On Topic Map Templates and Traceability

O. Drobnik, M. Ueberall

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

2006 / 10 / 11

Motivation and Objective

Problems in Software Development:

- participants (e.g., stakeholders, designers, programmers) with different objectives and domain knowledge (consistent conceptualisations)
- any loss of information to be considered fatal (traceability)
- semantics-preserving phase transitions

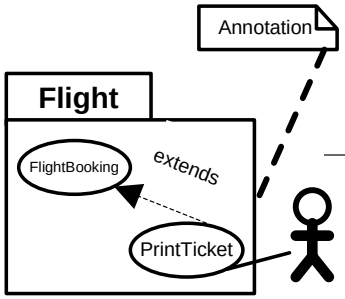
Own approach based on:

- use of lightweight representations (Topic Maps!)
- development process guidance by means of Templates

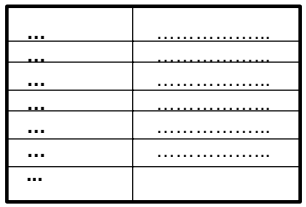
Approach: Overview

Requirements Engineering Phase

Use Cases



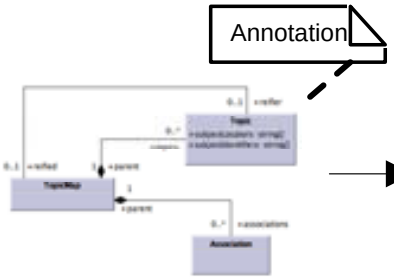
(forms-based template)

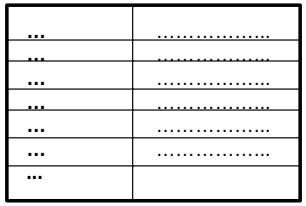


Stake-holders

Design Phase

Models Design Patterns

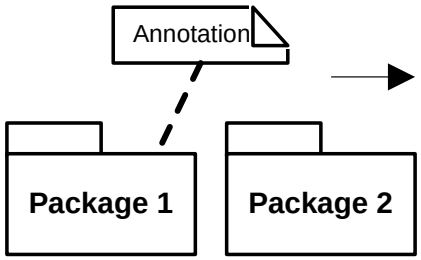


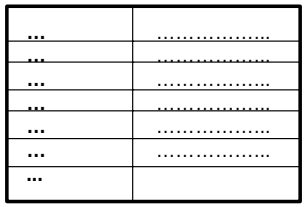


Designers

Implementation Phase

Packages Classes





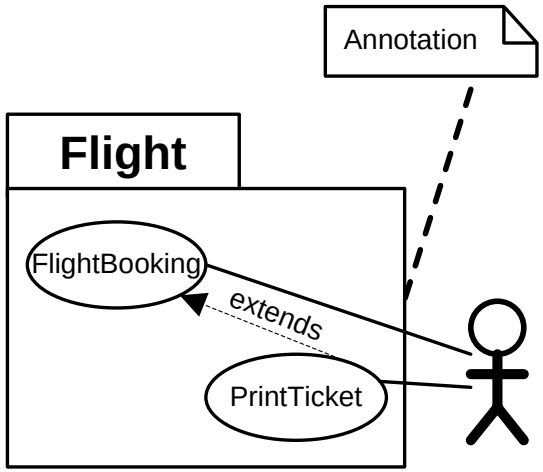
Program-mers

Use Case Specification

Concepts from the UML 2.0 Superstructure are, e.g.:

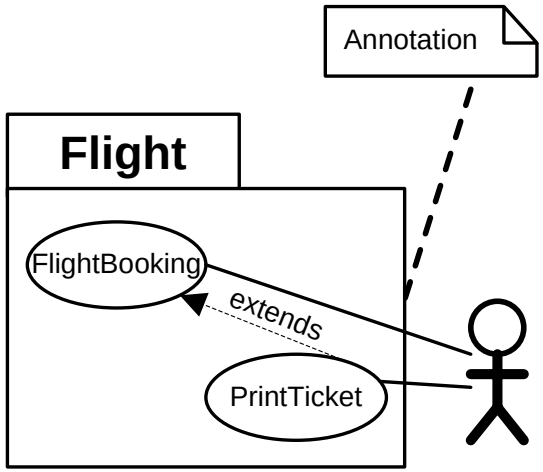
- *use case:* specification of a set of actions performed by a system, yielding an observable result of value for one or more actors
- *actor:* role played by a user or any other system that interacts with the subject
- *extend:* relationship specifying that the behaviour of a use case may be extended by that of another
- *include:* relationship defining that a use case contains the behaviour in another one

