

Requirements Engineering with ViewPoints

RE II Assignment 2 - Conflict Management

Robert Erdin, Philip Schaffner and Taya Goubran

Inconsistency management

Inconsistency:

- General: two parts of a specification that do not obey a relationship that should hold
- Specific: a rule that has been broken

Conflict: interference in the goals of one party caused by the actions of another party.

Why? - Inconsistency management

- inconsistent specifications **cannot be satisfied**
 - hard to achieve consistency in a large specification written by **a team**
 - Counter-productive to prevent change than to allow inconsistency.
- proper management leads to a **robust specification** and provides important clues about **missing information**

ViewPoints

“ViewPoints are loosely coupled, locally managed distributable objects which encapsulate **partial knowledge** about a system and its domain; [...]”

“A framework for distributed software engineering, in which **multiple perspectives** are maintained separately as distributable objects ”

Each ViewPoint has..

- representation style - notation for expressing the VP
- domain - area of concern
- specification - expressed statements of the domain
- work plan - set of actions to build the specification
- work record - history and performed actions

ViewPoints aren't required to be consistent with others!

Two ViewPoints - consistency test

A (owner=Anne; domain=telephone/calling)

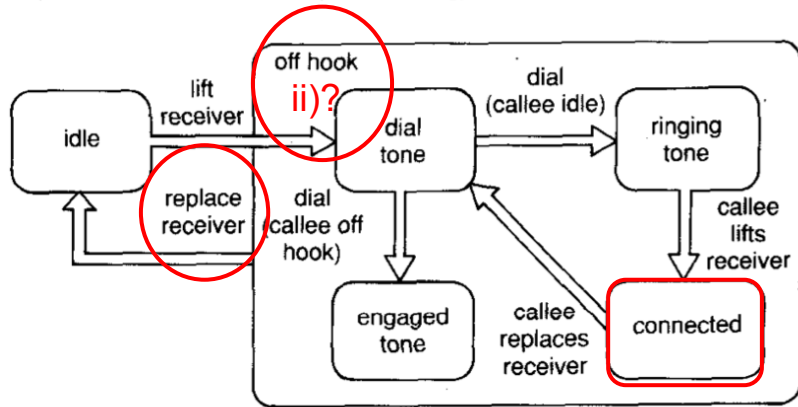


Fig. 1 Anne's initial ViewPoint specification for making a call

(i) If a transition between two states is described in one Viewpoint, and both states are described in the second Viewpoint, then the transition should also be described in the second Viewpoint.

(ii) If a state is shown as belonging to a super-state in one ViewPoint, and the same state is included in the second Viewpoint, then the super-state must also be included in the second Viewpoint.

B (owner=Bob; domain=telephone/incoming call)

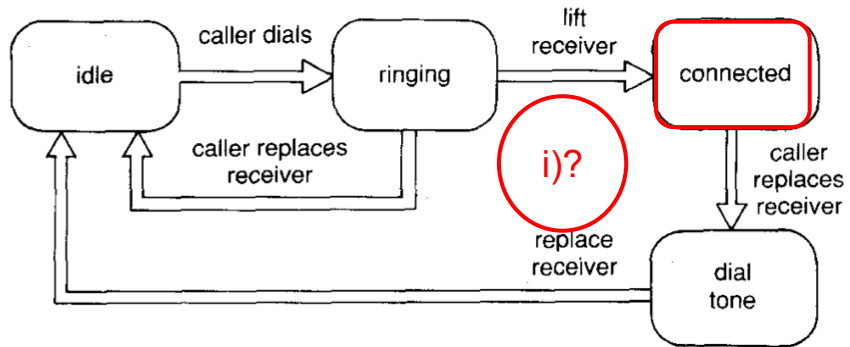
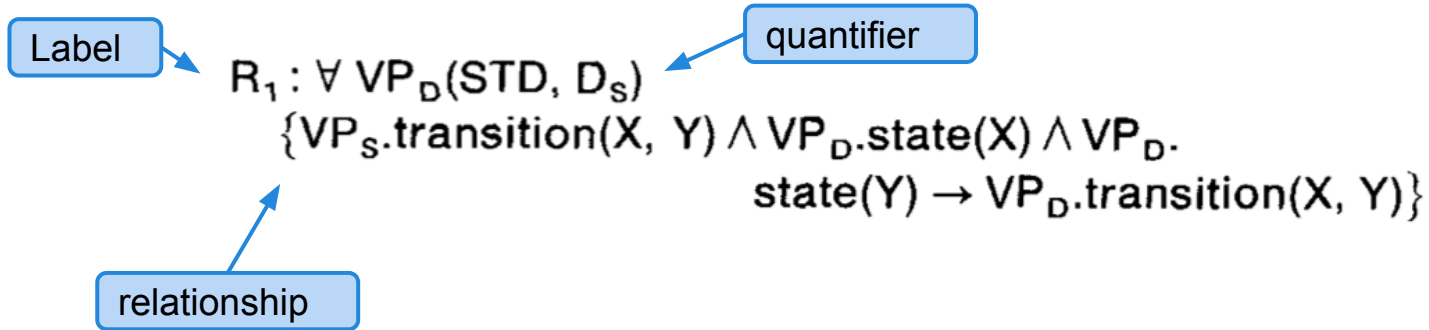


