

Collaborative Software Development and Topic Maps

O. Drobnik, M. Ueberall



Telematics Group, Institute of Computer Science,
J. W. Goethe-University, Frankfurt/Main, Germany

Overview

- Motivation and Objective
- Development Phases and Modelling
- Development Process Guidance
- Tool Support

Motivation and Objective

Problems in Software Development:

- Consistent semantics transfer of requirements between development phases
 - e.g., requirement analysis, design, implementation
- “Loss-free” communication between heterogeneous groups of participants
 - e.g., stakeholders, designers, programmers
- Tool-supported collaboration
 - e.g., version management and logging

State of the Art:

- Modelling Languages [Harel]
 - characterises the meaning of “semantics”
- Round-Trip-Engineering [Sendall/Küster]
 - identifies requirements, problems and risks
- Ontology-based Software Development [Oberle et al.]
 - targets application server based development using semantic web techniques

Our Approach

■ Development Phases (Modelling):

- Phase-dependent base ontologies, models, schemas
- Application-oriented information management

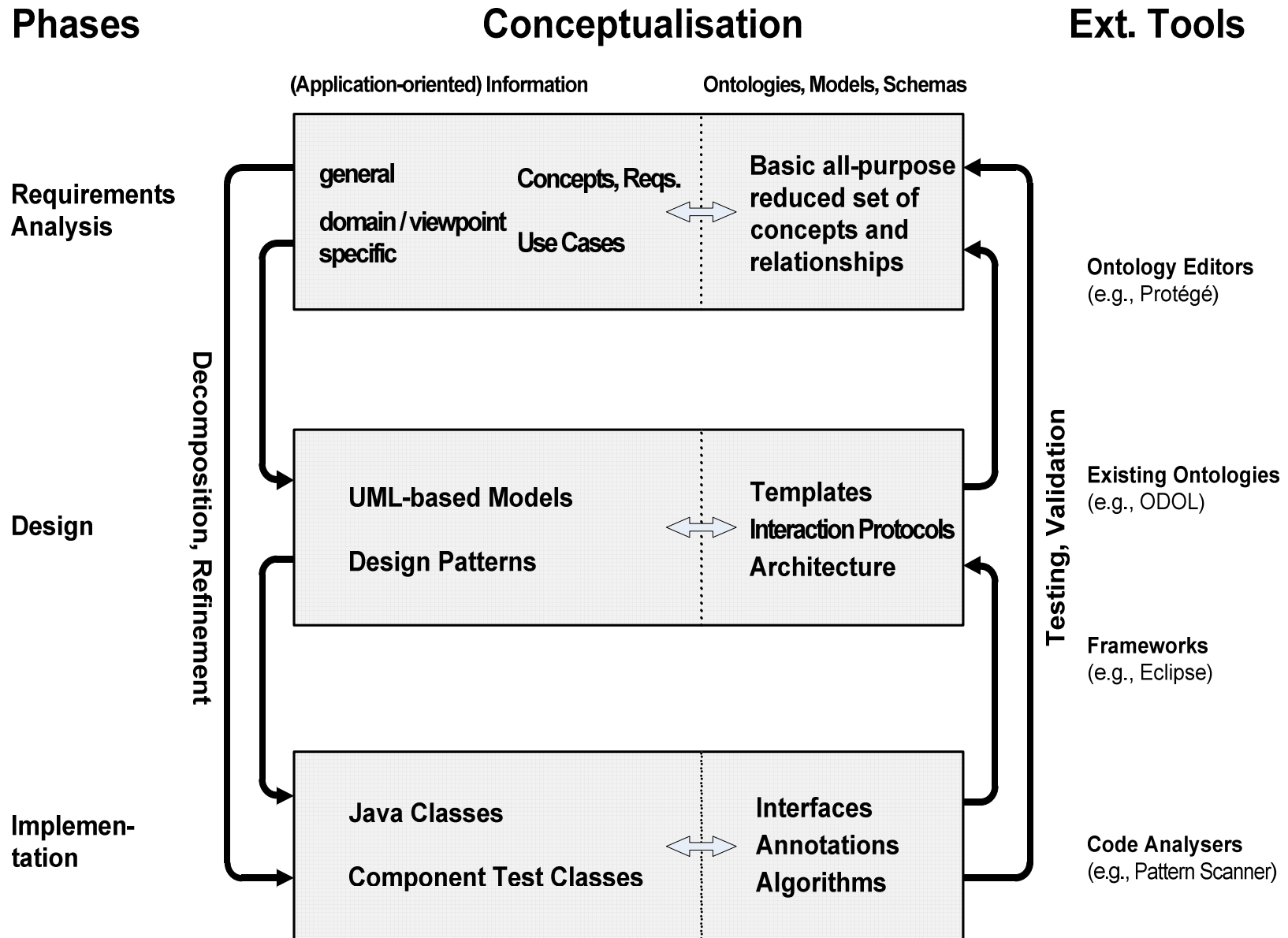
■ Development Process Guidance:

- Generic Process Model
- Visualisation

In the following:

How to adapt our methodology, why are Topic Maps particularly suited?

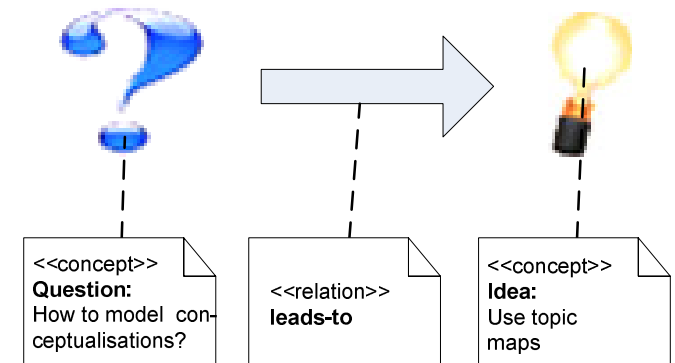
Development Phases (Simplified View)



Conceptualisation: Technical Realisation

■ Concepts and Relationships

```
[question: concept = "Question" @...]  
[idea: concept = "Idea" @...]  
...  
relationship(leads-to: relation, question, idea)  
...  
{question, comment, [[How to model conceptualisations?]]}  
{idea, comment, [[Use topic maps]]}  
...
```



■ Version Control

```
relationship(references: relation, source, destination1)  
    ~ internal-id1  
{internal-id1, human-readable-version, [[1.0]]}  
...  
relationship(references: relation, source, destination2)  
    ~ internal-id2  
{internal-id2, human-readable-version, [[1.1]]}  
superseded-by(internal-id1: outdated, internal-id2: current)  
...
```

