those methods.

7.2.2 Process Analysis in ASE Environments

The discussion on ASE environments analysis methods focus on the feasibility, accuracy and validation between designed process model and process model generated from actual engineering process event logs.

Signal Change Management

The initial process evaluation confirmed the implementation of the proposed change management process and enables a more detailed analysis for project management purposes, e.g., measuring the number of signal changes per event type. Table 12 summarizes the derived metrics from analyzing captured events. The data sets have been derived from a very early phase of the development project, i.e., system design phase, which is an explanation of the high variability of signals (1,211 changes and 89 similar signals in phase 1.2; the number of changes decreases to 288 and the number of similar signals increases to 432 in phase 1.3).

Table 12. Change Management Metrics based on Signal Comparisons (Winkler et al., 2011)

	Phase 1.1		Phase 1.2		Phase 1.3		Total	
	No	%	No	%	No	%	No	%
Similar Signals	0	0%	89	6.9%	432	60%	521	19.1%
Accepted Changes	708	100%	1,191	91.6%	276	38.3%	2,175	79.7%
Rejected Changes	0	0%	20	1.5%	12	1.7%	32	1.2%
Signal Comparisons	708	100%	1,300	100%	720	100%	2,728	100%

Note that we do not consider multiple changes of the same signals. The metrics are based on signal comparison values, i.e., the number of signals in the EDB and the number of signals captured during check-in sequences. See Figure 40 and Figure 41 for bar charts of individual check-in sequences.

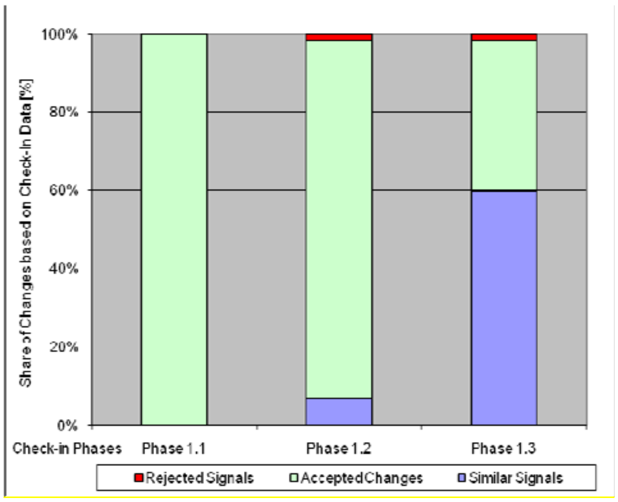


Figure 40. Accepted/Rejected Signals per Phase (Winkler et al., 2011)

Figure 40 presents the analysis results of accepted, rejected and unchanged (similar) signals per phase. The number of similar signals increases to 60% in phase 1.3 and the number of accepted changes decreases across the three check-in phases. Figure 41 and Table 13 present a more detailed view on signal changes based on accepted changes and on signal comparison activities. An interesting finding was that in phase 1.2 and phase 1.3 a high number of signals were removed, because sets of components have been replaced in early phases of development. Note that no new signal was introduced in phase 1.3.

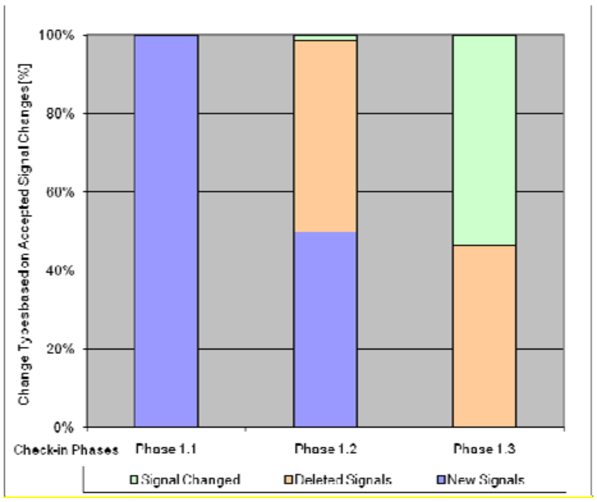


Figure 41. Change Type of Accepted Changes (Winkler et al., 2011)

Based on the observed events and captured data after the third check-in (i.e., summarizing phase 1.1, 1.2, and 1.3) we observed an overall number of 2,728 signal comparisons and a number of 2,175 accepted changes (new, removed, and changed signals). A more detailed analysis of changes showed an overall change acceptance rate after phase 1.3 of 79.7%. Experts estimated approximately 20% of signal changes along the overall project course – this seems to be contradictory. Nevertheless, explanations for this deviation are: (a) we applied signal lists based on a very early project phase, i.e., the systems design phase, with incomplete and unstable requirements and a basic systems architecture; (b) the signal lists does not cover all components of the plant but is limited to a small subset of components, i.e., most critical components.

Table 13. Signal Change Type of Accepted Signal Changes (Winkler et al., 2011)

	Phase 1.1		Phase 1.2		Phase 1.3		Total	
New Signals	708	100%	592	49.7%	0	0%	1,300	59.8%
Deleted Signals	0	0%	580	48.7%	128	46.4%	708	32.5%
Changed Signals	0	0%	19	1.6%	148	53.6%	167	7.7%
Accepted Changes	708	100%	1,191	100%	276	100%	2,175	100%

Nevertheless, we see the prototype evaluation as proof-of-concept of the proposed process, product, and project observation approach based on events that can support project managers in better understanding and analyzing the underlying processes and measuring generated products along the project course.

Project Progress and Risk Monitoring

This section summarizes the major findings of our risk-based approach for ASE projects based on the initial evaluation of real-world industry data derived from a large-scale engineering company in the hydro power plant domain. We identified three different risk groups: (a) Domain specific risks, (b) Collaboration risks, and (c) Project management risks.

While domain specific risks are typically addressed by domain specific tools and methods, e.g., RiskIt method in software engineering (Kontio, 1999), we observed strong limitations regarding risk assessment in the ASE domain focusing on heterogeneous engineering environments. Collaboration risks typically focus on the synchronization of data models, engineering objects (e.g., signals) and engineering artifacts where engineering propagate changes – the most critical engineering process in ASE projects, especially if various stakeholders from various disciplines are involved – to related engineers in other disciplines. In addition we observed strong limitations on a comprehensive view on the overall engineering project from the project management perspective (Project Management Risks).

Identify and assess risks by using engineering workflows in ASE projects. We identified the change management workflow in heterogeneous environments as the most critical process in ASE projects. Changes, even late in the project, can have a major impact on the project progress and success, even in a heterogeneous environment. Thus frequent synchronization of engineering artifacts is essential for successful collaboration and to enable a consistent project data for all related engineers. The Engineering Service Bus (Biffl, Schatten, et al., 2009) provides a middleware platform that enables technical integration of heterogeneous tools and semantic integration of data models coming from different sources (Biffl, Sunindyo, et al., 2009). Based on technical and semantic integration, project managers and engineers are able to synchronize data across disciplines more effective and efficient. In addition, metrics on the project progress become measurable, an important benefit for project managers. To address risk management we identified a set of metrics, enabling the observation and control of ASE projects.