Use Case Example

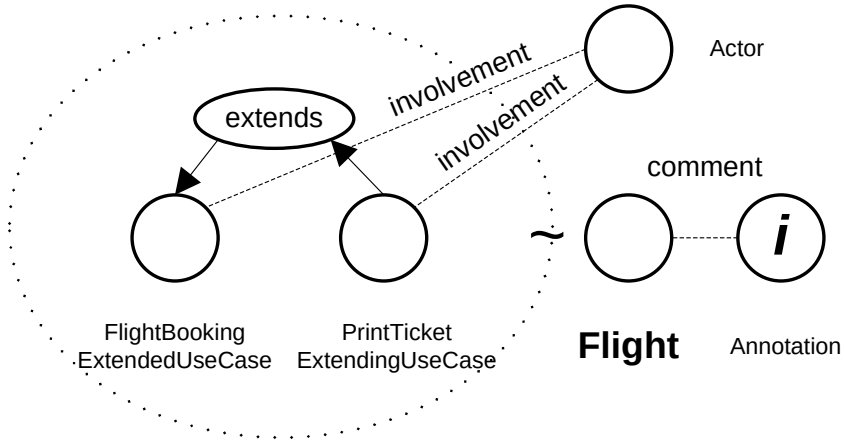


(a) UML notation

Use Case Example



(a) UML notation



(b) TM representation

Templates: Definitions

- A *Topic Map Template* is a Topic Map "that only consists of topics that are declared in order to be used as types in a class of topic maps" [ISO working group]
- *Association Templates* are models that introduce constraints [Biezunski/Newcomb]
- A single *Template* is "the formal declaration of a required *Pattern* for a given association type", a *Pattern* is "the structure of an individual association" [Vatant]

