Detection of Conflicting Functional Requirements in a Use Case-Driven Approach

Jan Hendrik Hausmann, Reiko Heckel and Gabi Taentzer, 2002

Presented by: Laura Walsh
Motivation

Find conflicting requirements as early as possible!
Motivation
Goal

- Analyse the requirements of the system before starting to build it, in order to identify whether there may be conflicting requirements.

- Add information to UML models which tell the modeller where there is the potential for conflicts.
Types of Consistency to Maintain

1. **Consistency of aspects**
   
   Use cases refer to situations from the problem domain which are not represented in the static model.

2. **Consistency of views**
   
   Semantic overlap between use cases expressing different requirements.
Running Example

Figure 1: Class diagram of the shop

Class diagram - to represent static requirements
Figure 3: Use case diagram of the shop

Use case diagram- to represent dynamic requirements
Figure 4: Action specifications for use case buy goods

Figure 5: Action specifications for use case sell goods

Action specifications - to represent functional requirements
Rules

Figure 7: Application of the rule bill good
Representing the Model

Typed graph transformation system $G = \langle TG, C, P, \pi \rangle$

$TG = \text{Type Graph (an abstract representation of the class diagram)}$

$C = \text{Constraints (what is allowable in the system)}$

$P = \text{Rule/action names}$

$\pi = \text{mapping between rule names (from P) and the expression of the rule in TG}$
What Causes a Conflict?

Parallel Independence: there can be no overlap in the items that are deleted by two transformations

Sequential Independence: there can be no overlap in the items that are created by two transformations

Figure 9: A conflict between pay bill and settle bill
Finding Conflicts

Find all critical pairs among transformations (can be done using graph transformation system AGG)
Strengths

- Simple implementation that has the potential for great improvement (of efficiency, cost cutting) to the requirements phase of software modelling

- Approach allows modeller to use their own CASE tool (along with AGG tool which already exists)
Weaknesses

- No study on whether their proposed additions to use case models would actually help modellers

- As the class diagram grows larger and more complicated, there will be many conflicts to sort through. Is it reasonable to expect modellers to manually review each flagged potential conflict?
Final Thoughts / Questions

- Small scope of the study

- Which (if any) techniques have been widely adopted since this paper was published?
Discussion

- How could the scope have been expanded?
- What are some ways that the researchers could have conducted a study to find out if their ideas had a significant impact?
- Do you think this process has the potential to be used by modellers? Why or why not?