Versioning of Topic Map Templates and Scalability

M. Ueberall, O. Drobnik

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

2007 / 10 / 12

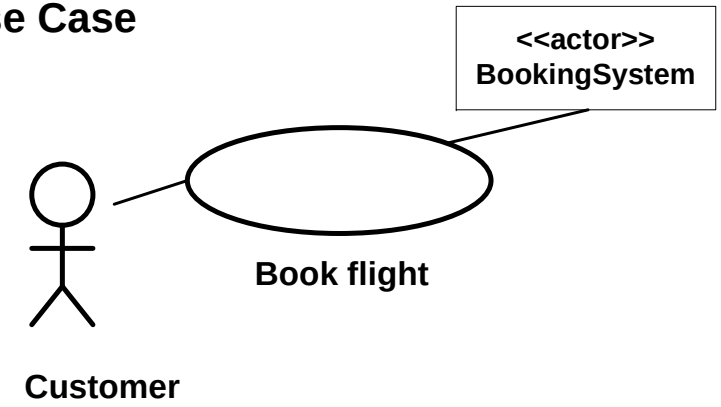
Motivation and Objective

- Problems in the context of software development processes:
  - participants use concepts from different knowledge domains → information overflow
  - insufficient communications → traceability deficits
- Our approach is based on
  - *Templates* consisting of Topic Map Objects → lightweight representation
  - version management → traceability
  - meta process model → communications support
  - role-based filtering → scalability

(Nested) Templates

Example

Use Case



Name:	**Customer**
Type:	**Person**
Description:	**...**

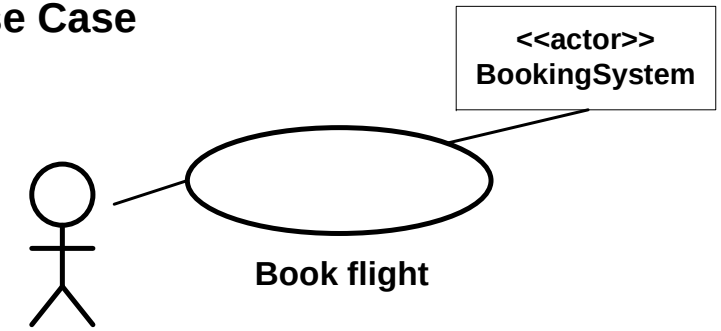
Semi-formal Representation
(Template containing Topic Map Objects)

Name	**...**
UseCase	**FlightBooking**
Actor	**...**
Precondition	**...**
Action	**...**
Postcondition	**...**

(Nested) Templates

Nesting Example

Use Case



Name:	**Customer**
Type:	**Person**
Description:	**...**

Semi-formal Representation

(Template containing Topic Map Objects)

Name	**...**
UseCase	**FlightBooking**
Actor	**Name: Customer** **Type: Person** **Description: ...**
Precondition	**...**
Action	**...**
Postcondition	**...**

Usage Restrictions for Templates

Constraints in LTM notation

```
[person : topic-type = "Person"]
[age : occurrence-type = "Age" = "Person's Age" /person ~toc1]

[toc1 : topic-occurrence-constraint =
    "Topic Occurrence Constraint Label for Occurrence 'Age'"]
{toc1, max-cardinality, [[1]]}
```

- embedded constraints as opposed to linked/embedded schemas allow for unified handling of versioning

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Faceted Classification

Browser Example

COCOONDEV Daisy 2.1-dev

cocoondev.org Home | Main Home

Search

User: guest Tools Recent Changes

Faceted Browser

1210 document(s) found.

☐ Limit to site collection (main) ☐ Limit to site variant (branch main, language en)

Order by: name - asc

Open this query in the query search page

by Document Type

Attachment	16
BookDefinition	5
CodeSample	6
DaisyCommWikiDocument	60
FAQ	23
FAQCategory	21
GlossaryEntry	6
Image	232
KnowledgeBaseArticle	13
KnowledgeBaseCategory	9

(3 more)

by Collections

daisy	24
daisydocs	955
daisykb	27
daisywiki	99
handbook	16
main	21
xreporter	64

by Branch

daisydocs-1_2	51

1-10 of 1210 Next >

.NET/C# implementation of the remote Java Daisy API
Document Type: SimpleDocument
Collections: daisywiki
Branch: main
Language: en
Last Modifier Login: cyberchand
Daisy is a comprehensive content management application framework, consisting of basically : a standalone repository server accessible through either a local Java API, an HTTP/XML API, or a high-level remote Java API (based also on an HTTP/XML API), and an extensive browsing and editing DaisyWi...

1.0.0 to 1.1.0 upgrade
Document Type: SimpleDocument
Collections: daisydocs
Branch: daisydocs-1_2
Language: en
Last Modifier Login: bruno
Changes (compared to Daisy 1.0.0) Features Introduced separate ACL permission for changing the version state (publish/draft). Added permanent deletion of documents. Added deletion of part, field and document types (only possible when no longer in use). Notification mails: instead of receiving m...

