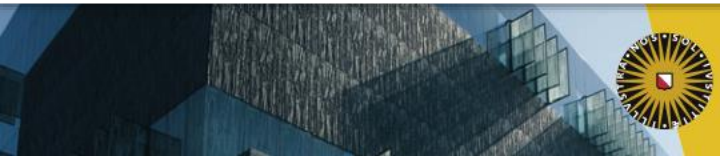
Fragments and method assembly

Session 7
11 March 2019

Universiteit Utrecht

Method assembly - agenda

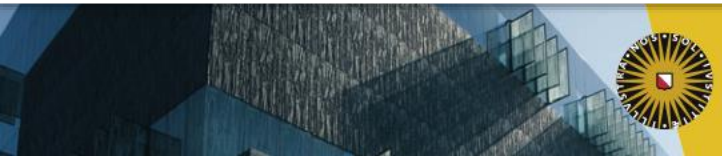
- Situationality and evolution of methods
- Method fragments
- Structuring the Method Base
- Method assembly: rules and guidelines
- An example of Method assembly



Outline

- Situational method engineering for informational system project approaches (Harmsen, Brinkkemper & Oei, 1994)

- Meta-modelling based assembly techniques for situational method engineering (Brinkkemper, Saeki, Harmsen, 1999



The poor ICT-worker…

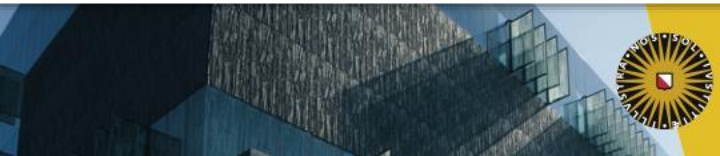
Many books on methods contain descriptions on phases and steps, but …

your project is always different

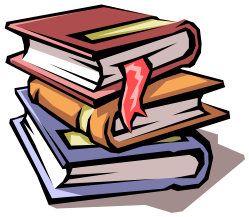
So:
how to adapt the process to suit your circumstance?

Diversity in IT projects

- Different IT systems
  - Information systems
  - Apps, web-applications, portals
  - Workflow, business intelligence
  - Product software, commercial software, freemium
  - Real-time systems, embedded software
- Different platforms
  - AWS, Azure, Google apps
  - Relational DBMS
  - Java, J2EE, Eclipse
  - MS Access, .NET,
  - LAMP (Linux, Apache, MySQL, PHP)
- Different domains
  - Consumer, gaming
  - Financial, insurance, banking
  - Educational
  - Government, public sector
  - Transport, logistics
  - IT services
  - Manufacturing, production
  - Service industry
  - Energy, oil, utilities



Situational Method



Method Description

Method base

Project Situation

Project Characterisation

Selection and Assembly of Method fragments

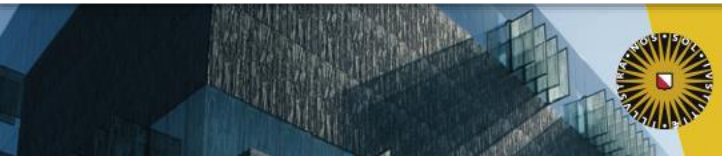
Situational Method

Situational

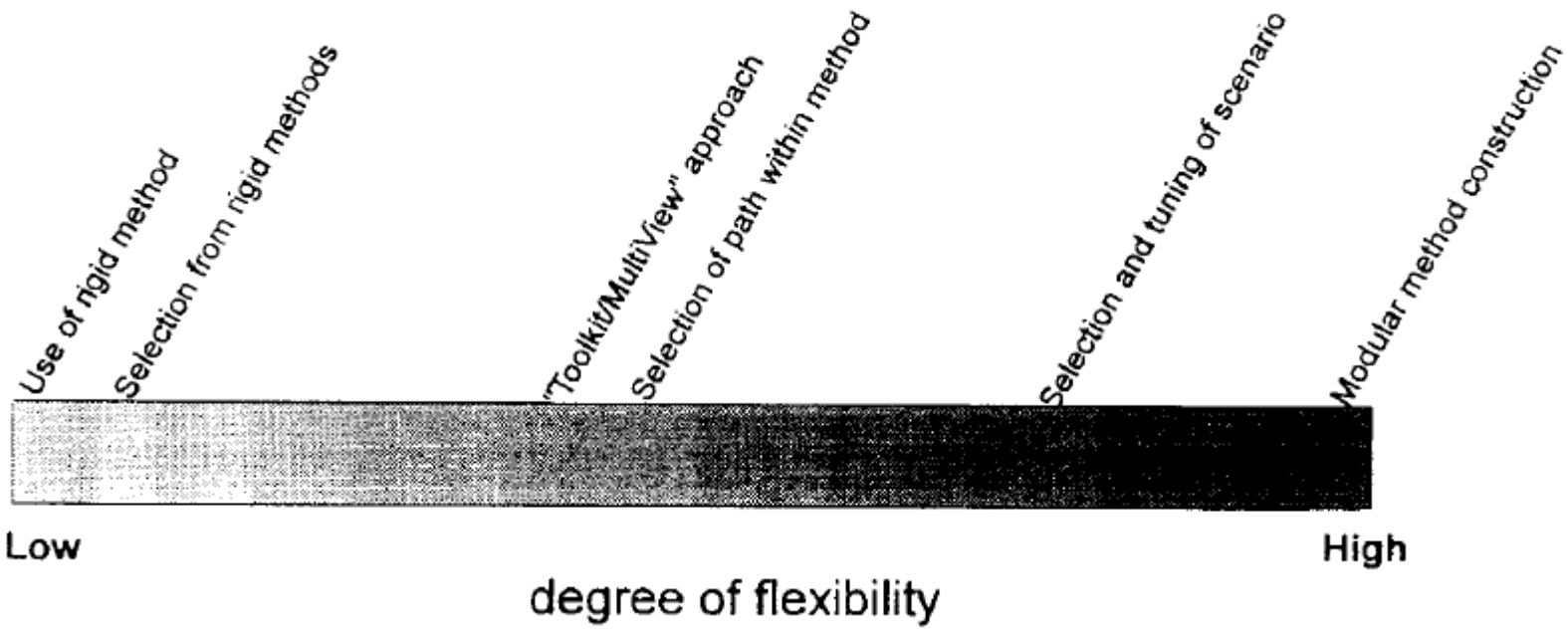
Definition

A *Situational method* is a method tuned to a specific situation.

Situational Method Engineering is the area of method engineering focusing on situational methods.

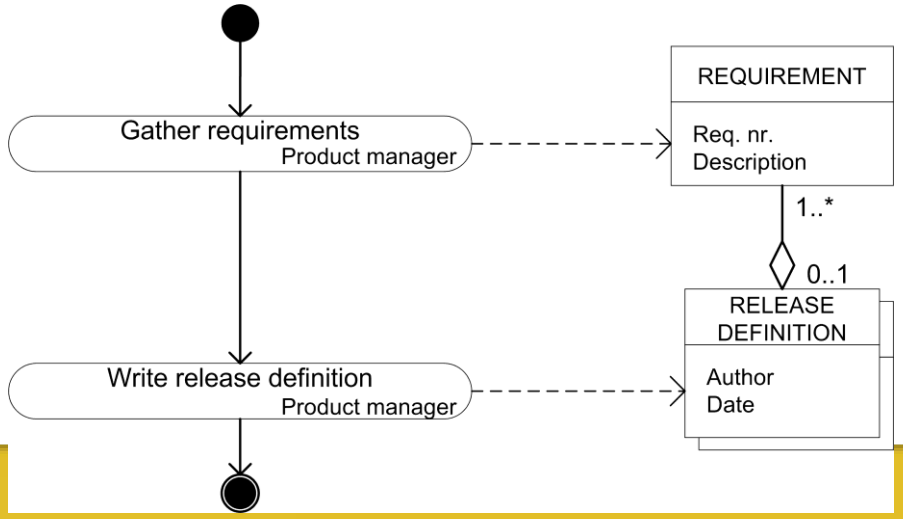


Situational method spectrum

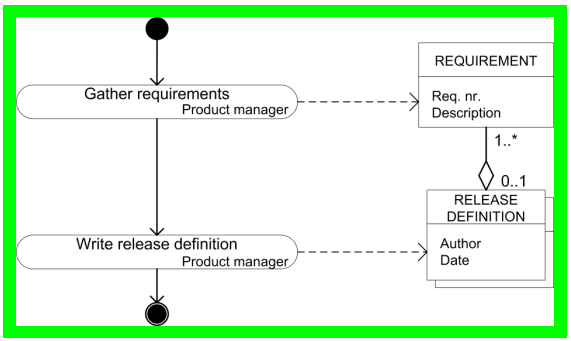


Situationality in software industry

- ERP software company
- Three years old
- 50 employees
- Problems:
  - Releases not completed on time
  - Stakeholders not satisfied with the implemented requirements
- Current Process (Incr. #0):
- (Note: more increments later)

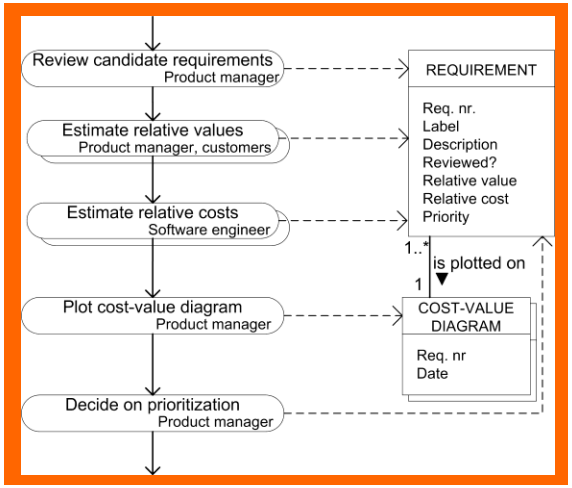


Method assembly for ERPComp

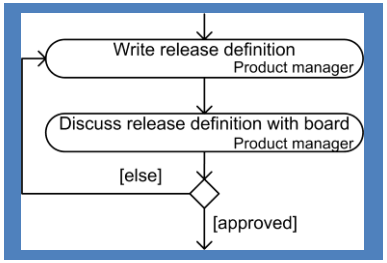


Increment #0

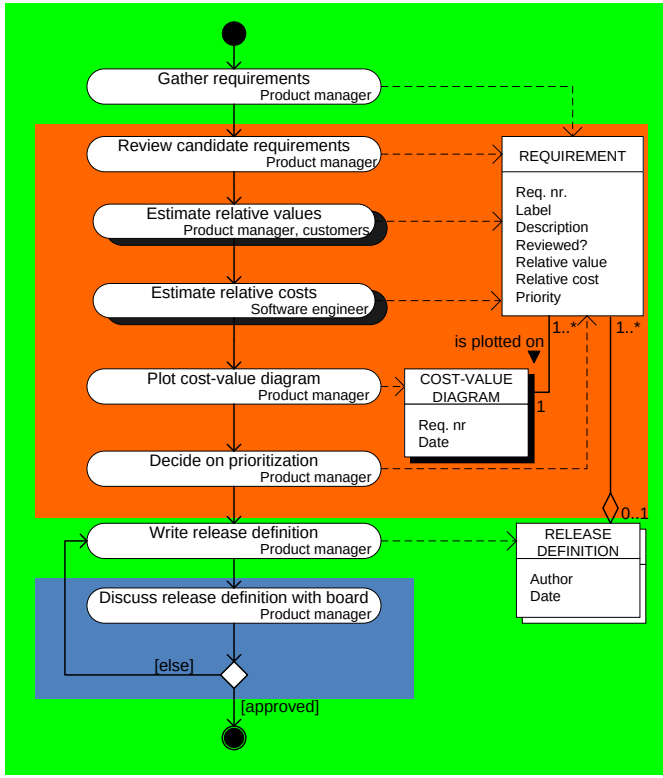
Method fragment
for requirements prioritization



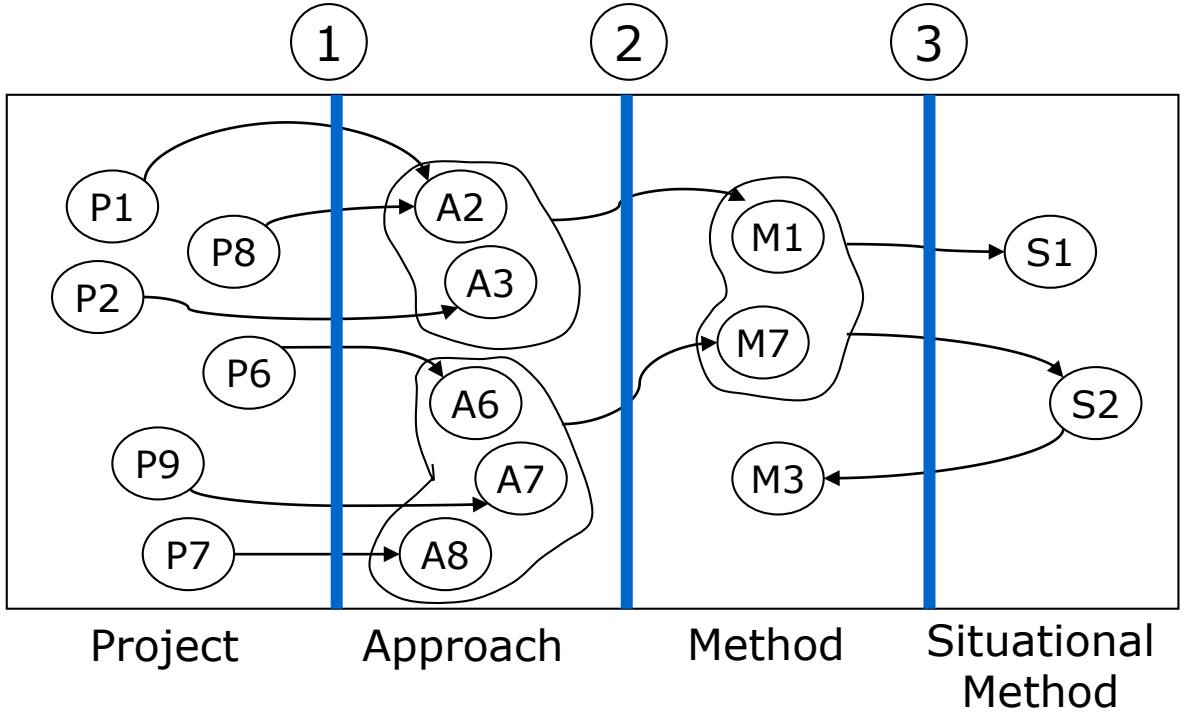
+ Method fragment
for release validation