The number of engineering objects (signals) is an important indicator regarding the identification of the project progress with respect to individual project risks. Figure 31 presented the number of signals per week and project phase in our initial evaluation study. One might assume an increasing number of engineering objects over time. Nevertheless the results showed a rapid decrease of the number of signals between week 18 and week 19. The main reason was that previously engineers reused large components from previous projects (some copy / paste approach). In week 18, an in-depth review takes place where it has been decided that the currently used solution approach was not appropriate. Thus, almost all parts of the components have been removed. Similar effects appear between week 21/22 and week 29/30. An analysis at the customer site identified some critical changes in a few components which have been fixed, i.e., exchanged by more appropriate components. Typically the analysis, illustrated in Figure 31 highlighted the risk of reusing components to a wide exchange to reduce effort and cost. Using wrong components will result in high rework effort. Thus, it is required to plan component reuse strategies appropriately.

Classify Risk Factors based on Different Types of Stakeholders. Based on observations at our industry partner and results from previous analysis results (Sadiq et al., 2004) we identified a set of metrics as promising candidate metrics for project risk assessment.

An overview on the Project Progress has already been applied to demonstrate the application of risk assessment, based on the number of engineering objects (i.e., signals) in the engineering database. Instable and the frequent changing number of available signals is an indicator for reusing components (copy/paste) approach or some unclear requirements which require high rework effort by engineers and experts.

The Impact of Changes from various Stakeholders (e.g., internal engineers or external stakeholder, i.e., the customer) is another important aspect in change management processes and can result in high risks (even if external changes come up frequently). Figure 32 presented the number of changes per stakeholder group. The results showed that the external stakeholders introduce changes every two months at the beginning and monthly at the end of the project. These analysis results help project-managers in better discussing the changes with the customer. In our initial evaluation the duration between external changes seems to be appropriate. It is notable that the last pile of changes takes effect in week 36, 2 months before project completion.

Impact on project phases. Signal changes, especially signal updates result in a reset of the current project phase based on a rather sequential engineering process. Thus an important information and consequence of changes is an analysis on the impact of changes per phase (see Figure 32). It is notable that until week 35 almost all signals are in the state “initial” or “drawing started”, early phases in the ASE project. After applying the last pile of customer changes (i.e., in week 36) the signal status develop to the project finalization time rapidly, e.g., more than one signal status update per week. Note that the evaluation focuses on snapshots (once a week) for analysis purposes.

Impact of Signal Operation. Finally, it is important to have an idea on the share and type of signal changes, i.e., added signals newly introduced to the system, modified (updated) signals, and – the most critical aspect – removed signals. As discussed before, three main risks apply, (a) in week 19 where almost 80% of signals have been removed; (b) in week 25; and (c) in week 30. Main reason for this large amount of deleted signals was the reuse of components and templates which have to be improved for future projects.

Process Model Validation

Evaluation of the business goal achievement in the production automation systems. We have introduced the business goal evaluation framework to enable the project/quality managers to analysis different processes types from different layers (e.g, business layer and process layer) and provide analysis results to the project/quality managers for further decision on the process improvement.

The evaluation shows some parameters in the test cases, e.g., the product types, number of products ordered, and classes of possible failures, affect the productivity of the systems, e.g., the number of finished products. This information is also useful for the project/quality managers to plan the configuration of workshop layout in the real situation.

Validation of the business processes with the production processes data in the production automation systems. The validation of different processes types from different layers is done by collecting the processes data from different layers, e.g, product trees from the business layer and processes event from the process layer. The collected data then is analyzed by using the conformance checking tool from Prom Alpha Algorithm plugin to compare the suitability between the process model generated from actual data and the designed process model in the process tree. The results show that the production processes are really follow the product trees and not violating the structure and orders on how the products are built from their parts.

Some failures have been introduced during the running of experiments in the SAW simulator, e.g., the machine failures, conveyor failures, and the combination of both failures. However, these classes of failures only impact on the number of finished products and do not impact on the form of products. Hence the project/quality managers can guarantee that the products delivered will always in the good form and conform their design, and the failures only effect on the processes and not the form of products.

8 Conclusion and Perspectives

In heterogeneous software and systems development environments which involves engineering systems and tools from several sources, the software is no longer be seen as a stand-alone system and delivered as "shrink-wrapped package", but embedded in larger context of systems, for example as part of some infrastructure where hardware and software components have to cooperate seamlessly (Biffl & Schatten, 2009).

The capabilities for effective and efficient integration of engineering systems (Issarny et al., 2007) and the semantic integration of engineering knowledge (Aldred et al., 2006) are key enablers for engineering process automation and advanced quality management (Weinberg, 1993). The next core question is how to observe and analyze engineering process in those heterogeneous software and systems development environments.

Current engineering process observation and analysis is limited to software-only development environments (Johnson, 2001) or focus on developers' action rather on the artifacts of the projects (Torii et al., 1999), making the results of observation and analysis are limited only to certain engineering domains.

This work proposes the Project Observation and Analysis Framework (POAF) for supporting engineering process observation and analysis in heterogeneous software and systems development environments. Since it is hard for the project manager to manage and monitor different engineering processes run by heterogeneous stakeholders in the project, the project manager needs a way to collect, integrate and analyze those heterogeneous engineering process and validate the results of analysis with the expected process model from the project manager.

The POAF consists of data collection, data analysis, and data presentation steps, and involves different methods, data sources, and automated/semi-automated tools that can help project managers and engineers to do their jobs. The semantic web, statistical analysis, and process mining technologies are used in designing and developing this framework. Key contributions of this work are industrial application and proof-of-concept of the proposed engineering process analysis and observation approach.

The research results were evaluated in two application domains, namely open-source software projects and automation systems engineering domains, regarding feasibility, efficiency, and effectiveness. The evaluation is based on the prototypes for a set of specific use cases of the two industrial application domains, as well as on empirical studies of beneficiary roles as proof-of-concepts. Major results of this work are the feasibility of the POAF, i.e., the process, method and tool support is usable and useful across engineering domains, as well as better accuracy, effectiveness and efficiency.

8.1 Highlights

In this section, the main results of the work done by the practitioners and the researchers in two application domains are summarized.

8.1.1 OSS Projects Monitoring

The highlights on OSS projects monitoring focus on the framework for OSS Data Analysis as an instantiation of the Project Observation and Analysis Framework in OSS application domain, integrated data model to support data analysis on heterogeneous data sources, bug history analysis method to analyze bug report data, and workflow validation analysis.

Framework for OSS Data Analysis

The framework for OSS Data Analysis is originated from the needs of OSS projects stakeholders for reliable and easy-to-determine project health indicators to predict whether the project is likely to sustain for a sufficient period of time in order to justify their investments into the project.

Section 5.2 of this thesis presented the Framework for OSS Data Analysis (FOSSDA) as an instantiation of the Project Observation and Analysis Framework (POAF) in OSS domain. An empirical study has been conducted with four Apache projects to investigate whether the FOSSDA can support efficient data collection and analysis from heterogeneous data sources.

By using this framework, the data collection efforts required for the traditional manual approach can be reduced up to 30%. With the support of Project Data Fetcher tool in FOSSDA, project and quality manager of OSS projects can collect, integrate and validate the data easily and then use the tool Project Monitoring Cockpit to analyze and assess health indicators of OSS projects within hours.

