Foundations for Event-Based Process Analysis in Heterogeneous Software Engineering Environments

Wikan Danar Sunindyo, Thomas Moser, Dietmar Winkler, Stefan Biffl
Christian Doppler Laboratory for
Software Engineering Integration for Flexible Automation Systems
Vienna University of Technology, Austria
{wikan, tmoser, winkler, biffl}@ifs.tuwien.ac.at

Abstract

For monitoring, controlling, and improving software development projects, project and quality managers need tool support to analyze engineering processes within development environments. Unfortunately, technical and semantic gaps between the engineering tools and related data models make it hard to observe and analyze the implemented tool-based engineering processes. In this paper we build on a service-oriented platform for technically and semantically integrating heterogeneous engineering tools and propose an approach to monitor, analyze, and improve tool-based engineering processes. We empirically evaluate the approach using the "continuous integration and test" process and discuss strengths and limitations. Major result was that the approach enabled comparing expected and real-life engineering processes with respect to process structure, performance of individual process steps, and risk of bottlenecks.
Keywords - software process, software validation and verification.

1. Introduction

Modern software environments often involve a wide range of engineering systems and tools that use different technical platforms and heterogeneous data models. Unfortunately, technical and semantic gaps between the engineering tools and their data models make it hard to observe and analyze the actual tool-based engineering processes. Current process analysis methods and techniques take high manual effort and are error-prone, often resulting in their negligence; for example using meta models for process mining, data handling requires additional effort for transforming implemented real data structures to meta models of process analysis tools [1]. Therefore, the tool-supported observation of software processes can benefit from effective approaches for technical and software integration that bridge technical and semantic gaps between the engineering tools and their data models. Technical gaps originate from heterogeneous tools and systems that use a variety of platforms, protocols, and data formats to communicate. Semantic gaps come from using different data formats, data structures, and terminologies for common concepts that are used in the communication between the tools [2]. Several approaches have been reported to integrate engineering tools and systems along the software life cycle [3].

We recently introduced the concept of the Engineering Service Bus (EngSB) [2], which builds on the Enterprise Service Bus concept (ESB) [4] and focuses on the technical integration of tools and systems along the Software Engineering (SE) lifecycle. The EngSB supports tools as components in SE processes, domain-specific services, and flexible software process prototyping. However, the EngSB needs to be extended to efficiently support monitoring and analyzing of software processes based on the events raised by the tools connected by the EngSB. The events can be used for process analysis and improvement by enabling performance analyses with focus on possible bottlenecks of processes based on the analysis on the collected events.

In this paper, we extend the concept of EngSB by adding semantic integration features on heterogeneous event log data. On the integrated event log data we can build the process monitoring and process analysis features in conceptual layers based on the technical integration. We build on the EngSB to provide foundations for event-based software engineering process analysis, and propose an approach to monitor,

analyze and improve tool-based engineering processes. By defining and applying a workflow model on the EngSB platform, we can support the execution of typical engineering processes and monitor these processes by analyzing the event log of the EngSB with process mining techniques. For example, usually the normal process will follow the workflow model as designed, but some deviations may occur during running the system. These exceptions can be captured in the event log that will be further analyzed by using process mining tool producing an actual process model. These foundations allow difference analyses of designed and real-life engineering processes, as well as performance analyses in order to identify potential bottleneck engineering process steps. We evaluate the approach with a real-world use case from agile SE projects and discuss strengths and limitations of the approach. We analyze the event logs coming from defined use cases and scenarios, e.g., the standard "continuous integration and test" (CI&T) process that involves development and testing processes of software [2].

The remainder of this paper is structured as follows: Section 2 summarizes related work on system integration technologies and on process analysis and validation. Section 3 identifies the research issues and motivates the research method. Section 4 develops the solution approach to integrate heterogeneous SE environments and enable the event-based analysis of SE processes. Section 5 describes the initial empirical evaluation based on the CI&T process. Section 6 discusses lesson learned and Section 7 concludes the paper.

2. Related Work

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.

2.1. 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 [5]. 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 [6]. 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 [4]. The ESB separates the business logic from the integration logic and provides a distributed integration platform. MULE (http://www.mulesoft.org) 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 [7]. 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 [2].

