Stijn Hoppenbrouwers

# Requirements Engineering, Lecture 2:

# Integral overview of key documentation items

Radboud University Nijmegen

## Key items

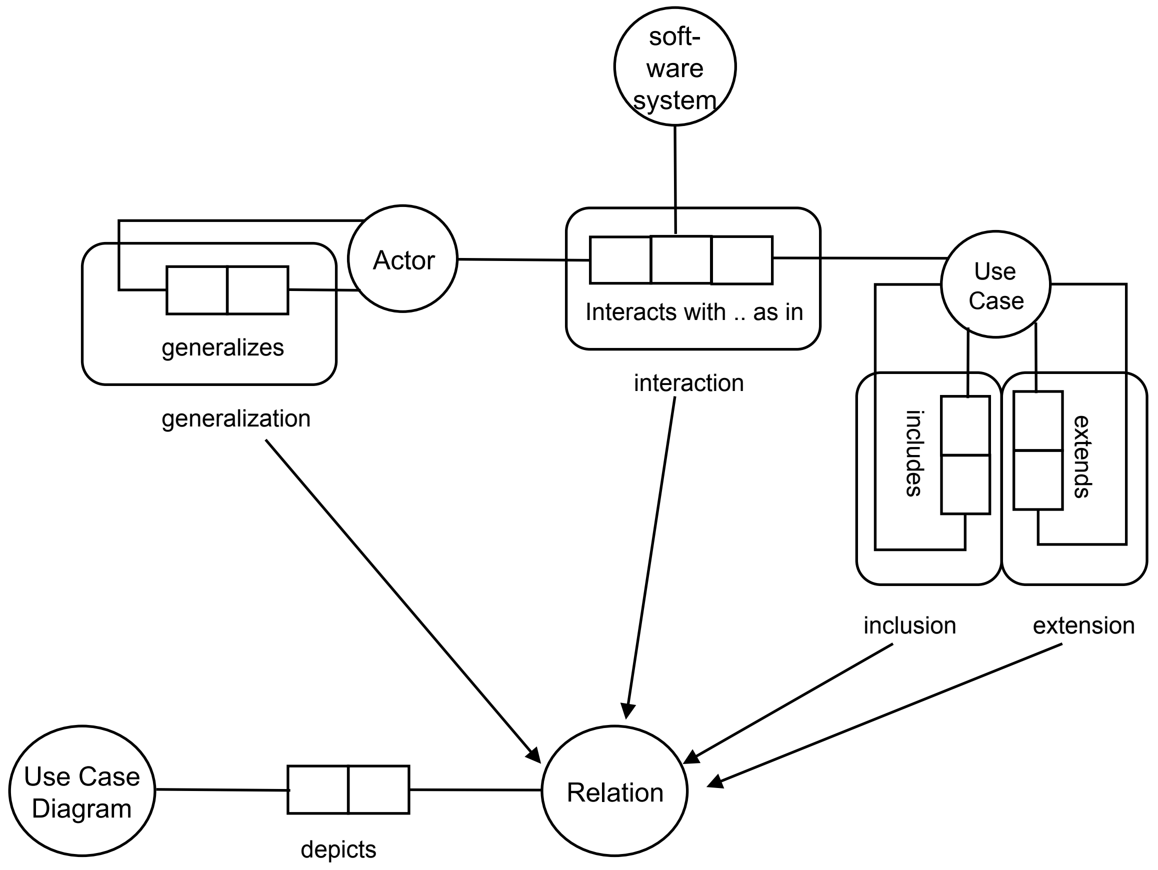
- Problem Statement
- Use Case Diagrams: Integrated
- Use Cases
- Use Case Survey
- Scenarios
- Domain Models (ORM): per UC (including example populations), and Integrated
- Business Rules: per UC, and Catalogue
- Terminological Definitions

## Problem statement

# Why?

Or: what’s the *problem*?

## Use Cases and Use Case Diagrams

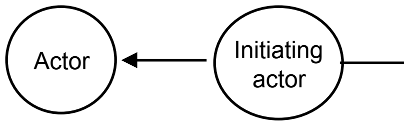


## UC Diagram: “Integrated”

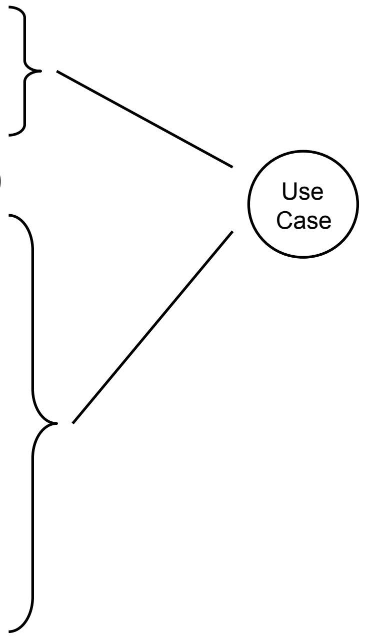
- Integrated UCD: comes with the UCS! It integrates all Ucs, actors, etc. in one diagram.

- Formerly, we also demanded a UCD for each individual UC. I have now dropped this requirement.