Integrated Data Model

The integrated data model is originated from the needs of OSS projects stakeholders to have a common view on heterogeneous data models of OSS data sources. This integrated data model is useful to support observation and analysis of OSS engineering processes over several data sources.

In section 5.3 of this thesis, the integrated data model for our research work is presented. This model comes from our observation and analysis on Apache projects and worked well to support process and project metrics for producing the health indicators and can be easily adapted to the project management standards in other OSS families, like SourceForge or RedHat. The flexibility in the data sources level makes the addition and modification of the data sources easy to handle, while the integrated data model in the process and project level remains stable make the observation of OSS project health indicators more robust against changes on the data source level.

Bug History Analysis Method

In section 5.5 of this thesis, the bug history analysis method is presented as an effort to improve the quality of OSS processes and products. Bugs are an important source for project observation. However, the use of advanced approaches, such as data mining approaches, to analyze bug report data, has not yet been intensively researched and therefore requires further investigations regarding its usefulness for OSS projects observation and quality improvement.

In this section, the contribution of an observation framework in improving the process quality in OSS projects has been explained, e.g., by observing the end states of bugs are not reopened frequently. The bug history data from RHEL projects have been used as a use case for our observation framework application and the Heuristics Mining (Weijters & Ribeiro, 2010; Weijters, van der Aalst, & de Medeiros, 2006) algorithm from the Process Mining (ProM) tool (de Medeiros & Weijters, 2009) have been used as our analysis tool. The analysis results on conformance checking of process models from RHEL bug history data can be used to improve the process quality.

Workflow Validation Analysis Method

In section 5.6, the workflow validation analysis method is presented to support a systematic analysis and improvement of software development projects. A service-oriented platform based on the EngSB has been introduced to integrate heterogeneous engineering tools and an approach to monitor and analyze tool-based engineering process has been proposed.

The approach has been initially evaluated with the best-practice “continuous integration and test” process. The analysis of the tool-based engineering process was demonstrated by comparing expected process model and actual process model that was obtained from applying process mining method on event logs provided by the EngSB platform.

Major result was that the approach allowed comparing expected (designed) and actual (real-life) engineering processes regarding their structure, performance, and risk of bottlenecks. We analyzed the performance of the processes, found surprising variations in the time needed to complete planned process steps and detected unexpected process paths. These findings can help the quality manager to plan focused and more detailed analyses and improve process control. Therefore, EngSB-based process analysis and validation can help project and quality managers to determine the status of running SE projects and measure key performance indicators.

8.1.2 Process Analysis in ASE Environments

The highlights on process analysis in ASE environments focus on signal change management, project progress and risk monitoring, and process model validation.

Signal Change Management

Collaboration and interaction between different engineering fields are critical issues in automation systems engineering (ASE) because individual disciplines apply different tools and data models. This heterogeneity hinders efficient collaboration and interaction between various stakeholders, such as mechanical, electrical, and software engineers. Based on the EngSB, the Virtual Common Data Model (VCDM) enables efficient data exchange based on common concepts, e.g., signals or engineering objects, as foundation for change management and process observation.

Process evaluation is required for process verification and validation, i.e., whether the designed (and implemented) workflow behaves like expected. Collecting and analyzing event data with data mining and analysis tools (e.g., ProM) enables the investigation of processes and event traces for process and workflow verification and validation purposes. Applying process measurement within the signal change management process, the results showed that (a) the designed change management process is appropriate in context of automation systems engineering projects and (b) the presented event definition, collection, and evaluation is a valuable approach for process evaluation.

In addition, event data (captured during process execution) can enable project monitoring and control for project management purposes. Implicit data, such as the number of changes per phase (and/or per time interval), can be made explicitly for decision makers to get an overview on the overall project and to implement counter-measures in case of project plan deviations.

Project Progress and Risk Monitoring

Engineers from different engineering fields, as occurring typically in large-scale Automation Systems Engineering (ASE) projects, rely on their own tools and data models to perform their specific tasks of their specific engineering fields. Furthermore, these engineers typically create and use their own specific engineering workflows for communicating, disseminating, and managing objects across the borders of different engineering fields. Thus, there is a need for flexible and comprehensive engineering process support across disciplines to allow risk-aware collaboration and interaction between disciplines, tools, and data models. With this focus on raising the risk awareness of object change management workflows, the key questions for project management and engineers are (a) how changes can be handled more efficient and (b) how relevant change requests can be passed to involved engineers.

This thesis presented the Engineering Service Bus (EngSB) framework to provide (a) an efficient change management process and (b) integrated views on heterogeneous engineering environments to better analyze and highlight upcoming risks. Based on real-world engineering project data from a hydro power plant systems integrator, the proposed approach is evaluated and discussed.

First results showed that – based on the change management workflow – the consideration of risk factors can enhance the overall engineering project quality and enables risk mitigation in ASE projects. Based on change management data, we identified four main risk factors context of the initial evaluation: (a) Overall number of signal data in the engineering base; (b) the impact of changes from different stakeholder groups, i.e., internal and external stakeholders; (c) Impact of Changes with respect to signal status within a defined engineering process; and (d) impact of different operations on engineering objects (i.e., add, update, delete, and status updates). The analysis results showed that these initially defined metrics are reasonable for assessing the current ASE project from engineers and management perspective.

Nevertheless, the presentation of data and analysis results is essential for individual stakeholders by providing individual views on the project, e.g., focus on a comprehensive view on the project from project management perspective or focus on individual disciplines from the perspective of individual engineers. In (Moser, Mordinyi, et al., 2011) we observed the engineering cockpit, a promising solution to present captured data from the change management process to related engineers and the project managers.

Process Model Validation

In this thesis, we have introduced the business goal evaluation framework to improve SbPV (Simulation-based Process Validation) approach in the production automation systems. The results showed that the business goal evaluation framework can help the project/quality managers to evaluate the business goal achievement and validate the business processes and the production processes in the production automation systems.

SbPV is laid as a part of the SAW simulator in the evaluation framework and support the project/quality managers to validate the simulation results that will be useful in for running manufacturing processes in the real world.

There are two findings which are interesting for the quality/project managers. First, the number of finished products is decreasing following the complexity of the product types and the failure classes. Second, the failure classes introduced in the test case of running experiment did not impact the form of products. So the project/quality managers can guarantee that the products produced are following the product design in the product trees.

8.2 Future Work

This section discusses the future work of engineering process observation and analysis in two directions, whether in single domain like OSS projects monitoring or multidisciplinary engineering fields, like process analysis in ASE environments.

8.2.1 OSS Projects Monitoring

The future work in OSS projects monitoring direction will include the extension of FOSSDA with new health indicators (e.g., communication metrics and interaction between the developers) and additional data sources to investigate robust OSS health status indicator in several environments, e.g., SourceForge and RedHat project management standards.

Bug report data is proven to be a good source for OSS data analysis and can be used to explain the historical behavior of the developers on resolving the bug reports during OSS project development period. Hence, the analysis on the bug report data should be intensified to show the relationships between the bug report data and other OSS artifacts to support the prediction and estimation of OSS project quality, for example by using process mining and other data mining approach.

