Tutorial Proposal for APSEC 2013

Model-based Transition from Requirements to High-level Software Design

Dr. Hermann Kaindl, Professor
Vienna University of Technology, ICT

Short Biography

Hermann Kaindl joined the Institute of Computer Technology at the Vienna Univ. of Technology in early 2003 as a full professor. Prior to moving to academia, he was a senior consultant with the division of program and systems engineering at Siemens AG Austria. There he has gained more than 24 years of industrial experience in software development and human-computer interaction. He has published five books and more than 150 papers in refereed journals, books and conference proceedings. He is a *Senior Member* of the IEEE, a *Distinguished Scientist* member of the ACM, a member of the AAAI, and is on the executive board of the Austrian Society for Artificial Intelligence.

He has previously held tutorials at CAiSE'00, RE'01, RE'02, HICSS-36, INCOSE'03, RE'03, IUI- CADUI'04, INCOSE'04, RE'04, HICSS-38, IRMA'05, INCOSE'05, AAAI'06, HCI'06, OOPSLA'06, HICSS-40, ICONS'07, IRMA-07, INCOSE'07, AAAI'07, IFIP Interact'07, OOPSLA'07, HICSS-41, ICCGI'08, RE'08, ICSEA'08, ICIW '09 , IFIP Interact'09, SMC'09, HICSS-43 ACHI'10, EICS'10, ICSEA'10, TdSE'10, HICSS-44, SAC'11, INCOSE'11, AAAI'11, RE'11, HICSS-45, SAC'12, ACM CHI'12, PROFES'12, BCS HCI'12, APSEC'12, HICSS-46, SAC'13, NexComm'13 and PROFES'13.

Tutorial purpose

The primary objective of this tutorial is to improve software development in practice regarding the difficult and important transition from requirements to high-level software design.

The participants will understand several key problems with current object-oriented (OO) methods and how they can be resolved. In particular, they will see how scenarios and use cases can be utilized for requirements engineering and software design. But they will also see the additional need to specify the functional requirements for the system to be built. In addition, they will be able to distinguish between domain objects and software objects. They will experience UML as a language for representing OO models, but also the need to be clear about what kind of objects are represented. This is important for the model-based transition from requirements to design.

Tutorial description

How can the application domain and the requirements be better understood using object-oriented (OO) modeling? How do scenarios / use cases fit together with functional requirements? How can a domain model be used for a model-based transition to a design model?

This tutorial addresses these questions in the following manner. It shows how each requirement given in natural language can be viewed itself as an object and modeled as such. This approach facilitates both a hierarchical organization (by grouping requirements instances into classes and subclasses) and explicit association (by relating requirements through OO associations). While scenarios / use cases can somehow illustrate the overall functionality, additionally functional requirements for the system to be built should be formulated and related to them appropriately. All

kinds of requirements make statements about the application domain, which should be first represented in a domain model of conceptual classes, in order to make the requirements better understandable. This tutorial explains a seamless transition to a high-level design model, where design classes are abstractions of implementation classes. This transition from requirements to software design is also investigated in model-driven terms, whether it is a transformation or just a mapping. In addition, the influence of non-functional requirements for selecting an architecture is explained.

The target groups are software development practitioners, such as requirements engineers, software designers and project leaders. Also educators will benefit from this tutorial. The assumed attendee background is some familiarity with object-oriented concepts as well as interest in requirements, analysis or software design.

Detailed Outline

- Introduction and Background
  — Requirements
  — Object-oriented core concepts
  — Use cases
- Business and domain modeling using objects and UML
  — Business process — Business Use Case
  — Domain model
- Functional requirements, goals and scenarios / use cases
  — Functional requirements
  — Goals
- Requirements and UML models
  — Types of requirements
  — Non-functional / quality requirements
  — Conflicts between quality requirements
  — OOA (object-oriented analysis) model
  — Requirements vs. requirements representation
  — Software Requirements Specification
- Transition to software design
  — Animated transition example
  — OOD (object-oriented design) model and architecture
  — Sequence diagrams for OOA vs. OOD models
  — Transition recommendations
  — Model-based transition
- High-level software design
  — Software architecture
  — Selection depending on non-functional requirements
- Summary and conclusion

Selected publications of the proposer related to this tutorial

1. Kaindl, H., How to Identify Binary Relations for Domain Models, In Proceedings of the Eighteenth International Conference on Software Engineering (ICSE-18), IEEE Computer Society Press, Los Alamitos, CA, 1996, pp. 28–36.

2. Kaindl, H., A Practical Approach to Combining Requirements Definition and Object-Oriented Analysis, Annals of Software Engineering, vol. 3, 1997, pp. 319–343.

3. Kaindl, H., Difficulties in the transition from OO analysis to design, IEEE Software, Sept./Oct. 1999, 94–102.

4. Kaindl, H., A Design Process Based on a Model Combining Scenarios with Goals and Functions, IEEE Transactions on Systems, Man, and Cybernetics (SMC) Part A 30(5), 2000, pp. 537–551.

5. Kaindl, H., Adoption of Requirements Engineering: Conditions for Success, Fifth IEEE Inter-national Symposium on Requirements Engineering (RE'01), Toronto, Canada, August 2001.

6. Kaindl, H., Is object-oriented requirements engineering of interest?, Requirements Engineering, vol. 10, 2005, pp. 81–84.

7. Kaindl, H., A Scenario-Based Approach for Requirements Engineering: Experience in a Tele-communication Software Development Project, Systems Engineering, vol. 8, 2005, pp. 197–210.

8. Kaindl, H., and Falb, J., Can We Transform Requirements into Architecture?, in Proceedings of the Third International Conference on Software Engineering Advances (ICSEA 2008), 2008, pp. 91–96, IEEE.

9. Kaindl, H., Kramer, S., and Kacsich, R., A Case Study of Decomposing Functional Requirements, in Proc. Third International Conference on Requirements Engineering (ICRE '98), April 1998, 156–163, IEEE.

10. Kaindl, H., and Svetinovic, D., On confusion between requirements and their representations, Requirements Engineering, vol. 15, 2010, 307–311.