Rule-based Detection of Inconsistency in UML Models

WenQian Liu
University of Toronto
October 1, 2002

Outline

- The Problem
  - inconsistency management in design models
- The Solution
  - knowledge base solution: production systems
- Discussion and Summary
The Problem

- Maintaining consistency in large evolving UML design models is hard.
- Change introduces inconsistency, i.e.
  - conflicting description of the system
  - violation of predefined constraints
Motivation

- Manual inconsistency management is tedious and error prone.
- Many existing solutions use preventive and batch approaches.
- We need computer assistance that can
  - react to change,
  - work incrementally, and
  - provide non-interruptive but timely feedback.
Example 1

Redundancy

Sequence diagram 1: Request a new meeting

- i: Initiator
  - initiateMeeting()
  - completed

- : OrganizeMeeting
  - new()
  - return m

- m: Meeting
  - return m

Sequence diagram 2: Request a specific meeting

- i: Initiator
  - initiateMeeting()
  - completed

- : OrganizeMeeting
  - new()
  - return m
  - addLocation(l, m)
  - addTime(t, m)

- m: Meeting
  - return m
Example 2

Conformance to Common Design Practices

Law of Demeter

class Borrower {
    ...
    Book[] p = getLibrarian().findBookByCallNumber().listCopies();
    ...
}
Classes of Design Inconsistency

Design Consistency

- Redundancy
  - Design Related
  - Data Related

- Conformance
  - Constraints
  - Standards

- Change
  - Edit blocks
Outline

- The Problem
- The Solution
  - knowledge base solution: production systems
- Discussion and Summary
Production Systems (1)

- Knowledge base (contains facts)
- Rule base (contains rules)
- Production system constantly applies rules to facts
- Rules modify facts
Production Systems (2)

● Facts
  ● definition of a fact
    (type attribute:a₁ ... attribute:aₙ)
  ● examples
    (transaction type:"debit" amount:50 accountid:00641)
    (account id:00641 balance:395)

● Rules
  ● definition
    IF condition THEN action
  ● examples
    IF (transaction type:"debit" amount:x accountid:a)
    (account id:a balance:y∧{≥x})
    THEN REMOVE 1
    MODIFY 2 (balance [y-x])

● Basic operation:
  1. recognize
  2. resolve conflict
  3. act
Production Systems (3)

- reactive, incremental, extensible
- suitable for open-ended tasks
- good for justification and explanation
Inconsistency Identification and Resolution

Four types of production rules
1. **Inconsistency**
   - identify violations of consistency properties
2. **Resolution**
   - resolve inconsistency upon receiving user’s choice
3. **Cleanup**
   - remove invalid inconsistency messages
4. **Dynamic control**
   - modify rule behaviors on the fly
Inconsistency Rule – Example 2

IF (sequenceMessage id:\(m_1\) from:\(T_1\) to:\(T_2\) return:\(c\) pid:\(p\))
(sequenceObject name:\(c\) type:\(T_3\)\(\{\neq T_2\}\) pid:\(p\))
(sequenceMessage id:\(m_2\) from:\(T_1\) to:\(T_3\) pid:\(p\))

THEN ADD (inconsistency id:[newId()]\(\land s\) ruleid:”sc-1”
  location:((sequenceMessage \(m_1\))
    (sequenceMessage \(m_2\))
    (sequenceObject \(c\)))
  msg:”Violation of the Law of Demeter.”)
ADD (userchoice id:[newId()] pid:\(s\) action:remove
  targetID:\(m_1\) targetType:sequenceMessage)
ADD (userchoice id:[newId()] pid:\(s\) action:remove
  targetID:\(m_2\) targetType:sequenceMessage)
Resolution Rule – Example 2

\[
\text{IF} \quad (\text{inconsistency id}: s) \\
\quad (\text{userchoice pid}: s \ \text{action}: \text{remove} \ \text{targetID}: m \\
\quad \quad \text{targetType}: \text{sequenceMessage}) \\
\quad (\text{userinput pid}: s \ \text{action}: \text{remove} \ \text{targetID}: m) \\
\quad (\text{sequenceMessage id}: m) \\
\]

\[
\text{THEN} \quad \text{REMOVE 1} \\
\quad \text{REMOVE 2} \\
\quad \text{REMOVE 3} \\
\quad \text{REMOVE 4}
\]
Outline

- The Problem
- The Solution
- Discussion and Summary
Discussion

- worst-case complexity \( O(W^{2C-1}) \)
  where \( W \) = size of the WM, 
  \( C \) = number of patterns in a rule
- declarative (e.g. xlinkit, design guidance)
- procedural (e.g. Argo/UML)
- classes of consistency
Summary

- **Contributions**
  - inconsistency classification
    - redundancy, conformance to constraints and standards, change
  - rule-based detection and resolution approach
    - incremental inconsistency detection and monitoring
    - single or multi-step resolution
  - integration with UML editors

- **Future Work**
  - classification scheme and standard solutions
  - verification of semantics over extensions to UML
  - analysis of inconsistency patterns over history of edits
Questions?