For generalization of our approach, we propose to apply the framework to other OSS projects and also for closed source software projects. New tools or improvement from previous tools can be introduced in collecting and integrating data sources. The semantic web technology is proven to be successful in collecting and integrating the data structures more efficiently and can be used to integrate a new data structure into previous data structure which is already available, e.g., in the form of ontology. The setting of tools parameters, like name and location of the data source and collecting period, make the data collection and integration is also faster than by using manual approach.

The analysis of challenging engineering processes and environments in a range of engineering domains will be covered. Our proposed foundations can be used and improved for analyzing processes in other engineering domains. Advanced process analysis can build on the combination of event data and knowledge models on the engineering process, learning both from SE process models and domain-specific practices.

8.2.2 Process Analysis in ASE Environments

The future work for process analysis in ASE environments can be done into three research directions, namely signal change management, project progress and risk monitoring, and process model validation.

In signal change management, future work will include (a) the integration of additional workflows aligned with a more detailed engineering process to better understand and automate engineering processes in general, (b) refining the change management process with respect to identify the number of changes per signal, i.e., detecting multiple changes per signal, and (c) an ongoing observation of product metrics with respect to better understand automation systems projects based on real-world project observations. In addition, a more sophisticated workflow component will allow a more flexible engineering process definition and implementation, and thus would make a large step towards agile signal change management in the automation systems engineering domain. Furthermore, we will focus on the introduction and adaptation of agile processes from software engineering to the automation systems engineering domain.

In project progress and risk monitoring, future work will include three different directions: (a) more detailed investigation of risk factors and the development / extension of the identified metrics to enable a better understanding of ASE projects; (b) additional evaluations and case studies to verify and validate the presented approach with respect to applicability and scalability; and (c) more detailed investigation on the current need of engineers, managers, and related stakeholders to learn more about ASE projects and the need for measurement, data collection, analysis and presentation with respect to develop an engineering cockpit for better supporting ASE projects.

In process model validation, the future work will include analysis on the relationships of structure and source of failures to the output of products by using organizational mining approaches or social network analysis method.

References

A. J. Rembert, & Ellis, C. (2009). An initial approach to mining multiple perspectives of a business process. *5th Richard Tapia Celebration of Diversity in Computing Conference: Intellect, Initiatives, Insight, and Innovations* (pp. 35–40). Portland, Oregon.

Aldred, L., van der Aalst, W., Dumas, M., & Hofstede, A. t. (2006). *Understanding the Challenges in Getting Together: The Semantics of Decoupling in Middleware*. Eindhoven, The Netherlands.

Ammon, R. v., Silberbauer, C., & Wolff, C. (2007). Domain Specific Reference Models for Event Patterns - for Faster Developing of Business Activity Monitoring Applications. *VIPSI 2007*. Lake Bled, Slovenia.

Baker, P., Zhen, R. D., Gabrowksi, J., & Oystein, H. (2008). *Model-Driven Testing: Using the UML Testing Profile*. Springer.

Basili, V., Caldiera, G., & Rombach, D. H. (1994). The goal question metric approach. In J. Marciniak (Ed.), *Encyclopedia of Software Engineering*.

Becker, P., Lew, P., & Olsina, L. (2011). Strategy to improve quality for software applications: a process view. *International Conference on on Software and Systems Process (ICSSP 2011)* (pp. 129–138). Waikiki, Honolulu, HI, USA: ACM. doi:10.1145/1987875.1987897

Biffl, S., & Schatten, A. (2009). A Platform for Service-Oriented Integration of Software Engineering Environments. *2009 conference on New Trends in Software Methodologies, Tools and Techniques (SoMeT'09)*. IOS Press.

Biffl, S., Schatten, A., & Zoitl, A. (2009). Integration of Heterogeneous Engineering Environments for the Automation Systems Lifecycle. *5th IEEE International Conference on Industrial Informatics (IndIn)* (pp. 576–581).

Biffl, S., Sunindyo, W. D., & Moser, T. (2009). Bridging Semantic Gaps Between Stakeholders in the Production Automation Domain with Ontology Areas. *21st International Conference on Software Engineering & Knowledge Engineering (SEKE 2009)* (pp. 233–239). USA.

Biffl, S., Sunindyo, W. D., & Moser, T. (2010a). Semantic Integration of Heterogeneous Data Sources for Monitoring Frequent-Release Software Projects. In L. Barolli, F. Xhafa, S. Vitabile, & H.-H. Hsu (Eds.), *2010 International Conference on Complex, Intelligent and Software Intensive Systems* (pp. 360–367). Krakow: IEEE. doi:10.1109/CISIS.2010.58

Biffl, S., Sunindyo, W. D., & Moser, T. (2010b). A Project Monitoring Cockpit Based On Integrating Data Sources in Open Source Software Development. *Proc. Twenty-Second International Conference on Software Engineering and Knowledge Engineering (SEKE 2010)* (pp. 620–627).

Boehm, B. W. (1991). Software Risk Management: Principles and Practices. *IEEE Software*, *8*(1), 32–41.

Booch, G., Rumbaugh, J., & Jacobson, I. (2005). *Unified Modeling Language User Guide* (2nd Editio.). Addison-Wesley Professional.

Brebner, P. (2009). Service-Oriented Performance Modeling the MULE Enterprise Service Bus (ESB) Loan Broker Application. *35th Euromicro Conference on Software Engineering and Advanced Applications (Euromicro SEAA 2009)* (pp. 404–411). Patras, Greece.

Brereton, P., Kitchenham, B., Budgen, D., Turner, M., & Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. *The Journal of Systems & Software*, *80*(4), 571–583.

Chappell, D. A. (2004). *Enterprise Service Bus*. O'Reilly Media.

Chen, P. P.-S. (1976). The entity-relationship model - toward a unified view of data. *ACM Trans. Database Syst.*, *1*(1), 9–36.

Doan, A. H., Noy, N. F., & Halevy, A. Y. (2004). Introduction to the special issue on semantic integration. *SIGMOD Rec.*, *33*, 11–13.

Ferreira, D. M. R., & Ferreira, J. J. P. (2004). Developing a reusable workflow engine. *J. Syst. Archit.*, *50*(6), 309–324.

Floyd, C. (1984). A systematic look at prototyping. (R. Budde, K. Kuhlenkamp, L. Mathiassen, & H. Züllighoven, Eds.)*Approaches to prototyping*, *1*, 1–18.

Freimut, B., Punter, T., Biffl, S., & Ciolkowski, M. (2001). State-of-the-Art in Empirical Studies. Virtuelles Software Engineering Kompetenzzentrum (Visek).

Gegick, M., Rotella, P., & Tao, X. (2010). Identifying security bug reports via text mining: An industrial case study. *7th IEEE Working Conference on Mining Software Repositories (MSR 2010)* (pp. 11–20).

Gousios, G., & Spinellis, D. (2009a). Alitheia Core: An extensible software quality monitoring platform. *IEEE 31st International Conference on Software Engineering (ICSE 2009)* (pp. 579–582). Vancouver, Canada: IEEE Computer Society.

Gousios, G., & Spinellis, D. (2009b). A platform for software engineering research. *6th IEEE International Working Conference on Mining Software Repositories (MSR '09)* (pp. 31–40). Vancouver, Canada: IEEE Computer Society.