Increment #1



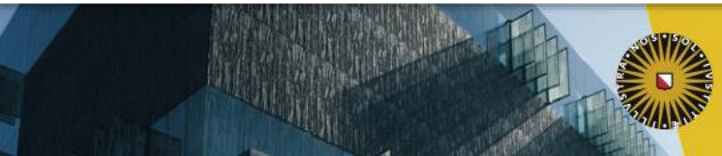
Method evolution



1. In project experiences generic activities are identified and new approaches are created
2. Approaches are codified into published methods
3. Situational methods are derived from published methods

Method assembly - agenda

- Situationality and evolution of methods
- Method fragments
- Structuring the Method Base
- Method assembly: rules and guidelines
- An example of Method assembly



Method Fragment

Definition

A *Method fragment* is any coherent building block of a method, technique or tool

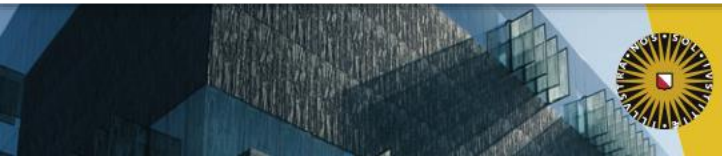
Examples of method fragments:

- **Phase**: Feasibility, Deployment, Beta Test
- **Activity:** Create Data model, Verify Release plan
- **Route map:** Packaged systems selection
- **Deliverable:** Functional Design, Test plan
- **Technique:** Use Cases, State diagram
- **Concept:** Entity, Class, Action, Test

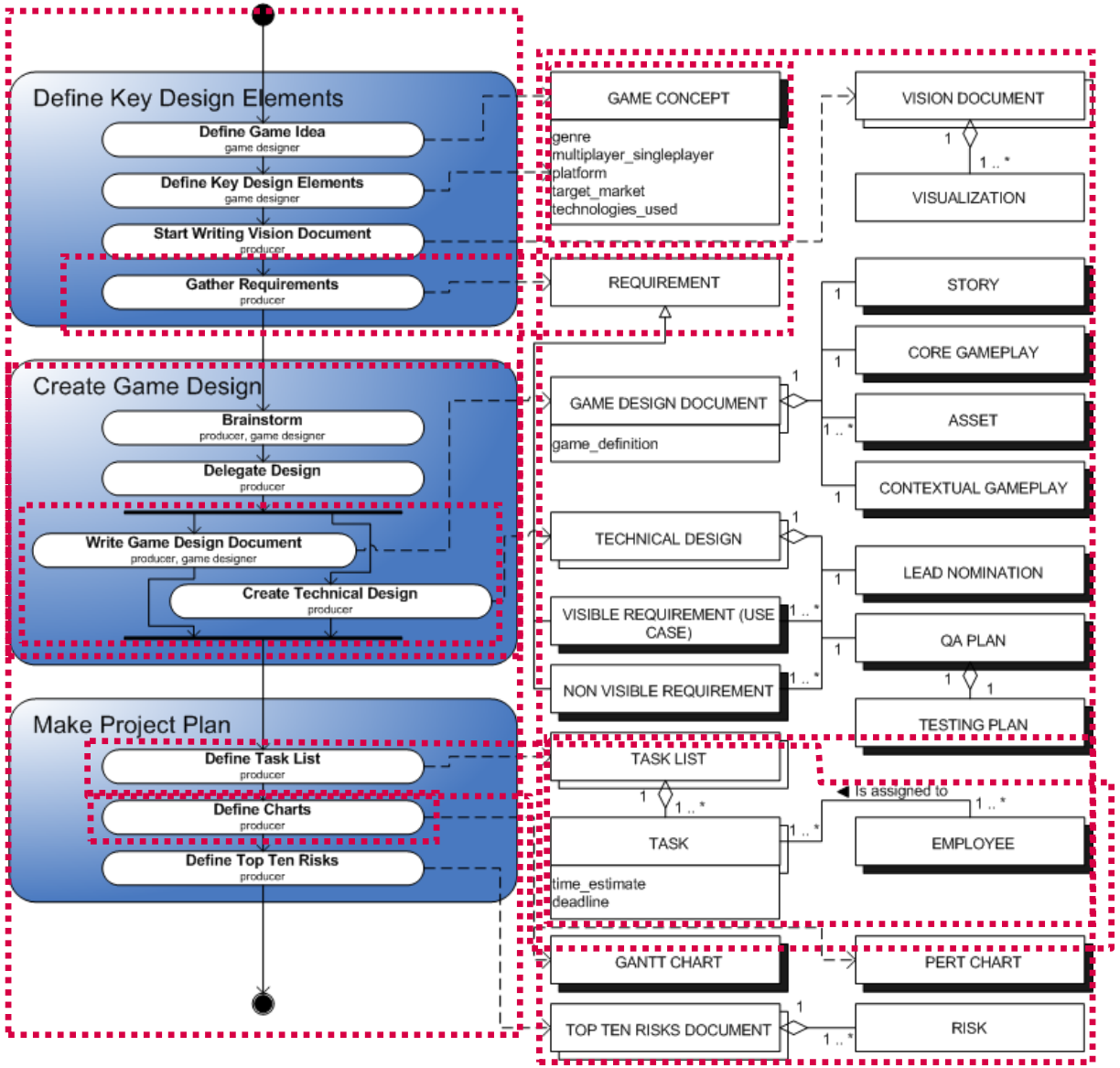
Process perspective

Product perspective

Method fragments can be at any abstraction level of a method, technique or tool.



Examples of method fragments



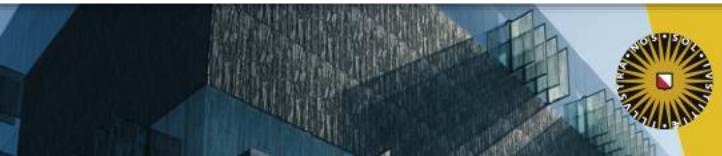
Types of method fragments

- Product fragments
  products and sub-products to be delivered by a method, such as deliverables, milestone documents, models, diagrams, etc.

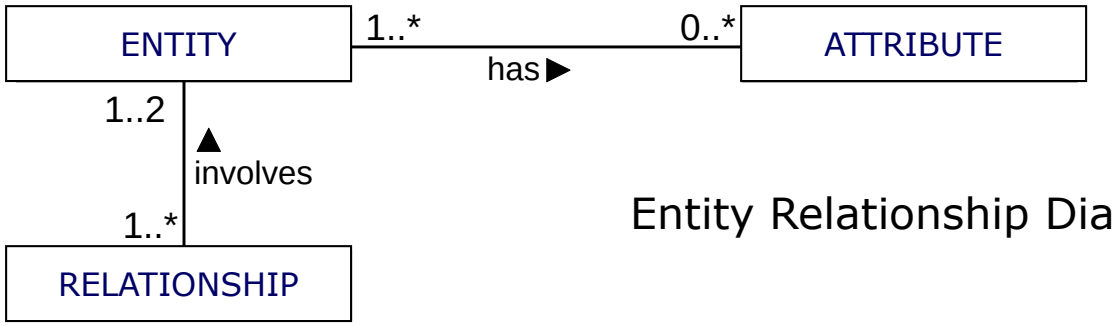
- Process fragments
  represent the stages, activities and tasks to be carried out in order to produce product fragments.

- Combination fragments
  contain both process fragments and product fragments, where the product is the output of the process

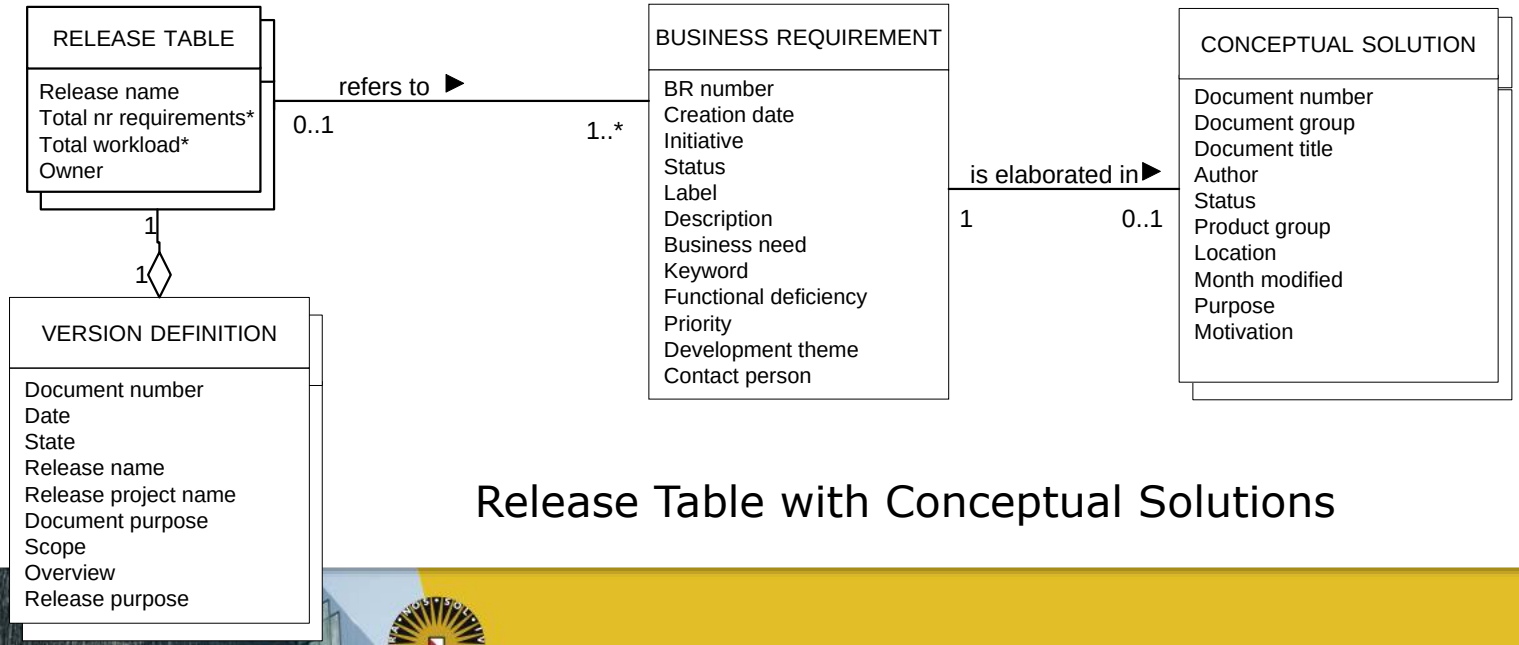
Meta-data models and meta-process models are used for the description and manipulation of method fragments