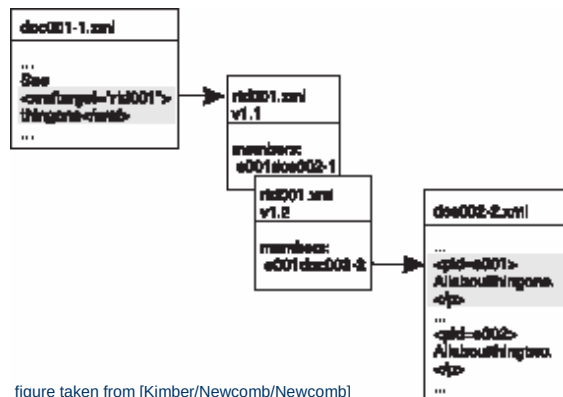
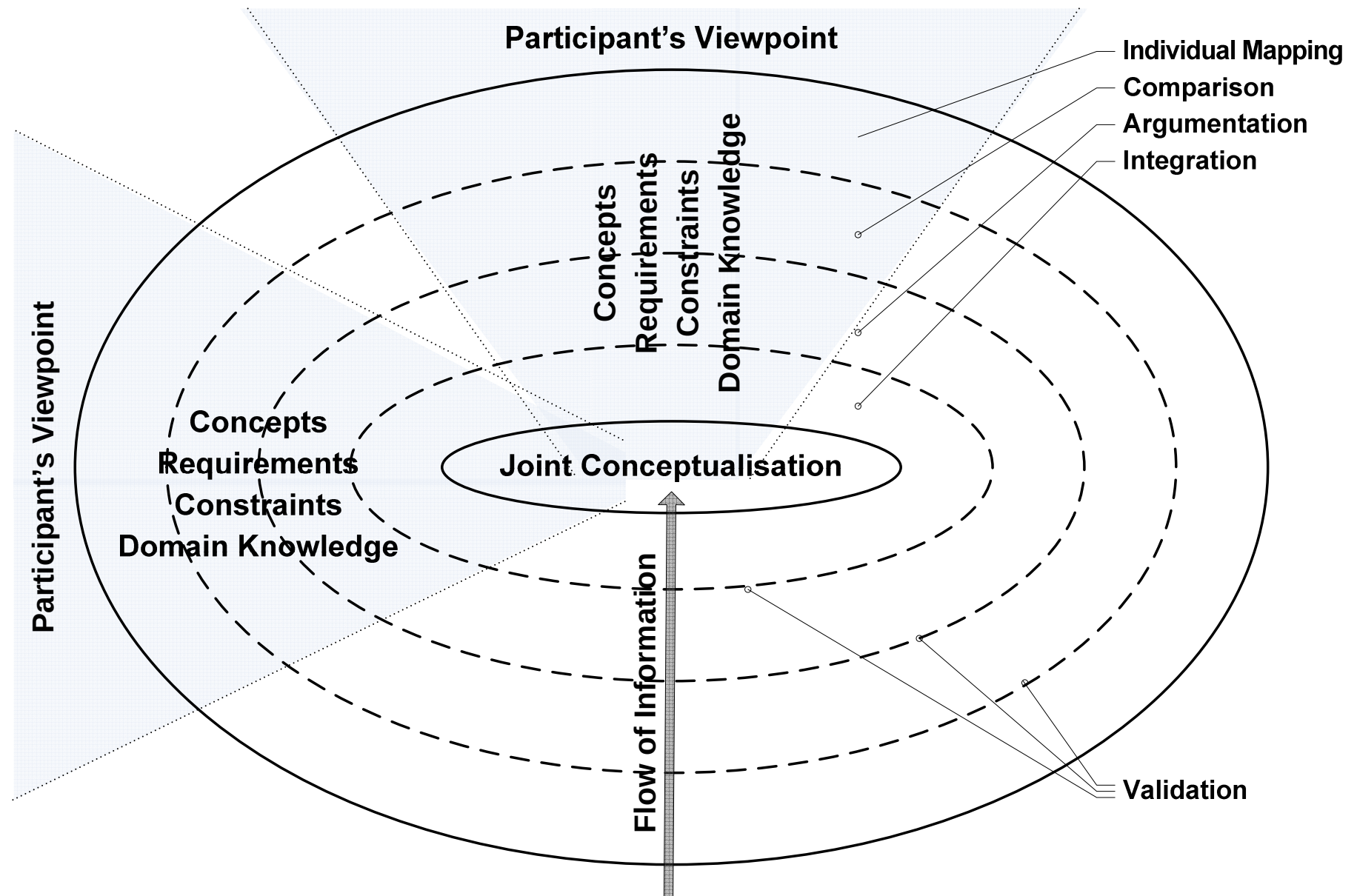


figure taken from [Kimber/Newcomb/Newcomb]

But how to make sure that changes get detected?

