PDD Applied: A Model Driven Approach

Helge Sören Klimek
Institute of Telematics
Hamburg University of Technology
Schwarzenbergstrasse 95
D-21073 Hamburg
Germany

helge.klimek@tu-harburg.de

Abstract: Property Driven Development is a methodology to check the validity of a distributed application and its conformance to specified requirements. A first prototype supporting this methodology is being built at the moment. It allows modeling of business processes and uses Model Driven Software Development (MDSD) technologies to transform the business process models to a mathematical specifications. While MDSD usually is used to create software artifacts, in this approach it is used solely to transform the models into their target form for further validation.

1 Introduction

Given a Service-Oriented Architecture (SOA) [MLM$^+$06] for a large virtual enterprise, the services in the architecture often represent business activities hidden behind well-defined interfaces. Services interact with each other by exchanging sets of messages. This composition of services leads to a new layer of abstraction, allowing simple business process composition, reorganization and also gives a clearer view on the business activities.

At the Institute of Telematics of the Hamburg University of Technology the Property Driven Development (PDD) methodology has been developed. It aims at enriching business process models with information needed to validate their fulfillment with respect to modeled requirements. This is achieved by expressing the essence of the business in terms of mathematical models. The model is enriched, for example, with safety properties, describing certain components legal states or temporal order of states. Since PDD allows validation of the fulfillment of requirements, it allows a better understanding and reflection of continuously changing business processes, thereby increasing the quality of the modeled systems and making development more convenient.

The focus in PDD is on the explicit modeling of systems under aspects of communication and service identification. Thus, the message exchange between participating services is important. After the services have been identified, other development methodologies can be applied for constructing the services, for example the Rational Unified Process (RUP) [JBR99]. The successful development of a system is only achieved in close dialog with stakeholders. In order to foster the dialog between stakeholders, for example domain experts, and developers, a common notation, understood by both, is used.

From the models drawn in that common notation, special formal models are derived, which are validated against the formal requirements. Model transformation and validation are automated by tools, making the transformation process transparent to the user. This way, the user does not need a deep understanding of formal methods while still being able to harness their power.

The methodology is a practical approach for developing and maintaining SOA based systems. It is especially designed to validate properties of systems and components and create provable results.

For a more comprehensive definition of PDD, please refer to [GVZ, KRVZ], the above information is taken from there.

In Model Driven Software Development (MDSD) models are part of the sources and development is centered around the models. They are used to generate parts of the system for other parts to rely on. Model transformation encapsules the process of translating a model into source code or other models for further processing. In model to code transformation, the user profits from the generation of invariant parts of the code. Variant parts ideally are computed from the model or otherwise are added later. In model to model transformation, aspects are taken from a model and are expressed in another one, allowing further refinement, for example by the user. The development is supported by tools and thus becomes more convenient, more efficient and less error prone. Generated source code and models can be reproduced using the original models and transformations and therefore are not considered to be source code. Model Driven Architecture (MDA) [MDA] is a special case of MDSD, having special requirements, for example the type of metamodel, transformations and constraint languages. In MDA model transformations are organized as stack of transformations, from platform independent to platform specific.

In the prototype created for this work, the more pragmatic MDSD approach is used. Model to model transformations are preformed, to transform a model created by stakeholders and developers into a mathematical specification of a system, in order to validate it against its formal requirements.

2 Current Work

Temporal Logic of Actions (TLA) [Lam02] is a language for describing behaviors of concurrent systems. It combines logic of actions with temporal logic. While the use of TLA was suggested in [GVZ, KRVZ], it first had to be shown, that this transformation can be performed automatically. In [Kno06] a method has been developed, how the transformation can be formalized and thus automatized. Therefore, this thesis laid the groundwork for the construction of a first prototype.

A first prototype application that utilizes the PDD approach is currently being developed. The models created by the stakeholders and developers are transformed into a TLA+ specification, representing the mathematical model.

In [GVZ, KRVZ] standard UML was proposed as notation for the business process models. In [Kno06] custom UML 2.0 Profiles have been introduced. However, for the prototype application both approaches have been dropped. Current business process modeling techniques provide a notation business experts are familiar with. Unfortunately, these notations, such as Business Process Execution Language (BPEL) [BPE] or Business Process Modeling Notation (BPMN) [BPM] are either too restrictive, not expressive enough or lack a formal metamodel.

Therefore an own metamodel has been developed. Additionally, a graphical tooling, especially designed to be simple and for domain experts intuitive to use, has been created. These two parts form the PDD Domain Specific Language (DSL). Both parts have been implemented using technology coming from the Eclipse ecosystem. The metamodel was implemented using the Eclipse Modeling Framework (EMF) [EMFa] and the graphical tooling was built using the Graphical Modeling Framework (GMF) [GMF].

The model transformation is performed in the OpenArchitectureWare (OAW) [OAW] enviroment. It is being used to transform the model, an instance of the PDD metamodel, to a textual representation of a TLA+ specification. The latter is used with the TLA Model Checker (TLC), which is part of the TLA+ Tools [TLC], to prove the fulfillment of the modeled requirements. For TLA and TLC Eclipse Plugins are available [GZ].

Other projects, for example AndroMDA [AND], use model transformations for generating source code artifacts as basis for further development. In contrast, PDD uses model transformations, to transform the model into a formal mathematical specification, with the only purpose of validating the same. Model transformation is used, to convey the model from one system to another, specialized and powerful system, in order to utilize its strength.