Semantic integration is defined as the solving of problems resulting from the intent to share data across disparate and semantically heterogeneous data [8]. 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 [9]. 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 [10]. The application of ontologies as semantic web technologies for managing knowledge in specific domains is inevitable. Noy and Guinness [11] 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.* [12] 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 [12] and also SE. Further, Biffl *et al.* [19] 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 [13, 14] 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 [2]. 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 1 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.

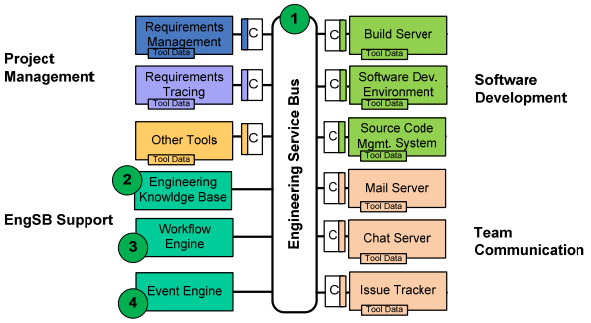

Figure 1. High-level view on tool connections with the EngSB [2].

2.2. 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 [15].

Ferreira and Ferreira [16] 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. [15]. 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 (http://www.processmining.org) 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 [17] 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 [18] introduced complex event processing (CEP) 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.

3. Research Issues

Project and quality managers need to understand the actual processes conducted in the engineering team. Unfortunately, technical and semantic gaps between the engineering tools and their data models make it hard to observe and analyze the actual tool-based software engineering processes. From this challenge, we derive the following research issues to provide a foundation for event-based analysis of SE processes.

RI-1: Observation of tool-based engineering processes. The tools that support SE processes already provide a wealth of events. However, semantic gaps between the engineering tools data models hinder effectively using these tool contributions from an engineering process point of view. Event recording should be on processes rather than on tool level. The EngSB can provide a foundation to connect engineering tools for collecting events on process level rather than for each individual tool. Additionally, the EngSB also supports logging of manual process steps using engineering tickets, e.g., in Trac (http://trac.edgewall.org/). Therefore, the first research issue investigates options to design an observation platform that collects events from tools in a SE environment. A key question is how to integrate the events from a range of heterogeneous tools into a consolidated event framework for further analysis of SE processes.

RI-2: Process analysis based on integrated data. The capability to observe tool-based engineering processes can enable model-based analysis of engineering processes in order to compare expected and actual process variants as foundations for planning and tracking the improvement of engineering processes. Thus, a key question is which kinds of relevant SE processes can be observed with an event-based platform in sufficient detail to support SE process analysis and improvement.

We will discuss the requirements for SE process analysis and improvement and demonstrate the approach in a research prototype using the Continuous Integration and Test (CI&T) process [19] to show the capability of performing process analysis and improvement. The CI&T process will implement continuous processes of applying quality control frequently, which consists of numbers of build, test and deploy of different software modules which will be integrated with existing modules. This empirical experience can later be used for more complex processes such as change management across tools and data models for multi-disciplinary engineering projects. The process analysis focuses on the comparison of planned and real projects, which are hard to observe in practice. Note that these observations are the foundations for process improvement based on measurement and data. The configuration of event collection will be discussed in Section 4.

Requirements for SE process analysis and improvement are as follows. First, design events as system outputs and collect all events from the system for further usage. Second, filter and transform collected events to the format of the process analysis tool, so that they can be used for further analysis. Third, define what kind of process analysis methods we use for analyzing event logs. Fourth, use the process analysis tool for analyzing event logs and use the results for improving processes.

4. Solution Approach

This section describes the platform architecture, the analysis and improvement process and the event log transformation for process measurement, analysis, and improvement.

4.1. Platform Architecture for Technical/ Semantic Integration

Figure 2 illustrates the platform architecture, consisting of three layers, (a) tool level, (b) process level, and (c) process analysis and improvement level.

In order to analyze engineering processes on a SE process level rather than on the level of the individual participating engineering tools, all important events (related to the specified engineering process) need to be identified and collected. The Engineering Service Bus (EngSB) bridges technical gaps between engineering models and tools for quality and process improvements for SE and engineering disciplines that interact with SE [2]. All events produced or consumed by engineering tools connected to the EngSB use the bus as communication platform, and therefore enables central data collection and storage, e.g., in an event log database (as shown in Figure 2).