1.0.0 to 1.1.0 upgrade
Document Type: SimpleDocument
Collections: daisydocs
Branch: daisydocs-1_3
Language: en
Last Modifier Login: bruno
Changes (compared to Daisy 1.0.0) Features Introduced separate ACL permission for changing the version state (publish/draft). Added permanent deletion of documents. Added deletion of part, field and document types (only possible when no longer in use). Notification mails: instead of receiving m...

Faceted classification

LTM notation (1)

```
// cf. [Ahmed2003] (Proc. Extreme Markup Languages)
#PREFIX tmhd @"http://www.techquila.com/psi/hierarchy/#"
#PREFIX tmtd @"http://www.techquila.com/psi/thesaurus/#"
#PREFIX tmfd @"http://www.techquila.com/psi/faceted-classification/#"

[all-templates : tmfd:facet = "Set of all Templates"]
[template-facet : tmfd:facet = "Template Facet"]

tmfd:facet-has-hierarchy-type(template-facet : tmfd:facet,
    tmfd:subcategory-supercategory : tmfd:facet-hierarchy-type)
tmfd:facet-has-root(template-facet : tmfd:facet,
    all-templates : tmfd:facet-root)
```

Faceted classification

LTM notation (2)

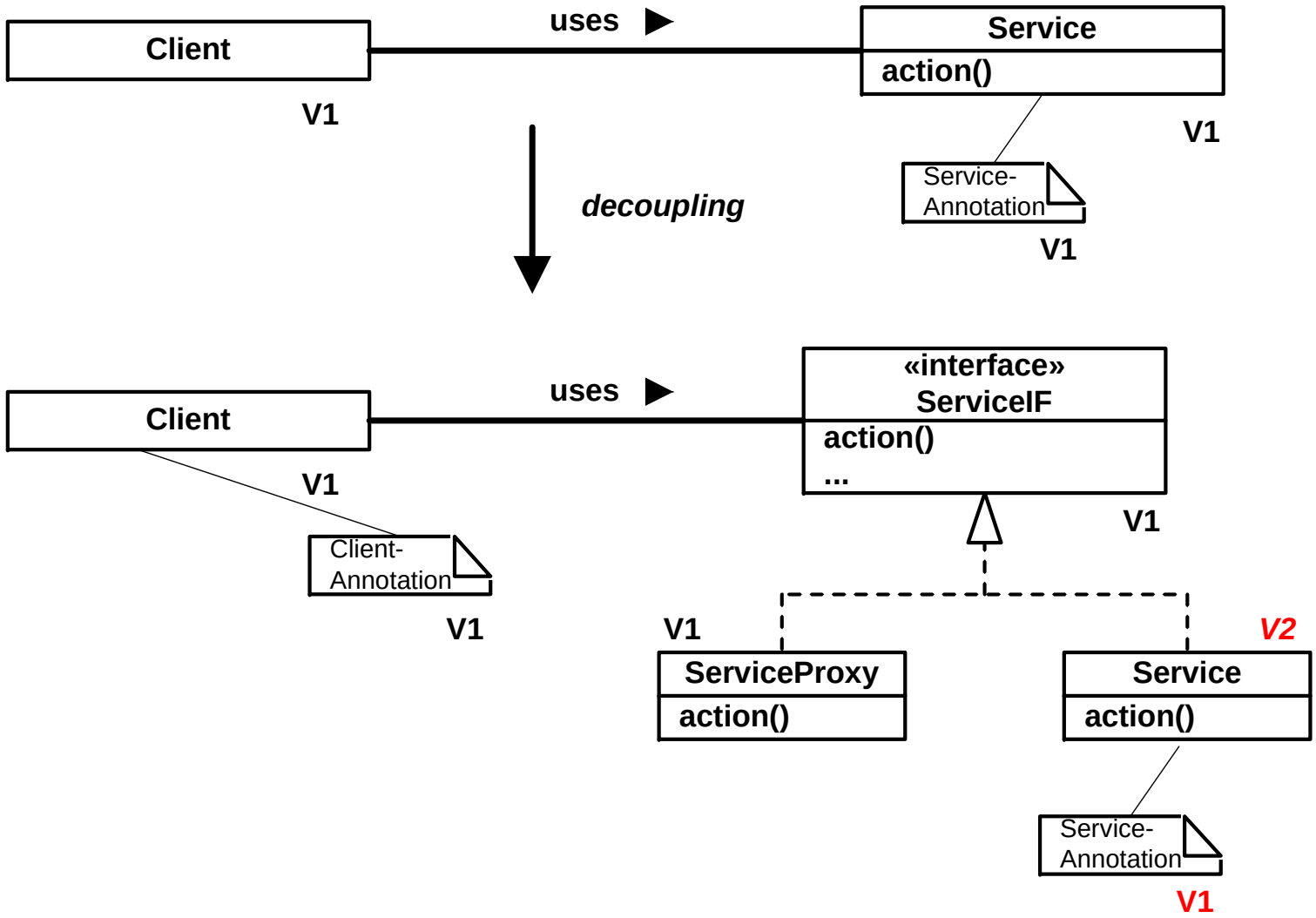
```
// taken from the UML Superstructure specification, cf. [UD06]
[actor : template-class = "Actor"]

tmfd:subcategory-supercategory(actor : tmhd:subcategory,
    all-templates : tmhd:supercategory)
tmfd:subcategory-supercategory(person : tmhd:subcategory,
    actor : tmhd:supercategory)
tmfd:subcategory-supercategory(legal-entity :
    tmhd:subcategory, actor : tmhd:supercategory)

tmtd:part-whole(actor : tmtd:whole, role : tmtd:part)
tmtd:part-whole(person : tmtd:whole, age : tmtd:part)
```

