Streamlined business processes with clabject-based software modeling

Susanne Altendorfer, Wolfgang Pree
C. Doppler Laboratory Embedded Software Systems
University Salzburg
Jakob-Haringer-Str. 2, 5020 Salzburg, Austria
firstname.lastname@cs.uni-salzburg.at
www.uni-salzburg.at/CDL

Abstract. The paper sketches how a program model in general and the resulting software architecture in particular influence business processes. We argue that monolithic software architectures impede business processes to adapt to changing requirements whereas a so-called clabject-based modeling approach, that unifies the notion of classes and objects, offers more flexibility and thus forms the basis of a business process reorganization that enables major improvements. We outline the core modeling concepts, the envisioned research contributions and sample application domains.

Keywords: component-oriented software architectures, model-driven system development, clabjects, business software, business processes, process streamlining.

1 Introduction

Testbed automation systems, so-called testbeds, are specific automation systems for operating combustion engines under controlled conditions. Our research partner is an engineering company that develops and sells engine testbed systems, primarily for the automotive industry. A testbed system basically measures, records, and visualizes numerous values provided by sensors according to test plans. The testbed requires appropriate parameterization for that purpose. Due to the various different use cases of testbeds, testbeds have to be adapted according to customer-specific requirements. Typical testbeds consist of hundreds of thousands of components. According to [1] "an engine test facility is a complex of machinery, instrumentation and support services, housed in a building adapted or built for its purpose. For such a facility to function correctly and cost-effectively, its many parts must be matched to each other while meeting the operational requirements of the user and being compliant with various regulations". The goal of our research cooperation is to radically simplify the configuration and operation of testbeds. As a consequence, the associated business processes should also be significantly streamlined.

This paper is structured as follows: Section 2 provides an overview of the current architecture of testbed system software and the business processes used by our cooperation partner. Section 3 sketches the clabject-based modeling approach. Section 4 briefly outlines the feasibility of this modeling approach for testbed systems and its impact on the business process structure.

2 Architecture of testbed automation systems and associated business processes

This section describes the monolithic architecture of the current software system and the business processes which are not supported at all by the testbed software.

2.1 Monolithic architecture with unstructured parameters

A testbed system needs to be tailored to customer demands in a straight-forward way, which is not well supported by the current monolithic software architecture. "To build or substantially modify a modern engine test facility requires coordination of a wide range of specialized engineering skills; many technical managers have found it to be an unexpectedly complex task" [1]. The existing software has evolved over the past decades

and comprises ca. 4.5 million lines of code, mainly written in C++ and C, which makes adjustments quite difficult.

The current system requires error-prone editing of parameters (in the order of tens of thousands) in spreadsheet-like tables, as well as the adaptation of configuration files and scripts scattered in the file system. Using this representation and finding specific parameters is difficult as parameters associated with a physical entity of the testbed, such as the engine under test, are not grouped accordingly. If such an entity is changed the user has to know by heart which parameters are affected. Configuration files and scripts exist in various standard and proprietary formats and different editors are required for each file type. Thus the handling is time-consuming and expensive.

Above all, the current software does not support the whole range of business processes. For example, due to its complex user interface and its focus on technical details, it is a system that can only be adapted and operated by experts. It is a standalone software tool that is not integrated at all in the overall business process chain.

2.2 Current business process landscape

The broad product portfolios as well as adaptations of products by customers mean that every delivered system is unique. Although customers might regard this high degree on individualism as advantage, the implied costs are enormous and have to be reduced. In other words, the potential for cost cutting is significant.

The current software is restricted to the project execution process without having any connection to the up- and downstream processes, which are essential for the overall project success. Figure 1 illustrates this situation. The processes Sales, Project Execution, and After Sales Support, are completely separated processes. The handover points from one process to the other are without any tool support. Thus relevant information might be lost throughout the process chain, and information needed in the progression of the project is not available, as the importance of these data might not be clear in an earlier phase.

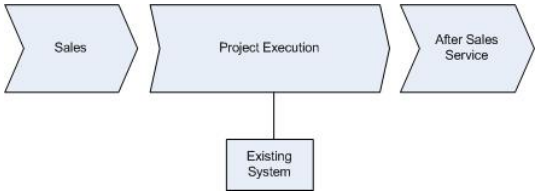

Fig. 1: Position of the current software system

Currently, each process uses different tools, like Excel sheets or proprietary tools. These gaps cause a significant information loss and cause extra manual conversion and transfer efforts. As the existing software systems became complex and do not interact with each other, the actual situation can no longer be changed through incremental adaptations of the existing software systems. The required changes are neither possible with the existing software architecture nor is any potential effort for writing code for transforming the data justifiable.