Templates: Generic Basic Types

Question

Information

Theory

Problem

Topic Types

Comment

Creator

Version

Occurrence Types

supports

contradicts

leads-to

connected-with

Association Types

Flight Booking Example

LTM notation

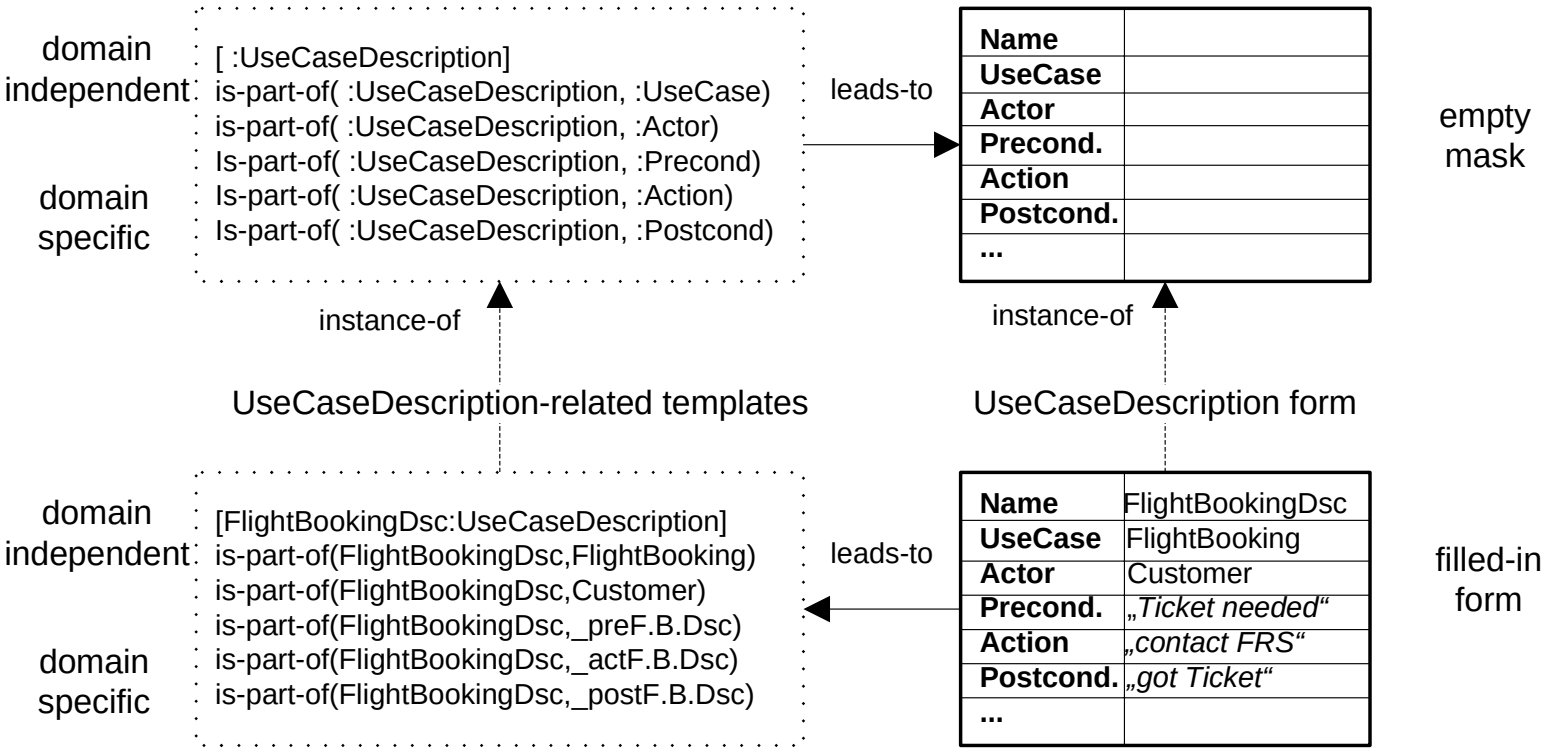
```
// topic types
[Actor = "Actor" @"http://.../UML_Superstructure/UseCases/#Actor"]
[UseCase = "UseCase" @"http://.../UML_Superstructure/UseCases/#UseCase"]

// role definitions
[placeholder = "placeholder" @"http://.../Templates/#Placeholder"]
[ExtendingUseCase : UseCase = "ExtendingUseCase" @"http://..."]
[ExtendedUseCase : UseCase = "ExtendedUseCase" @"http://..."]
[IncludingUseCase : UseCase = "IncludingUseCase" @"http://..."]
[IncludedUseCase : UseCase = "IncludedUseCase" @"http://..."]

// association types
[extends = "extends" @"http://.../UML_Superstructure/UseCases/#Extend"]
[includes = "includes" @"http://.../UML_Superstructure/UseCases/#Include"]

// "examples" of associations
extends(placeholder : ExtendedUseCase, placeholder : ExtendingUseCase)
includes(placeholder : IncludingUseCase, placeholder : IncludedUseCase)
```

Nested Templates Example



Flight Booking Example (I)

LTM notation

```
// initial use case with actor
[FlightBooking : UseCase = "FlightBooking" @"http://.../#FlightBooking"]
[Customer = "Customer" @"http://.../#Customer"]
tmdm:supertype-subtype(Actor, Customer)

// a complete use case description links use cases and actors
[UseCaseDescription]
exmpltm:is-part-of(placeholder:UseCaseDescription, placeholder:UseCase)
exmpltm:is-part-of(placeholder:UseCaseDescription, placeholder:Actor)

// a customer is an individual person with certain attributes
[Person = "Person" @"http://.../#Person"]
[Address = "Address" @"http://.../#Person_Address"]
[Age = "Age" @"http://.../#Person_Age"]
[BankingAccount = "BankingAccount" @"http://.../#Person_BankingAccount"]
exmpltm:has-attribute(placeholder : Person, placeholder : Address)
exmpltm:has-attribute(placeholder : Person, placeholder : Age)
exmpltm:has-attribute(placeholder : Person, placeholder : BankingAccount)
tmdm:supertype-subtype(Customer, Person)
```

Flight Booking Example (II)