Process Guidance: Generic Process Model



Process Model, Roles: Technical Realisation

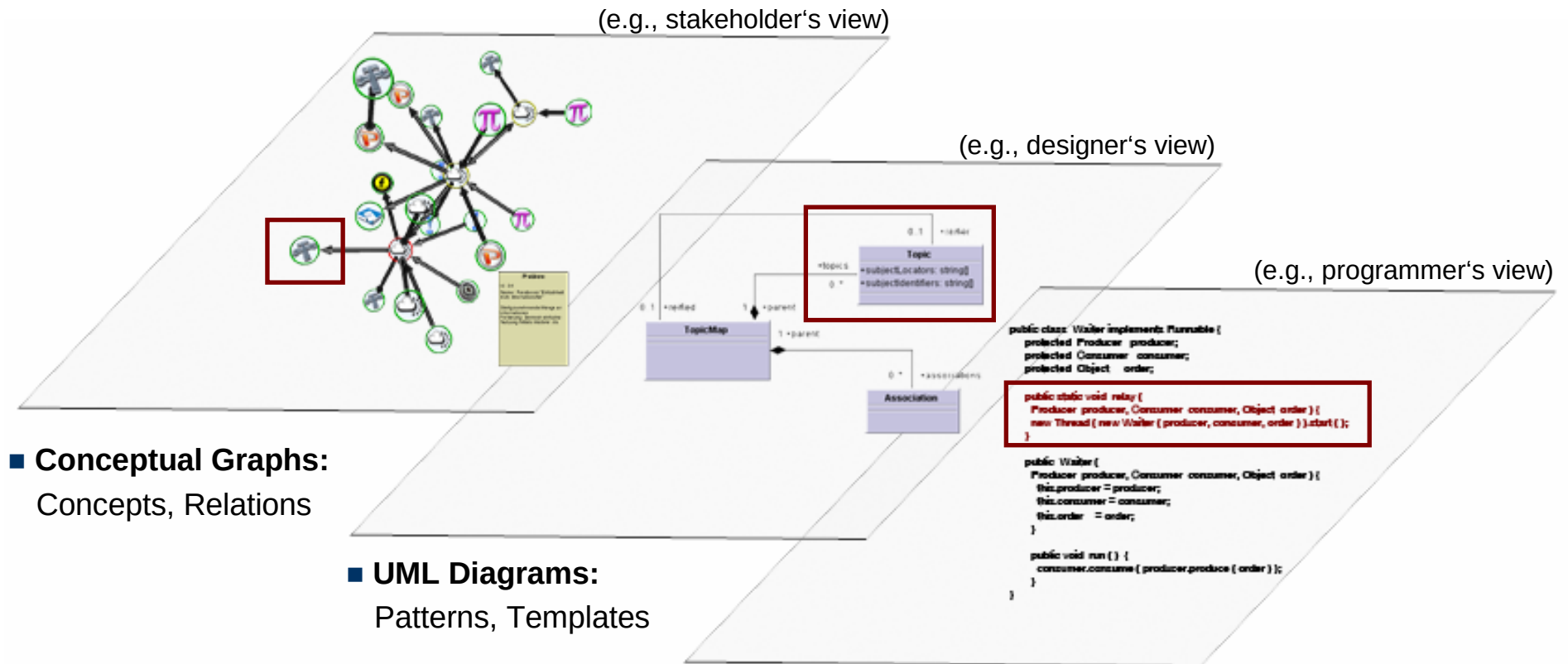
- Anything can and should be modelled using topic maps

```
[developer: role = "Developer" @...]    /* cf. stakeholder, programmer, tester, ... */
[individual-mapping: subprocess = "'Individual mapping' phase" @...]
...
[mue: person = "Markus Ueberall"]
[task-id1: task = "Task name"]
...
has-role(... :person, ...: role, ... :subprocess, ... :task)
...
```

- How to associate the needed (application-specific) business logic?

```
associated-business-logic(...: java-class, ...: subprocess)
associated-business-logic(...: java-class, ...: task)
associated-business-logic(...: java-class, ...: role)
...
[my-java-class: java-class = "check whether references changed" % "http://.../my-java.class"]
    /* better use a proxy resource here... */
```


Visualisation: Semantic Zooming, Filtering



■ **Conceptual Graphs:**
Concepts, Relations

■ **UML Diagrams:**
Patterns, Templates

■ **Java Code:**
Classes, Methods, Java-specific Annotations

Visualisation: Technical Realisation

■ How to connect / map concepts?