3 Clabject-based modeling

The hypothesis is that a software system that allows a stepwise modeling of a testbed automation system would support the overall business process chain and thus form the basis for process streamlining and cost cutting. This means that the sales process starts with modeling a coarse-grained testbed that is refined by the subsequent process steps. So every detail of the testbed that is relevant for the various business processes as well as for its operation needs to be modeled. Thus, we refer to this kind of modeling as deep virtualization of the corresponding real-world testbed.

By examining the examples of how to model an engine and its attached sensors, we identified problems of conventional modeling languages such as UML [7]. Instead we decided to go for what Colin Atkinson et al. have called clabject-based modeling [5]. The idea of unifying the concept of classes and objects goes back to an experimental object-oriented language called SELF [4].

Modeling with clabjects
In order to harness the existing automation system, the aim is to generate the parameters for the automation system from the testbed model. This requires a testbed model comprising all essential hardware and software parts, as sketched in figure 2.

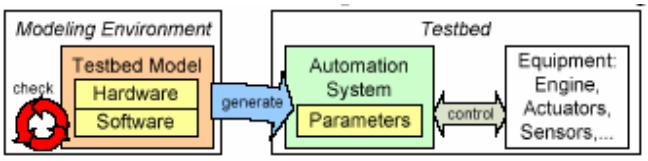

Fig. 2: Model processing

Processing of testbed models involves three distinct phases. (1) In the *modeling phase* a user defines a testbed model. (2) Once a model is valid, the corresponding automation system parameters are generated in the *generation phase* by a separate transformation system. (3) In the *execution phase* the automation system is started with a defined sequence of test steps. While the first two phases happen during *modeling time*, the last phase happens during *execution time*. Checks in the modeling phase ensure that only valid models are used for parameter generation, thus preventing execution-time errors and in turn severe damage to equipment.[2]

Clabjects—uniform representation of classes and objects.
In the context of domain models, Atkinson and Kühne define accidental complexity as introduced due to mismatches between the problem and the modeling means to represent the problem [6].They argue that modeling languages that allow using only two levels, such as UML with the class and object level, induce accidental complexity when modeling domains that inherently require more modeling levels. The solution they offer is the concept of a *clabject*, a modeling entity that has a class facet as well as an object facet. Figure 3 shows the clabject based approach for the domain of testbed automation systems. The notation used here is similar to that of the original clabject concept. Each model element has a compartment for the name, and a combined compartment for the type facet and the instance facet. The dashed arrows between the levels represent the "instance of" relationship. With a uniform representation of type facets and instance facets, our example can be modeled in a natural way. By definition, the clabjects at the top-level only have a type facet, whereas the clabjects at the bottom level only have an instance facet. [2]

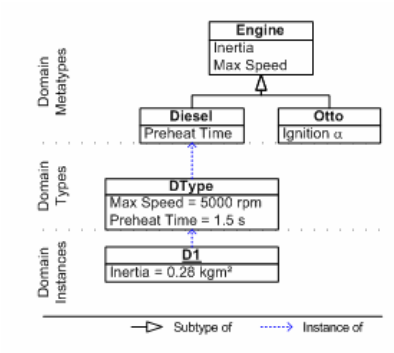

Fig. 3: Engine hierarchy with clabjects

For the domain of testbed automation systems we extended the clabject notion for handling association inheritance and instantiation, which are essential for bridging the gap between theoretical research and industrial applications. Describing all modeling formalism would go beyond the scope of the paper.

4 Process Reorganisation

To show the feasibility of these concepts another kind of requirements also has to be considered: the business processes. It is crucial that a software system matches the requirements of an organization and that the software enables the business processes to adapt to business requirements. The overall complexity of testbeds demands

for a business process that applies stepwise refinement. The aim is that a unified method and tool set is established, with which everybody throughout the process chain can interact. The resulting model of the overall testbed corresponds to the real-world testbed in all its relevant details. We refer to the tool set as DeepVTool. All relevant information should be added gradually to the testbed model in each process phase according to the particular requirements. Thus the level of detail, represented by the model, increases over the progression within a project. A further advantage is that errors or dependencies in the design and development of testbeds can be identified already at an early stage in the sales phase, avoiding costs that are due to late detection and fixing of such errors in the subsequent project execution phase.

