Implementation of Collaborative Data and Schema Conflict Resolution in EA Repositories

Guided Research: Introductory Presentation; 8.7.2013

Referee: Björn Kirschner

Supervisor: Sascha Roth

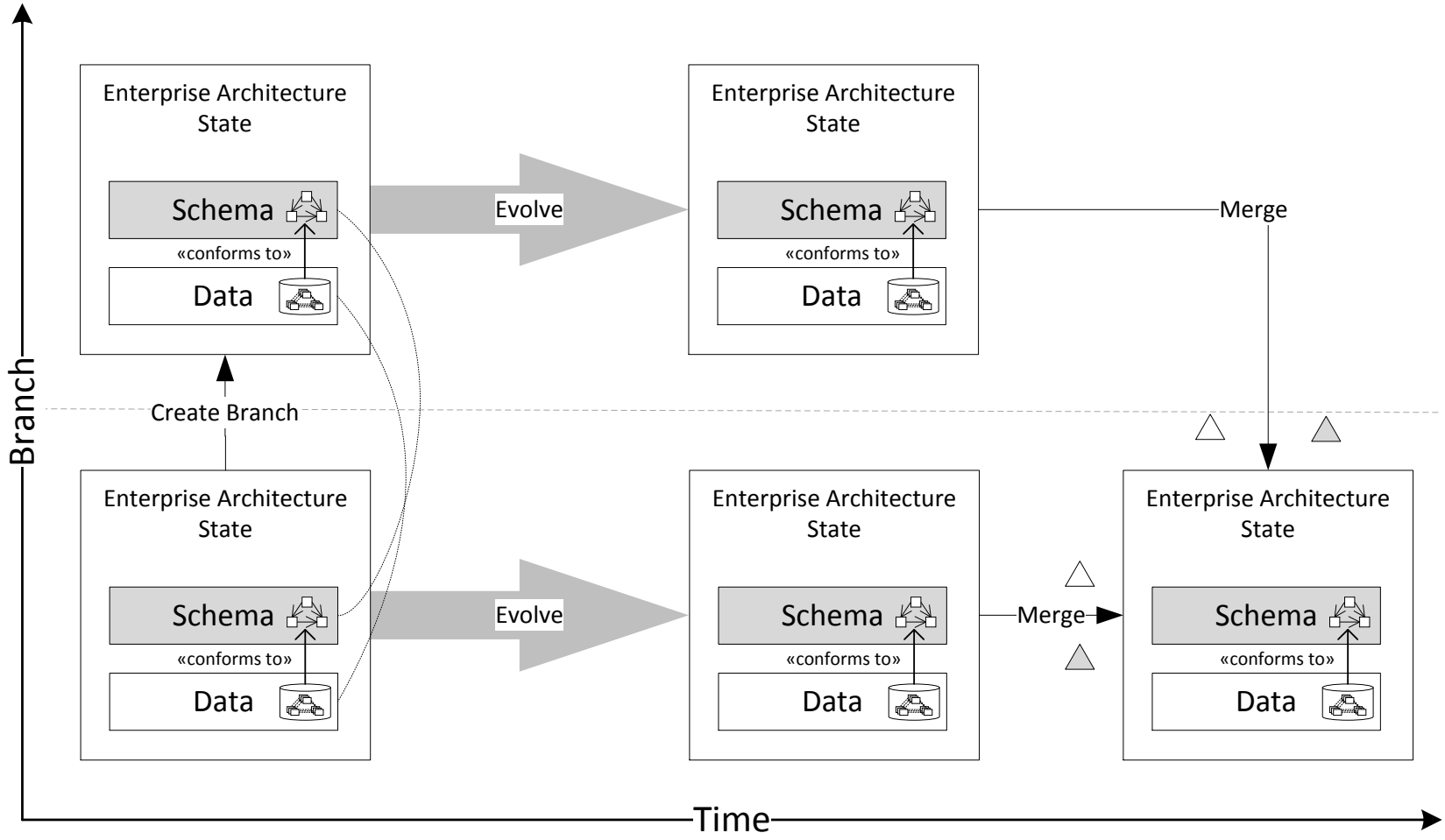
Software Engineering betrieblicher Informationssysteme (sebis)
Ernst Denert-Stiftungslehrstuhl