Halevy, A. (2005). Why Your Data Won't Mix. *Queue*, *3*(8), 50–58. doi:http://doi.acm.org/10.1145/1103822.1103836

Hohpe, G. (2006). 06291 Workshop Report: Conversation Patterns. In F. et al. Leymann (Ed.), *The Role of Business Processes in Service Oriented Architectures. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI).* Schloss Dagstuhl, Germany.

Hohpe, G., & Woolf, B. (2003). *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Hu, D., & Zhao, J. L. (2008). A Comparison of Evaluation Networks and Collaboration Networks in Open Source Software Communities. *14th Americas Conference on Information Systems (AMCIS 2008)* (pp. 1–8). Toronto, ON, Canada: Association for Information Systems (AIS).

Hull, R., & King, R. (1987). Semantic database modeling: survey, applications, and research issues. *ACM Comput. Surv.*, *19*(3), 201–260.

Humphrey, W. (1996). Using A Defined and Measured Personal Software Process. *IEEE Software*, *13*, 77–88.

Humphrey, W. (2000a). The personal software process: status and trends. *Software, IEEE*, *17*(6), 71–75.

Humphrey, W. (2000b). *The Team Software Process (TSP)*. Pittsburgh, Pennsylvania 15213.

Humphrey, W., Chick, T. A., Nichols, W. R., & Pomeroy-Huff, M. (2010). *Team Software Process (TSP) Body of Knowledge (BOK)* (pp. 1–150).

IEEE. (2007). IEEE Recommended Practice for CASE Tool Interconnection - Characterization of Interconnections. *IEEE Std 1175.2-2006*, 1–45 ST – IEEE Recommended Practice for CASE Tool.

IEEE. (2011). IEEE Guide--Adoption of the Project Management Institute (PMI(R)) Standard A Guide to the Project Management Body of Knowledge (PMBOK(R) Guide)--Fourth Edition. *IEEE Std 1490-2011*. doi:10.1109/IEEESTD.2011.6086685

Issarny, V., Caporuscio, M., & Georgantas, N. (2007). A Perspective on the Future of Middleware-based Software Engineering. *2007 Future of Software Engineering, International Conference on Software Engineering* (pp. 244–258). Washington, DC.

Jakoubi, S., & Tjoa, S. (2009). A reference model for risk-aware business process management. *Fourth International Conference on Risks and Security of Internet and Systems (CRiSIS 2009)* (pp. 82–89).

Johnson, P. (2001). Project Hackystat: Accelerating adoption of empirically guided software development through non-disruptive, developer-centric, in-process data collection and analysis. Honolulu, HI: Department of Information and Computer Sciences, University of Hawaii.

Juristo, N., & Moreno, A. (2001). *Basics of Software Engineering Experimentation*. Boston, MA: Kluwer Academic Publishers.

Kitchenham, B. (2007). *Guidelines for performing Systematic Literature Reviews in Software Engineering*.

Kitchenham, B., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., Emam, K. E., & Rosenberg, J. (2002). Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering*, *28*(8), 721–734.

Kontio, J. (1999). Risk Management in Software Development: a technology overview and the RiskIt method. *21st ICSE conference* (pp. 679–680).

Lawrence, P. (1997). *Workflow Handbook 1997* (p. 508). Chichester, UK, UK: John Wiley & Sons, Ltd.

Luckham, D. (2002). *The Power of Events* (p. 376). Boston, MA, USA: Addison-Wesley.

Lüder, A., Peschke, J., & Reinelt, D. (2006). Possibilities and Limitations of the Application of Agent Systems in Control. *International Conference On Concurrent Enterprising (ICE)*.

Merdan, M., Moser, T., Wahyudin, D., & Biffl, S. (2008). Performance evaluation of workflow scheduling strategies considering transportation times and conveyor failures. *IEEE International Conference on Industrial Engineering and Engineering Management (IEEM 2008)* (pp. 389–394).

Mockus, A., Fielding, R. T., & Herbsleb, J. (2000). A case study of open source software development: the Apache server. *22nd International Conference on Software Engineering*. Limerick, Ireland: ACM. doi:http://doi.acm.org/10.1145/337180.337209

Mockus, A., Fielding, R. T., & Herbsleb, J. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Trans. Softw. Eng. Methodol.*, *11*(3), 309–346. doi:http://doi.acm.org/10.1145/567793.567795

Mordinyi, R., Moser, T., Kühn, E., Biffl, S., & Mikula, A. (2009). Foundations for a Model-Driven Integration of Business Services in a Safety-Critical Application Domain. *35th Euromicro Conference on Software Engineering and Advanced Applications (Euromicro SEAA 2009)* (pp. 267–274). Patras, Greece.

Mordinyi, R., Pacha, A., & Biffl, S. (2011). Quality Assurance for Data from Low-Tech Participants in Distributed Automation Engineering Environments. In Z. Mammeri (Ed.), *Proceeding of the 16th IEEE International Conference on Emerging Technologies and Factory Automation* (pp. 1–4). doi:10.1109/ETFA.2011.6059149

Moser, T. (2009). *Semantic Integration of Engineering Environments Using an Engineering Knowledge Base. Faculty of Informatics*. Vienna University of Technology, Vienna, Austria.

Moser, T., & Biffl, S. (2010). Semantic Tool Interoperability for Engineering Manufacturing Systems. *15th IEEE International Conf. on Emerging Technologies and Factory Automation (ETFA)*. Bilbao, Spain.

Moser, T., Biffl, S., Sunindyo, W. D., & Winkler, D. (2010). Integrating Production Automation Expert Knowledge Across Engineering Stakeholder Domains. In L. Barolli, F. Xhafa, S. Vitabile, & H.-H. Hsu (Eds.), *Proceedings of the 4th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2010)*. IEEE Computer Society.

Moser, T., Biffl, S., Sunindyo, W. D., & Winkler, D. (2011). Integrating Production Automation Expert Knowledge Across Engineering Domains. *International Journal of Distributed Systems and Technologies (IJDST)*, *2*(3), 88–103.

Moser, T., Mordinyi, R., Mikula, A., & Biffl, S. (2009). Making Expert Knowledge Explicit to Facilitate Tool Support for Integrating Complex Information Systems in the ATM Domain. *International Conference on Complex, Intelligent and Software Intensive Systems (CISIS'09)* (pp. 90–97). Fukuoka, Japan: IEEE Computer Society.

Moser, T., Mordinyi, R., Winkler, D., & Biffl, S. (2011). Engineering project management using the Engineering Cockpit: A collaboration platform for project managers and engineers. *IEEE 9th International Conference on. Industrial Informatics (INDIN'2011)* (pp. 579–584). doi:10.1109/INDIN.2011.6034943

Moser, T., Waltersdorfer, F., Winkler, D., & Biffl, S. (2011). Version Management and Conflict Detection across Tools in a (Software+) Engineering Environment. *Proceedings of the Software Quality Days 2011* (pp. 1–4).

Noy, N. F., Doan, A. H., & Halevy, A. (2005). Semantic Integration. *AI Magazine*, *26*, 7–10.

Noy, N. F., & McGuinness, D. L. (2001). Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory, Stanford Medical Informatics.