Product fragments

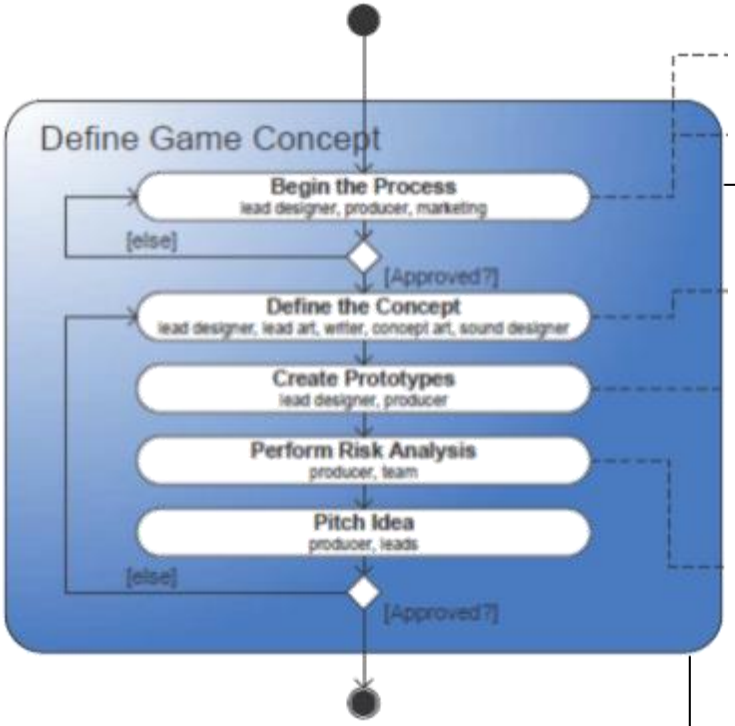


Entity Relationship Diagram

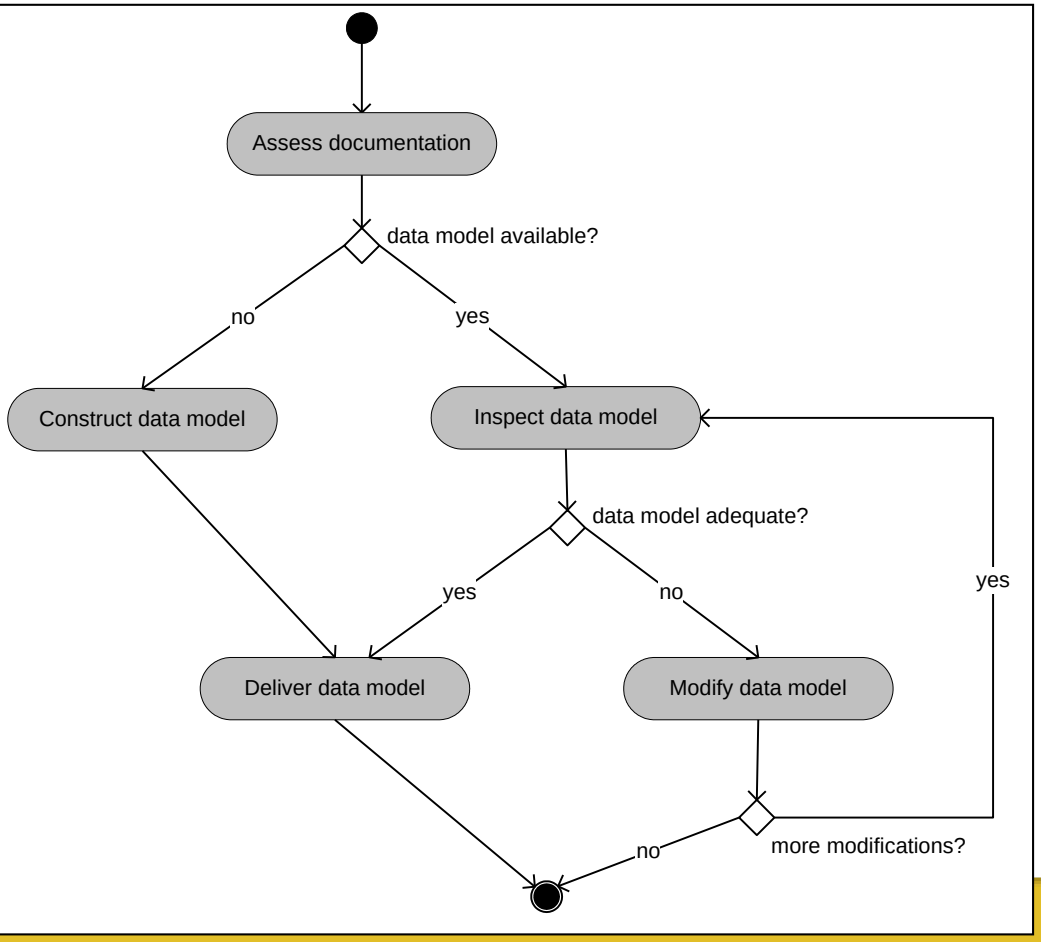


Release Table with Conceptual Solutions

Process fragments



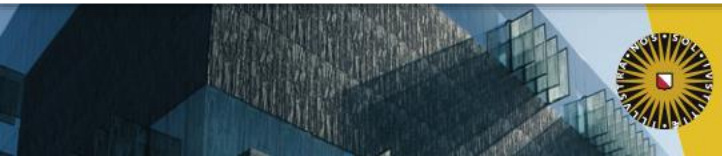
Process: Define game concept



Process: Make data model for new database

Method assembly - agenda

- Situationality and evolution of methods
- Method fragments
- Structuring the Method Base
- Method assembly: rules and guidelines
- An example of Method assembly



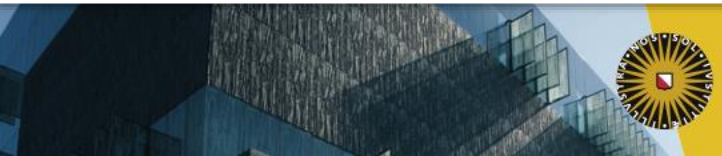
Method base

Three types of operations on method fragments in method base

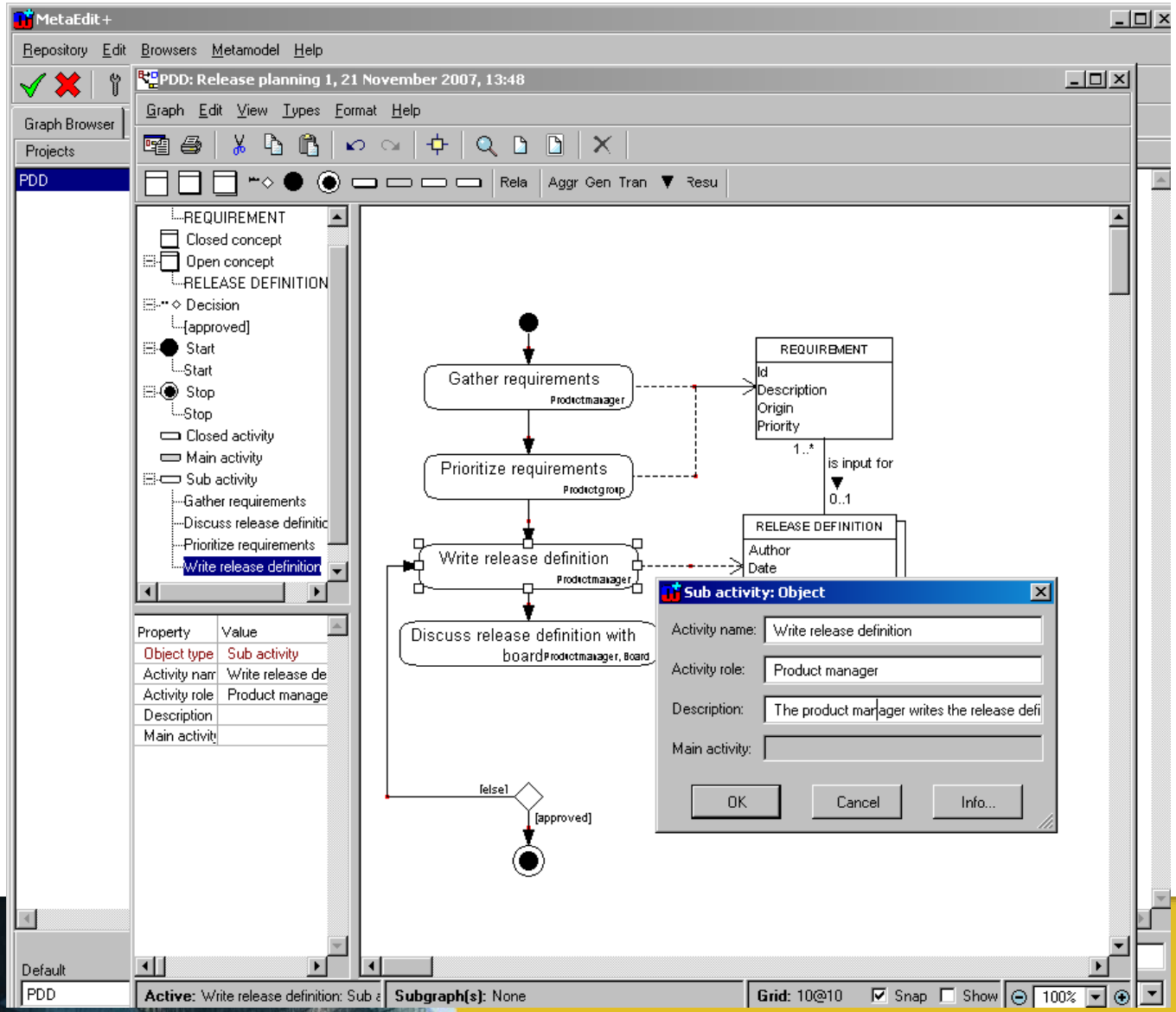
- Administration operations: insertion and modification of MFs; PDDs and descriptors
- Selection operations: query for situational MFs
- Assembly operations: creation of a new MF out of existing MFs

Problem:

How to store the MFs?



CAME tool


MetaEdit+
Repository Edit Browsers Metamodel Help
PDD: Release planning 1, 21 November 2007, 13:48
Graph Edit View Types Format Help
Graph Browser
Projects
PDD
REQUIREMENT
Closed concept
Open concept
RELEASE DEFINITION
Decision
[approved]
Start
Start
Stop
Stop
Closed activity
Main activity
Sub activity
Gather requirements
Discuss release definitic
Prioritize requirements
Write release definition
Property Value
Object type Sub activity
Activity nam Write release de
Activity role Product manage
Description
Main activity
Gather requirements
Productmanager
Prioritize requirements
Productgroup
Write release definition
Productmanager
Discuss release definition with board Productmanager, Board
[else]
[approved]
REQUIREMENT
Id
Description
Origin
Priority
1..*
is input for
0..1
RELEASE DEFINITION
Author
Date
Sub activity: Object
Activity name: Write release definition
Activity role: Product manager
Description: The product manager writes the release defi
Main activity:
OK
Cancel
Info...
Default
PDD
Active: Write release definition: Sub a
Subgraph(s): None
Grid: 10@10
Snap
Show
100%

Storing Method Fragments in the Method Base

- How would the storage structure of a Method Base look like?
- Discuss with your neighbour

- What is the Mx MOF level?

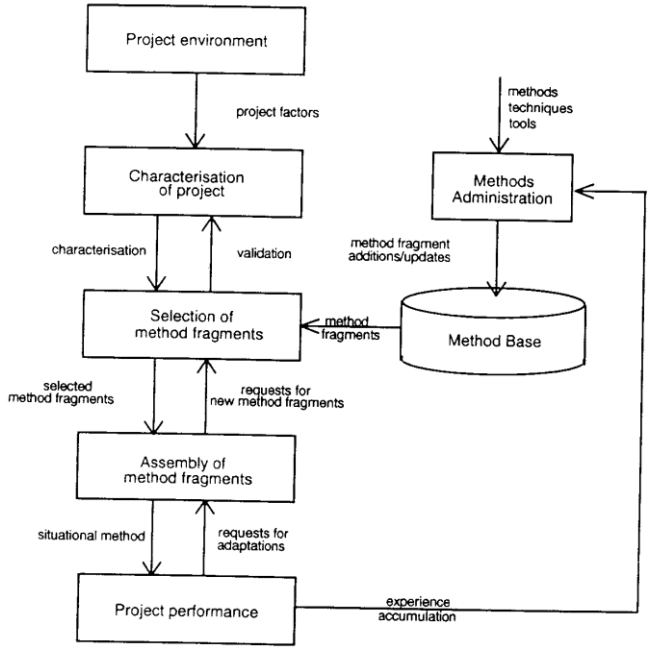
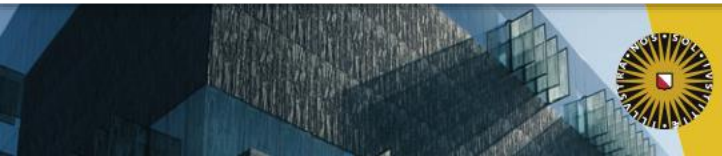


Figure 1 The process of configuration of situational methods

Method engineering levels

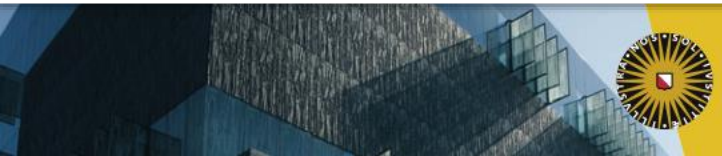
- Macro level of systems development organizations
- Meso level of systems development projects
- Micro level of systems development techniques

Harmsen, Brinkkemper & Oei (1994)



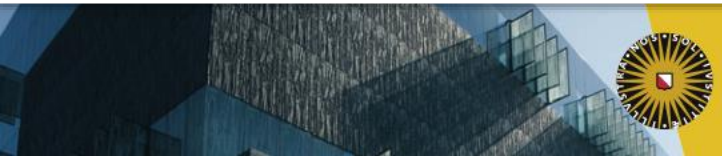
Method Engineering Framework

- Granularity layer
  - Method: Information Engineering, UML
  - Stage: Business Analysis, Technical Design
  - Model: Data Model, UI model
  - Diagram: Object diagram, Class Hierarchy
  - Concept: State, Event, Identify States



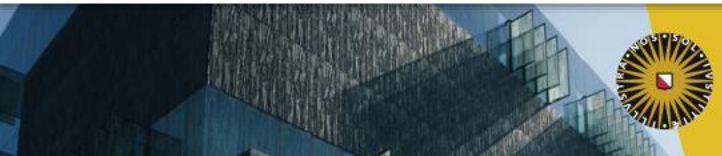
Method Engineering Framework

- Granularity layer
- Perspective
  - Product: deliverables, milestone documents, models
  - Process: stages, activities, tasks