Fig. 2 Bob's initial ViewPoint specification for receiving a call

Rule Notation

(i) If a transition between two states is described in one Viewpoint, and both states are described in the second Viewpoint, then the transition should also be described in the second Viewpoint



Rule Application

didn't agree about the state “connected”

→ delay resolution

→ apply rule R_2

→ (unintentional)
resolution of R_1

B (owner=Bob; domain=telephone/incoming call)

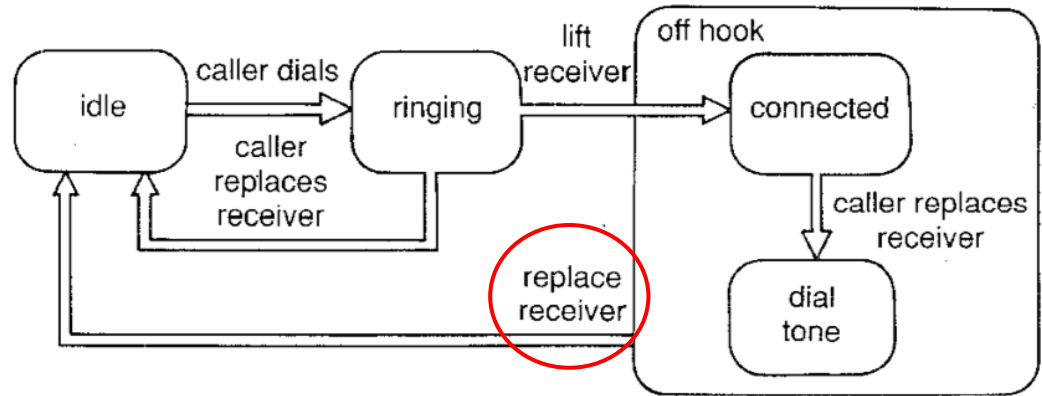


Fig. 3 Bob's ViewPoint specification after resolution

Evolving Specification

Each ViewPoint maintains a local list of **known unresolved inconsistencies**

- effect of changed ViewPoints (adding super-state)
- only Bob knows about the side effect
- no guarantee that resolved inconsistencies of the relationship expressed rules still hold
- warning when inconsistencies re-occur via work record

View Merging

Framework for merging multiple views that tolerates inconsistency between the views:

- trace back from merge to original view
- trace back to contributing stakeholders
- determine the degree of knowledge

interconnection diagram

- expressing a hypothesis about how a group of objects are related
- set of objects of a particular category and a subset of possible mappings

→ Set merge: $S_1 = \{x, y\}$, $S_2 = \{x, t\} \rightarrow \{x_{s1}, y_{s1}, x_{s2}, t_{s1}\}$

