Flexible Support for Adaptable Software and Systems Engineering Processes

Richard Mordinyi, Thomas Moser and Stefan Biffl
Christian Doppler Laboratory “Software Engineering Integration for Flexible Automation Systems”
Vienna University of Technology
Vienna, Austria
{*firstname.lastname*}@tuwien.ac.at

Deepak Dhungana
Siemens AG
Vienna, Austria
deepak.dhungana@siemens.com

***Abstract*—Typical complex software and systems engineering projects involve a set of different engineering disciplines and therefore heavily rely on systems integration approaches. Service-oriented architecture and Enterprise Service Bus provide a valuable basis for systems integration; however often depend on tedious manual work or invasive and therefore inflexible adaptations in case of even minimal changes to engineering processes or engineering tools. In this paper we describe the Engineering Service Bus (EngSB), an integration platform which not only integrates tools and systems, but also provides a bridge between engineering processes and existing engineering tools. We evaluate the EngSB by implementing a standard software engineering process for multi-engineering environments. Major results were that the EngSB allows for non-invasive and flexible integration of well-established tools in multi-engineering environments, while enabling the integration to take place on the level of engineering processes.**

Keywords-(software+) engineering, non-invasive integration, component-based architectures.

I. INTRODUCTION

Todays real-world and large-scale software and systems engineering projects such as the engineering of water power plants or steel mills typically involve a wide range of different engineering disciplines such as mechanical engineering, electrical engineering and software engineering [1]. Each of the disciplines already provides its specific engineering tools and engineering systems to manage specific engineering processes. Each of these processes heavily relies on different technical platforms and heterogeneous data models. Well-established software engineering methods and approaches (e.g., iterative development processes, issue tracking systems etc.) could provide additional value for building and managing such information systems engineering projects, but require careful integration and seamless collaboration with other engineering fields to achieve industrial acceptance. Therefore, this kind of cooperation can be considered as "(software+) engineering" [2].

There is already individual tool-support available for each engineering process step, but the integration of those tools to form a single integration solution heavily relies on manual work and invasive changes to tools, and thus does not provide an automated and non-invasive solution including support for the integration of entire engineering processes. Furthermore, current rigid integration solutions introduce complete tool sets supporting the whole engineering process, but are inflexible regarding even minimal changes in standard engineering processes or tool sets. However, typical real-life engineering processes very often comprise specific and highly specialized tools and also often alter existing standard engineering processes according to their needs. Therefore, large-scale engineering projects seek for a flexible framework that facilitates a non-invasive support of multi-disciplinary engineering processes in case of tool exchangeability, as well as reusability and adaptation of process descriptions without changing existing tools.

In this paper, we describe in detail the Engineering Service Bus (EngSB) integration platform we have successfully used in several industrial projects for the integration of heterogeneous automation systems engineering tools. The EngSB is an integration platform based on the ESB and not only integrates different tools and systems, but also provides a bridge between engineering processes and existing engineering tools [1, 3]. The so-called engineering workflow is used as basis for the non-invasive engineering tool integration and to automate the integration; this workflow is directly related to the original engineering process and represents the implementation of the engineering process in the EngSB platform. 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.

In this work we focus on the application of the EngSB to multi-engineering environments and describe the detailed architecture of the relevant EngSB integration platform components. Furthermore, we discuss the EngSB concept as a technical platform for the integration of different engineering tools and for bridging engineering processes and engineering tools. Additionally, we evaluate the EngSB by applying it to a standard software engineering process for the information systems engineering domain, the Continuous Integration and Test (CI&T) process [4], and comparing the results to other related approaches, like with the general Application lifecycle management (ALM) approach and as a specific implementation of the process with Hudson[1]. The specific configuration of the EngSB with respect to the given process results in the concrete OpenCIT[2] (a continuous integration and test server) implementation.

The remainder of this paper is structured as follows: Section 2 summarizes related work on aspects of technical integration and related implementations of research groups or industrial projects. Section 3 describes the use case scenario. Section 4 details the EngSB implementation, which is evaluated and discussed in section 5. Finally, section 6 concludes the paper, and identifies further work.