Versioning

Example: Proxy Design Pattern

Versioning

Example: Proxy Design Pattern



Versioning

Granularity of Objects

Name	...
UseCase	FlightBooking
Description	Statement A Statement B Statement C
Precondition	...
Action	...
Postcondition	...

Name	...
UseCase	FlightBooking
Description	
Description1	Statement A
Description2	Statement B
Description3	Statement C
Precondition	...
Action	...
Postcondition	...

- rule-of-thumb: any statement which is explicitly referenced has to be reified
- ordering accomplishable by means of “sortkey” concept, cf. [Grønmo93]

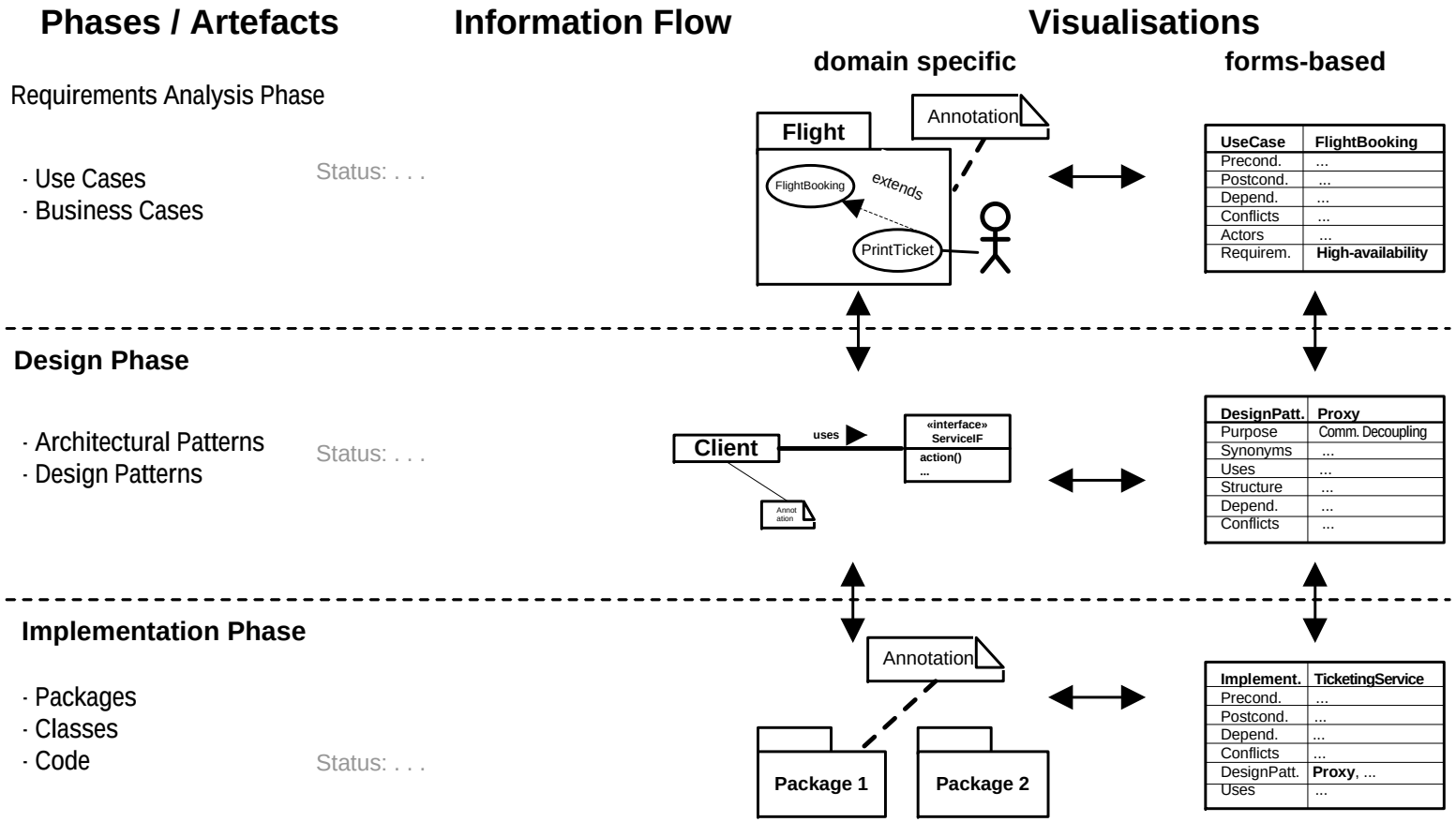
Versioning

Metadata for Units of Information, LTM notation

```
// <occurrence | association> ~object-ref

// dc: cf. http://dublincore.org/documents/dces (Dublin Core)
{object-ref, dc:creation-date, [[2007-06-04T2359:59+01:00]]}
{object-ref, dc:version, [[version-id]]}
{object-ref, dc:author, [[user-id]]}
// skos: cf. http://w3.org/2004/02/skos (SKOS)
{object-ref, skos:changeNote, [[description]]}
...

is-replaced-by(object-ref_1 :old-obj, object-ref_2 :new-obj)
is-deprecated(object-ref_3 :obj)
is-deleted(object-ref_4 :obj)
```