Example

merge node-sets:

$$f_{\text{node}} = \{u_1 \rightarrow x_1, u_2 \rightarrow x_2\}$$

$$g_{\text{node}} = \{u_1 \rightarrow n_1, u_2 \rightarrow n_2\}$$

merge edge-sets:

$$f_{\text{edge}} = \{v_1 \rightarrow p_1\}$$

$$g_{\text{edge}} = \{v_1 \rightarrow e_1\}$$

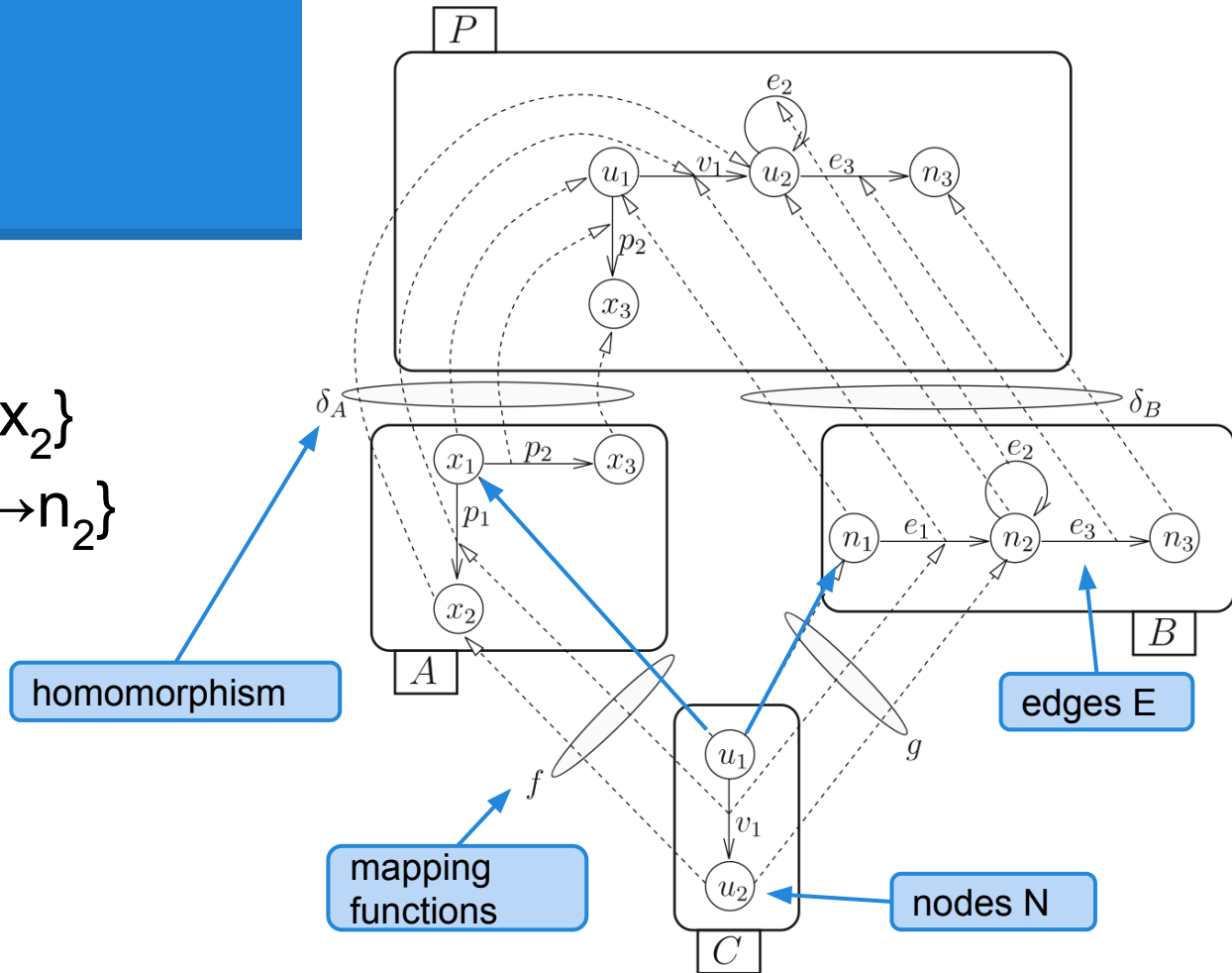