wwwmatthes.in.tum.de

Agenda

1. Motivation
2. Research Questions
3. Proposed Meta-Information Model
4. Related Work
5. Timeline
6. Conclusion

Motivation: Co-Evolution of Models in an EA

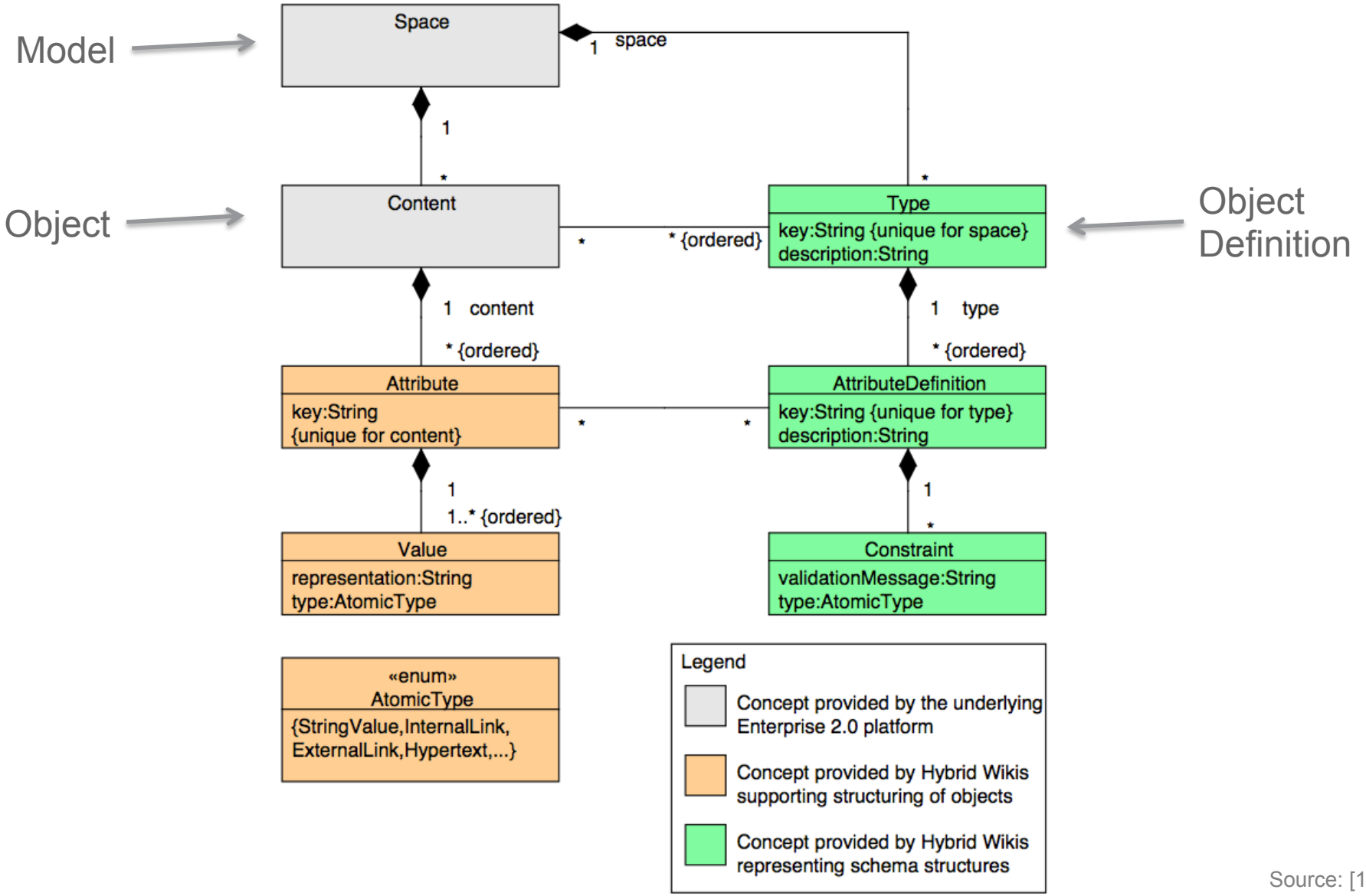


Research Questions

How can model conflicts in hybrid wiki workspaces be solved collaboratively?