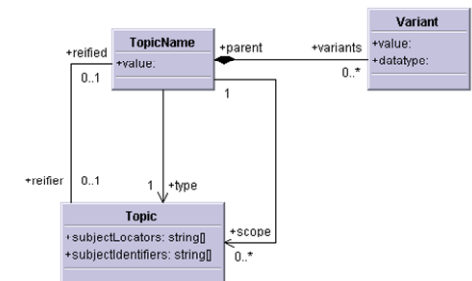
```

{view, description, "roughly corresponds to a layer in the preceding figure" }
{layouter, description, "determines how to arrange objects within a view" }
...
{template, description, "describes how to render a (set of) topic(s)" }
{editor, description, "how to edit (a set of) concept(s)" }
...
contained-in(...: concept, ...: template)
applicable-to(...: template, ...: view)
editable-by(...: template, ...: editor)
...
relationship(references: relation, ...: concept, ...: template)
...
associated-business-logic(...: java-class, ...)
...

```



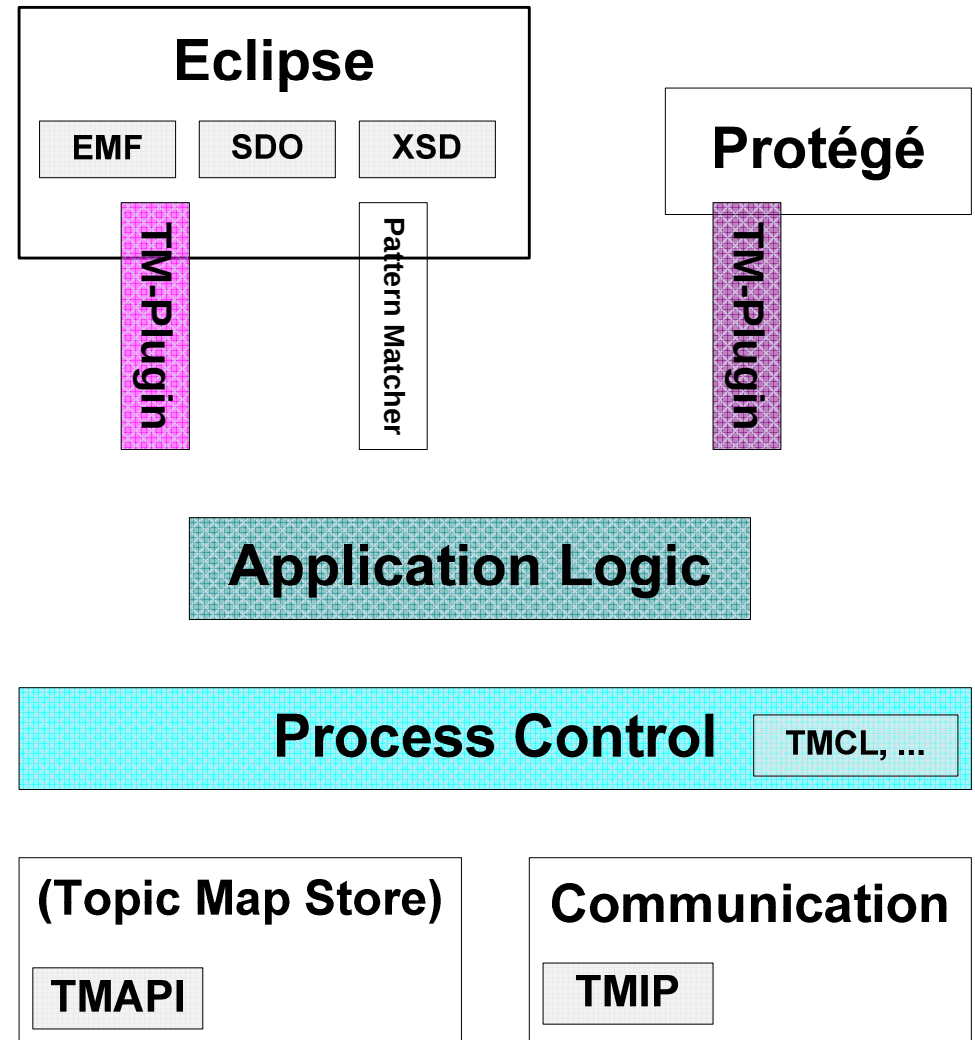
•
•
•



•
•
•

Tool Support

- Both generic process model and visualisation support have to be evaluated
- Objective / Requirement: Integration of existing tools for ontology maintenance and round-trip engineering support
- Currently: Designing / implementing an Eclipse 3.1 Plugin