Method Engineering Framework

- Granularity layer
- Perspective
- Abstraction level
  - Conceptual level: descriptions, meta-models
  - Technical level: tools (e.g. Statemate)



Fragment relations: Production

Production

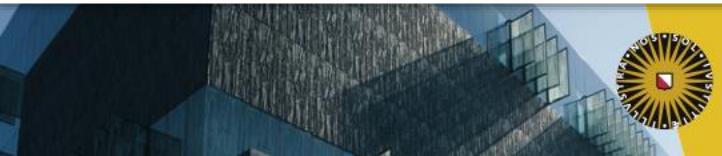
- Process fragments produce product fragments (output relationship)

  Example: *Create Functional Design* produces *the Functional Design document.*

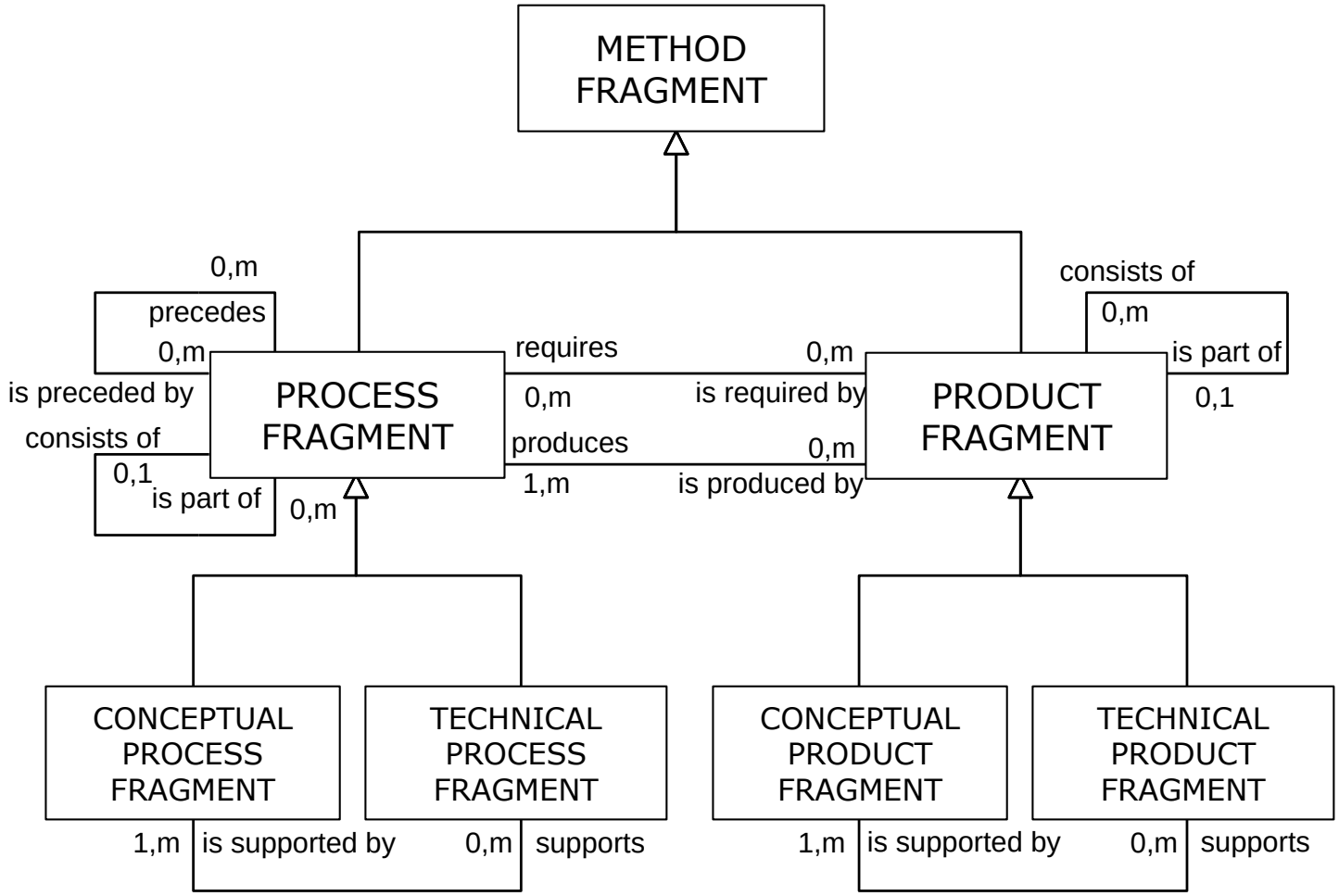
- Product fragments are required by process fragments (input relationship)

  Example: *Functional Design* is required *by Develop Interaction Workflows.*

- So: process fragments precede other process fragments (implied relationship)



Fragment relations: Structure of Method Base



Fragment relations: Containment

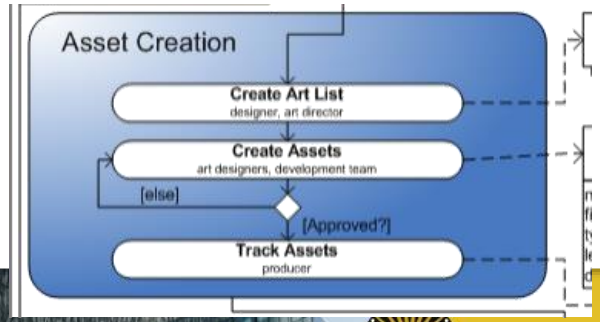
Containment

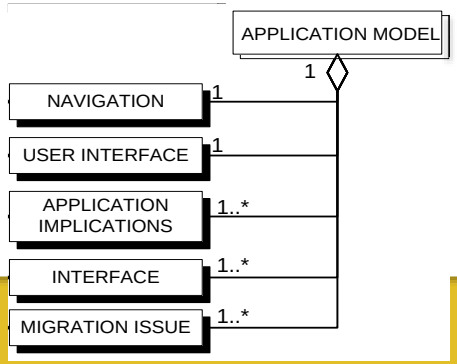
- A process fragment may consist of other process fragments

  Example: *The Asset Creation* consists of *Create Art List, Create Assets,* and *Track Assets.*

- A product fragment may consist of other product fragments

  Example: *The Application model document* consists of *a Navigation, the User Interface, Application implications, Interface,* and *the Migration Issue.*





Fragment relations: Support

Support

- Conceptual product fragment consists of the concepts, descriptions and philosophy behind the method fragment

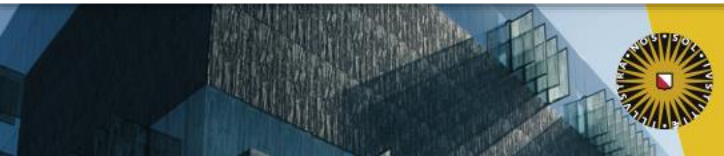
  Example: *Class diagramming in the book of RUP of Kruchten.*

- Technical product fragment is a support tool

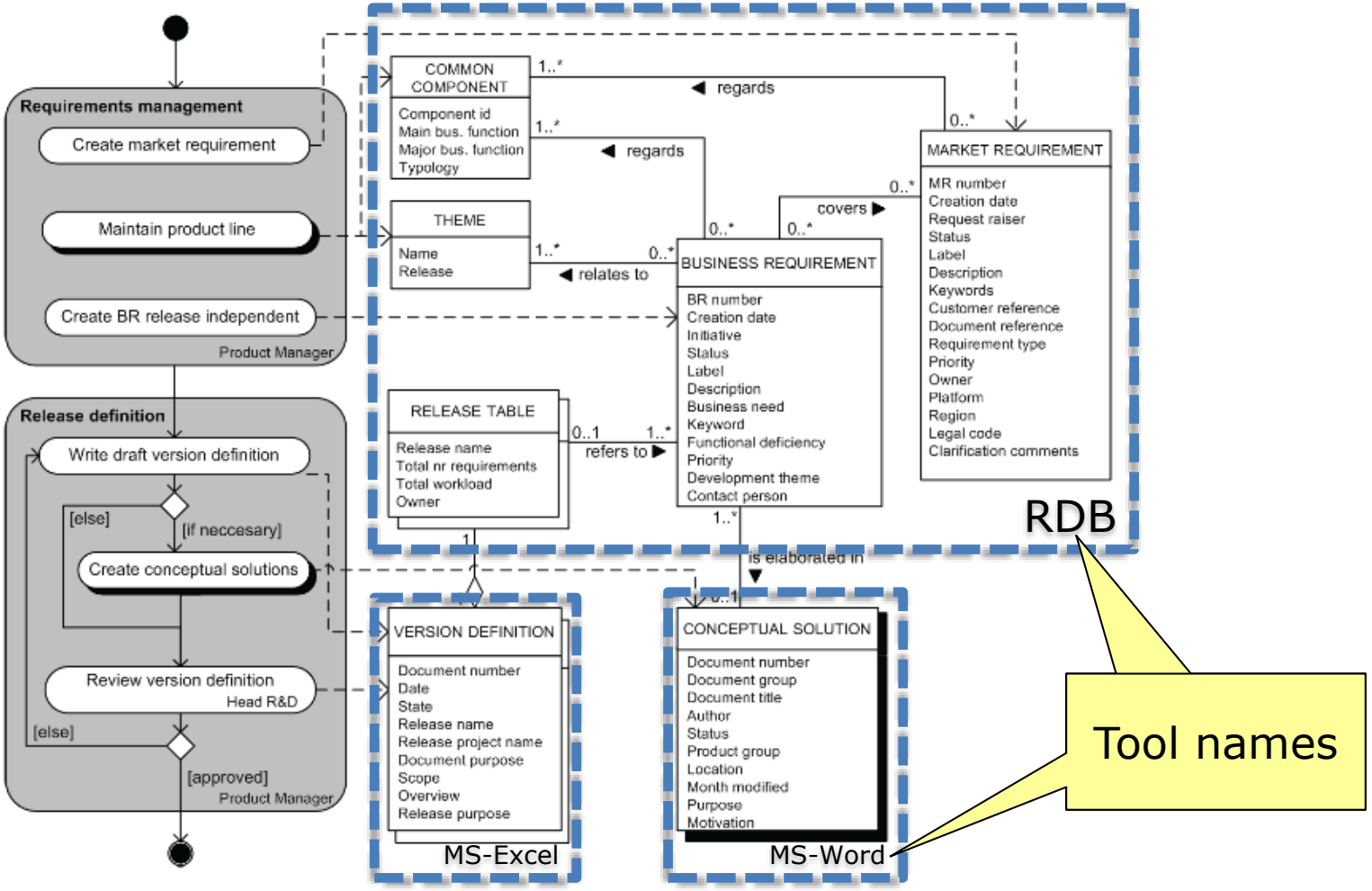
  Example: *Visual Studio 2005 Class Designer*

- Technical product fragments support conceptual product fragments

  Example: *Visual Studio 2005 Class Designer* supports *Class diagramming.*



Recall: notation for support by Technical Product Fragment

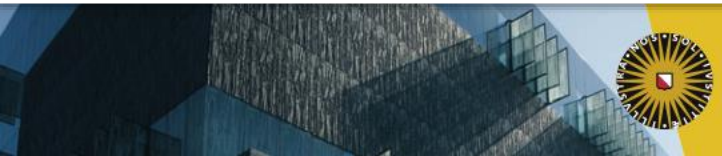


Support tools are listed on the deliverable side

Consistency checking

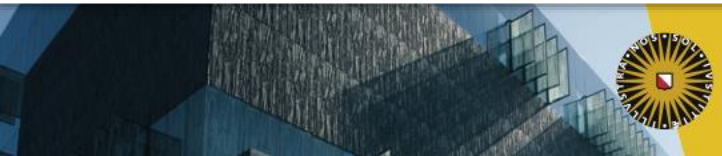
- Consistency between granularity levels of the situational method
- Precedence consistency
- Input/output consistency
- Support consistency

Note, these are all present in the data model of the method base.