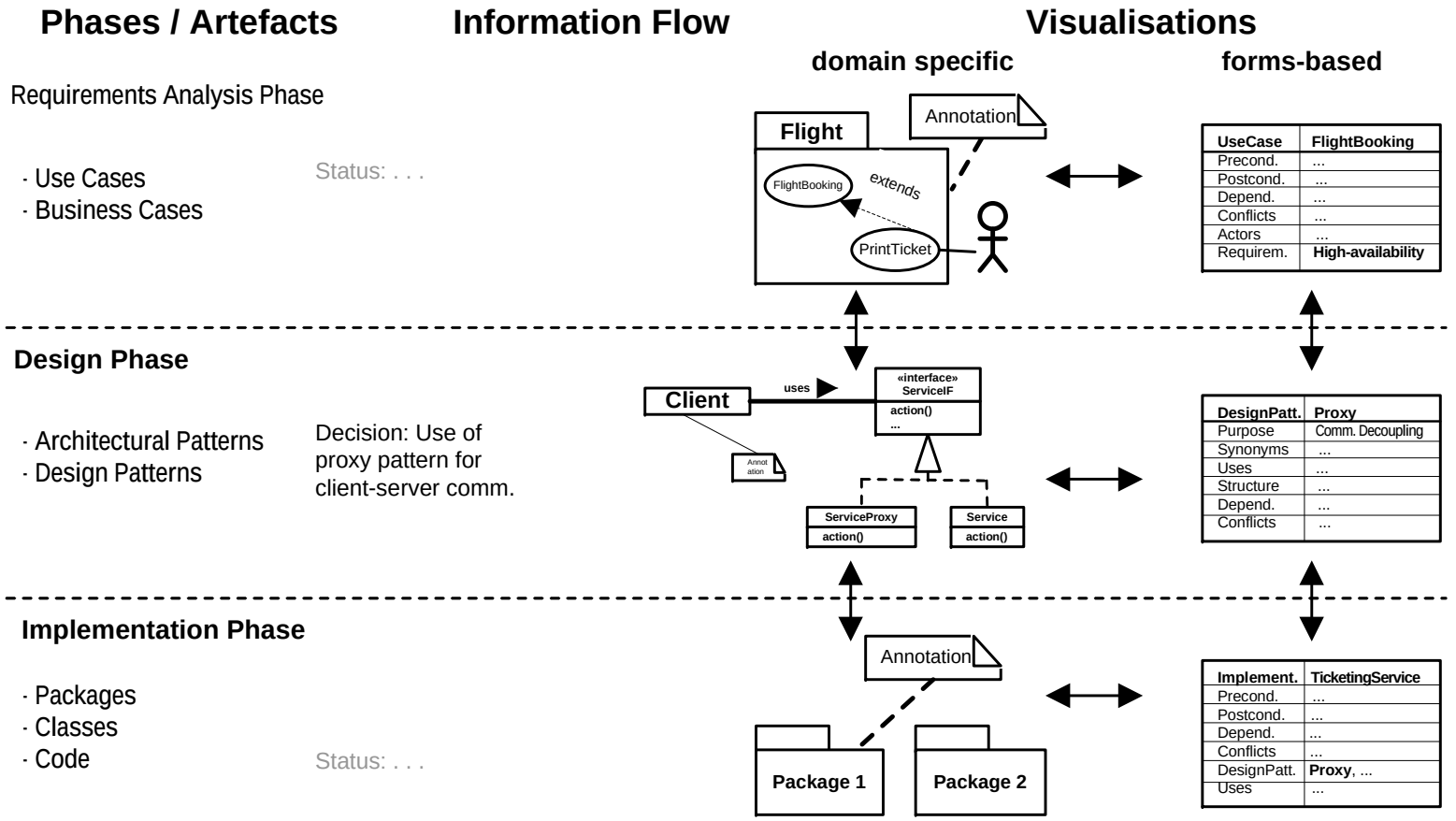
Versioning and Scalability

Software Development Scenario



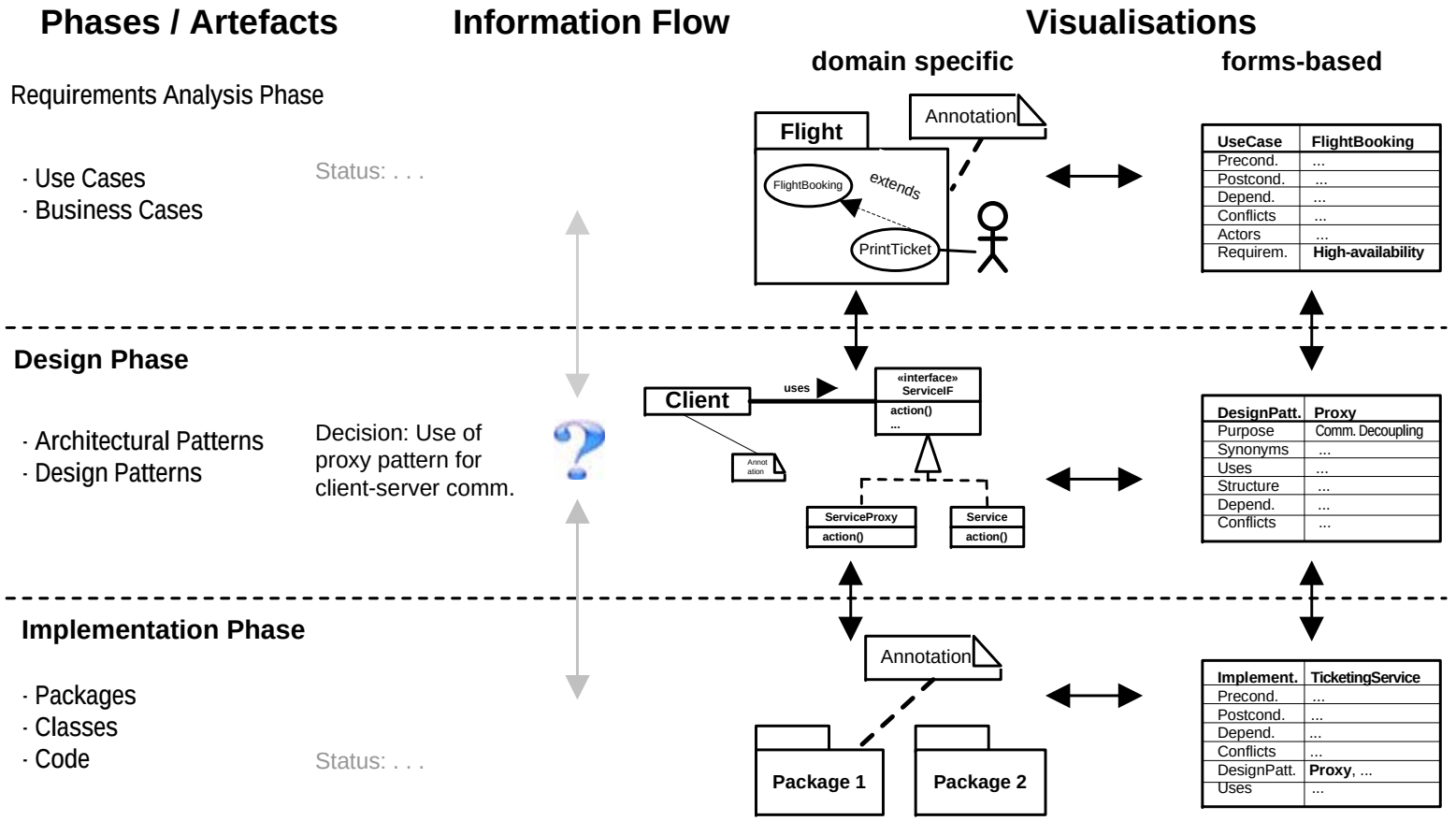
Versioning and Scalability

Software Development Scenario



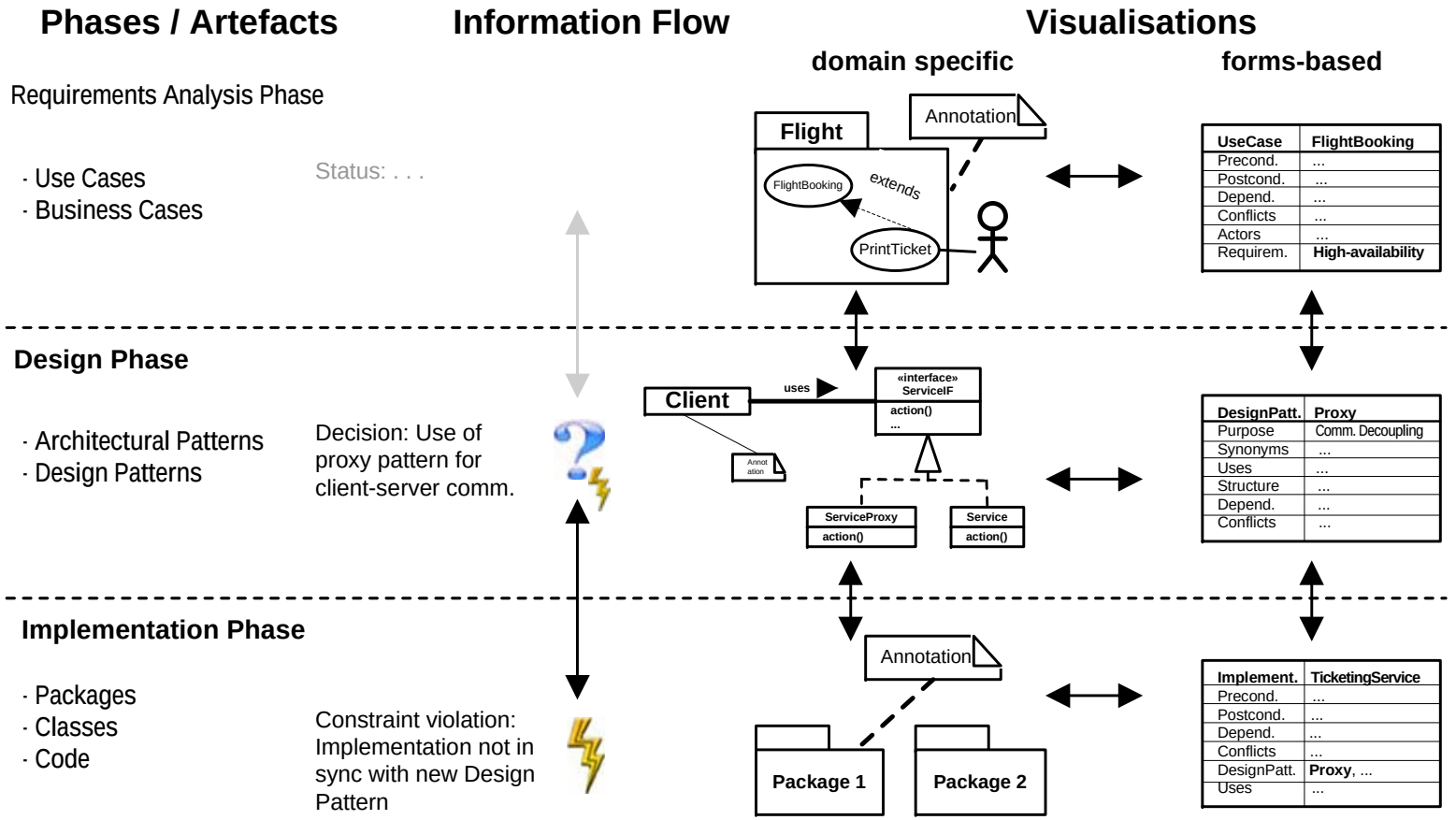
Versioning and Scalability

Software Development Scenario



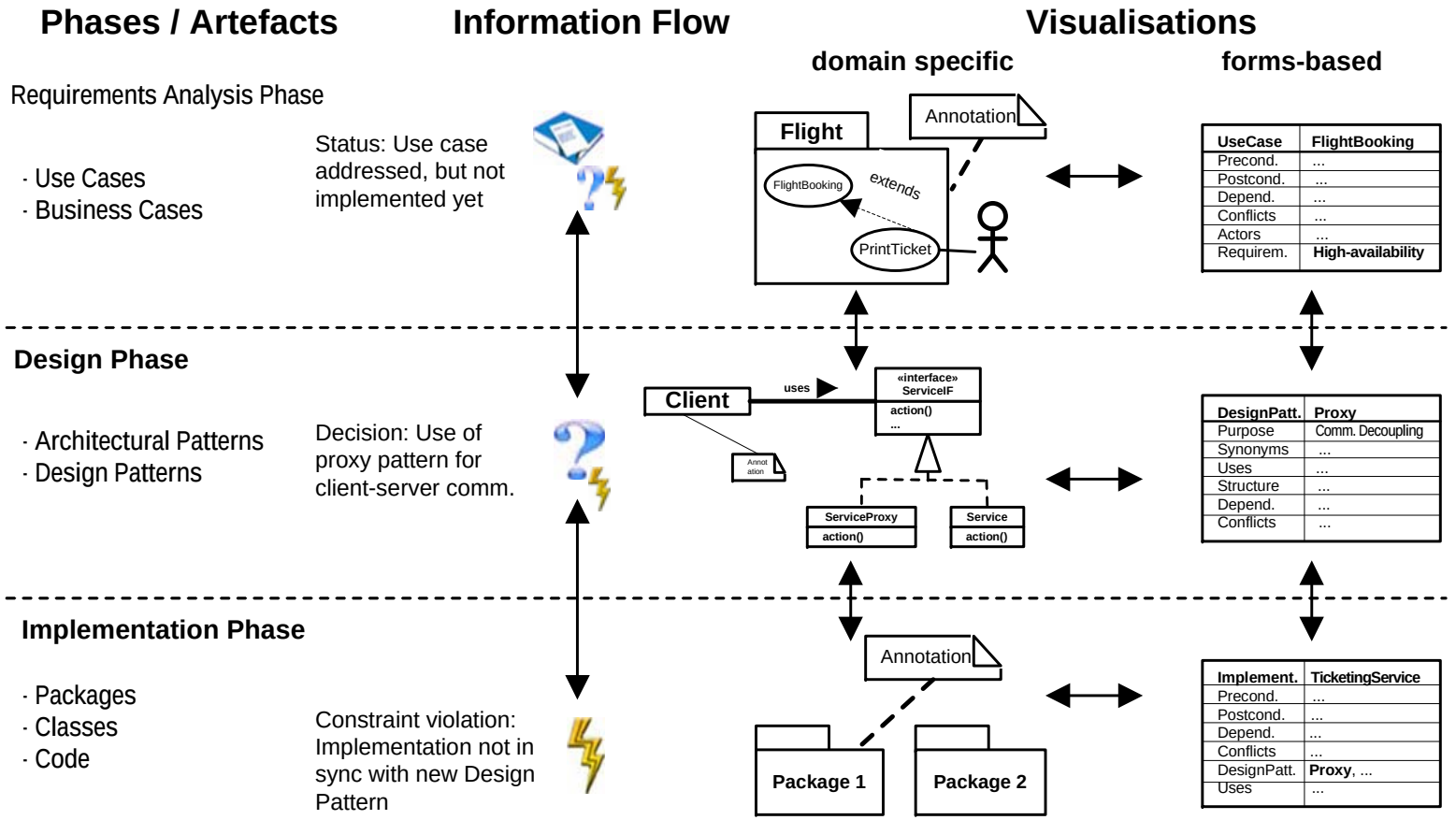
Versioning and Scalability

Software Development Scenario



Versioning and Scalability

Software Development Scenario



Versioning and Scalability

- The key to Scalability is always some kind of "divide-and-conquer" method