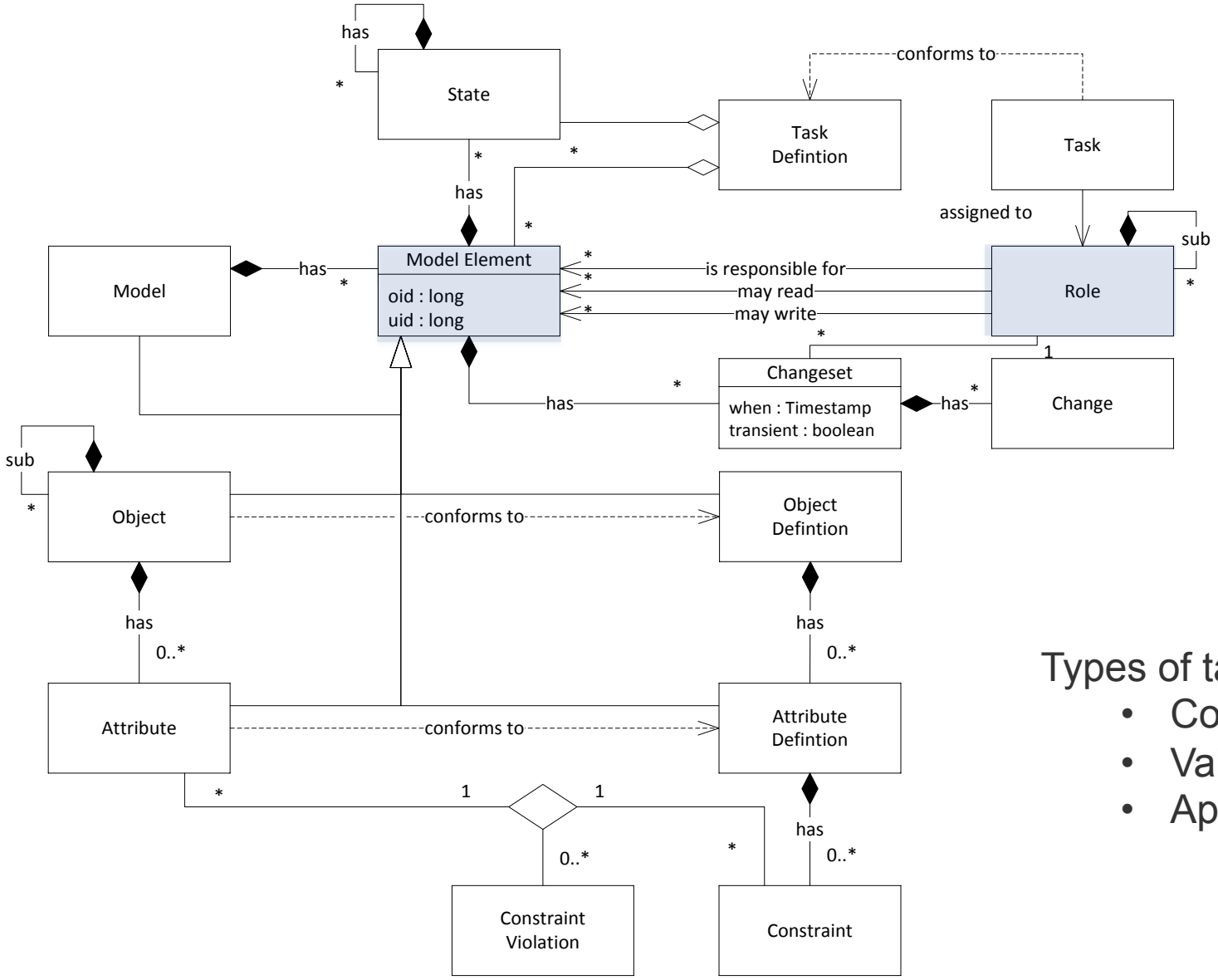
- How does a meta-information model, which supports the collaborative solving, look like?
- How would an algorithm for merging co-evolving EA models look like?
- How can users be involved for the collaborative solving process?

Meta-Information Model in Hybrid Wikis



Source: [1]

Proposed Meta-Information Model



Types of tasks:

- Conflict
- Validate
- Approve

Related Work: Collaborative Handling of Conflicts

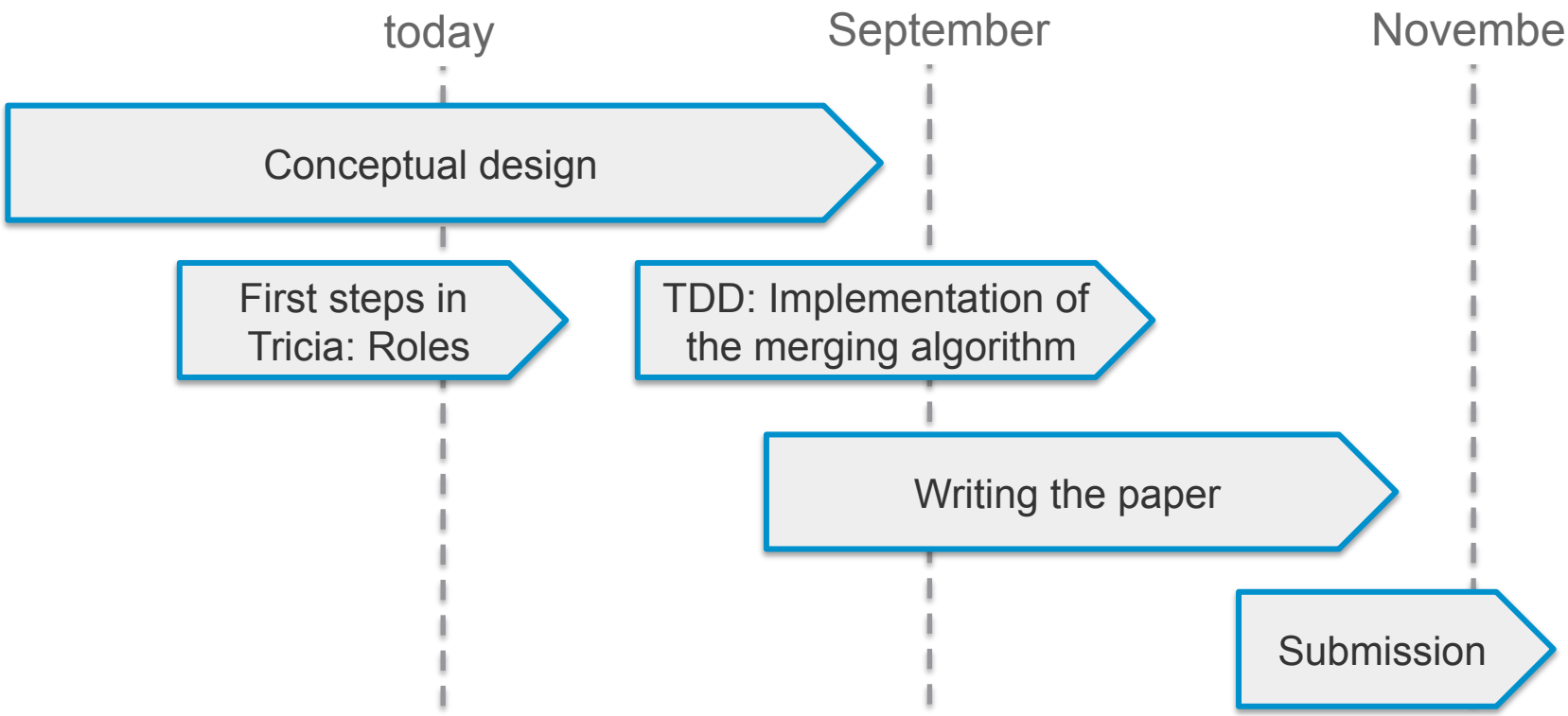
Insert	Delete	Update	Use	Move	
					Insert
		×	×	×	Delete
		×	~	~	Update
				~	Use
				×	Move

× ...*Conflict*

~ ... *Warning*

Source: [2]

Timeline


today
September
November
Conceptual design
First steps in Tricia: Roles
TDD: Implementation of the merging algorithm
Writing the paper
Submission

Conclusion

Prospects

- Publication of a conference paper
- Technical evaluation via a prototype implementation in Tricia
- Tobias‘ thesis: Task handling via visualizations
- For a future master‘s thesis: evaluation of the concept in the industry