Thank you!

e-mail to:

ueberall@tm.informatik.uni-frankfurt.de

Development Phases: Example

- Business Case: *Timekeeping System*
 - Employee hours are tracked by a time-clock system
- Requirements Specification
 - System must document the hours employees start/stop working and take breaks
 - Supervisors must be able to review the hours of subordinates
 - Timekeeping system must transmit employee hours to the payroll system
- Development of High-Level Essential Use Cases
 - Employee uses timekeeping terminal
- Object-Oriented Analysis
 - Conceptual modelling (think ER diagrams)
- Object-Oriented Design
 - Conceptual modelling results are translated into class diagrams
- Implementation
 - Ideally: identify, use and document well-known patterns
- Validation
 - Check whether initial requirements are met

Process Guidance: Example

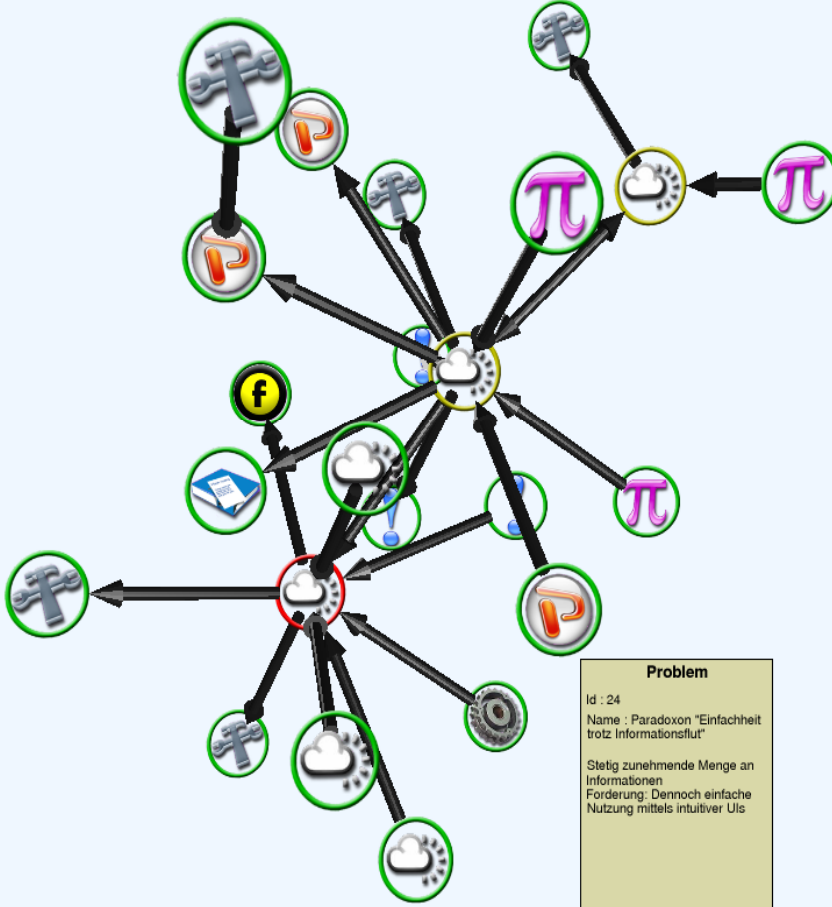
- Stakeholders come up with the forementioned business case, emphasising their own point of view
 - Using the base ontologies, activities can be both expressed and linked
 - Requirements and use-cases can be drafted
- Designers start with their object-oriented analysis and design
 - During each revision of the resulting conceptual models, available constraints (i.e., the forementioned requirements) must not be violated
- Reviewers check whether the mapping between the stakeholders' points of view and the designers' models is sound
 - Without this step, no further model revisions can be committed
- Programmers work with the resulting interfaces ("class/method signatures")
 - Any signature change will be reported and enforces another review step
- Testers check whether the initial functional and non-functional requirements are met
 - Again, deficiencies will be reported „upstream“