II. Related Work

This section summarizes related work on current methods, concepts and approaches to integrate distributed environments, architecture concepts for tool integration, and integration framework specifications.

A. Tool Integration in Distributed Environments

The most common integration patterns have been summarized in [5] whereas messaging pattern results in a huge number of middleware frameworks available [6], mostly with the Enterprise Service Bus (ESB) concept [7]. The ESB has been successfully applied as agile integration platform for back-end services in distributed heterogeneous business software environments [7]. Key strengths of the ESB [8] are providing distributed integration and separating the description of the business logic from the integration logic in contrast to design patterns such as client-server architecture [9] or Messaging. Nevertheless it has to be taken into consideration that too much abstraction does not help software engineers integrate their environments [10, 11] as the reusability of the different components is limited. ESBs provide components which can be reused to integrate different functionality or protocols, but only little research is done regarding tool and process abstraction, requiring system integrators to start again nearly from the scratch for each new project.

The ESB is also used as the underlying infrastructure for Service-oriented Architectures (SOA) [11] which intend to support service composition and application evolution [12] aiming at higher reusability, shorter time to market and lower costs. Although the concepts are well adapted [11, 13], the large number of different standards, specifications, technologies and high requirements in analyzing the organization make SOA hard to understand and complex to implement. Additionally, SOA itself does not provide any concepts for business process event handling. SOA notifications are basically implemented by defining services which directly call other services. This requires rewriting the service, notifying the system, for each logical change. With the help of service orchestration the logic can be extracted from the services themselves into a meta layer. However, this only moves the problem into a higher level of abstraction as the meta-layer has to be rewritten every time an additional service has to be notified.

[1] http://hudson-ci.org

[2] http://opencit.openengsb.org

B. Integration Environments

In general, analyzing the different concepts for integration developed over time, three approaches were able to become defacto standards for integration in software engineering:

Script based integration approaches describe the build process of software only. Modern approaches like Maven [14] also integrate test-, deployment-, announcement- and documentation environments to cover the complexities of current software engineering projects. Although Maven is one of the most used build systems for software development in the Java world, its reusability and flexibility is limited: first, the Maven workflow is hard coded into its core. It can be extended by attending plug-ins to the different lifecycle goals, but it cannot be changed completely and adapted for other uses.

Another approach of tool integration is to focus on the developer within its **Integrated Development Environment** (IDE) [15] like in Eclipse[3] or Microsoft Visual Studio[4]. They provide integration modules like Mylyn which supports a complete, task centered integration of issues retrieved from a wide range of different issue trackers. Although an IDE brings the entire engineering environment to the developer the logic and process models have to be implemented with each environment. Furthermore, they do not support automation of complex processes over multiple, different IDEs.

Application lifecycle management (ALM) based integration approaches usually comprise source control system and integration workflows, deployment, monitoring, project management and testing, while at the same time covering the entire project lifecycle [16]. An ALM is the Eclipse Application Lifecycle Framework (ALF) [17] which handles integration challenges by introducing a central negotiator that manages interactions between applications. By using an intermediate communication format the event manager prevents to integrate applications several times with several other applications. It allows a single integration on the central node, which carries out communication with other ALM applications through orchestration of their corresponding web services. However, ALF is no longer supported by the Eclipse Foundation [18].

III. Use Case

This section presents a (software+) engineering industrial use case from signal engineering that has been retrieved from an industrial partner developing, creating, and maintaining hydro power plants. The standard mo del of the Continuous Integration and Test (CI&T) process known from software engineering consists of a set of activities: checking out source code artifacts from a source code management system, building the system, running tests, performing analyses regarding the outcome of these tests, and finally reporting test results to interested roles. The CI&T use case shows a key feature of an iterative software development process: if parts of a system or engineering model get 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.