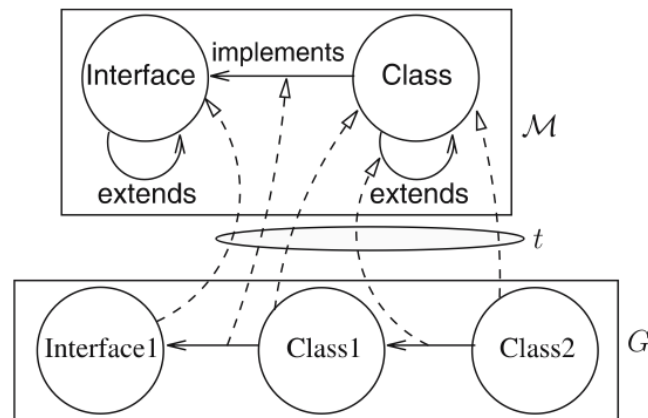
Fig. 8 Three-way merge example for graphs

Enforcement of Types

- types are not supported
- typing map: assigning a type to every element in the graph



(a)



(b)

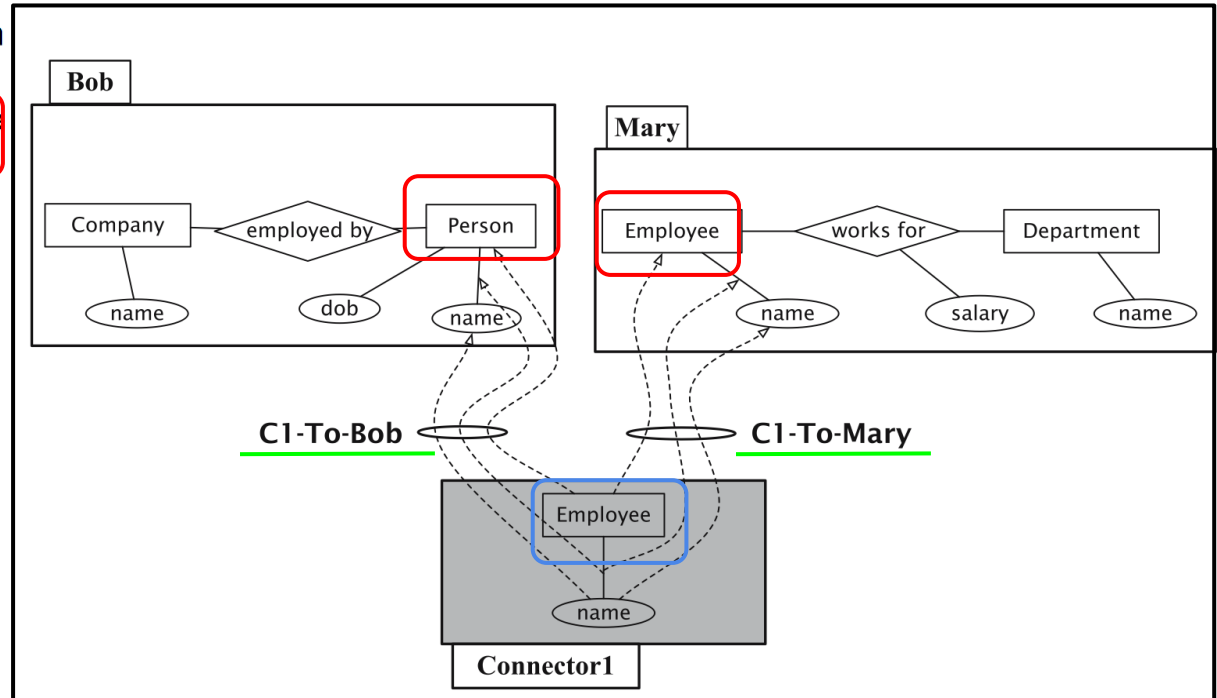
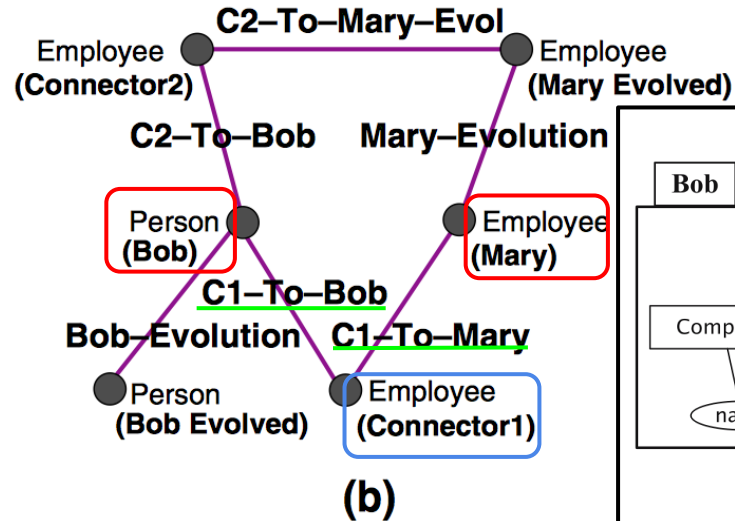
Fig. 9 Example of typed graphs