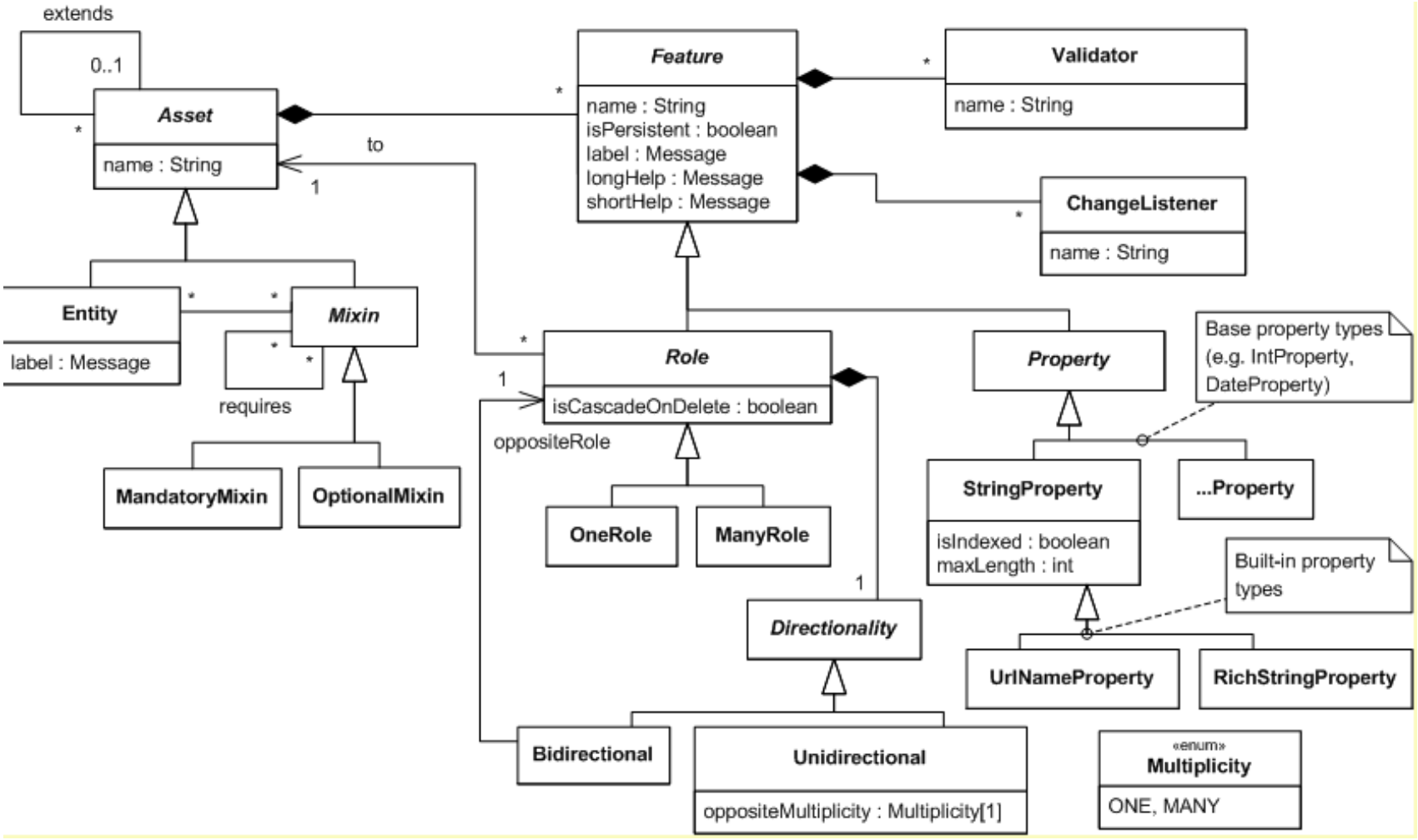
Sources

[1] C. Neubert. Facilitating Emergent and Adaptive Information Structures in Enterprise 2.0 Platforms. PhD thesis, Technical University Munich, München, Germany, 2012.

[2] K. Wieland, P. Langer, M. Seidl, M. Wimmer, and G. Kappel. Turning conflicts into collaboration - concurrent modeling in the early phases of software development. Computer Supported Cooperative Work: The Journal of Collaborative Computing, tba:1–52, 2012.

Implementation: Complications

Tricia Object Model:



Merging Algorithm

Algorithm 1: n-way merging of models

Data: Model $M_{1..n}$, Baseline Time t_b, Target Model M_t
Result: Task $T_{1..n}$

```
1: conflicts ← (key : ∅, value^{t_b} : ∅, value : {∅})
2: approve ← (key : ∅, value^{t_b} : ∅, value : {∅})
3: validate ← (key : ∅, value^{t_b} : ∅, value : {∅})
4: changesets ← {∅}
5: // 1.  collect operations
6: foreach element ∈ M_{1..n} do
7:     foreach changeset ∈ element do
8:         if changeset.when > t_b then
9:             changesets ← changesets ∪ changeset
10: // 2.  consolidate operations
11: foreach c_1 ∈ changesets do
12:     foreach c_2 ∈ changesets do
13:         conf = detectConflicts(c1, c2)
14:         // see Algorithm 2 and 3
15:         if conf ≠ ∅ then
16:             if ∃c_1.modelElement ∈ conflicts then
17:                 chgConf ←
                    conflicts.getAndRemove(c_1.modelElement) ∪
                    conf
18:                 conflicts ← conflicts
                    ∪(c_1.modelElement, c_1.modelElement^{t_b}, chgConf)
19:             else
20:                 conflicts ← conflicts ∪ (c.modelElement,c)
21: foreach e ∈ conflicts.keys do
22:     foreach c ∈ changesets do
23:         if e ≡ c.modelElement then
24:             changesets.remove(c)
25: // 3.  apply to target model
26: // 4.  check consistency/schema conformance
27: // 5.  user intervention
28: // 6.  create conflict resolution tasks
29: notify role of e role of change unify them in a group, notify
    group
```