LTM notation

```
// use case refinement
[RC : UseCase = "RegisterCustomer" @"http://..."]
[DT : UseCase = "DeliverTicket" @"http://..."]
...

includes(FlightBooking : IncludingUseCase, RC : IncludedUseCase) ~ INCL
extends(FlightBooking : ExtendedUseCase, DT : ExtendingUseCase) ~ EXTD
```

Traceability: Definitions

- In the general sense, *traceability* denotes the ability to chronologically interrelate the uniquely identifiable entities to ensure the completeness of information about every step within a process chain
- In software development, the term refers to the ability to link the requirements set forth at the beginning of a project to the corresponding design artefacts, the resulting software, and associated test cases

Traceability and Requirements Engineering

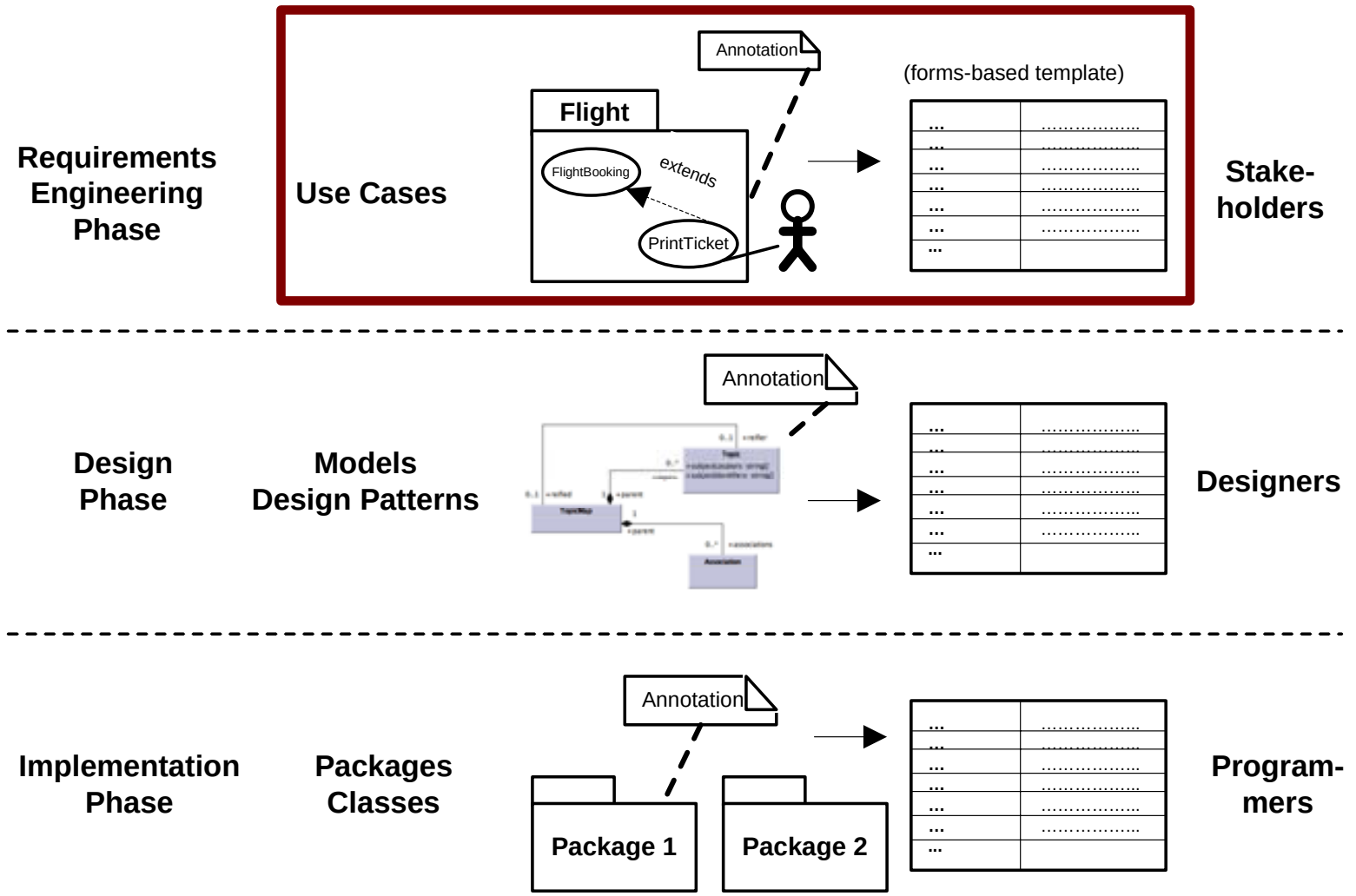
Individual development steps must be comprehensible