Rademakers, T., & Dirksen, J. (2008). *Open-source ESBs in action*. Manning Publications.

Rangan, R. M., Rohde, S. M., Peak, R., Chadha, B., & Bliznakov, P. (2005). Streamlining Product Lifecycle Processes: A Survey of Product Lifecycle Management Implementations, Directions, and Challenges. *Journal of Computing and Information Science in Engineering*, *5*, 227–237 ST – Streamlining Product Lifecycle Proce.

Rozinat, A, & van der Aalst, W. (2008). Conformance checking of processes based on monitoring real behavior. *Information Systems*, *33*(1), 64–95.

Rozinat, Anne, & van der Aalst, W. (2006). Conformance Testing: Measuring the Fit and Appropriateness of Event Logs and Process Models. In C. Bussler & A. Haller (Eds.), *Business Process Management Workshops* (Vol. 3812, pp. 163–176). Springer Berlin / Heidelberg. doi:10.1007/11678564_15

S. Heinonen. (2006). *Requirements Management Tool Support for Software Engineering in Collaboration*. University of Oulu.

S. Heinonen, Kääriäinen, J., & Takalo, J. (2007). Challenges in Collaboration: Tool Chain Enables Transparency Beyond Partner Borders. *Enterprise Interoperability II*, 529–540.

Sadiq, S., Orlowska, M., Sadiq, W., & Foulger, C. (2004). Data flow and validation in workflow modelling. *15th Australasian database conference (ADC 2004)* (pp. 207–214).

Schafer, W., & Wehrheim, H. (2007). The Challenges of Building Advanced Mechatronic Systems. *Future of Software Engineering (FOSE '07)* (pp. 72–84). Washington, DC, USA: IEEE Computer Society. doi:DOI=10.1109/FOSE.2007.28 http://dx.doi.org/10.1109/FOSE.2007.28

Sharma, S., Sugumaran, V., & Rajagopalan, B. (2002). A framework for creating hybrid-open source software communities. *Information Systems Journal*, *12*(1), 7–25.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2010). *Database System Concepts* (p. 1376). McGraw-Hill.

Simpson, J., & Weiner, E. (1989). *The Oxford English Dictionary* (p. 22000). Oxford University Press.

Sunindyo, W. D. (2011). Observing and Validating Heterogeneous Workflows in Multidisciplinary Engineering Environments. In D. Dhungana (Ed.), *Proceedings of International Doctoral Symposium on Software Engineering and Advanced Applications (IDoSEAA 2011)* (pp. 1–7).

Sunindyo, W. D., Moser, T., Dhungana, D., Winkler, D., & Biffl, S. (2012). Improving Open Source Software Process Quality based on Defect Data Mining. In S. Biffl, D. Winkler, & J. Bergsmann (Eds.), *Software Quality Days 2012 - Research Track* (pp. 84–102). Vienna, Austria: Springer-Verlag. doi:10.1007/978-3-642-27213-4

Sunindyo, W. D., Moser, T., & Winkler, D. (2012). Process Model Validation for Heterogeneous Engineering Environments. *Software Quality Days 2012 - Practical Track*. Vienna, Austria.

Sunindyo, W. D., Moser, T., Winkler, D., & Biffl, S. (2010a). Foundations for Event-Based Process Analysis in Heterogeneous Software Engineering Environments. *Proc. 36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2010)* (pp. 313–322).

Sunindyo, W. D., Moser, T., Winkler, D., & Biffl, S. (2010b). A Process Model Discovery Approach for Enabling Model Interoperability in Signal Engineering. In J. Bezivin, R. M. Soley, & A. Vallecillo (Eds.), *Proceedings of the First International Workshop on Model-Driven Interoperability (MDI 2010)* (pp. 15–21). ACM Press.

Sunindyo, W. D., Moser, T., Winkler, D., & Biffl, S. (2012). Analyzing OSS Project Health with Heterogeneous Data Sources. *IJOSSP SEPA*, 1–23.

Sunindyo, W. D., Moser, T., Winkler, D., & Mordinyi, R. (2013). Project Progress and Risk Monitoring in Automation Systems Engineering. *Software Quality Days 2013* (p. 24). Vienna, Austria.

Sunindyo, W. D., Moser, T., Winkler, D., Mordinyi, R., & Biffl, S. (2011). Workflow Validation Framework in Distributed Engineering Environments. *Proceedings of 3rd International Workshop on Information Systems in Distributed Environment (ISDE'11)* (pp. 1–10).

T. Muller, & Knoll, A. (2009). Virtualization Techniques for Cross Platform Automated Software Builds, Tests and Deployment. *Fourth International Conference on Software Engineering Advances (ICSEA '09)* (pp. 73–77).

Terzic, I., Merdan, M., Zoitl, A., & Hegny, I. (2008). Modular assembly machine - ontology based concept. *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2008)* (pp. 241–244).

Thalheim, B. (2000). *Entity-Relationship Modeling: Foundations of Database Technology* (p. 627). Berlin, Heidelberg: Springer.

Torii, K., Kenichi, M., Kumiyo, N., Yoshihiro, T., Shingo, T., Kazuyuki, S., & Kenichi, M. (1999). Ginger2: An Environment for Computer-Aided Empirical Software Engineering. *IEEE Transactions on Software Engineering*, *25*, 474–492.

Van Solingen, R., & Berghout, E. (1999). *The Goal/Question/Metric Method*. McGraw-Hill Education.

Wahyudin, D. (2008). *Quality Prediction and Evaluation Models for Products and Processes in Distributed Software Development*. Vienna University of Technology.

Wahyudin, D., Mustofa, K., Schatten, A., Biffl, S., & Tjoa, A. M. (2007). Monitoring the "health" status of open source web-engineering projects. *International Journal of Web Information Systems*, *3*(1/2), 116–139.

Wahyudin, D., Schatten, A., Mustofa, K., Biffl, S., & Tjoa, A. M. (2006). Introducing "Health" Perspective in Open Source Web-Engineering Software Projects, Based on Project Data Analysis. *International Conference on Information Integration, Web-Applications and Services (IIWAS 2006)*. Yogyakarta Indonesia: Austrian Computer Society.

Weijters, A. J. M. M., & Ribeiro, J. T. S. (Joel). (2010). *HeuristicsMiner 6.0: Users Guide*. Eindhoven, The Netherlands: Department of Technology Management, Eindhoven University of Technology.

Weijters, A. J. M. M., van der Aalst, W., & de Medeiros, A. K. A. (2006). *Process Mining with the HeuristicsMiner Algorithm*. *BETA Working Paper Series*. Eindhoven: Eindhoven University of Technology.

Weinberg, G. M. (1993). *Quality Software Management* (p. 346). New York, USA: Dorset House Publishing.

Winkler, D., & Biffl, S. (2012). Improving Quality Assurance in Automation Systems Development Projects. In P. M. Savsar (Ed.), *Quality Assurance and Management* (pp. 20–40). Intec Publishing. doi:10.5772/33487

Winkler, D., Moser, T., Mordinyi, R., Sunindyo, W. D., & Biffl, S. (2011). Engineering Object Change Management Process Observation in Distributed Automation Systems Projects. *Proceedings of 18th European System & Software Process Improvement and Innovation (EuroSPI 2011)* (pp. 1–12).