Method assembly - agenda

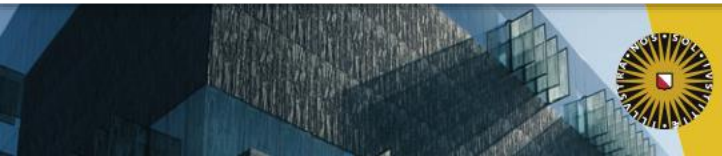
- Situationality and evolution of methods
- Method fragments
- Structuring the Method Base
- Method assembly: rules and guidelines
- An example of Method assembly



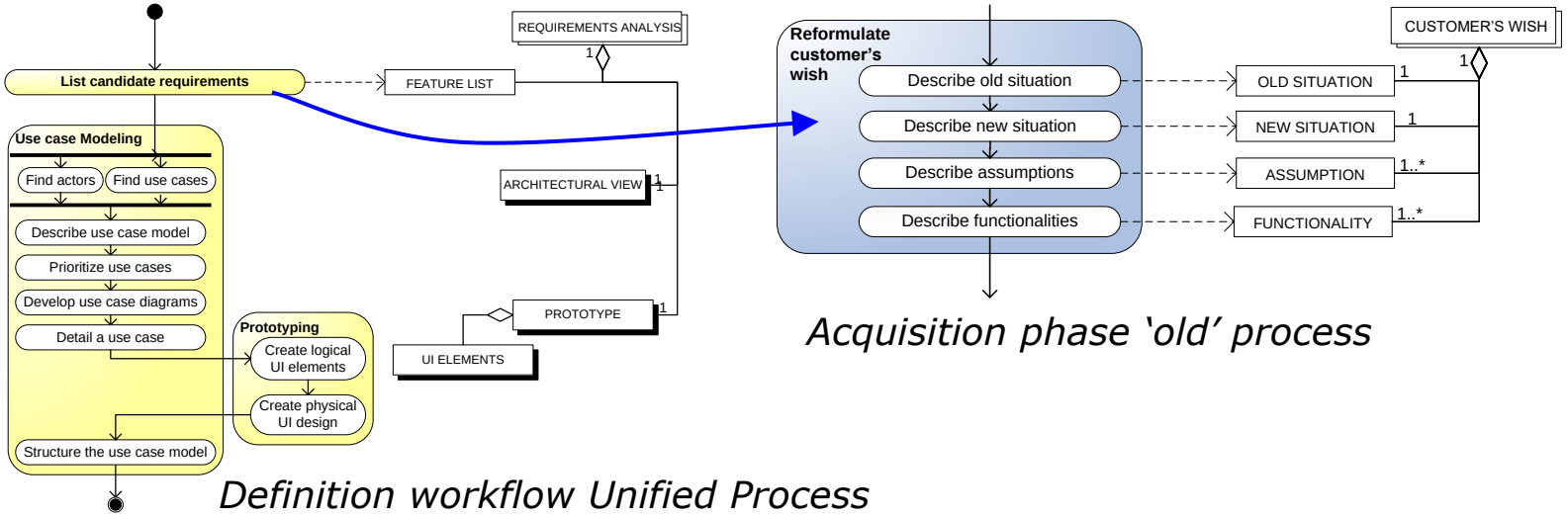
Engineering methods

- Insertion, Change or Removal of
  - Concept
  - Activity
  - Association
  - Rule
  - Etc.

- Assembly of fragments
  - Meta-data models of product fragments
  - Meta-process models of process fragments

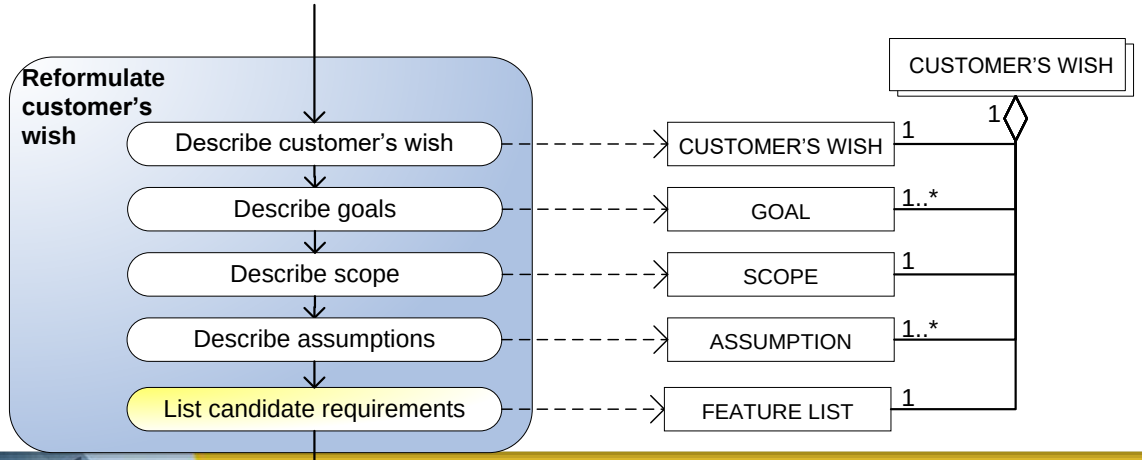


Insertion of an activity and concept



Definition workflow Unified Process

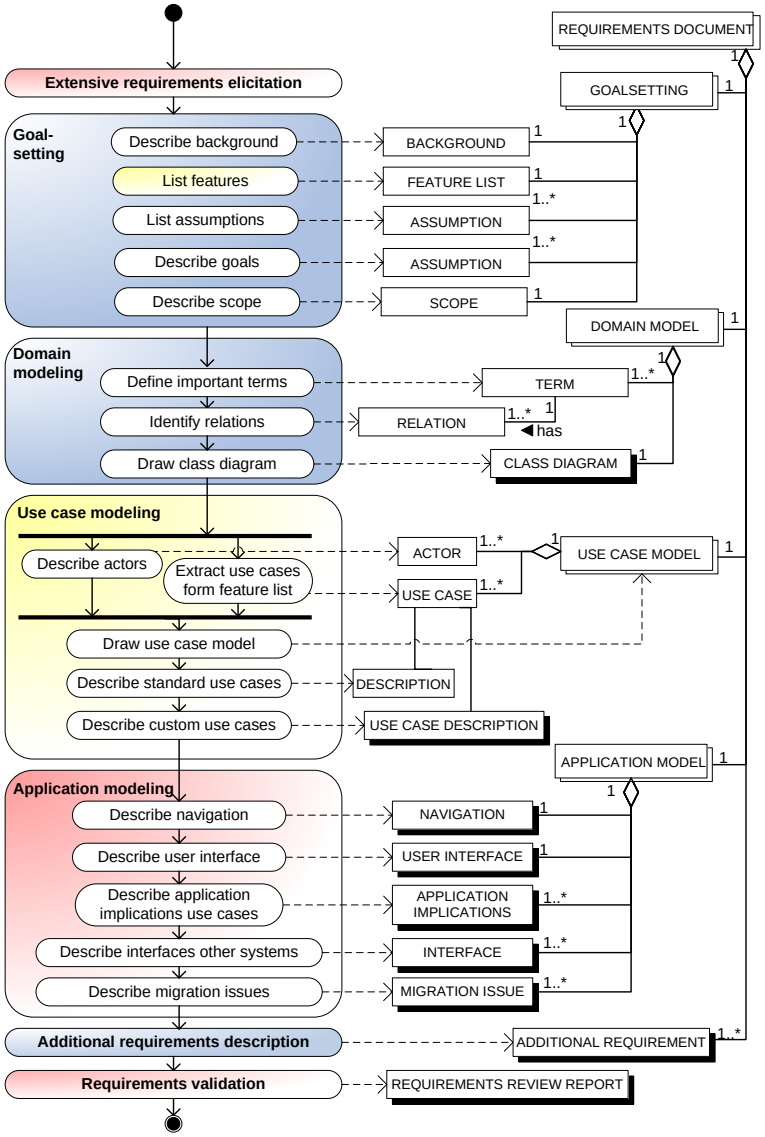
Acquisition phase 'old' process

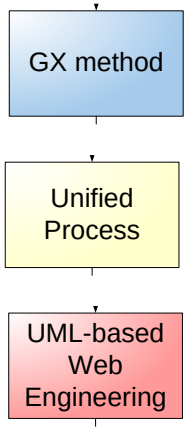


Acquisition phase after insertion method fragment

Method assembly

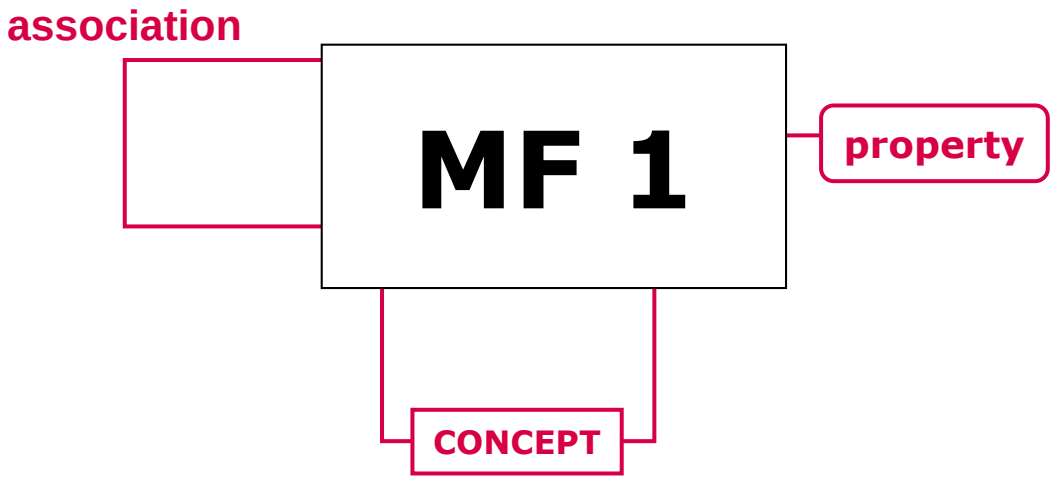
Resulting Complex Definition phase after assembly at GX





Assembly rules - 1

- At least one concept, association or property should be newly introduced to each method fragment to be assembled, i.e. a method fragment to be assembled should not be a subset of another.



Assembly rules - 2

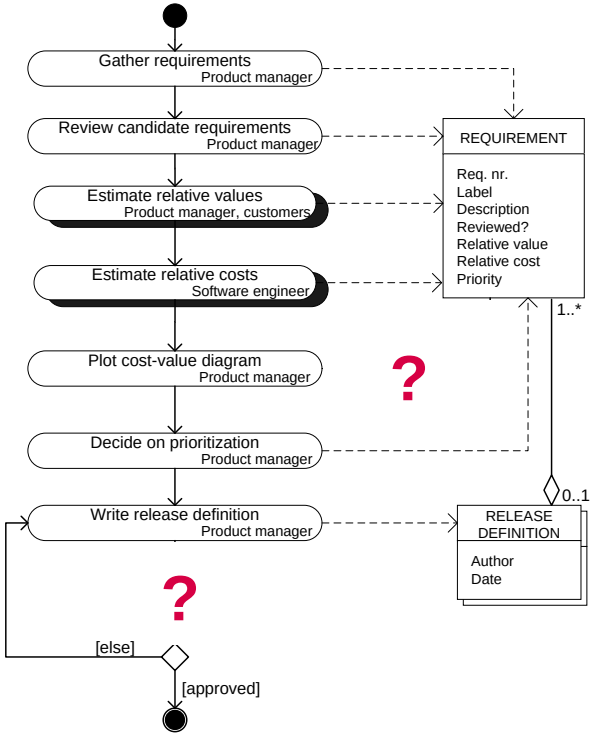
- If we add new concepts, they should be connectors to both of the assembled method fragments

Assembly guidelines

- Completeness
- Consistency
- Efficiency
- Reliability
- Applicability

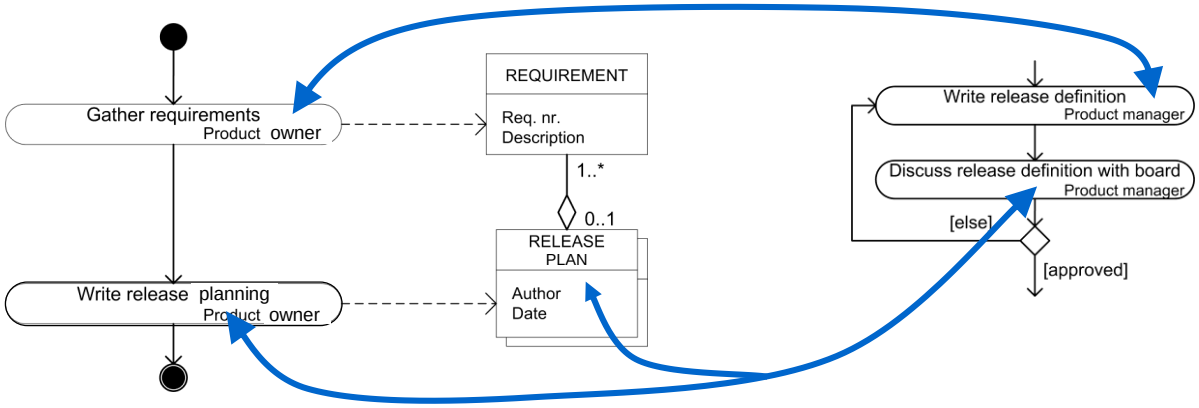
Completeness guideline

- The situational method contains all the method fragments that are referred to by other fragments in the situational method.
  - i.e. there are no gaps in the assembled method.



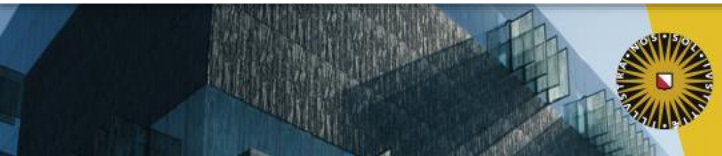
Consistency guideline

- All activities, products, tools and people plus their -mutual- relationships in a situational method do not contain any contradiction and are thus mutually consistent.
  - Names of concepts, roles, deliverables, activities are made consistent
  - Abstraction level is made consistent



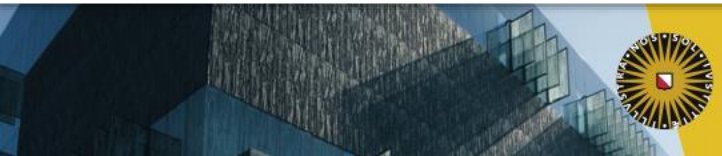
Efficiency guideline

- The method can be performed at minimal cost and effort.
  - No superfluous activities
  - Each deliverable adds value to the method



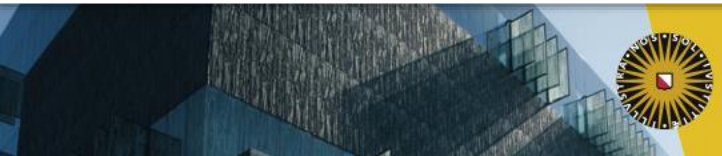
Reliability guideline

- The method is semantically correct and meaningful.
  - All concepts have been defined correctly and have well accepted meaning
  - Same for activities, roles, etc.