However, a major challenge in current industrial development and research approaches is insufficient semantic model integration between individual expert disciplines collaborating in a project [20]. Different and partly overlapping terminologies, used in individual expert disciplines, hinder understanding and collaboration between disciplines. Consequently, the weak tool support for semantic integration of the expert knowledge across domain boundaries hinders flexible engineering process automation and quality management, leading to development delays and risks for system operation. The EKB framework [12] addresses the challenges of semantic integration and enables efficient data exchange and collaboration. The general mechanism of the EKB framework uses common engineering concepts as foundation for mapping proprietary tool-specific engineering knowledge and (more generic) domain-specific engineering knowledge to support transformation between these engineering tools. The EKB framework acts as a component in the proposed EngSB, performing transformation services based on the messages transmitted on the EngSB.

The collection and storage of these events takes place on the process level rather than on the tool level. By using the EKB that transforms the tool-level event data into process-level event data, the EngSB can provide a basis for collecting events on process level rather than for each individual tool and in addition also supports the logging of manual process steps by e.g., engineering tickets.

4.2. Process Analysis Steps

Figure 2 presents event models on two different levels of events, (a) the tool level events and (b) the process level events. Tool level events focus on the collection of event data from different tools, e.g., build servers,

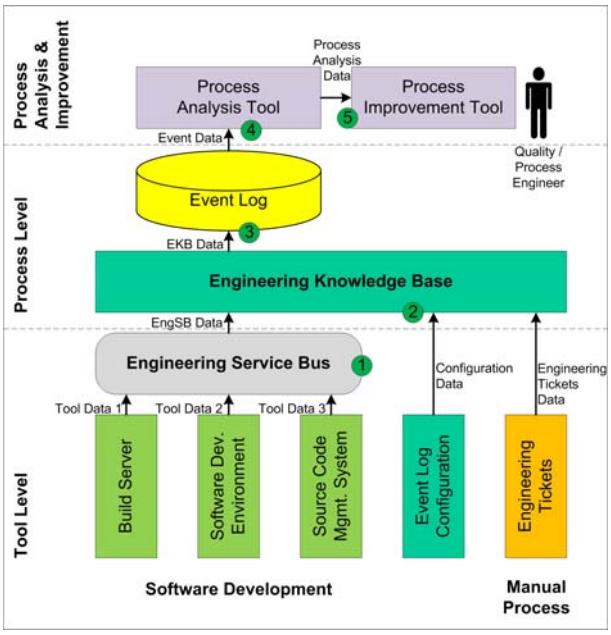

Figure 2. Platform architecture of process analysis and improvement.

software development environment, and source code management systems. On the process level, process level events integrate 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. We perform three types of analysis [21]: (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.

The analysis process consists of the following steps: (1) *process definition* based on available software development tools and their events, like build server, software development environment and source code management system via the Engineering Service Bus; (2) *event implementation* using Event Data Configuration and integrated data from the Engineering Knowledge Base, (3) *event observation* using event data storage; (4) *engineering process analysis* using a process analysis tool, e.g., based on *ProM*; and (5) *process improvement* based on the analysis results conducted by a quality/process engineer. These analysis processes contain the

capability to observe the tool-based engineering process that enables model-based analysis of the engineering process in order to compare expected and actual process variants as foundations for planning and tracking engineering process improvement.

4.3. Event Log Transformation for Process Analysis

The EngSB event log is the foundation for the process analysis. The event log is generated by the EngSB event engine. The structure of event logs produced by the EngSB event engine is illustrated in Figure 3. Activities in the EngSB are recorded in XML style as event logs, which consist of collections of XML files. These collections include attributes that explain the origin of the log, time of creation and modification, owner, type and information of activities, and the outcome of the activities.

Process mining is based on the minimal amount of information that needs to be present. The event log should follow these requirements i.e., each logged event should be an event that occurred at a *defined point in time*, each logged event should refer to *one single activity* only, each logged event should contain a *description of the event* that happened with respect to the activity, each logged event should refer to a *specific process instance* (case), and each process instance should *belong to a specific process*. ProM is an open-source tool for implementing process mining techniques in a standard environment, which allow the extraction of information from event logs. An example transformation result can be seen in Figure 4.

The MXML format consists of two parts, the source and processes. Each process consists of process instances that contain detailed audit trail entries. Each audit trail entry consists of workflow model element, event type, timestamp and originator. Additionally, we can add the "data" element for storing arbitrary textual data, and contains a list of Attribute" elements.

