Stijn Hoppenbrouwers

Requirements Engineering, Lecture 3:

Three Phase Approach

Radboud University Nijmegen

Three Phases

1. Facade – outline and high-level descriptions.

2. Filled – broadening and deepening.

3. Focused – narrowing and pruning.

We shouldn't be too rigid about sticking to a specific phase
for a specific use case,
even though we organize the case in "three phases"

Kulak & Guiney

- Read the book!
- The book provides more info than this lecture, and you *do have to know what's in the book*, for case project *and* exam!
- For deliverables, look at *my* Deliverables Table and *stick to that*. In this respect [K&G] are confusing.
- Also, because we do not actually realize a system in the case project, our case approach is a bit different.
- For the spirit of the (difference between) the three iterations, [K&G] are excellent, and also for lots of good tips, tricks, and examples.

Guiding Principles

- Reduce risk
- Focus on business interactions
- Reduce volume
- Reduce duplicates and inconsistencies
- Create requirements users understand easily
- Create requirements that are useful for designers, developers, and project managers
- Leave a requirements trail
- Leave (technical) design for later
- Keep long-term plan in mind

Facade

- create placeholders for each major interaction of an actor with application.
- use cases: minimum of info, as placeholder.
- identify initiating actor + other actors.

Steps towards Façade Use Cases

1. Create a problem statement.
2. Identify and review existing documentation.
3. Get Exec. Sponsor viewpoint.
4. Review business process definitions (if available).
5. Identify users, user group management, stakeholders, customers, owners of data.
6. Interview stakeholders.

Rest of the way

- Create stakeholders list.
- Find the actors.
- Create use case survey (list of Facade use Cases).
- Start non-functional req. survey.

7. Start business rules catalog: try and find *existing* (explicit) business rules if any are around at all

Rest of the way

9. Create a risk analysis.
10. Create a statement of work.
11. (Begin looking at GUI metaphors).
12. (Begin GUI story boards).
13. Get informal approval from executive sponsor (= evaluation after deadline).

Filled iteration

- Break out detailed use cases
- Provide alternative solutions
- Create Filled use cases
- Add domain models
- Add terminological definitions
- Add business rules (i.e. *create* them where relevant).
- Test Filled use cases.
- **Include various options** if relevant
- Put some things off (really? ... yes really!)

Breaking out

- Some Facade use cases may be too general.
- Need to be broken down into several Filled use cases (use extensions/inclusions, but only if needed).
- Not always a one-to-one relationship between Facade -> Filled use cases.
- What size should a use case be?

Does size matter?

- This does not refer to amount of documentation supporting use case.
- Refers to how you define boundaries between system functionality.
- How do we split system functionality into use cases?
- How many use cases needed to describe system?
- Loosely use the 7 +/- 2 rule here as a guideline for how many elements you want to pack into one “component” of your decomposition.
- Make sure your decomposition remains **transparent**!

Steps towards Filled Use Cases

1. Identify triggers.
2. Identify preconditions.
3. Refine the use case name.
4. Refine the actors.
5. Specify basic course of events.
6. Indicate repetition.
7. Document alternatives and exceptions.
8. Start with domain models
9. Refine use of language; write term definitions
10. Create (sketches of) business rules

Put some things off...

- Filled iteration is time to expand detail of documented requirements.
- Not time to increase/decrease scope dramatically.
- Wait on identifying common behaviour and merging use cases.
- Wait on handling detailed exceptions.
- Keep gathering what, and not how!!!

Focused iteration

- You have a collection of use case documents.
- Filled iteration provided large amount of requirements.
- Focused iteration is time to select the best options identified in Filled iteration.
- Include only these best options in the project scope.
- Clears the path through paperwork and leaves clear project requirements (we hope!).

Steps towards Focused Use Cases

1. Merge duplicate processes.
2. Bring focus to each use case.
3. Manage scope changes during this iteration.
4. Manage risks and assumptions.
5. Review.

Merging steps

1. Identify duplicates.
2. Transform duplicate piece into its own use case.
3. Update the rest of the documentation!

- merging of duplicates critical at this point
- causes serious problems when beginning to implement code
- complicates maintenance, single change means several processes to update...

Merging paybacks

Detection/merging duplicate processes enhances ability to perform number of tasks:

- choosing better candidate use cases
- simplifying application documentation
- prioritizing use cases
- creating accurate estimates (possible reuse use cases elsewhere)
- improving development process
- planning project iterations

Review

- Reviews are essential in this iteration!
- Stakeholders, technical architects, business domain experts.
- Look at use cases individually and as part of system.
- Review for accuracy and completeness.
- Collection use cases must describe an adequate system.
- If you are reviewing the Focused iteration, what are you looking for?

Phased Deliverable Table (*my version*)

Introduction	Contextual	Preliminary version	Preliminary version	Complete	Keep it short at first, only a sketch; nice, clear and complete at the end.
Problem statement	**Key deliverable**	As good as possible	As good as possible	Complete	
Stakeholder list/analysis	Contextual	As good as possible	As good as possible	Complete	
Mission-Vision-(Values)	Contextual	Complete	Complete	Complete	
Statement of Work	Contextual	Complete, and *up-to-date*	Complete, and *up-to-date*	Complete, and *up-to-date*	
Risk Analysis	Contextual	Complete, and *up-to-date*	Complete, and *up-to-date*	Complete, and *up-to-date*	
Use Case Survey	**Key deliverable**	As good as possible	Nearly complete	Complete	
Integrated UC Diagram	**Key deliverable**	Complete (though preliminary)	Complete	Complete	One for whole project
Use Cases	**Key deliverable**	Not yet!	"Filled" level	Complete	
Scenarios	**Key deliverable**	Not yet!	Several for each UC	Complete ("focused" level)	
Domain Models	**Key deliverable**	Not yet!	Partially complete	Complete	One for each UC
Business rules per UC	**Key Deliverable**	Not yet!	Partially complete	Complete	
Integrated Domain Model	**Key deliverable**	Not yet!	First draft	Complete	One for whole project
Busines Rules Catalogue	**Key deliverable**	Not yet!	Partially complete	Complete	
Non-functional Requirements	**Key deliverable**	Notes	Partially complete	Complete	
Terminological Definitions	**Key deliverable**	Notes	Partially complete	Complete	
Executive sponsor viewpoint	Implicit deliverable	Complete	Complete	Complete	Integrated in M-V-(V)!
Use case tests	Implicit deliverable	Notes	As good as possible	Complete	Integrated in scenarios; to be done, but not written down explicitly as a separate item
Busienss process definitions	Optional appendix	If available / relevant	If relevant	If relevant	Use existing definitions/ models or make them yourself: optional!
GUI metaphors / storyboards	Optional appendix	If relevant	If relevant	If relevant	

link

Excercise: Domain Models

- Create an ORM model for:
  - The concepts (information) used in *interaction* in the “Borrow Item” Use Case as discussed previously. Again, it’s the form that counts most here.
  - Include in that ORM model the concepts needed to capture the following business rules:

1. At any time, no more than fifteen items may be borrowed by a library member
2. If the maximum number of 5 prolongations [verlengingen] of the loan of an item is reached, no more items will be loaned to the library member concerned