Modeling degree of knowledge

K-respecting homomorphism $\rightarrow$ knowledge preserved

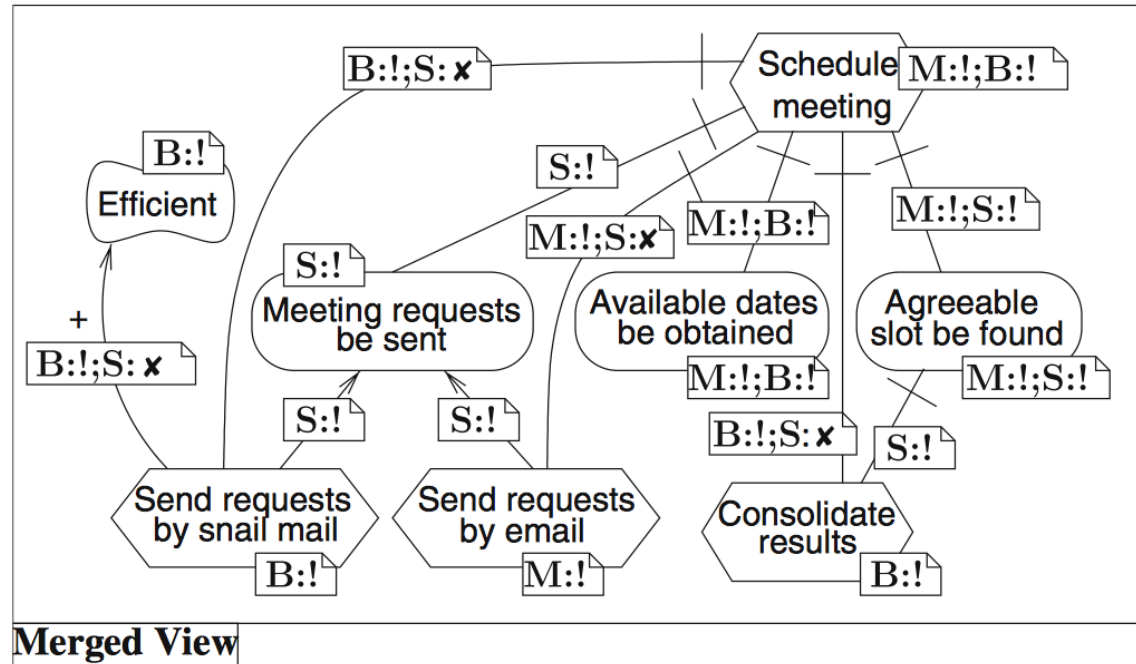
Symbol	meaning	well-conceived
!	proposed	maybe
✗	repudiated	no
✓	affirmed	yes
⚡	disputed	maybe

Traceability Links



different Stakeholder
on same view!

Stakeholder + knowledge!



Conclusion

- managing inconsistencies helps eliciting information
- manage inconsistencies with respective rules
- continuous process - evolving specification
- traceability of merged views and involved stakeholders