A typical business process chain covers aspects from product management, sales, project execution to the after sales support. Each process has its special requirements and associated roles on the tool. With the introduction of the new business tool, the process landscape could be streamlined in terms of execution times, which is shown in figure 4. First estimations show that savings between ten and fourteen per cent can be achieved only in the project execution process. With a process-wide support even more savings can be achieved-

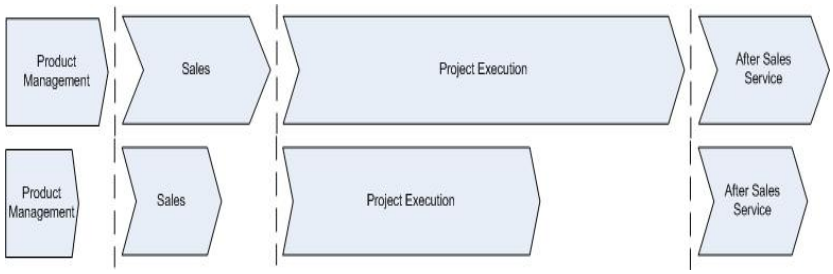

Fig. 4: Process Chain streamlining

Thus we call this modeling approach “deep virtualization”. Figure 5 demonstrates the core benefit of such a modeling tool. It shows the integrated approach of the new modeling tool, DeepVTool, across the process chain. Through the integration the tool becomes a business tool as it supports all essential processes and it will have pre-defined interfaces to other tools, which are relevant for the processes and where an interaction is essential. While the DeepVTool is the technical backbone, the other tools, indicated as tool 1 till tool 4, are mainly used for economical matters, like a sales tool. All technical solution components from the DeepVTool could be automatically exported to the sales tool. Thus, the workflow is well supported and needed information can be easily accessed.

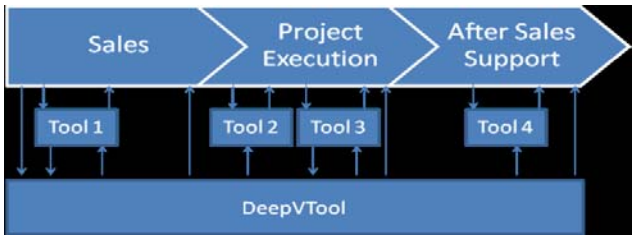

Fig. 5: Envisioned tool supported process chain

A further advantage of the DeepVTool is that it is not intended to be a tool for experts only, as the current system is. Employees working in business areas—from sales to testbed operators—can use those aspects of the tool that is relevant for their tasks. The granularity and level of detail of information however will be adapted to the role concept which is supported by the DeepVTool.

5 Research Method

The applied research method can be separatd into three major methods: (1) Starting with an investigation of the current processes current system weaknesses in terms of modularity and usability are identified. This is mainly done by face to face interviews and process analysis. (2) From the gained process understanding and the existing tools the envisioned processes will be specified with our DeepVTool approach. Therefore business process

modelling notation from [7] will be used in combination with RASI chart extensions. (3) After the process reorganisation and the implementation of the DeepVTool final results will be evaluated according to pre-defined key factors.

6 Future Work

In close cooperation with our industry partner we are currently modeling real-world testbed examples, and first results show that the modeling method as well as the DeepVTool is applicable in practice. The analysis of the existing processes is also almost finished and we are concentrating on the integration of the DeepVTool. Thus adaptations within the process structure have to be analyzed and evaluated. Furthermore interface requirement specifications will be defined to establish interfaces to other tools.

A prototype testbed case study will be used to demonstrate the feasibility of the new approach and the DeepVTool. This forms the basis for verifying or falsifying the hypothesis that the new approach and tool set have a positive qualitative and quantitative effect on the business processes. As the change might cause massively cultural and political resistance, the change has to be accompanied by an appropriate communication model, which has to be defined in this context. Savings can also be realized as the customer satisfaction can be directly affected by the DeepVTool. Thus the new modeling approach and the DeepVTool will have a significant impact on the whole company and its organizational structure.

7 Conclusion

In this paper we have demonstrated the need for an appropriate modeling formalism to support model-driven engineering in the domain of testbed automation systems.Our clabject based approach is now being integrated in the business processes and with the prototypical implementation across the process chain, mentioned in section 6, the feasability of the new approach is confirmed.

8 Acknowledgements

The authors would like to thank Thomas Aschauer and Gerd Dauenhauer for their comments on an early draft.

9 References

[1] Martyr, A.: Engine Testing: Theory and Practice, Butterworth Heinemann, 3rd revised Edition, 2007.
[2] Aschauer T., Dauenhauer G. Pree W.: Multi-Level Modeling for Industrial Automation Systems.2009
[3] The Economist: From blueprint to database, Online Edition June 2008.
[4] Ungar, D. and Smith, R. B. “Self: The power of simplicity”, *Proceedings of OOPSLA ’87*, ACM SIGPLAN Notices, Vol. 22, 1987, pp. 227–242.
[5] C. Atkinson and T. Kühne, “Model-Driven Development: A Metamodeling Foundation”, *IEEE Software*, Vol. 20, No. 5, pp. 36-41, 2003.
[6] C. Atkinson and T. Kühne, “Reducing accidental complexity in domain models”, *Software and Systems Modeling*, vol. 7, no. 3, 2007, pp. 345–359.
[7] Object Management Group, Unified Modeling Language Infrastructure, version 2.1.2, 2007.