## Use Cases and the Use Case Survey



- Use case number
- Use case name
- Initiating actor (so not all related actors)
- Description (=summary)
- Completeness (is everything there?)
- Maturity (well thought over?)
- (Dependency)
- Source
- Comments



## Use Case: Template

- Use case name
- Iteration
- Description ( “summary”)
- Basic course of events
- Alternative paths (to avoid IF-THEN-ELSE bog)
- Exception paths
- (Extension points)
- Triggers (when or why does an actor enter the use case)
- Assumptions (ref. “non-formalized assumptions” in B&B)
- Preconditions (ref. “formalized system assumptions” in B&B)
- Postconditions (ref. “formalized system commitments” in B&B)
- Related business rules
- Author
- Dates

## Use Cases and Scenarios

- •Use case name
- •Iteration
- **•Description**
- **•<u>Basic course of events</u>**
- **•Alternative paths**
- **•Exception paths**
- **•Triggers**
- **•Assumptions**
- **•Preconditions**
- **•Postconditions**
- **•Related business rules**
- •Author
- •Dates

• Cover all relevant paths of BCoE etc. in scenario

• Do this at *instance level*, i.e. using concrete examples (creating an “example population”)

• Keep clear relation between the structure of the use case (steps) and its scenarios visible

• Keep terminology (type/instance level) of *entire* use case and scenario *<u>consistent</u>*

• Often (but not necessarily!), scenarios are developed *after* use cases.

## Scenarios as a Requirements Gathering Technique

- “Instance level” scenarios are useful but
- Sometimes, a less formal, broader for of scenario is also helpful:
- Asking a stakeholder to “just do a walk-through” of some task.
- This is more aimed at requirements gathering.
- Such scenarios you can use (not obligatory) and even include in the documentation, but please keep the well apart from the “test” scenarios.

## Use cases, Scenarios, and Domain Models

- Though [K&G] do not include them, it is often an excellent idea to use a Domain Modeling technique (ORM or perhaps ER or even UML class diagrams) of use cases. We use ORM.

- Mainly of BCoE (data models may be derived) but possibly also of other use case items. Note that if I say “BCoE” I *also* mean the related alt. and exc. paths.

- Include the following concepts:
  – All key concepts in the UC, capturing the UoD of the UC
  – In addition, concepts needed to define the business rules that apply to the UC.

## Terminological Definitions

- Traditional *Term Definitions* typically have three components:
- a *definiens* (that which is defined)*,*
- a *genus* (it's "supertype")
- and some *differentiae* (what distinguishes this subtype of other subtypes).
- For example: a *dog* (definiens) is an *animal* (genus) that *can bark* and *wags its tail a lot* (two differentiae).

## Use Cases, Domain Models, and Business Rules

- Business rules should be kept consistent with *all* documentation items mentioned
- There is a particularly strong relation between Domain Models and Business Rules
- Many business rules can be seen as “complex constraints on domain models”
- ORM is a technique used in the international Business Rule community (and also used for data modeling)
- If you use ORM for basic business rule specification or even just for domain modeling, you are at least half way in *formalizing* the business rules: it is a form of logic!

## Domain Models, ORM, and Term Definitions

- In ORM, providing instance level examples (example population) is recommended, and often even part of the elicitation process. The population examples should correspond to what instance-level concepts are used in the scenarios.
- Note that (informal) term *definitions* can be created either in parallel to ORM specification or even apart from ORM. ORM does not include them as such! (fact type definition is *not* concept definition)
- Business Rules Mantra: Ruls consist of Facts, Facts consist of Terms.
- Rules = Logic; Facts = ORM; Terms = TDs.

## Some useful Links and References

- ORM
  - Domain Modeling course in 1st year
  - Book Terry Halpin (2001): Information Modeling and Relational Databases
  - www.orm.net

- Business Rules
  - Business Rule Manifesto (see RE website)
  - www.orm.net
  - www.brcommunity.com

## Informal en Formal specification

- There is a gray area between “informal” and “formal” texts. In this RE course, in principle we work with “informal” specifications. However, if you create a well-structured and above all conceptually and terminologically *consistent* informal set of deliverables, you are well on the way to formalizing it.
- The standards for coherence and consistency for documentation in RE are not unlike similar demands in the Beweren & Bewijzen course.
- An easy way to score an **insufficient mark** in the Case Project is to deliver messy, inconsistent requirements documentation.
- Business rules can have several degrees of formalization, from wholly informal, to just some formal elements, to all-formal.

## Additional Documentation Items

- Introduction
- Stakeholders list/analysis/demography
- Mission – vision – (values): why-what-how of the *project* (so not of the *system!!*)
- Non-functional requirements (*not* use cases)
- Statement of work: work plan (p57 [K&G])
- Risk analysis (*project* risks!)

- Business process definitions: optional appendix
- GUI metaphors / storyboards: optional appendix

- (Prototype (*not*) )

## Excercise

- Provide scenarios for the Use Case we worked out last time (“item uitlenen”)
- Don’t worry about the exact UC too much, just provide a 2-3 good scenarios. It’s the form that counts most now, not the content.
- Also try to write a scenario from scratch and then create a UC from it
- Also try out how a “Use Case Test” works (p 102 K&G)