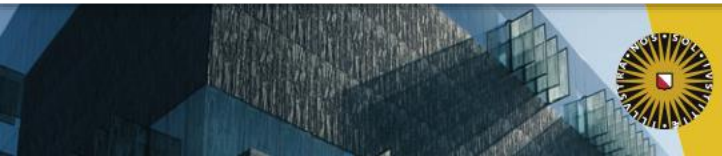
Applicability guideline

- The developers are able to apply the situational method
- Learnability of the MFs
- History of the composite MFs, so that component MFs are already known



Method assembly - agenda

- Situationality and evolution of methods
- Method fragments
- Structuring the Method Base
- Method assembly: rules and guidelines
- An example of Method assembly



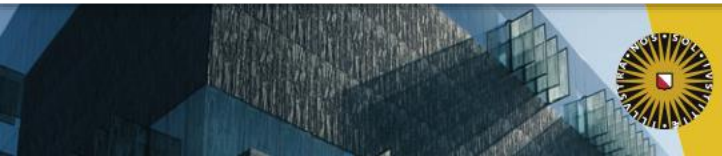
Method assembly of techniques

- Assembly in the product perspective
- Assembly in the process perspective

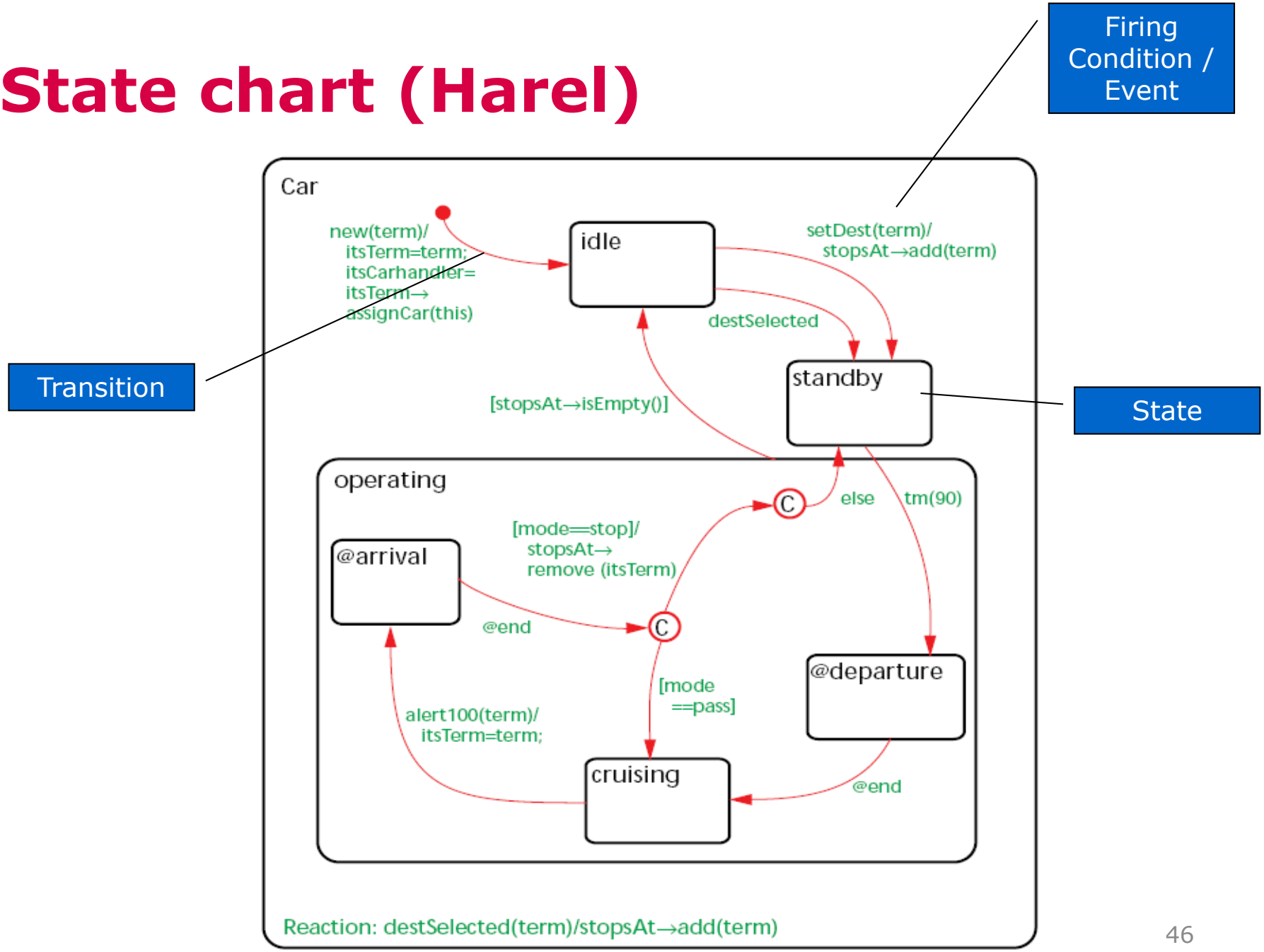
Example:

Statechart (Harel 90) +
Object model (Coad&Yourdon 91) →
Objectchart (Coleman 92)

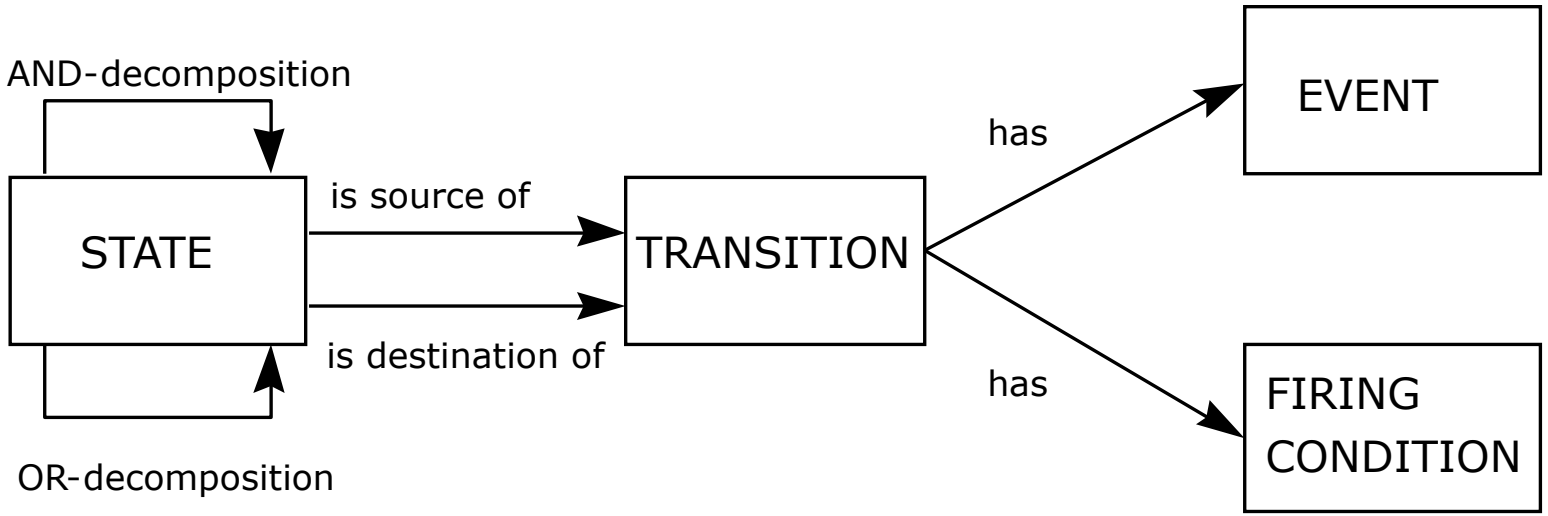
Note: a similar notation as PDD is being used in this example



State chart (Harel)



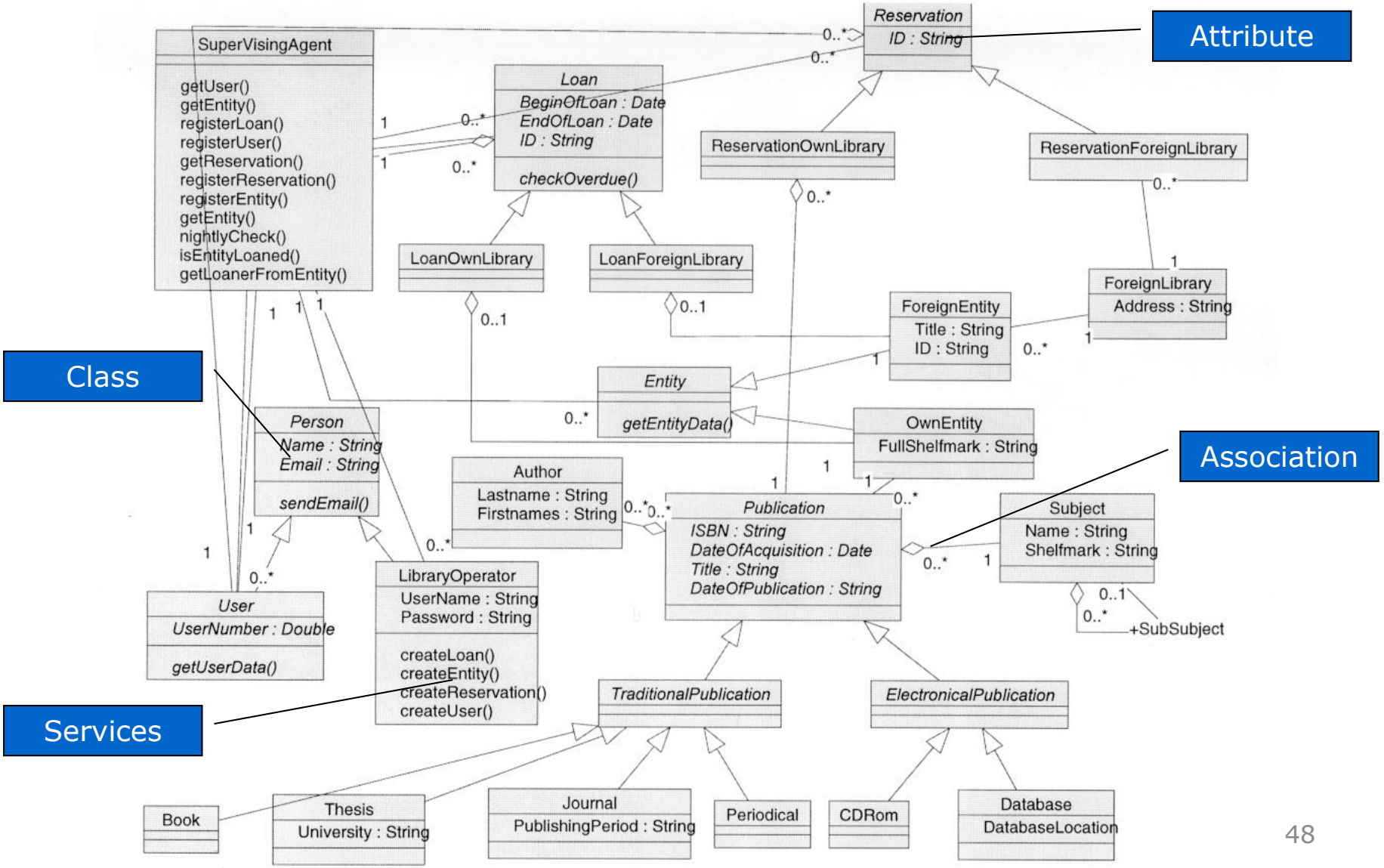
State chart: meta-data model



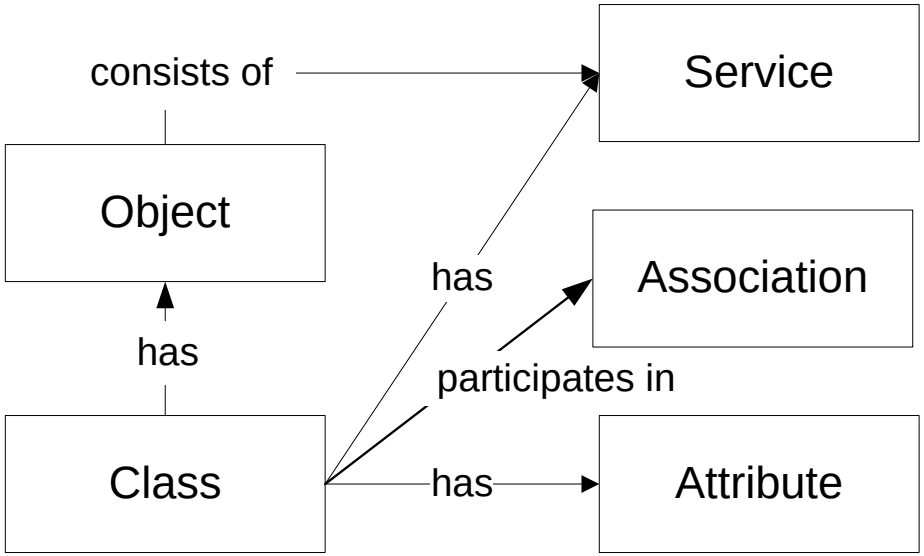
Product fragment State chart

Note: we use the shorthand → *notation instead of* 0..1 0..*

Object model (Coad & Yourdon)