[3] http://www.eclipse.org/

[4] http://msdn.microsoft.com/en-us/vstudio/default.aspx

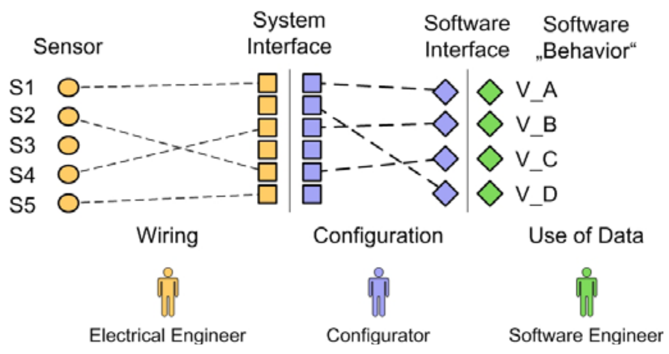

Figure 1: Overview End-to-End Analysis [19, 20].

In distributed multi-engineering environments (see Figure 1), typically a set of different tools and data models are used along the engineering chain. In difference to single system models, where changes of model values have direct impacts on other model elements, model value changes in cross-domain modeling require additional transformation and checks before the actual impact of a value change can be estimated or measured. To cooperate, the engineers have to exchange relevant parts of the data structures in their tools with each other with the goal of a consistent overall view on certain aspects in the project, e.g., when producing a specification for a subcontractor. This requires exhaustive communication between engineers to keep models consistent and thus introduces a variety of error sources as well. Currently, every role uses organization-, domain-, and tool-specific data formats and terms, thus the data exchange also takes considerable expert knowledge on the receiving end to make sense of the incoming data, typically as large PDF document or tool-specific import file. Applying the automated continuous integration and testing process in such an environment increased consistency and minimize the number of potential error sources.

IV. EngSB Implementation

This section gives a description of the components (Figure 2) of the Engineering Service Bus (EngSB) architecture:

Core Components. The concept of Core Components contains additional developed components providing services which could not be provided by external tools or which are not usable from external tools as required. The two most important components for the architectural need of the EngSB are the registry and the workflow component. Since different projects may be running in parallel on the EngSB, each project requires carrying a context which contains at least the project identifier allowing the ***registry*** to map onto endpoints for each accessed tool. The ***workflow component*** is responsible for engineering rule, process and event management in the EngSB. The Drools[5] business logic integration platform builds the backbone for the workflow management system since it provides a consistent solution for BPM, knowledge and event management. It includes Drools Expert as rule engine, Drools Fusion for complex event processing, Drools Flow for business process management and finally Drools Guvnor as a centralized knowledge repository to manage processes and rules centralized. The important part of the workflow component is its indirection layer between engineering rules and processes and the EngSB Tool Domains.

[5] http://www.jboss.org/drools/

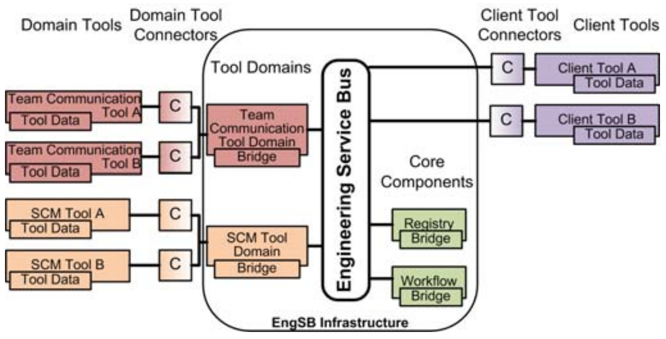

Figure 2: Components of the Engineering Service Bus.

Tool Domains. Although each tool provider gives a personal touch to its product their design is driven by a specific purpose. For example, there are many different issue tracker available, each having its own advantages and disadvantages, but all of them can create issues, assign and delete them. Tool Domains [3] are based on this idea and distill the common functionality for such a group of tools into one Tool Domain interface. They can contain code, workflows, additional logic and data, but they are useless without a concrete implementation.

Bridge. JBI does not allow components that are not deployed in the JBI infrastructure to directly interact with services. This means that tools deployed to the EngSB can directly access neither external components nor the other way round. The bridge provides a connection between these two worlds.

Client Tools. Client Tools in the EngSB concept are tools which do not provide any services, but consume services provided by Tool Domains and Core Components instead.