User Interface (old CLE prototype)

Lokale Sitzung Kollaborative Sitzung Bearbeiten Hinzufügen Wissensgraph Präsentation Text Hilfe

ubicom_joern

3D-Welt



Problem

Id : 24

Name : Paradoxon "Einfachheit trotz Informationsflut"

Stetig zunehmende Menge an Informationen

Forderung: Dennoch einfache Nutzung mittels intuitiver UIs

Wissenstext

Letzteres mag etwas absurd erscheinen oder nach Science-Fiction klingen - tatsächlich ist es aber nicht ganz einfach, sich auszumalen, was in einer Welt aus informatisierten und miteinander vernetzten Alltagsdingen unter Berücksichtigung ökonomischer und gesellschaftlicher Bedingungen möglich und akzeptabel ist und welche neuen Anwendungen und Dienste sowie Geschäftsfelder sich herausbilden könnten, wenn etwa Dinge sich genau lokalisieren können oder aus der Ferne identifizierbar sind oder wenn Gegenstände ein (in das Internet ausgelagertes) episodisches Gedächtnis besitzen, wodurch ihr sensorischer Input für andere abfragbar ist. Prinzipiell scheint dies jedenfalls bald machbar, genauso wie vielleicht das elektronisch beschreibbare "smart paper", welches manchen Computern dann ein radikal anderes Aussehen, etwa als zusammenfaltbare Straßenkarte, verleihen könnte.

In den Konsequenzen zu Ende gedacht, dürfte eine Welt aus "smarten" Dingen jedenfalls zu einer deutlich veränderten Wahrnehmung unserer Umgebung führen, größere gesellschaftliche und ökonomische Auswirkungen haben und damit letztendlich sogar von politischer Relevanz sein. Mit Sicherheit ist dabei die Privatsphäre im Sinne von Datenschutz und "Privacy" betroffen; die weitergehenden Folgen in kultureller und wirtschaftlicher Hinsicht erscheinen derzeit allerdings noch relativ unklar.

Verswindende Technologie

Die hier angedeutete langfristige Vision wurde von dem 1999 früh verstorbenen Mark Weiser, seinerzeit leitender Wissenschaftler am Forschungszentrum von XEROX in Palo Alto, propagiert, der dafür bereits vor mehr als 10 Jahren den Begriff "Ubiquitous Computing" prägte [5]. Weiser sieht Technik als reines Mittel zum Zweck an, die in den Hintergrund treten sollte, um eine Konzentration auf die Sache an sich zu ermöglichen - der PC als Universalwerkzeug sei dafür der falsche Ansatz, da dieser aufgrund seiner Komplexität die Aufmerksamkeit zu sehr in Anspruch nehme. Er schreibt u.a.: "As technology becomes more imbedded and invisible, it calmes our lifes by removing the annoyances... The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it."

Ob das scheinbar Paradoxe gelingt, nämlich trotz zunehmender Menge und Allgegenwart von Information diese dann - etwa mittels intuitiver Schnittstellen und impliziter Informationsverarbeitung - auch einfacher zu nutzen, bleibt abzuwarten. Die von Weiser avisierte "verschwindende Technologie" hat jedenfalls der neuen "Disappearing-Computer"-Forschungsinitiative der EU [6] nicht nur den Namen verliehen, sondern auch ganz wesentlich deren Forschungsprogramm und Ziele beeinflusst. Während allerdings Weiser den Begriff "Ubiquitous Computing" eher in akademisch-idealistischer Weise als eine unaufdringliche, humanzentrierte Technikvision verstand, die sich erst in der weiteren Zukunft realisieren lässt, hat die Industrie dafür später den Begriff "Pervasive Computing" mit einer leicht unterschiedlichen Akzentuierung geprägt: Auch hier geht es um die überall eindringende und allgegenwärtige

Bereit

- 24 - Paradoxon: Einfachheit trotz Inf...