- In our case → exploitation of combined known advantages of both Topic Map based representations and hierarchical structuring:
  - *Querying and Filtering:* each participant is presented a view which contains concepts from his domain of knowledge
  - *Interlinked Representations:* all changes are propagated (and thereby "mapped" according to pre-defined/user-supplied constraints) *and require reconcilliation*
  - *Communication Complexity:* i.e., modifications can be exchanged by means of Topic Map Fragments → Meta Process Model

Meta Process Model

- Simple approach consists of two alternating subphases:
  - *summarisation:* annotation of own modifications
  - *exploration:* finding of others' modifications
- Conflicts can be resolved using, e.g.,
  - computer-driven matchings of concepts
  - computer-aided reconcillation

Prototype: Ongoing Work

- modularised querying/filtering support for preliminary prototype
  - evaluation with regard to more complex development scenarios
  - combination with other Eclipse plugins, e.g., pattern scanners
- medium-term objective: support for coping with ontology changes and/or mergers within meta process model subphases
- long-term objective: support of operations as needed for decentralised version management (n-way merge)

Thank you!

e-mail to:
ueberall@tm.informatik.uni-frankfurt.de

(Nested) Templates

Definition

- A *Template* consists of any combination of related individual Topic Map Objects
  - used for instantiation of concepts as identifyable objects
  - important component: human-readalbe description of the underlying semantics
- *Nested Templates* are Templates which include references to other Templates
  - derivation of new definitions through extension and/or combination of existing (base) concepts (analogous to the underlying principle for DITA (Darwin Information Typing Architecture), though the latter does not support, e.g., typed associations