Every level can additionally store information regarding the environment the log was created in. The event model, integration platform, and process analysis method provide the foundation for demonstrating event-based analyses.

Figure 3. Structure of EngSB event logs.

5. Evaluation and Results

This section describes the design and results of the process analysis and evaluation 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 fulfills requirements defined in section 3 and follows process analysis and mining guidelines [1] 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.

```
<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 4. Transformation of event logs for further process analyses.

The expected SE process model for the CI&T use case implemented in EngSB systems in our lab as shown in Figure 5 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 [2]. 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.

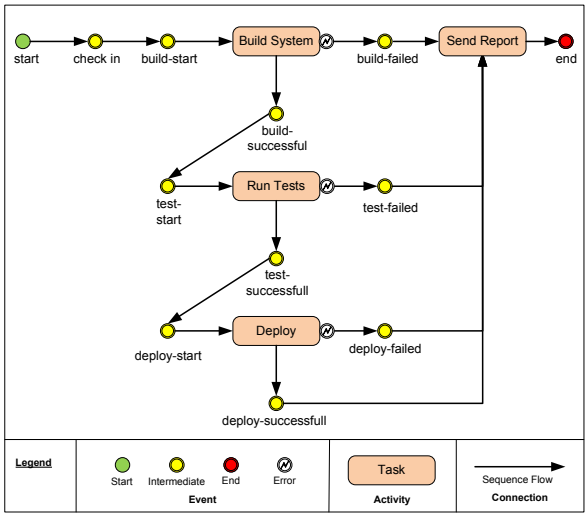

Figure 5. Model of the expected CI&T process.

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 5 are the model of expected CI&T process that is compared with the actual model derived from SE process-level events in Figure 6.

We collected 360 event log files from running a CI&T configuration of the EngSB system with the structure as shown in Figure 3. The collected event logs were transformed into MXML format as shown in Figure 4 for further analysis. From the transformed event logs, we performed process model verification and process performance analysis by using the ProM tool. The process model verification is done by comparing the real process with respect to the expected designed process. Figure 6 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 paper, we focus on the relevant performance information of the ProM plug-in: the place time metrics and activity metrics.

The *transition time metric*, based on the Petri Net model, is illustrated in Table 1. 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.

Figure 6 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 2. 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 6.

Table 1. Transition time metric for CI&T process steps and overall process.

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**

Table 2. Activity metric for the CI&T process steps.

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

6. Discussion

In this paper, 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 with respect to the introduced research issues.

RI-1: 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

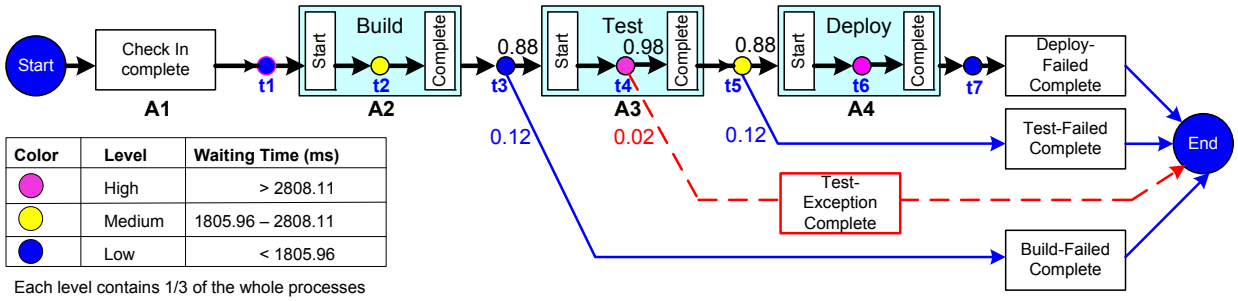

Figure 6. Performance analysis of 300+ CI&T process runs.

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.

RI-2: Process analysis based on the integrated data. For process analysis we focused in this paper 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. Conclusion and Further Work

Project and quality managers in SE projects need tool support to systematically analyze and improve software development projects. In this paper we

introduced a service-oriented platform based on the EngSB for integrating heterogeneous engineering tools and proposed an approach to monitor and analyze tool-based engineering processes. We initially evaluated the approach with the best-practice "continuous integration and test" process. We demonstrated the analysis of the tool-based engineering process 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.

Further work will be to analyze challenging engineering processes and environments in a range of engineering domains. 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.

Acknowledgments