Wohlin, C., Runeson, P., Höst, M., Ohlsson, N., Regnell, B., & Wesslen, A. (2000). Experimentation in Software Engineering - An Introduction. *The Kluwer International Series in Software Engineering*. Kluwer Academic Publishers.

Yin, J., Chen, H., Deng, S., Wu, Z., & Pu, C. (2009). A Dependable ESB Framework for Service Integration Internet Computing. *IEEE*, *13*, 26–34.

de Medeiros, A. K. A., & Weijters, A. J. M. M. (2009). *ProM Framework Tutorial* (p. 47). Eindhoven, The Netherlands.

van Dongen, B. F., & van der Aalst, W. (2005). A Meta Model for Process Mining Data. *CAiSE'05 WORKSHOPS* (pp. 309–320).

van der Aalst, W. (2005). Business Alignment: Using Process Mining as a Tool for Delta Analysis and Conformance Testing. *RE Journal*, *10*(3), 198–211.

van der Aalst, W. (2011). *Process Mining - Discovery, Conformance and Enhancement of Business Processes* (p. 352). Heidelberg: Springer.

van der Aalst, W., Weijters, A. J. M. M., & Maruster., L. (2004). Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, *16*(9), 1128–1142.

Wikan Danar Sunindyo Curriculum Vitae

Wikan Danar Sunindyo, M.Sc. conducted his PhD at the Institute of Software Technology and Interactive Systems (ISIS) with an Indonesian Directorate General of Higher Education scholarship since November 2008. He received his master degree in Computational Logic at the Dresden University of Technology, Germany in 2007. He received his bachelor degree in Informatics at the Bandung Institute of Technology (ITB), Indonesia in 2000. Later, he works as a PhD researcher at TU Vienna, Austria in the research area “Complex Systems” since 2008. His main research areas include Open Source Software, automation systems, process observation and analysis, and semantic web technologies to better integrate heterogeneous engineering environments. He is also an associated researcher at the Christian Doppler Laboratory for “Software Engineering Integration for Flexible Automation Systems” (CDL-Flex).

Personal Data

Name	: Wikan Danar Sunindyo
Date of Birth	: January 10th, 1977
Sex	: Male
Place of Birth	: Magelang, Indonesia
Nationality	: Indonesian
Current position	: Ph.D student at Vienna University of Technology Institute of Software Technology and Interactive Systems

Postal Address

Street, Number	: Dürergasse 7/12
Postal Code, City	: A-1060 Vienna
Country	: Austria
Email	: wikan.danar@gmail.com
Phone	: +436505870648
Website	: http://www.isis.tuwien.ac.at/wikan

Education

University (postgraduate)
Master Program in Computational Logic
Faculty of Informatics
Dresden University of Technology
Dresden, Germany
2004 - 2007

University (undergraduate)
Department of Informatics
Bandung Institute of Technology (ITB)
Bandung, Indonesia
1995 – 2000

Research Experience

Christian Doppler Laboratory
Software Engineering Integration for Flexible Automation Systems
Vienna University of Technology (TU Wien)
Vienna, Austria
as Associated Researcher
2010 - now

Program Insentif
(*Incentive Program*)
Open Source Software Competence Center
Bandung Institute of Technology (ITB)
Bandung, Indonesia
as Researcher
2007 - 2008

Master Thesis Research
Title: Toward Software Transactional Memory and Mobility
Master Program in Computational Logic
Dresden University of Technology
Dresden, Germany
2006 – 2007

Project Research
Title: An Introductory Tutorial for the use of Mobility Workbench
Master Program in Computational Logic
Technische Universität Dresden
2006

Riset Unggulan Strategis Nasional (Rusnas)
(*National Research of Excellence and Strategic Product*)
on Software Component Development
as Software Developer
2001 – 2004

Riset Unggulan Terpadu (RUT) IX
(*Integrated Excellence Research*)
on Format Integrator Development
Bandung Institute of Technology (ITB)
Bandung, Indonesia
as Researcher
2001 - 2004

Publications

Book Chapters

1. **Wikan Danar Sunindyo**, Thomas Moser, Dietmar Winkler, Richard Mordinyi, Stefan Biffl (2012) Workflow Validation Framework in Collaborative Engineering Environments. In Distributed Computing Innovations for Business, Engineering and Science (in print).
2. Thomas Moser, Richard Mordinyi, **Wikan Danar Sunindyo**, Stefan Biffl (2010) Semantic Service Matchmaking in the ATM Domain Considering Infrastructure Capability Constraints, 133-157. In Canadian Semantic Web: Technologies and Applications.

Journal Papers

1. **Wikan Danar Sunindyo**, Thomas Moser, Dietmar Winkler, Stefan Biffl (2011) Analyzing OSS Project Health with Heterogeneous Data Sources, 1-23. In International Journal of Open Source Software and Pocesses (IJOSSP), October-December 2011, Vol. 3, No. 4 (in print).
2. Thomas Moser, Stefan Biffl, **Wikan Danar Sunindyo**, Dietmar Winkler (2010) Integrating Production Automation Expert Knowledge across Engineering Domains, 1-15. In International Journal of Distributed Systems and Technologies (IJDST).

Conference Papers

1. **Wikan Danar Sunindyo**, Thomas Moser, Dietmar Winkler et al. (2013) Project Progress and Risk Monitoring in Automation Systems Engineering, 1-24. In Software Quality Days 2013 (to be published).
2. **Wikan Danar Sunindyo**, Thomas Moser, Dietmar Winkler, Deepak Dhungana (2012) Improving Open Source Software Process Quality based on Defect Data Mining, 84-102. In Software Quality Days 2012.
3. Dietmar Winkler, Thomas Moser, Richard Mordinyi, **Wikan Danar Sunindyo**, Stefan Biffl (2011) Engineering Object Change Management Process Observation in Distributed Automation Systems Projects, 1-12. In Proceedings of 18th European System & Software Process Improvement and Innovation (EuroSPI 2011).

4. Inah Omoronyia, G Sindre, T Stalhane, Stefan Biffl, Thomas Moser, **Wikan Danar Sunindyo** (2010) A Domain Ontology Building Process for Guiding Requirements Elicitation, 188-202. In Proc. 16th International Working Conference on Requirements Engineering (RefsQ 2010).
5. Stefan Biffl, **Wikan Danar Sunindyo**, Thomas Moser (2010) A Project Monitoring Cockpit Based On Integrating Data Sources in Open Source Software Development, 620-627. In Proc. Twenty-Second International Conference on Software Engineering and Knowledge Engineering (SEKE 2010).
6. **Wikan Danar Sunindyo**, Thomas Moser, Dietmar Winkler, Stefan Biffl (2010) Foundations for Event-Based Process Analysis in Heterogeneous Software Engineering Environments, 313-322. In Proc. 36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2010).
7. Thomas Moser, Stefan Biffl, **Wikan Danar Sunindyo**, Dietmar Winkler (2010) Integrating Production Automation Expert Knowledge Across Engineering Stakeholder Domains. In Proceedings of the 4th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2010).
8. Stefan Biffl, **Wikan Danar Sunindyo**, Thomas Moser (2010) Semantic Integration of Heterogeneous Data Sources for Monitoring Frequent-Release Software Projects, 360-367. In 2010 International Conference on Complex, Intelligent and Software Intensive Systems.
9. Stefan Biffl, **Wikan Danar Sunindyo**, Thomas Moser (2009) Bridging Semantic Gaps Between Stakeholders in the Production Automation Domain with Ontology Areas, 233-239. In 21st International Conference on Software Engineering & Knowledge Engineering (SEKE 2009).
10. Thomas Moser, Richard Mordinyi, **Wikan Danar Sunindyo**, Stefan Biffl (2009) Semantic Service Matchmaking in the ATM Domain Considering Infrastructure Capability Constraint, 222-227. In Proceedings The 21st International Conference on Software Engineering & Knowledge Engineering (SEKE 2009).