- within the same phase of development (intra-phase)
- between (consecutive) phases of development (inter-phase)

First, prevent incorrect use of concepts (cf. “extend” example earlier on)

- Templates have to be augmented by usage restrictions (using constraints)
- Validation rules can be expressed in terms of (boolean) predicates or queries

Intra-Phase Consistency



Intra-Phase Consistency: Using Constraints

OSL notation

```
[Constraint = "Constraint" @"http://.../Restrictions/#Constraint"]
{extends, Constraint, [[<association>
  <instanceOf><internalTopicRef href="#extends"/></instanceOf>
  <role min="1" max="1">
    <instanceOf><internalTopicRef href="#ExtendingUseCase"/>
      </instanceOf>
    <player><internalTopicRef href="#UseCase"/></player>
  </role>
  <role min="1" max="1">
    <instanceOf><internalTopicRef href="#ExtendedUseCase"/>
      </instanceOf>
    <player><internalTopicRef href="#UseCase"/></player></role>
  </association>]]}
```

Intra-Phase Consistency: Using Predicates

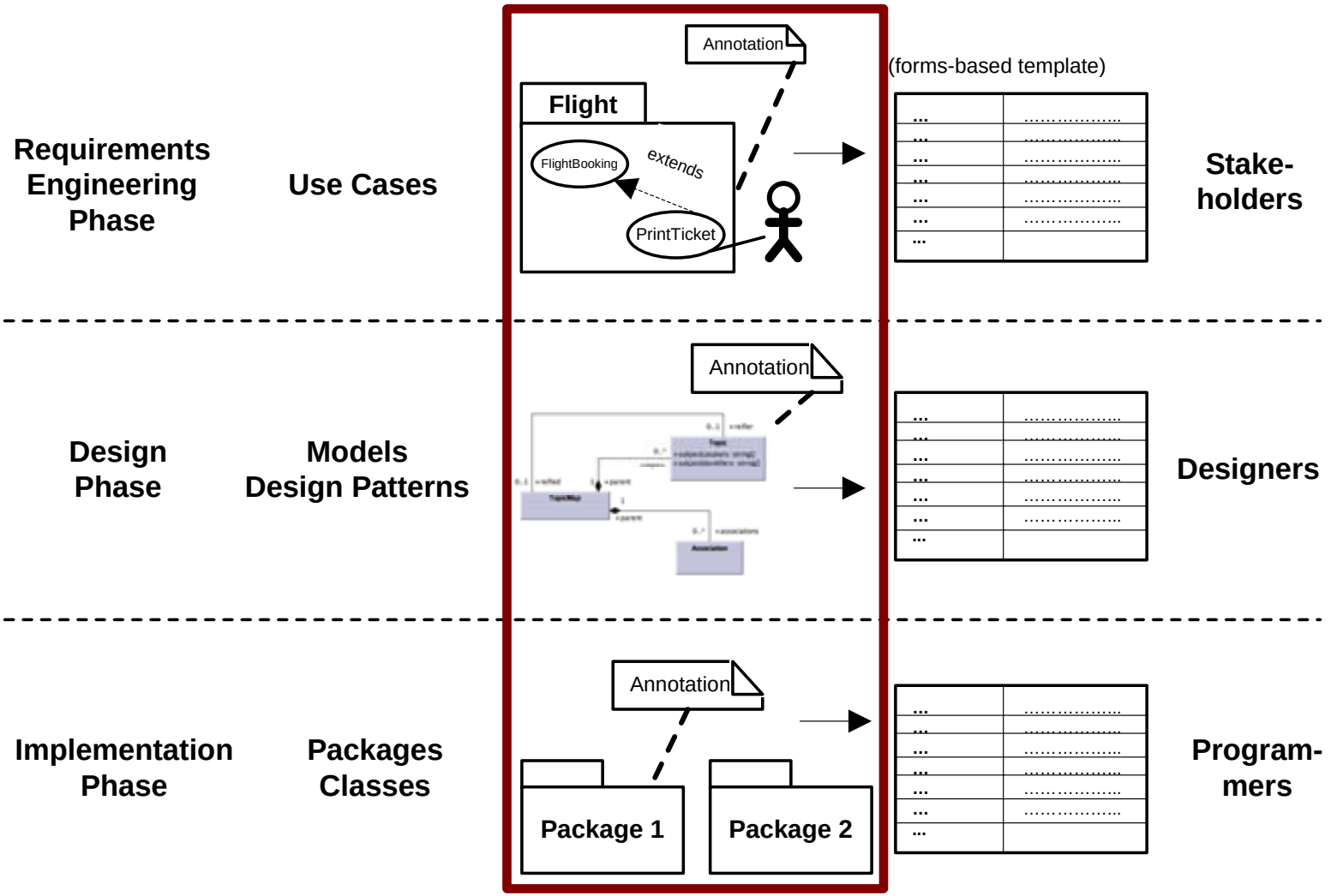
TMQL/AsTMA= notation

```
[Predicate = "Predicate" @"http://..."]
{UseCase, Predicate, [[is-valid-indirect isa tmql_predicate where: """
  is-valid (UseCase: $usecase) &
  not (extends(ExtendedUseCase: $usecase',
               ExtendingUseCase: $usecase) &
  not(is-valid-indirect(UseCase: $usecase'))) &
  not (includes(IncludedUseCase: $usecase'',
                IncludingUseCase: $usecase) &
  not(is-valid-indirect(UseCase: $usecase''))) """
]]}
```

