SUPPORTING AGILITY IN MDE THROUGH MODELING LANGUAGE RELAXATION

Rick Salay and Marsha Chechik
Department of Computer Science, University of Toronto, Canada

Sept. 29, 2013
The Agility Conflict in MDE

Models are used by humans and programs

BUT

Humans want expressive freedom

Programs need well-defined constraints
We focus in two types of agility

- **Omission agility**
  - Allowing the expressive freedom to omit information that is not relevant, certain, etc.

- **Clarity Agility**
  - Allowing the expressive freedom to present information in ways that are easier to understand.

Agility requires the *relaxation* of the language
Example

- Omission agility
  - Target of association is omitted because it is not yet known
• Clarity agility
  • Classes in hierarchy are aligned to indicate how instances are paired
  • ... rather than expressing this using OCL
• Omission Agility
  • The exact class of “park(location)” operation is not known

• Clarity Agility
  • Use a single mention of the operation external to but linked to both

Example
Example

- Clarity Agility
  - Put name of class outside of box to avoid clutter.
Example

- Omission Agility
  - The precise relation types between these classes is irrelevant and so is omitted.
Summary

• Omission Agility
  - drop info
  - provide alternatives
  - use abstraction

• Clarity Agility
  - adding notation
  - leveraging visual conventions
Generalizing from examples

Observation: Supporting agility in MDE requires transformations on concrete syntax!
Relaxation and tightening: language aspects

Modeling languages have a vocabulary and well-formedness constraints

- Relaxation: Extending vocabulary - Tightening: Translate the extension
- Relaxation: Weakening constraints - Tightening: Repair violation
Applying to examples

- Relaxation: Extend vocabulary; Tighten: Translate extension
  - oval → operator
  - link → containment
  - dashed link → rel
  - alignment → OCL

- Relaxation: Weakening constraints; Tighten: Repair violation
  - add class
  - remove ownership
  - move class name
Applying to examples

• Relaxation: Extend vocabulary; Tighten: Translate extension
  oval → operator
  link → containment
  dashed link → rel
  alignment → OCL

• Relaxation: Weakening constraints; Tighten: Repair violation
  add class
  remove ownership
  move class name
  Some tightening requires choice
Alternative to choice: Partial modeling

- Rather than choosing one possibility, use a partial model to express all possibilities
- Partial models represent sets of models
  - Modal Transition Systems [Larsen and Thomsen ’88]
    - Much follow-on work: Chechik, Uchitel, Ben-David, etc.
  - MAVO [Salay, Famelis and Chechik ‘12]
    - Generalizes from behavioral models
- But ..
  - Applying programs to partial models requires lifting the algorithm to sets of models
  - e.g., lifting transformations to partial models
    - “Transformation of Models Containing Uncertainty” [Famelis et. al. Models’13]
MAVO partial modeling example

- The V annotation means: treat the class C like a “variable class”
  - represents all possible well-formed models obtained by instantiating variable C with a particular class
MAVO partial modeling example

- The M annotation means: the link may or may not exist
  - Represents all possible well-formed models in which some of the links are present.
Towards Tool Support for Relaxation/Tightening

Relaxation

- Use a general drawing tool (e.g., Visio) that allows constraints to be selectively disabled or deferred.

Tightening

- For extended vocabulary
  - Identify: new symbols being used
    - Only possible automatically when it causes a concrete syntax change. In other cases, evident spatial relations may be a clue
  - Tighten: provide a tool for translating the new language construct in terms of existing ones (e.g. using ATL)

- For weakened constraints
  - Identify: constraint violation
  - Tighten: use existing approaches for computing the minimal repair to a constraint violation
    - e.g. [Xiong et. al 2009], [Reder et. al. 2012], etc.

- Optional: use partial modeling to handle choice
Summary

• There is an agility conflict in MDE
  • Humans want freedom; programs need structure
• We propose an approach to allow freedom and structure
  • Relax for humans; tighten for programs
    • Optionally use partial models to address choice
  • Focus on two kinds of agility:
    • Omission agility: freedom to leave out information
    • Clarity agility: freedom to express clearly
  • Start of a theory: vocabulary extension/constraint weakening
• Explored approach using examples
• Potential tool support with existing technologies
• Just the beginning …
  • Develop theory, test feasibility, extend to other kinds of agility, etc.
THANK YOU