Domain Tools. Domain Tools, compared to Client Tools, denote the other extreme of only providing services. Classically, single purpose server tools, like issue tracker or chat server, match the category of Domain Tools best.

Domain and Client Tool Connectors. Although most (back-end) tools provide interfaces to integrate and automate them in distributed environments, they do not directly fit the needs of the EngSB. Instead, they have to be wrapped up to allow external, distributed tools to access them via the EngSB. These required bridges are called Tool Connectors. Tool Connectors wrap tools as Domain Tool Connectors to provide their services to accommodate the relevant Tool Domain with the expected interface. As Client Tool Connectors they provide a Client Tool with an access to the EngSB services. Tools can be integrated with more than one connector to act in many different domains.

V. Evaluation and Discussion

In this section we evaluate the capabilities of the EngSB related to flexibility and non-invasive support for integrating software engineering processes by reporting on conducted feasibility studies. The evaluation is based on the OpenCIT framework by applying the standard software engineering

CI&T process (see left hand side of Figure 3) to multi-engineering environments.

With respect to the CI&T process, the right hand side of Figure 3, shows OpenCIT integrating common tools via five different domains, namely Source Code Management, Build, Test, Analysis and Notification. For each of these domains, a concrete tool instance from each used engineering environment (e.g., electrical engineering and software engineering) is connected using a tool connector. In addition, the core Engineering Workflow Rules component is used to configure the defined CI&T process as a workflow on the EngSB and to orchestrate the specific runtime events of the individual tools accordingly.

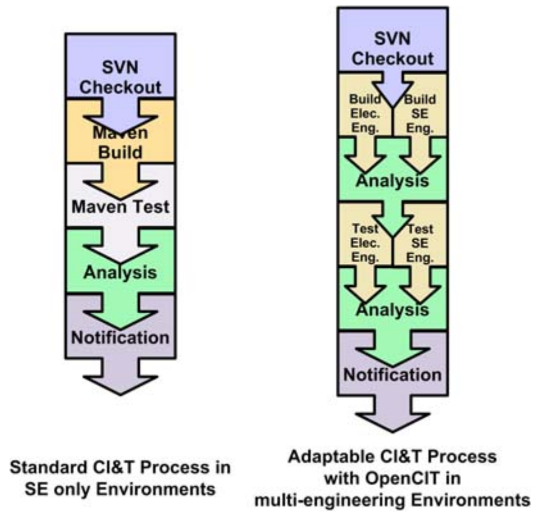

Figure 3. Overview CI&T and OpenCIT.

The evaluation has been conducted with two different tools in two phases. First, we compare OpenCIT with a highly popular CI&T tool suite (Hudson) to demonstrate how EngSB can address standard software engineering integration problems. Second, we compare EngSB with a general purpose integration framework (Eclipse ALF), which is a set of tools designed to support an enterprise application from its initial inception through its deployment and system optimization. The high level goal is to analyze the effort of adopting a new CI&T tool suite, from the viewpoint of software administrators, building tool connectors and defining/customizing the process.

Phase 1. In order to compare the effort required for integrating new tools to the frameworks, we compared the effort required for developing a connector and the mechanisms available for extending the core in Hudson and EngSB. We examine initial adoption effort and the possibility to make changes to standard CI&T workflow. Hudson provides extension points, where new plugins can be added to extend the application logic. This mechanism is used to integrating new tools to the framework. EngSB provides interfaces, which can also be extended. The extensions provide the functionality via JMS, or REST, or Web Services, allowing programming language independent integration facilities. As Hudson is meant for use in software engineering, event processing is specialized and hardcoded and may be customized only via tools integrated using extensions points. Workflows are hardcoded, which means that the core has to be extended for this purpose. In EngSB events can be designed for domains but also completely independent, which allows one to decide how tight the tools should be integrated to the workflows. Workflows are basically defined in drools.