Consistency-preserving Phase Transitions

- Phase Transitions require a consistent mapping (e.g., requirements have to be mapped to design artefacts)
- This kind of mapping can be supported by additional regulations concerning Templates from both phases (e.g., any use case instance has to be associated with at least one design pattern instance)
- Violations of these regulations have to be handled differently (temporarily invalid constructs should not be rejected at once)
- Regulations (i.e., constraints) can also be used to make sure that certain required steps during development phase transitions are performed

Inter-Phase Consistency



Consistency-preserving Phase Transitions: Using Queries

```
[Query = "Query" @"http://..."]
[DesignPattern = "DesignPattern" @"http://..."]
{UseCase, Query, [[
 SELECT $usecase WHERE
 applies-to($usecase : UseCase, $designpattern :
            DesignPattern)]]}

[Component = "SoftwareComponent" @"http://..."]
{UseCase, Query, [[
 SELECT $usecase WHERE
 applies-to($usecase : UseCase, $designpattern :
            DesignPattern) AND
 is-implemented-by($designpattern : DesignPattern,
                   $comp : Component)]]}
```

Navigation and Views

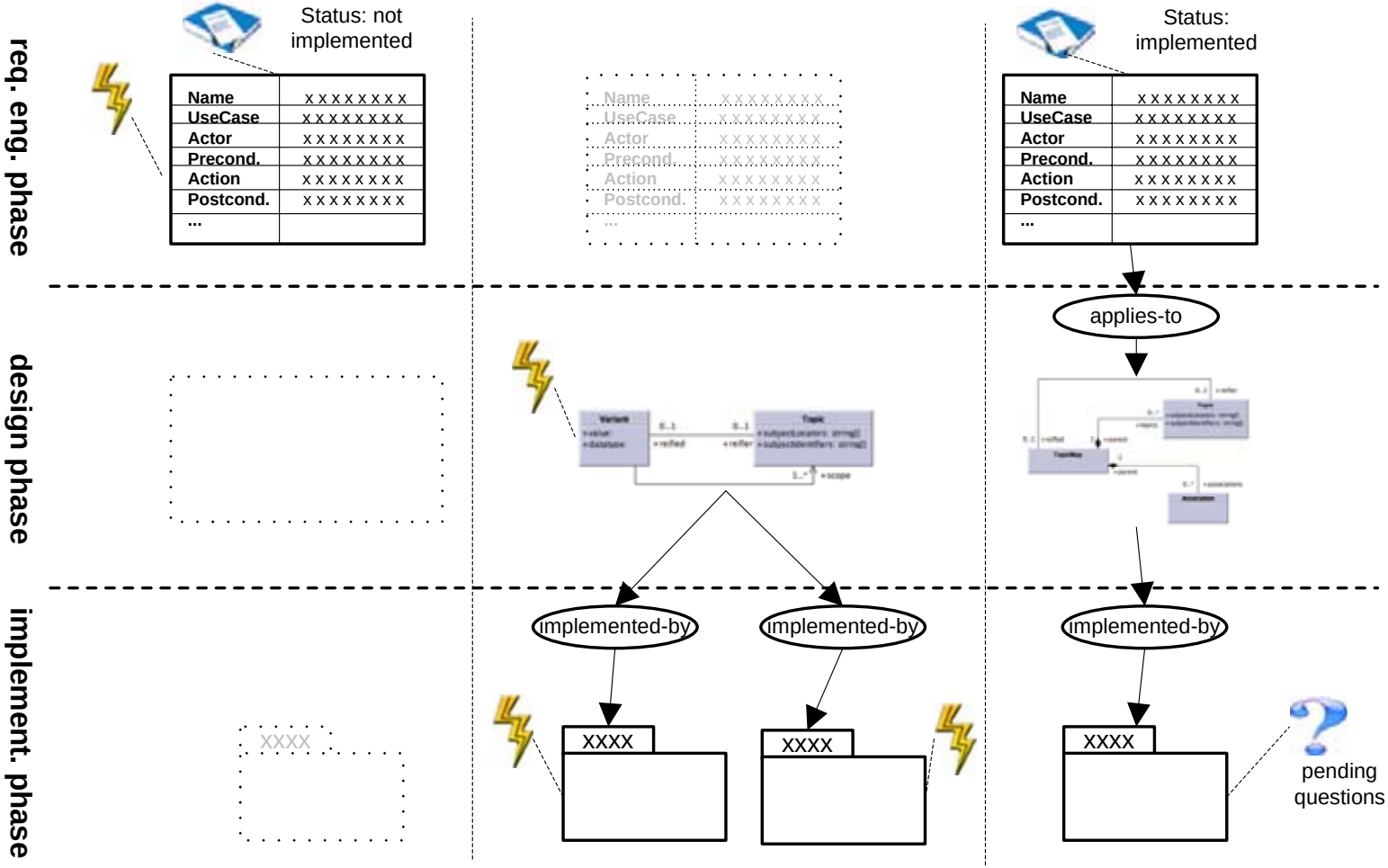
Different views and objectives of participants involve, e.g.,

- different (phase-dependent) concepts
- different needs of details
- different notations

Therefore, flexible visualisation, (semantic) zooming and filtering capabilities are needed

- possible visualisations: Conceptual Graphs, UML diagrams, code fragments
- predicate-based queries (also address scalability aspects)