Object model: meta-data model

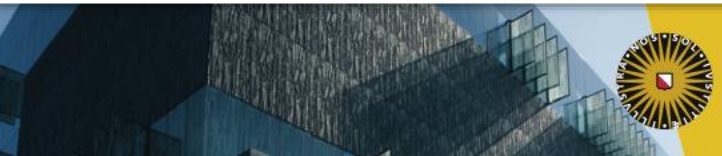


Product fragment Object Model

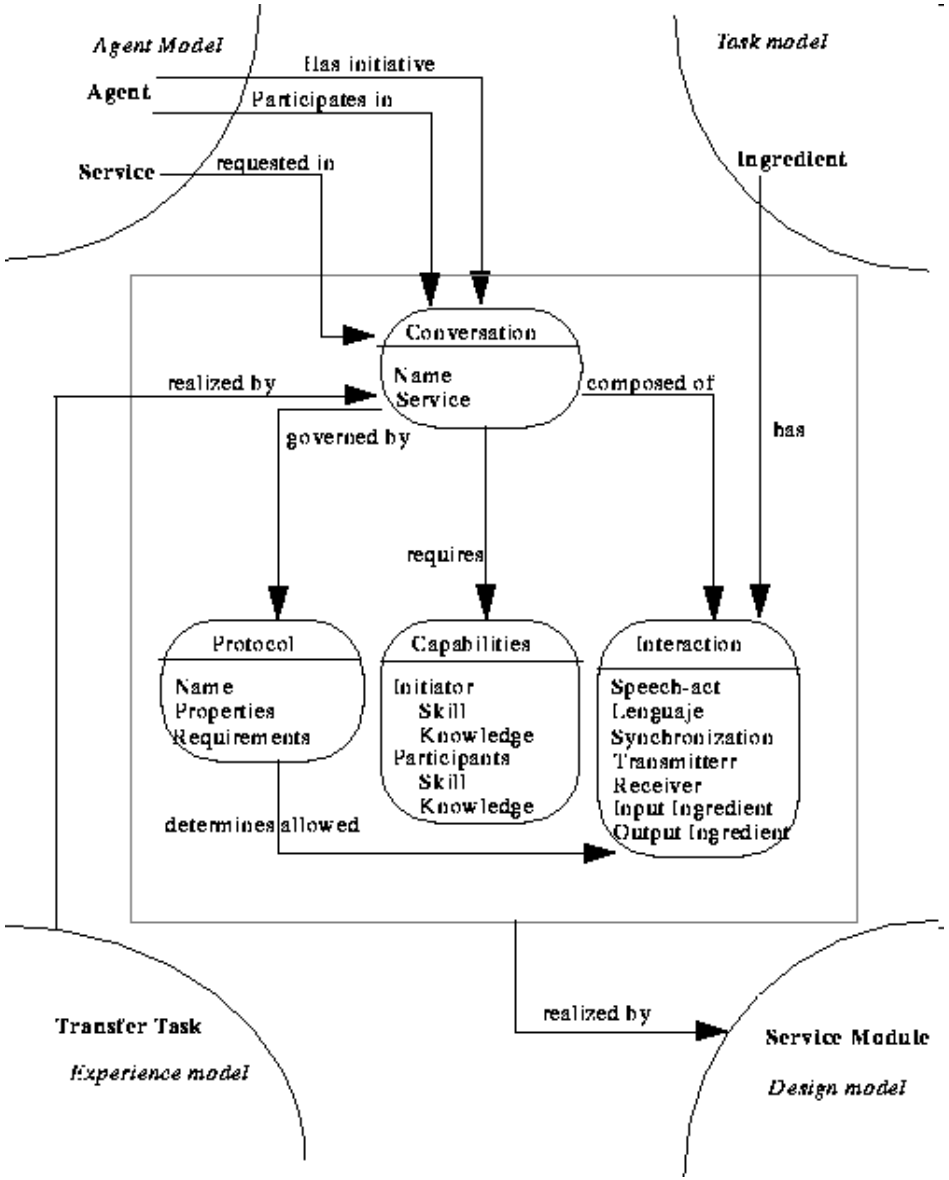
Creation of object chart

1. A Class has a Statechart, which specifies its behaviour
2. Attributes of a Class may be annotated to States in its Statechart. This indicates which attribute values are meaningful or visible in a specific State.
3. An Event issued during a Transition is a request of a Service to the other Object
4. A Transition may change an Attribute value of an Object

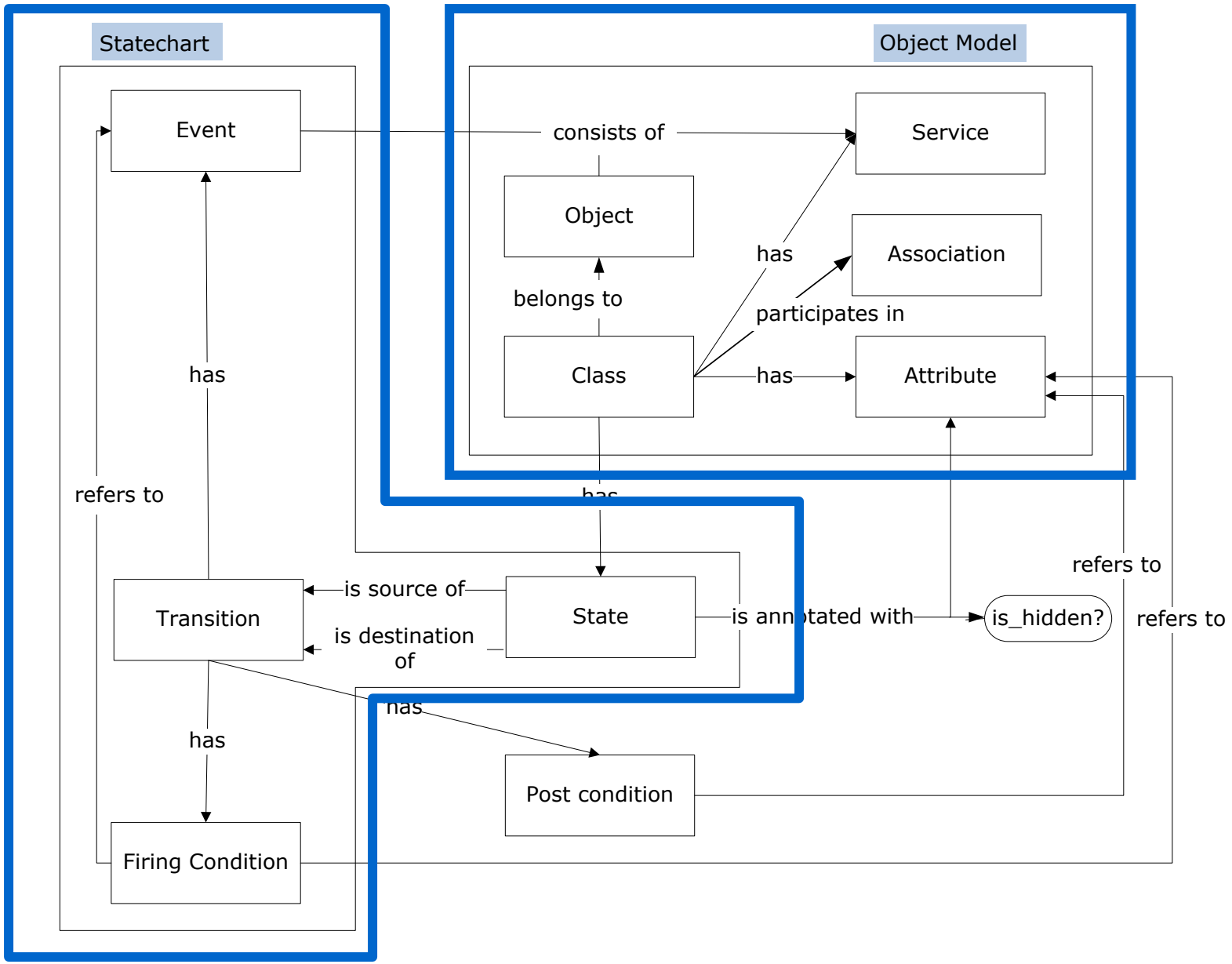
- New Associations due to 1, 2, and 3
- New Concept: Post-condition due to 4
- New Property: 'is hidden' due to 2