Workshop Papers

1. **Wikan Danar Sunindyo**, Thomas Moser, Dietmar Winkler (2012) Process Model Validation for Heterogeneous Engineering Environments. In Software Quality Days 2012 - Practical Track.
2. **Wikan Danar Sunindyo**, Martin Melik-Merkumians, Thomas Moser, Stefan Biffl (2011) Enforcing Safety Requirements for Industrial Automation System at Runtime - Position Paper, 37-42. In Proceedings of 2nd International Workshop on Requirements@Run.Time (RE@RunTime 2011).
3. **Wikan Danar Sunindyo** (2011) Observing and Validating Heterogeneous Workflows in Multidisciplinary Engineering Environments, 1-7. In Proceedings of International Doctoral Symposium on Software Engineering and Advanced Applications (IDoSEAA 2011).
4. Thomas Moser, **Wikan Danar Sunindyo**, Munir Merdan, Stefan Biffl (2011) Supporting Runtime Decision Making in the Production Automation Domain Using Design Time Engineering Knowledge, 9-22. In Proceedings of 1st International Workshop on Ontology and Semantic Web for Manufacturing (OSEMA 2011).
5. **Wikan Danar Sunindyo**, Stefan Biffl (2011) Validating Process Models in Systems Engineering Environments, 25-32. In Proceedings of the Workshop on Industrial Automation Tool Integration Project Automation.
6. **Wikan Danar Sunindyo**, Thomas Moser, Dietmar Winkler, Richard Mordinyi, Stefan Biffl (2011) Workflow Validation Framework in Distributed Engineering

Environments, 1-10. In Proceedings of 3rd International Workshop on Information Systems in Distributed Environment (ISDE'11).

7. **Wikan Danar Sunindyo**, Thomas Moser, Dietmar Winkler, Stefan Biffl (2010) A Process Model Discovery Approach for Enabling Model Interoperability in Signal Engineering, 15-21. In Proceedings of the First International Workshop on Model-Driven Interoperability (MDI 2010).
8. **Wikan Danar Sunindyo**, Stefan Biffl, Richard Mordinyi, Thomas Moser, Alexander Schatten, Mohammad Tabatabai Irani, Dindin Wahyudin, Edgar Weippl, Dietmar Winkler (2010) An Event-Based Empirical Process Analysis Framework, 1-2. In Proc. 4th International Symposium on Empirical Software Engineering and Measurement (ESEM 2010) - Poster Sessions.
9. **Wikan Danar Sunindyo**, Stefan Biffl (2010) Bridging Semantic Heterogeneities in Open Source Software Development Projects with Semantic Web Technologies, 285-286. In Proceedings of the Junior Scientist Conference 2010.
10. **Wikan Danar Sunindyo**, Stefan Biffl, Christian Frühwirth, Richard Mordinyi, Thomas Moser, Alexander Schatten, Sebastian Schrittwieser, Edgar Weippl, Dietmar Winkler (2010) Defect Detection Using Event-Based Process Analysis in (Software+) Engineering Projects, 1-2. In Proc. 36th Euromicro Conference Software Engineering and Advanced Applications (SEAA 2010) - Work in Progress Session.
11. **Wikan Danar Sunindyo** (2010) Observability of Software Engineering Processes in Open Source Software Projects Domain, 1-9. In Proceedings of the 5th International Doctoral Symposium on Empirical Software Engineering.
12. Christian Frühwirth, Stefan Biffl, Alexander Schatten, Dietmar Winkler, **Wikan Danar Sunindyo** (2010) Quantitative Software Security Measurement in an Engineering Service Bus Platform, 1-2. In Proc. 4th International Symposium on Empirical Software Engineering and Measurement (ESEM 2010) - Poster Sessions.
13. Christian Frühwirth, Stefan Biffl, Alexander Schatten, Sebastian Schrittwieser, Edgar Weippl, **Wikan Danar Sunindyo** (2010) Research Challenges in the Security Design and Evaluation of an Engineering Service Bus Platform, 1-2. In Proc. 36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA) - Work in Progress Session.

Work Experience

Department of Informatics
Bandung Institute of Technology (ITB)
Bandung, Indonesia
as lecturer staff
2001 - 2008

Tim Informatika (Workshop)
Bandung Institute of Technology (ITB)
Bandung, Indonesia
as staff
2000 – 2001

Quantum E-Commerce College
Bandung, Indonesia
as lecturer staff
2000 – 2001

Diploma Program PT Pos Indonesia
under supervision of Department of Informatics
Bandung Institute of Technology (ITB)
Bandung, Indonesia
as lecturer staff
2000

Diploma Program PT Pos Indonesia
under supervision of Department of Informatics
Bandung Institute of Technology (ITB)
Bandung, Indonesia
as teaching assistant
1998 – 1999

Department of Informatics
Bandung Institute of Technology (ITB)
Bandung, Indonesia
as teaching assistant
1997 - 1999

Professional Experience

Lembaga Penelitian dan Pemberdayaan Masyarakat (LPPM)
(Institute of Research and Community Development)
Bandung Institute of Technology (ITB)
Description: development of official website of LPPM ITB, http://www.lppm.itb.ac.id
as Software Developer
2002 - 2004

Department of Culture and Tourism
The Republic of Indonesia
Description: development of official website of Department of Culture and Tourism of the Republic of Indonesia, http://www.depbudpar.go.id
as Software Developer
2001 – 2002

Department of Marine Exploration and Fishery
General Directorate of Caught Fishery
The Republic of Indonesia
Description: development of fishery information system based on fish resources and market demand, http://www.pelabuhanperikanan.or.id
as Software Developer
2000 – 2001

PT Sumarno Pabottingi
(private company)
Description: development of e-commerce website, http://www.aspri.net
as Software Developer
2000 – 2001

PT Telkom
(Indonesian National Telecommunication Company)
ProBIS division
Description: development of web-based application, RPL On Line
as Software Developer
1998

PT Schlumberger Geophysics Nusantara
Description: development of official website of PT Schlumberger Geophysics Nusan-
tara, Batam Supply Base
as Software Developer
1998

Technical Expertise

Programming Language / Tools
Basic, Pascal, C, Delphi, Fortran, LISP, Prolog

Database
Oracle, MS Access, MS SQL Server, PostgreSQL, MySQL

Operating System
Linux, Windows

Miscellaneous
Power Designer

Language

Mother Tongue: Bahasa Indonesia, Javanese
Foreign: English (good), German (basic)

Vienna, November 26th, 2012

Wikan Danar Sunindyo