Navigation and Views: Overview



View Definitions

LTM notation

```
[View = "View" @"http://..."]
[JAR = "JavaArchive" @"http://.../Application/#JavaArchive"]
[Filter : JAR = "FilterLogics" @"http://.../Application/#FilterLogics"]
[Display : JAR = "DisplayLogics" @"http://.../Application/#DisplayLogics"]
[Editor : JAR = "EditorLogics" @"http://.../Application/#EditorLogics"]

uses-layouter(placeholder:View, placeholder:Layouter)
has-filter(placeholder:View, placeholder:Filter)
has-display(placeholder:View, placeholder:Template, placeholder:Display)
has-editor(placeholder:View, placeholder:Template, placeholder:Editor)
is-connected(placeholder:View, placeholder:View)

[FormsEditor:Editor = %"http://.../Application/FormsEditor.jar"]
[UseCaseDescView:View = @"http://.../Application/#UseCaseDescriptionView"]
[DesignPatternView:View = @"http://.../Application/#DesignPatternView"]
has-editor(UseCaseDescView, UseCaseDescription, FormsEditor)
is-connected(UseCaseDescView, DesignPatternView)
...
```

State of Implementation

- (Re-)Implementing a lightweight Topic Maps engine with needed parsing/serialising facilities (using AOP techniques)
- Investigating rules to
  - generate forms-driven dialogs from template definitions
  - support traceability
  - facilitate navigation and generate reports (“executive summaries”)
- Evaluating different query languages (cf. earlier slides)
- Medium-term goal: Eclipse 3.2 Plugin
  (Prototype: Q1/2007)

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Thank you!

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