Current work on the prototype focusses on creating Eclipse plug-ins that allow the (visual) creation of business process PDD diagrams, enriching them, transforming them to TLA+ specifications and validating them using the TLC. In the first step, this will be loosely bound together.

The PDD metamodel comprises the most important elements of common business process notations such as business processes, business objects, participants, start- and stop-elements, elements to add safety properties and links to connect those elements – to name a few. During the transformation process, the model elements and their connections are analyzed and TLA actions, variables, invariants and so on, are generated. Safety properties are added to the actions or are added as invariants to the specifications. The generated specifications are simulated using the TLC.

3 Problems

The transformation from the EMF based PDD model instance to a textual TLA+ specification works as planned.

However, it is desireable to transform directly between two metamodels (M2) of the same metametamodel (M3). Currently there is no EMF based metamodel for TLA+. Having such a metamodel would allow performing a model to model transformation, where both models are EMF based. As result, the transformed model could be fed directly into the model checker, saving the indirection of writing a TLA+ specification and then loading and parsing it. Such a transformation is expected to be much simpler and easier to adapt to changes.

Safety properties and constraints can be added to the model, in terms of invariants, pre- and post-conditions. Initially it was planned to use the Object Constraint Language (OCL) [Obj06] to express constraints and safety properties. This is reasonable, because OCL is well suited for this kind of task. It is well known in developer circles and there is an OCL implementation from the EMF Technologies (EMFT) [EMFb] project, where EMF is used as metamodel.

The problem using OCL is, another metamodel is involved and needs to be transformed to the TLA metamodel. Additionally OCL does not provide temporal operators, which TLA does. This mismatch would lead to a narrowing of the expressiveness and powers of the PDD approach. This is of course undesired.

4 Future Work

From the current point of view, there are two main focusses for optimization and further research, both have been mentioned in the section above. One is the optimization of the model transformations, the other one concerns the way safety properties and constraints are expressed.

As stated, a model to model transformation is desireable. Currently no EMF based metamodel for TLA exists. For the future, it should be considered to take the effort and create such a metamodel. This will improve the overall model transformation process and its adaptability. However, having the metamodel alone will not solve the transformation problem. The metamodel needs to be bound to the backend of the TLC compiler, this is going to be difficult.

Further research is needed in order to determine how safety properties and constraints are used in practice, in order to find an optimal language to express those. That language has to be interpreted, translated to TLA and integrated into the existing TLA+ specifications, too. An EMF based metamodel for that language would be supportive as well.

For convenience a tighter integration of the different parts is desireable. For the long term, it is not necessary to expose the TLA+ specifications to the user. This requires that error messages can be traced and transported from TLA to the PDD diagram. Complete transparency of the transformation and validation process seems feasible and should be targeted.

Also, it seems a promising idea to reuse the transformations from this work in existing products or to have custom transformation plug-ins plugging into the prototype, allowing the generation of further artifacts. One could think of an integration into an existing development architecture, allowing the validation of a models formal requirements, as well as the generation of code obeying those requirements.

References

[AND] AndroMDA. http://andromda.org (last checked: 2007-02-21).

[BPE] Web Services Business Process Execution Language. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-specification-draft.pdf (last checked: 2007-02-21).

[BPM] Business Process Modeling Notation. http://www.bpmn.org/Documents/OMG %20 Final %20Adopted %20BPMN %201-0 %20Spec %2006-02-01.pdf (last checked: 2007-02-21).

[EMFa] Eclipse Modeling Framework. http://www.eclipse.org/emf (last checked: 2007-02-21).

[EMFb] Eclipse Modeling Framework Technologies. http://www.eclipse.org/emft/projects (last checked: 2007-02-21).

[GMF] Graphical Modeling Framework. http://www.eclipse.org/gmf (last checked: 2007-02-21).

[GVZ] Boris Gruschko, Friedrich H. Vogt, and Simon Zambrovski. Business Activities in an Industrial Context.

[GZ] Boris Gruschko and Simon Zambrowski. TLA+ Eclipse IDE Plugin. http://www.techjava.de (last checked: 2007-02-21).

[JBR99] Ivar Jacobson, Grady Booch, and James Rumbaugh. *The Unified Software Development Process*. Addison-Wesley, 1999.

[Kno06] Arne Knorr. Property-Driven Development - Modellierung als Methode. Master's thesis, Hamburg University of Technology, Telematics Institute, Germany, December 2006.

[KRVZ] Arne Knorr, Robert D. Russel, Friedrich H. Vogt, and Simon Zambrovski. Property-Driven Development: A verification approach for distributed processes.

[Lam02] Leslie Lamport. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley, 2002.

[MDA] Model Driven Architecture Specifications. http://www.omg.org/mda/specs.htm (last checked: 2007-02-21).

[MLM$^+$06] Matthew MacKenzie, Ken Laskey, Francis McCabe, Peter Brown, Rebekah Metz, and Booz Allen Hamilton. *Reference Model for Service Oriented Architecture 1.0, Committee Specification 1*. Organization for the Advancement of Structured Information Standards, August 2006.

[OAW] Open Architecture Ware. http://www.openarchitectureware.org (last checked: 2007-02-21).

[Obj06] Object Management Group. *Object Constraint Language, OCL 2.0*, October 2006.

[TLC] TLA+ Tools. http://research.microsoft.com/users/lamport/tla/tools.html (last checked: 2007-02-21).