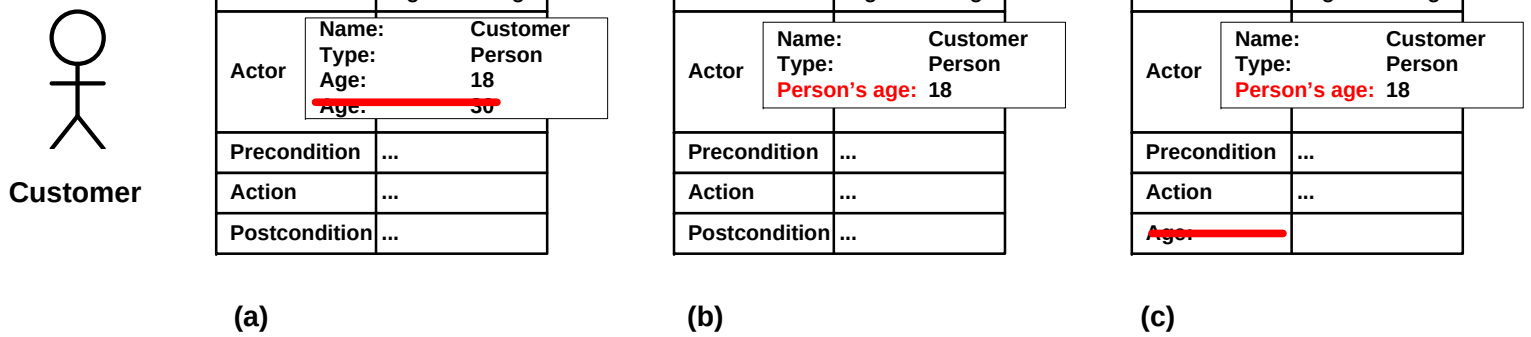
Constraints

Definition

- constraints are particularly *usage restrictions*
- basic types of constraints include:
  - *cardinality constraints:* define the allowed cardinalities of instances of a Topic Map object (e.g., a certain role within an association)
  - *type/role constraints:* ensure that Topic Map objects are used according to their initial semantic definition (cf. NCPL `occurrence-type`, `topic-type`)
  - *scope:* can be seen as extension of the forementioned item, if applicable using the `scope` operator (e.g., context-sensitive description of roles: `person's age`, `age`)

Usage Restrictions for Templates

Constraints Example



(a) cardinality constraint for "Age"

(b) context-sensitive labeling

(c) occurrence-type cannot be used as template attribute

Hierarchical Structuring

- Hierarchical structuring facilitates the underlying application logics, which are needed for nested template support (e.g., for dealing with context-sensitive handling of inputs)
- the TMDM (Topic Map Data Model) specification only lists basic relations like `supertype-subtype` which are not flexible enough (for our purposes)
- in particular, the *faceted classification system* comes in handy

Hierarchical Structuring

Faceted classification: Definition

- A *faceted classification system* allows the assignment of multiple classifications to an object
- this enables the classifications to be ordered in multiple ways, rather than in a single, pre-determined, taxonomic order
- The most prominent use of faceted classification is in faceted navigation systems that enable a user to navigate information hierarchically, going from a category to its sub-categories, but choosing the order in which the categories are presented

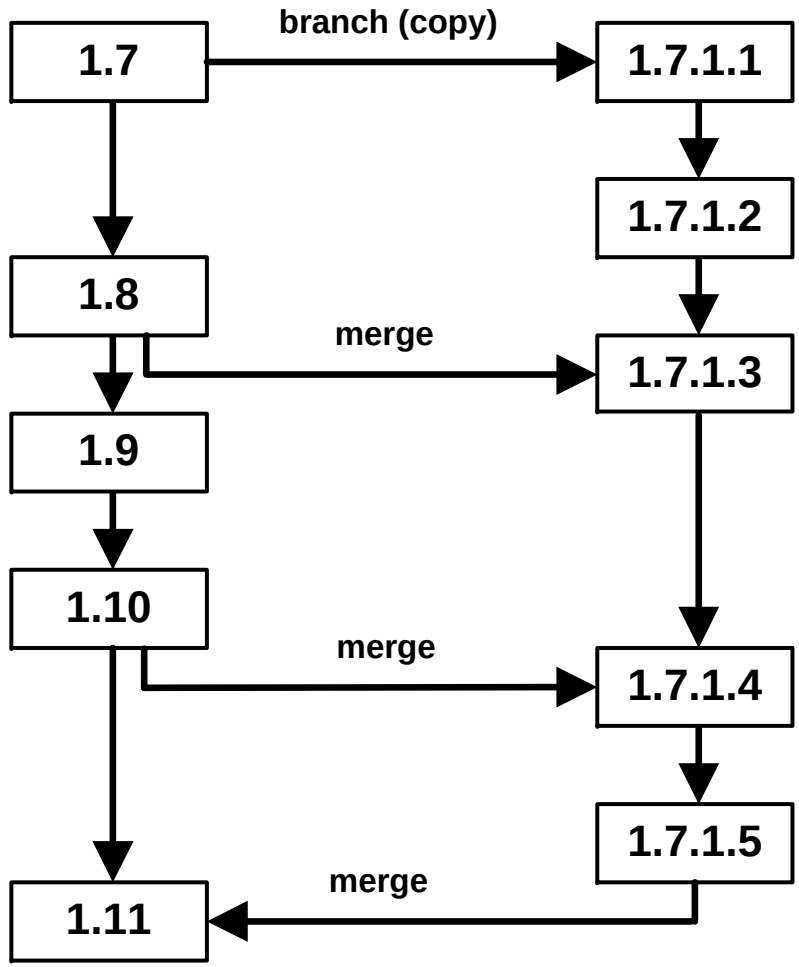
Versioning

Definition

- Versioning is conceived as management of multiple revisions of the “same” unit of information
  - in our context: unit of information = template/instance
- Minimum requirement: unique version id (not necessarily human-readable) and ancestor information
  - in practice (collaborative environments), additional metadata (author, date, comment, ...) is *essential*

Versioning

Importance of the Directed Acyclic Graph structure



(cf. `http://www.kerneltraffic.org/kernel-traffic/kt20030323_210.txt`)