This work has been supported by the Christian Doppler Forschungsgesellschaft and the BMWFJ, Austria. This work has been partially funded by the Vienna University of Technology, in the Complex Systems Design and Engineering Lab.

References

[1] B. F. van Dongen, and W. M. P. van der Aalst, "A Meta Model for Process Mining Data," in CAiSE'05 WORKSHOPS, 2005, pp. 309-320.

[2] S. Biffl, and A. Schatten, "A Platform for Service-Oriented Integration of Software Engineering Environments," in Eight Conference on New Trends in Software Methodologies, Tools and Techniques (SoMeT 09), 2009, pp. 75-92.

[3] I. Thomas, and B. A. Nejmeh, "Definitions of tool integration for environments," *IEEE Software,* vol. 9, pp. 29-35, 1992.

[4] D. A. Chappell, *Enterprise Service Bus - Theory in Practice*, p. 247, Sebastopol, CA, USA: O'Reilly, 2004.

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

[6] T. Muller, and A. Knoll, "Virtualization Techniques for Cross Platform Automated Software Builds, Tests and Deployment," in Fourth International Conference on Software Engineering Advances (ICSEA '09) 2009, pp. 73-77.

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

[8] A. Halevy, "Why your data won't mix," *Queue,* vol. 3, no. 8, pp. 50-58, 2005.

[9] N. F. Noy, A. H. Doan, and A. Y. Halevy, "Semantic Integration," *AI Magazine,* vol. 26, no. 1, pp. 7-10, 2005.

[10] A. Doan, N. F. Noy, and A. Y. Halevy, "Introduction to the special issue on semantic integration," *SIGMOD Rec.,* vol. 33, no. 4, pp. 11-13, 2004.

[11] N. F. Noy, and D. McGuinness, *Ontology Development 101: A Guide to Creating Your First Ontology*, Stanford Knowledge Systems Laboratory, 2001.

[12] T. Moser, S. Biffl, W. D. Sunindyo, and D. Winkler, "Integrating Production Automation Expert Knowledge Across Engineering Stakeholder Domains," in 4th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2010), Krakow, Poland, 2010, pp. 352-359.

[13] S. Heinonen, "Requirements Management Tool Support for Software Engineering in Collaboration," Master's Thesis, Department of Electrical and Information Engineering, University of Oulu, 2006.

[14] S. Heinonen, J. Kääriäinen, and J. Takalo, "Challenges in Collaboration: Tool Chain Enables Transparency Beyond Partner Borders," *Enterprise Interoperability II*, pp. 529-540, 2007.

[15] W. M. P. van der Aalst, A. J. M. M. Weijters, and L. Maruster., "Workflow Mining: Discovering Process Models from Event Logs," *IEEE Transactions on Knowledge and Data Engineering,* vol. 16, no. 9, pp. 1128-1142, 2004.

[16] D. M. R. Ferreira, and J. J. P. Ferreira, "Developing a reusable workflow engine," *J. Syst. Archit.,* vol. 50, no. 6, pp. 309-324, 2004.

[17] A. J. Rembert, and C. Ellis, "An initial approach to mining multiple perspectives of a business process," in 5th Richard Tapia Celebration of Diversity in Computing Conference: Intellect, Initiatives, Insight, and Innovations, Portland, Oregon, 2009, pp. 35-40.

[18] R. v. Ammon, C. Silberbauer, and C. Wolff, "Domain Specific Reference Models for Event Patterns - for Faster Developing of Business Activity Monitoring Applications," in VIPSI 2007, Lake Bled, Slovenia, 2007.

[19] P. Duvall, S. Matyas, and A. Glover, *Continuous Integration: Improving Software Quality and Reducing Risk*, p. 336: Addison-Wesley, 2007.

[20] W. Schäfer, and H. Wehrheim, "The Challenges of Building Advanced Mechatronic Systems," in 2007 Future of Software Engineering, 2007, pp. 72-84.

[21] W. M. P. van der Aalst, "Business Alignment: Using Process Mining as a Tool for Delta Analysis and Conformance Testing," *Requirements Engineering Journal,* vol. 10, no. 3, pp. 198-211, 2005.