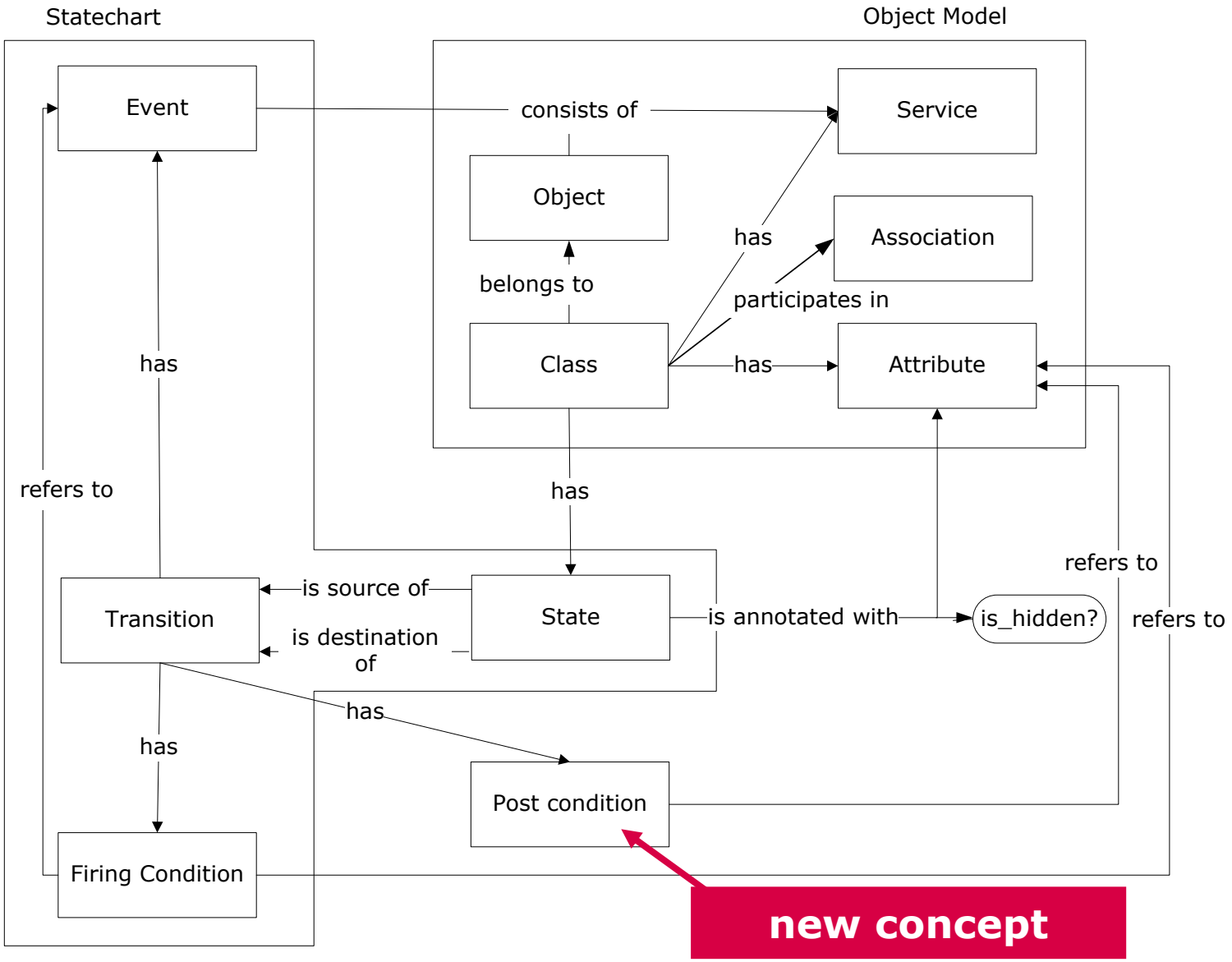
Assemble fragment: object chart



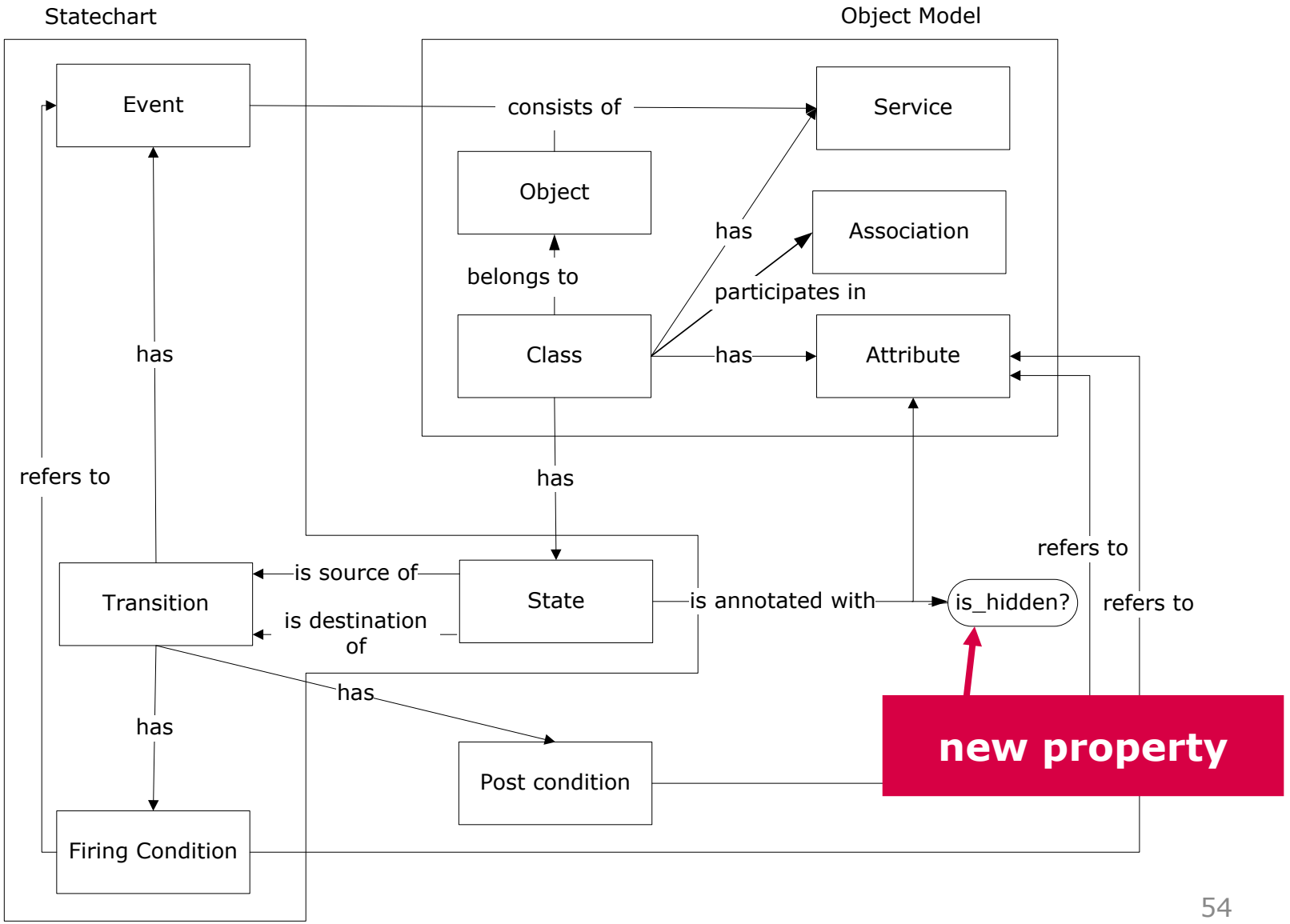
Meta-model of the resulting assembly



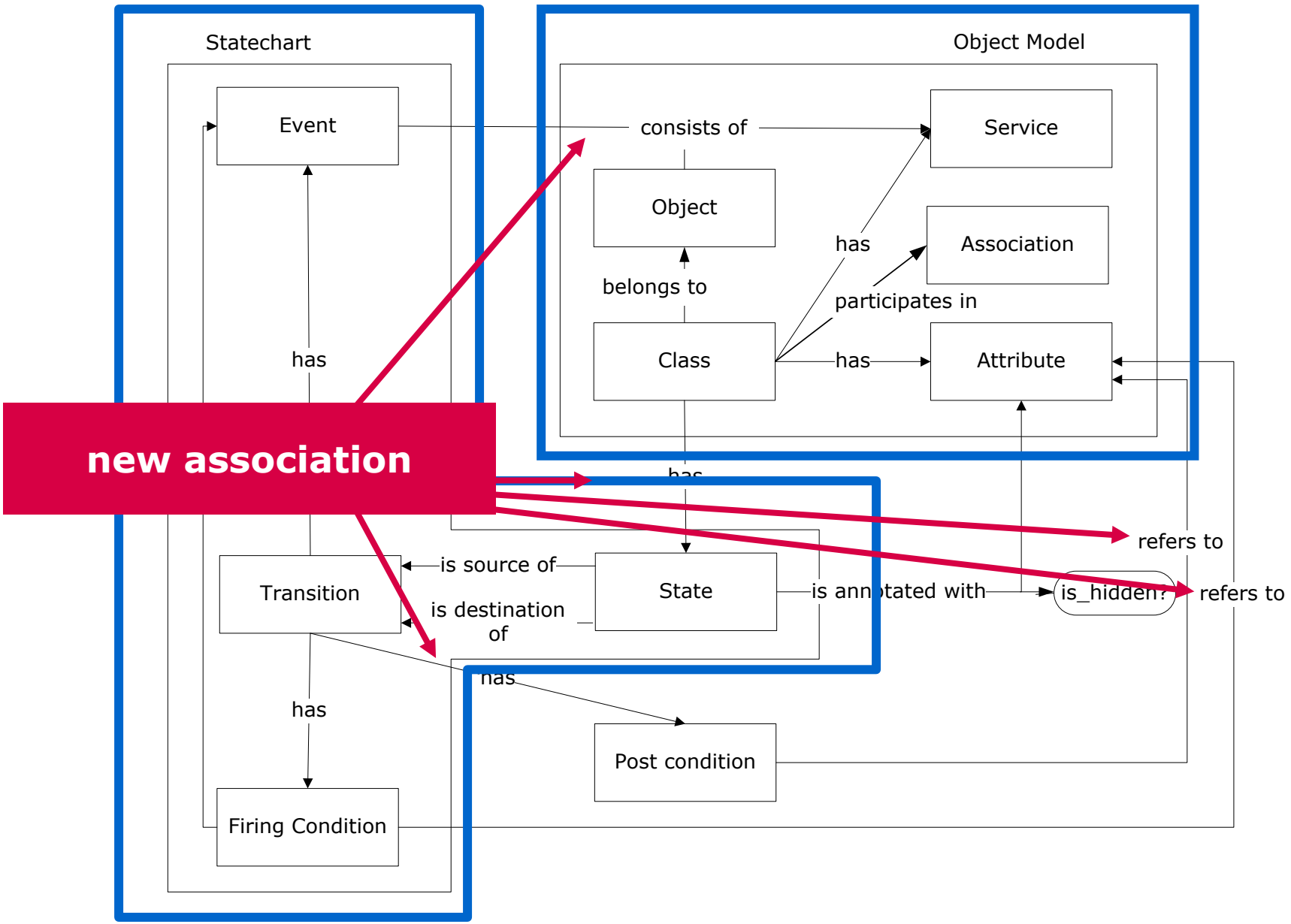
Inserted concept



Inserted property



Inserted associations

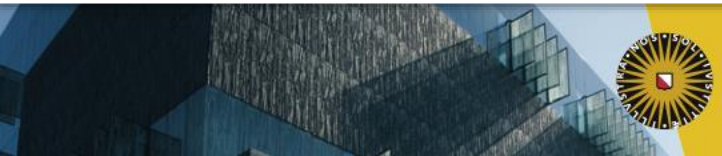


Method assembly in the process perspective

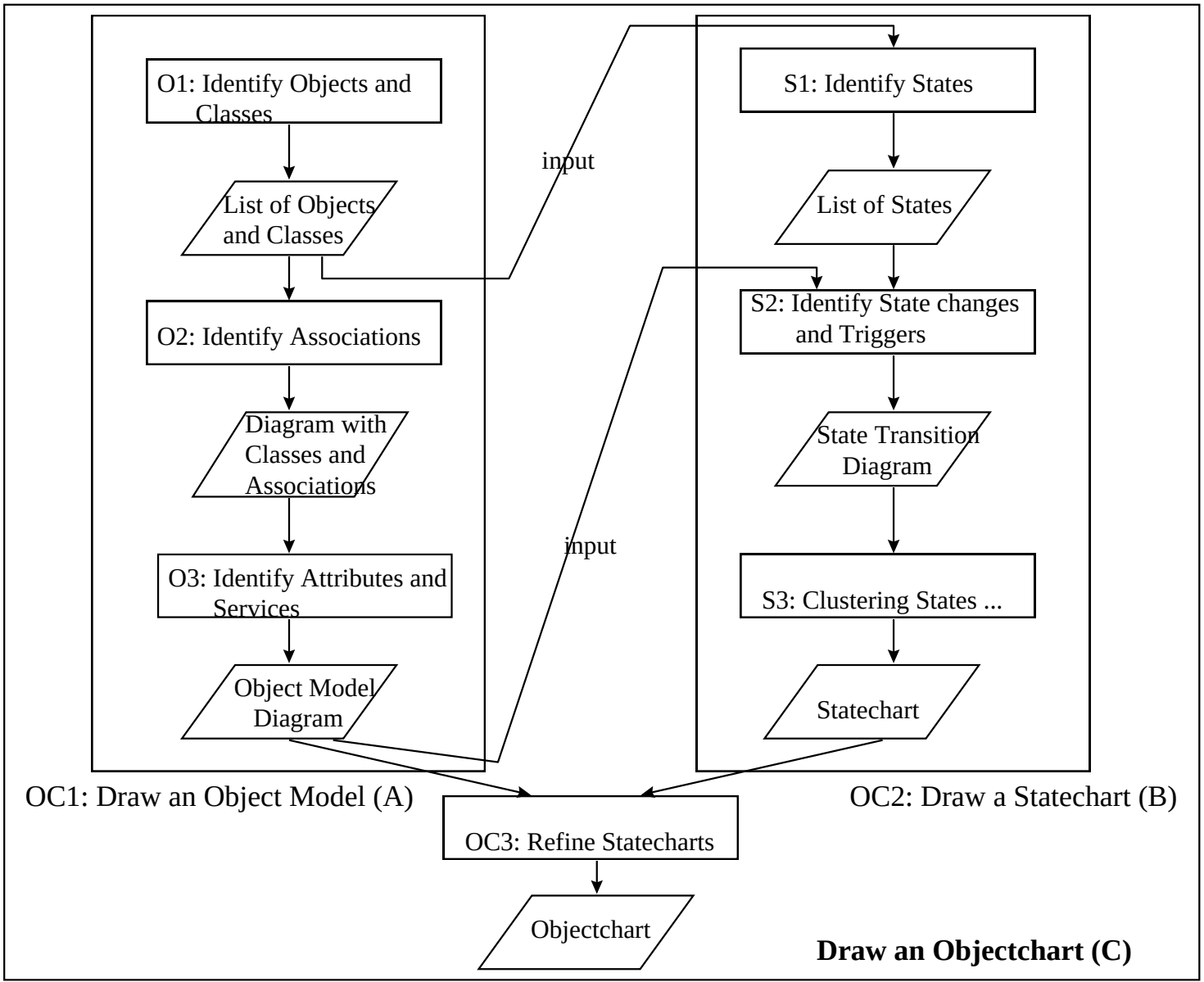
A. Draw an Object Model
   - Identify objects & classes
   - Identify relationships
   - Identify attributes and services

B. Draw a Statechart
   - Identify states
   - Identify state changes and their triggers
   - Cluster states, and so on

C. Draw an Objectchart
   - Draw an Object Model
   - For each significant class, Draw a Statechart
   - Refine statechart to an objectchart by adding conditions and states of the statechart with attributes



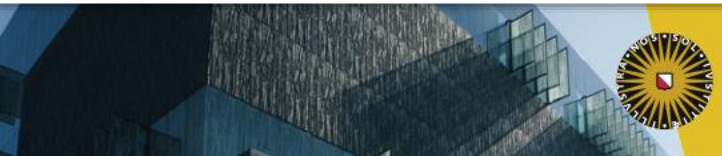
Process perspective



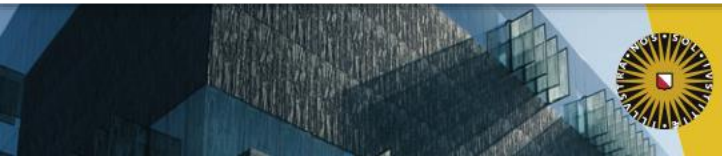
Concluding

- Meta-data modeling and meta-process modeling provide simple means for the documentation and communication of methodical processes and deliverables.

- Various applications of meta-modeling:
  - Situational methods and Method assembly (week 11)
  - Formalization of methods (week 11)
  - Method rationale (week 12)
  - Incremental method engineering (week 12)
  - Method association for product implementations (week 13)



QUESTIONS?



References

Kevin Vlaanderen, Fabiano Dalpiaz, Geurt van Tuijl, Sandor R. Spruit, Sjaak Brinkkemper (2014). Online Method Engine: A Toolset for Method Assessment, Improvement and Enactment. *Int Journal Information Systems Modeling and Design* 5(3): 1-25.

Brinkkemper, S., Saeki, M., & Harmsen, F. (1999). Meta-modelling based assembly techniques for situational method engineering. *Information Systems 24*(3), 209-228.

Harmsen, F., S. Brinkkemper, H. Oei, Situational Method Engineering for Information System Project Approaches. In: A.A. Verrijn Stuart and T.W. Olle (Eds.), Methods and Associated Tools for the Information Systems Life Cycle. Proceedings of the IFIP WG 8.1 Working Conference, Maastricht, Netherlands, September 1994, IFIP Transactions A-55, North-Holland, 1994, ISBN 0-444-82074-4, pp. 169-194.