Challenges in applying the concept of aspect-orientation

Mehmet Aksit
The TRESE Group
Department of Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede, The Netherlands
aksit@cs.utwente.nl
http://trese.cs.utwente.nl

To receive suggestions/directions to the questions please send an email to aksit@cs.utwente.nl and let me know what you are interested in!

Table of contents

• Early aspects
• Analysis and design models
• AOP languages
Early aspects

Aspect oriented requirement analysis is important but not sufficient. We need ways to map requirements to (technical) solutions.

- Are there aspects that are specific to certain domains?
- Are there aspects that are common to all domains?
- How can we model qualities (performance, reliability, adaptability) as domains, and what are the aspects in these domains?
- How can we combine aspects from requirements and aspects from domains?

Analysis and design models

Extending UML (etc) is important but not sufficient. We need ways to represent aspects at a higher level of abstraction than UML.

- How can we model patterns at a higher level abstraction than AOPL’s?
- How can we model patterns with crosscutting concerns (CC) at a higher-level abstraction than AOPL’s?
- How can we model patterns with CC concerns + semantic constraints at a higher level abstraction than AOPL’s?
- How can we transform (these) patterns to AOPL’s?
Analysis and design models (cont’d)

Aspect oriented modeling aims at addressing decomposition and composition problems. Different qualities require different compositions-decompositions and possibly different aspects!

- How can we model quality factors so that we can reason with the quality models to determine the necessary compositions-decompositions and aspects?
- How can we optimize models for certain qualities?
- How can we adapt models to changing context and requirements; how can we keep models optimal in case of these changes?

AOP languages

- Most AOP languages are general purpose and define advices (CC behavior) in a programming language like Java. However, we need domain specific (or high-level) languages to ultimately express and compose the concerns in that domain and also be able to verify programs.
- How can we model domain specific aspects in an AOP language?
- How can we compose multiple domain specific aspects together?
- How can we compose domain specific aspects with general purpose aspects?
- How can we verify the composed aspects?
- How can we define aspects independent of implementation platforms (language, compile time, run-time transparency)?