Phase 2. We investigated the key characteristics of Eclipse ALF, a system that allows reusable process definition and adaption regardless of low-level technical details. We analyzed the flexibility regarding event processing, effort needed to modify workflows, and tool exchangeability capabilities. In Eclipse ALF events are SOAP messages which are defined in the core. New tools may react to the SOAP messages to achieve workflow modifications. In EngSB events can be designed for domains but also completely independent, which allows one to decide how tight the tools should be integrated to the workflows. In Eclipse ALF workflows are defined in BPEL leading to a heavy-weight solution which has to be defined programmatically and cannot be modified or extended at runtime. In EngSB workflows are defined in Drools. While in Eclipse ALF exchanging tools requires redefinition of the processes and event handling, EngSB supports tools to be freely exchangeable, as the workflow is defined on the level of tool domains.

While CI&T suites such as Hudson enable the integration of software engineering tools, there is no CI&T suite available for enabling CI&T processes for multi-engineering environments such as the scenario presented in Figure 1. As shown in the right hand side of Figure 3, OpenCIT can be used for migrating the CI&T process, which originally comes from the software engineering domain, to multi-engineering environments. This we define as (software+) engineering, as software engineering provides additional value to systems engineering. The tool domain concept of the EngSB allows for a non-invasive exchange of single tool instances, since tool domains define and provide the domain-specific functionality usually provided by single engineering tools. Therefore, OpenCIT has no restrictions regarding the tool sets to be used, but relies on the tool domain concept of the EngSB to specify and integrate already existing tool functionalities.

In order to enable the CI&T process for multi-engineering environments, the standard CI&T process is adapted as shown in the right hand side of Figure 3. In comparison to the standard software engineering CI&T process, the multi-engineering CI&T implementation of OpenCIT allows the definition of multiple tool domains (often from different engineering disciplines) to define the same CI&T process step (e.g., the *build* process step). When this process step is executed, OpenCIT triggers the functionalities of all tool domains that are linked to this specific process step. Furthermore, different strategies can be implemented related to the order of the triggering, e.g., the workflow engine of the EngSB can be used to implement and enforce project-specific reporting strategies regarding the outcome of a specific process step (e.g., a build failure notification). The tool domain concept of the EngSB can additionally be used to simulate engineering tool instances on the level of an engineering process. This feature can be used to test engineering process (e.g., whether they work as planned), without the need to have individual tool instances available and connected to the EngSB.

Additionally, EngSB showed more flexibility regarding the continued use of favored tools by engineers, whereas standard tool suites allows the integration of some standard software engineering tools, which may or may not be used by the engineers in a certain organizational setting. The possibility to use known tools (thorough the possibility of integrating arbitrary tools) results in minimal training effort for project participants, since the used engineering tools are already familiar to them.

VI. Conclusion

Today's large-scale information systems engineering projects typically involve a range of different engineering disciplines with their specific engineering tools. However, for building and organizing information systems engineering projects to manage specific engineering processes a cooperation of each discipline is required to form an integrated engineering system. Although, there is already individual tool-support available for each engineering process step, the integration of those tools to form a single integration solution heavily relies on manual work and invasive changes to tools. However, typical real-life engineering processes very often comprise specific and highly specialized tools and also often alter existing standard engineering processes according to their needs. In this paper, we presented the Engineering Service Bus (EngSB) as an integration platform based on the ESB that does not only integrate different tools and systems, but also provides a bridge between engineering processes and existing engineering tools. We demonstrated the use of the EngSB to migrate a standard software engineering process - Continuous Integration and Test (CI&T) – to multi-engineering environments and compared it with popular tools, like Hudson or Eclipse ALF. The comparison showed that OpenCIT, the multi-engineering environment integration of CI&T using the EngSB, can replace an integration framework like Hudson for projects involving multiple engineering disciplines. Adaptations to the standard workflow are more flexible in the case of EngSB because it allows arbitrary extensions through flexible event handling and exchangeability of tools. Mutual restrictions between tools and processes can be minimized through the use of workflow rules that are formulated on the level of tool domains. Furthermore, we compared EngSB with Eclipse ALF to demonstrate the EngSB provides higher abstraction level for tool integration, which makes it easier to adopt For future work we intend to perform empirical evaluations in case studies and implementations as well as creating domain-specific process definition tools to support non-experts in defining information systems engineering processes to be deployed in the EngSB.

