Flexible Engineering Environment Integration for (Software+) Engineering Teams

Andreas Pieber (Faculty Mentor: Stefan Biffl)
Christian Doppler Laboratory for SE Integration for Flexible Automation Systems
Institute of Software Technology and Interactive Systems
Vienna University of Technology
Vienna, Austria
Email: {andreas.pieber,stefan.biffl}@tuwien.ac.at

Abstract — *For the development of complex software-based systems multi-disciplinary engineering teams have to cooperate. While there are elaborate software tools for each engineering discipline, there is surprisingly little work on flexible and efficient component-based integration of engineering tools across disciplines. In this paper we present the "Engineering Service Bus", an integration platform for (software+) engineering systems based on the established Enterprise Service Bus concept for business software systems. Based on real-world software engineering use cases adapted to the (software+) engineering domain, we show how the Engineering Service Bus allows prototyping & reuse of engineering processes and discuss strengths and limitations of the proposed concept.*

I. Introduction

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 (SE). We call these teams (software+) engineering teams as software engineering adds much to the added value of these systems but depends on the seamless collaboration with other systems engineering disciplines. A major challenge in (software+) engineering projects is the week integration of the software tools that the engineers use: Each discipline uses elaborate software tools for design and development but there is surprisingly little work on the integration of these tools and their data to support process automation on the project level. Each project has specific requirements, which can be covered by a set of tools but rarely by the tool set provided by a single vendor. Although there exist integration approaches for business software systems, (software+) engineering projects need the adaptation of integration approaches.

In this paper we extend the concept of the Engineering Service Bus (EngSB) [1] to bridge the technical gaps between engineering tools for quality assurance and process automation for engineering disciplines that interact with software engineering.

II. Research Issues

Although there exist approaches for integrating component-based systems in business environments such as the enterprise service bus (ESB) [2], engineering systems mostly are designed rigid and do not allow to integrate with other domains and allow (software+) engineering systems to integrate.

We describe and extend the architecture of the EngSB with so-called tool domains that support the flexible composition [4] of engineering tools and business systems to automate engineering processes.

We empirical validate the comparing implementations of traditional SE approaches and the EngSB approach for the real-world use case "continuous integration and test" (CI&T), a standard SE process that is implemented unnecessarily rigidly even in modern SE environments.

III. The Integration Architecture

Figure 1 shows the integration of tools from different engineering environments and their extension by engineering integration tools via the EngSB. In the EngSB *Apache Servicemix* is used as the ESB based on the JBI standard. In addition, the EngSB consists of tool domains, connectors and core tools [1].

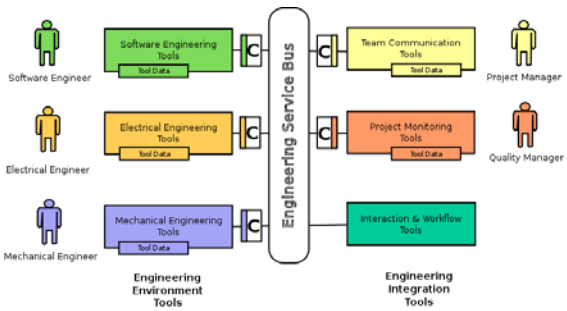

Figure 1: The Engineering Service Bus concept.

Tool domains describe abstract service interfaces [4] of the tools, which are to be integrated. Services are always called on tool domains so the component implementations never know about an explicit tool

instance. As bridge between tool domains and the integration of tools so-called tool connectors are used (see "C" in Figure 1).

Core components extend the capabilities of the EngSB such as a workflow component, event logging for process analysis and validation, and poject user/role administration. All other engineering integration tools like monitoring and issue management are integrated via tool connectors and domains.

IV. (SOFTWARE+) ENGINEERING USE CASE

The CI&T use case is a standard life cycle process for SE consisting of the following steps: 1. On change of a code unit, build the source code; 2. Test the built source code; 3. Package the compiled source code. The build result gets published on a project web homepage and if there are errors, a notification mail gets sent to a configured list of recipients. This process is implemented by integration build server like *Hudson* or *Continuum* in a rigid way.

For implementing the same process in the EngSB environment we created tool domain for the SCM, build, test, and notification tools. The base process is implemented as *Drools* flow. Finally we added *SVN* as the SCM tool, *Apache Maven* for build, test and deploy and Email for notification. The cooperation of these tools enable the classical CI&T for the EngSB.

Now the specific tools can easily be replaced behind the tool domains. However, the process is still rigidly designed. By extending the base workflow to call a statistics component after each run we enabled the process to gather and present statistics.

We reused the CI&T process infrastructure as the change management use case for signals in (software+) engineering. Signals are technically collections of key-value pairs occurring in the different engineering domains. Therefore, we were able to design a signal-specific data source as the SCM system, to reuse the build for an electrical plan tool, and to design a specific validation concept between signals.

Lessons learned from implementing the use cases were: 1. Designing tools as components in the EngSB context worked well; 2. Implementing specific tool instances behind abstract tool domains greatly simplified the process of changing tool instances; 3. Using business process modeling [5], rule engines, and event processing [4] allowed a flexible process definition and workflow extension. Processes could be flexibly reused in (software+) engineering scenarios. Limitations of our approach are the added complexity to the tool environment from middleware configuration and administration.

V. CONCLUSION AND FUTURE WORK

In this paper we extended the concept of the "Engineering Service Bus" (EngSB) that integrates components in office-like design environments with tool domains. Based on real-world engineering use cases "continuous integration and test" (CI&T) and signal change management we showed how the EngSB allows prototyping new variants of SE processes, which could be reused in different domains as a change management process, e.g., for signal engineering in (software+) engineering.

Major results were that even initial states of the EngSB implementation can show significant benefits to a heterogeneous engineering environments, better team awareness and quality assurance. Advanced stages of the EngSB implementation can facilitate a global view on tools and systems along the life cycle as basis for the optimization of the engineering process. Even the initial prototype was able to demonstrate significant benefits in a heterogeneous (software+) environment like flexible and efficient configuration of (software+) engineering processes and stable team process even if tool instances change.

From the experiences with the prototypes implemented we extracted the need for graphical configuration on engineering domain level abstracting the complexity of the EngSB.

REFERENCES

[1] S. Biffl, A. Schatten, and A. Zoitl: 'Integration of Heterogeneous Engineering Environments for the Automation Systems Lifecycle'. Proc. IEEE Industrial Informatics (IndIn) Conf., 2009, pp. 576-581

[2] S. Biffl, A. Pieber, and A. Schatten: 'Service-Oriented Integration of Heterogeneous Software Engineering Environments'. Technical Report, QSE-2009-11, ISIS, Vienna Univ. of Tech, 2009.

[3] D. Chappell, Enterprise Service Bus; O'Reilly Media, 2004.

[4] G. Hohpe and B. Woolf, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions: Addison-Wesley, 2003.

[5] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, Service-Oriented Computing: State of the Art and Research Challenges, Computer 40 (2007), 38-45.