

Conflict Management with Goal Oriented Requirements Engineering

Requirements Engineering II

Index

- Conflict Management
- Goal-oriented approach
- KAOS Framework
 - Language
 - Elaboration Method
- Managing Conflicts

Conflict Management

Introduction:

- Stakeholder have different requirements
- All software projects affected
- Critical success factor

How-to:

- Identify
- Analyse
- Resolve

Goal-oriented RE

Approach:

- Derive requirements from goals
 - every requirement belongs to a goal
- Convert goal conflicts to new requirements
=> otherwise not discovered

Benefits:

- Goals more stable than requirements
- Goals provide completeness criterion for specification

KAOS Framework

(Knowledge Acquisition in autOmated Specification)

- Key Questions:

"How can incompleteness, contradictions and ambiguities of requirements be handled?"

"How can requirements specification be checked against set of desired properties?"

"How can also non-functional requirements be defined formally?"

=> define formal framework based on goal-oriented approach

KAOS Language

- Ontology:
Object, Operation, Agent, Goal, Requirement,
Assumption & Scenario
- Time dimension:

○	(in the next state)	●	(in the previous state)
◇	(some time in the future)	◆	(some time in the past)
□	(always in the future)	■	(always in the past)
$\mathcal{W}$	(always in the future <i>unless</i>)	$\mathcal{U}$	(always in the future <i>until</i>)

KAOS Language

- Constructs:
 - outer semantic layer
 - inner formal assertion layer

```
Goal Achieve[ParticipantsConstraintsKnown]
  Concerns Meeting, Participant, Scheduler
  Refines MeetingPlanned
  RefinedTo ConstraintsRequested, ConstraintsProvided
  InformalDef A meeting scheduler should know the
    constraints of the various participants invited
    to the meeting within some deadline d after
    invitation.
  FormalDef  $\forall m: \text{Meeting}, p: \text{Participant}, s: \text{Scheduler}$ 
    Invited(p, m)  $\wedge$  Scheduling(s, m)
     $\Rightarrow \Diamond_{\leq d} \text{Knows}(s, p.\text{Constraints})$ 
```

Elaboration Method

- Goal elaboration
 - "Why" questions for parent goals
 - "How" questions for child goals
 - Object capture
 - Operation capture
 - Operationalization
 - Responsibility assignment
- + always use Metalevel-Knowledge !

Managing Conflicts

- Key Questions:

"When exactly should divergences be identified?"

"When exactly should an identified divergence be resolved?"

=> definitive answer out-of-scope

Managing Conflicts

Goals:

- Define / study relationship of inconsistency types
- Formal techniques for identifying goal conflicts
- Method to formally resolve goal conflicts

Types of inconsistencies

A set of descriptions is inconsistent if there is no way to satisfy those descriptions all together

What inconsistencies are there?:

- Terminology Clash
- Designation Clash
- Structure Clash
- Conflict
- Divergence

Divergence

Conflicts only if certain boundary conditions are met

- 1) $\{\text{Dom}, B, \bigwedge_{1 \leq i \leq n} A_i\} \vdash \mathbf{false}$ (*logical inconsistency*)
- 2) for every i : $\{\text{Dom}, B, \bigwedge_{j \neq i} A_j\} \not\vdash \mathbf{false}$ (*minimality*)
- 3) there exists a scenario S and time position i such that
 $(S, i) \models B$ (*feasibility*)

Divergence

- Often resource conflicts
- Conflicts of interest (maximize vs minimize)
- Special cases:
 - Competition
 - Two instances require the same resource
 - Obstruction
 - One assertion may not hold

Detection

Regressing Negated Assertions

- Try to find preconditions for negative assertion
- Derive domain constraints

Using Divergence Patterns

- Identify frequent divergence patterns
- Achieve-Avoid
- Retraction

Resolving divergence

Assertion transformation

- Avoid boundary conditions
- Goal restoration
- Conflict anticipation
- Goal weakening
- Alternative goal refinement
- Divergence Resolution Heuristics
 - Arbitrator to solve competition
 - Reservation systems to solve resource conflicts

Resolving divergence

Object transformation

- Object refinement
 - Create subtypes for objects
- Agent refinement
 - Create subtypes for agents

Case Study - Key findings

- Interesting approach to visualize conflicts
- conflicts are difficult to identify
- lack of real requirements
- very difficult for novices
- not suitable for every project

Conclusion

- Benefits:
 - "all-inclusive" approach
 - Allows formal proof of specification
 - Suitable for mission-critical software
 - Artifacts provide full overview of project
- Drawbacks
 - High complexity due to formalism
 - Methodology experts needed
 - Limited use within industry yet

Questions?

Thanks for your attention.....

Backup Slides

Viewpoints

Different actors, different concerns & perceptions

Goals:

- 1) restrict the visibility of conceptual features to specific contexts associated with corresponding actors, and
- 2) allow product-level inconsistencies to be captured and recorded for later resolution.

Implemented as facets

Example of viewpoints

Entity Meeting

Has Date: *Calendar*; Location: *String*

Entity MeetingToBeScheduled

FacetOf Meeting **SeenBy** MeetingOrganizer

Has ExcludedDates: **SetOf**[*Calendar*]

RequiredEquipment: **SetOf**[*String*]

DomInvar $\forall m: \text{Meeting}$

$m.\text{Date not in } m.\text{ExcludedDates}$