Acknowledgments

This work has been supported by the Christian Doppler Forschungsgesellschaft and the BMWFJ, Austria. We are thankful to Alexander Schatten and Andreas Pieber for their valuable input in several discussions.

References

[1] Biffl, S., and Schatten, A.: 'A Platform for Service-Oriented Integration of Software Engineering Environments'. Proc. Eight Conf. on New Trends in Software Methodologies, Tools and Techniques, 2009

[2] Biffl, S., Pieber, A., and Schatten, A.: 'Service-Oriented Integration of Heterogeneous Software Engineering Environments', TechnicalReport QSE, ISIS, Vienna University of Technology, 2009

[3] Biffl, S., Schatten, A., and Zoitl, A.: 'Integration of Heterogeneous Engineering Environments for the Automation Systems Lifecycle', Proc. IEEE Industrial Informatics (IndIn) Conference 2009

[4] Duvall, P., Matyas, S., and Glover, A.: 'Continuous Integration: Improving Software Quality and Reducing Risk' (Addison-Wesley, 2007. 2007)

[5] Hohpe, G., and Woolf, B.: 'Enterprise Integration Patterns : designing, building, and deploying messaging solutions' (Addison-Wesly) 2004

[6] Du, Y., Peng, W., and Zhou, L.: 'Enterprise Application Integration: An Overview'. In Proc. of the 2008 Int. Symposium on intelligent information Technology Application Workshops, Washington, DC, USA2008 pp. Pages

[7] Chappell, D.: 'Enterprise Service Bus' (O'Reilly Media, Inc.), 2004

[8] Chappell, D.: 'ESB Myth Busters: 10 Enterprise Service Bus Myths Debunked', (Last visited February 22, 2010, online: http://soa.sys-con.com/node/48035)

[9] Berson, A.: 'Client/server architecture (2nd ed.)' (McGraw-Hill, Inc.), 1996

[10] Woolf, B.: 'ESB-oriented architecture: The wrong approach to adopting SOA', (Last visited February 22, 2010, online: http://www.ibm.com/developerworks/webservices/library/ws-soa-esbarch/)

[11] Engels, G., and Assmann, M.: 'Service-Oriented Enterprise Architectures: Evolution of Concepts and Methods'. Proc. of the 2008 12th Int. IEEE Enterprise Distributed Object Computing Conference, Washington, DC, USA, 2008

[12] Yin, J., Chen, H., Deng, S., Wu, Z., and Pu, C.: 'A Dependable ESB Framework for Service Integration', IEEE Internet Computing, 2009, 13, (2), pp. 26--34

[13] Mike, P.P., and Willem-Jan, H.: 'Service oriented architectures: approaches, technologies and research issues', The VLDB Journal, 2007, 16, (3), pp. 389-415

[14] Sonatype: 'Maven: the definitive guide, 1st edition' (O'Reilly & Associates, Inc., 2008. 2008)

[15] Cheng, L.-T., de Souza, C.R.B., Hupfer, S., Patterson, J., and Ross, S.: 'Building Collaboration into IDEs', Queue, 2004, 1, (9), pp. 40--50

[16] Chappell, D.: 'What is Application Lifecycle Management?' (Last visited March 22, 2011, online: http://www.davidchappell.com/WhatIsALM--Chappell.pdf)

[17] Buss, T., and Caroll, B.: 'ALF Architecture - Draft: A Platform for ALM Tools Integration', 2005

[18] Manchester, P.: 'Eclipse kills open-source SOA projects', (Last visited February 26, 2010, online: http://www.theregister.co.uk/2008/11/07/eclipse_kills_soa_projects/)

[19] Biffl, S., Moser, T., and Winkler, D.: 'Risk Assessment In Multi-Disciplinary (Software+) Engineering Projects', Int. Journal of Software Engineering and Knowledge Engineering, Special Issue: Software Risk Assessment, 2011, 21, (2), pp. 1-25

[20] Moser, T.: 'Semantic Integration of Engineering Environments Using an Engineering Knowledge Base'. PhD thesis, Vienna University of Technology, 2009