Detection of Conflicting Functional Requirements in a Use Case driven Approach

Static analysis technique based on graph transformation

Jan Hendrik Hausmann
Reiko Heckel
Gabi Taentzer
University of Paderborn, 2002

Presented by Christina Christodoulakis
University of Toronto, Fall 2012
Requirements Capturing

I'll need to know your requirements before I start to design the software.

First of all, what are you trying to accomplish?

I'm trying to make you design my software.

I mean what are you trying to accomplish with the software?

I won't know what I can accomplish until you tell me what the software can do.

Try to get this concept through your thick skull: the software can do whatever I design it to do!

Can you design it to tell you my requirements?

© Scott Adams, Inc./Dist. by UFS, Inc.
The Authors Propose:

- Formal interpretation of use case models consisting of use case, activity and collaboration diagrams based on concepts of graph transformation.
- Make precise the notions of conflict and dependency between functional requirements expressed in different use cases.
Outline

- Introduction
- Motivation
- Formal Interpretation of Models
- Conflicts and Dependencies
- Analysis Method and Tool Support
- Conclusions
Introduction

➢ Use cases are used to model requirements
  • Dynamic (sequence | activity diagrams)
  • Functional (natural language)

➢ Consistency problems
  • Of aspects (requirements are not captured in static domain model, executing a use case may violate constraints)
  • Of views (semantic overlap due to dependencies or conflicts)
Introduction

- Diagrammatic specification of pre and post conditions of actions by UML collaborations formally interpreted as graph transformation rules.
- Formalize intuitive notions of conflict and dependency of use cases.
Aim of the paper:

NOT formal verification of consistency, but detection of potential inconsistency.
Outline

- Introduction
- Motivation
- Formal Interpretation of Models
- Conflicts and Dependencies
- Analysis Method and Tool Support
- Conclusions
Motivation

- The running example:

*Figure 1: Class diagram of the shop*
Motivation

Figure 3: Use case diagram of the shop

Dynamic Model
Motivation

The missing link:

- Functional requirements hidden in natural language (names in use case and action states reference classes)
- Authors propose a rule based specification of pre and post conditions and effects of actions by means of UML collaborations
Motivation

Collaboration rules

Figure 4: Action specifications for use case buy goods

Figure 5: Action specifications for use case sell goods
Outline

- Introduction
- Motivation
- Formal Interpretation of Models
- Conflicts and Dependencies
- Analysis Method and Tool Support
- Conclusions
Graphs

“A graph consists of a set of vertices V and a set of edges E such that each edge e in E has a source and a target vertex s(e) and t(e) in V respectively”

Graphs whose edges and vertices are decorated with textual or numerical information are attributed graphs.
A class diagram is represented by type graph TG plus set $C \subseteq \text{Constr}(TG)$ of constraints over TG.

The class of instance graphs over TG is denoted by $\text{Inst}(TG)$ while we write $\text{Inst}(TG, C)$ for subclass satisfying constraints $C$. 

Formal interpretation of models
Graph Transformation Rules

$p : L \rightarrow R$

preconditions

postconditions

$L, R$ are TG instance graphs such that union is defined.

Edges which appear in both $L, R$ are connected to same vertices in both graphs etc.
Use Cases as views

- A Use Case represents a view of the overall model corresponding to requirements of a particular actor or group of actors.

- In a TG, a view is represented by a subgraph and subset of rules.
Outline

- Introduction
- Motivation
- Formal Interpretation of Models
- Conflicts and Dependencies
- Analysis Method and Tool Support
- Conclusions
Conflicts and Dependencies

- Analysis based on notion of independence
  - Parallel (absence of conflicts)
  - Sequential (absence of causal dependencies)
From pre state G to post state H : $G \xrightarrow{p(o)} H$

- Find an occurrence $o|L$ of the left-hand side L in the current object graph G.
- Remove all the vertices and edges from G which are matched by $L \setminus R$.
- Glue D with $R \setminus L$ to obtain the derived graph H.
Conflicts and Dependencies

- Given sequence of transformations, if the second transformation is not independent of the first, the first causes the second.

- If the second transformation does not delete objects needed for the first, the applications can be exchanged without affecting overall result.
Outline

- Introduction
- Motivation
- Formal Interpretation of Models
- Conflicts and Dependencies
- Analysis Method and Tool Support
- Conclusions
Analysis Method and Tool Support

- Given two use cases, make the modeler aware of potential conflicts or dependencies between functional specifications.
- Display these as annotations on the model, and provide small counter examples.
Implemented in the AGG tool using critical pair analysis

Pair of transformations which are in conflict, and such that graph $G$ is minimal

Set of critical pairs represents all potential conflicts
AGG offers a graphical user interface to browse through computed pairs of critical pair analysis
Analysis Method and Tool Support

Left hand side of both rules

Overlapping graphs
Graph transformation and critical pairs are stored in XML based language and Critical Pair Exchange format.

Computed pairs could be analyzed by AGG but it is more natural to view analysis results on original models.

Computed pairs are exported and transformed to add annotations of conflicts and dependencies.
Analysis Method and Tool Support

Figure 11: Conflicts between use cases buy goods and sell goods
Bill can be created even if customer has not selected a good

Figure 12: Causal dependencies between use cases buy goods and sell goods
Casual dependencies are depicted together with essential objects or links.
Ultimately it is the modeler’s job to decide which dependencies and conflicts actually represent errors or inconsistencies.

Result in either changing the model or better documentation of the decision.
Activity of paying now includes the returning of cart
Outline

- Introduction
- Motivation
- Formal Interpretation of Models
- Conflicts and Dependencies
- Analysis Method and Tool Support
- Conclusions
Conclusions

- There must exist a tradeoff between understandability and expressivity of modeling concepts.
- The approach is formal without being limited to trained professionals, and can be used by modelers to remove conflicts and facilitate dependencies at an early stage in the design.