

Announcements

Please hand in HW

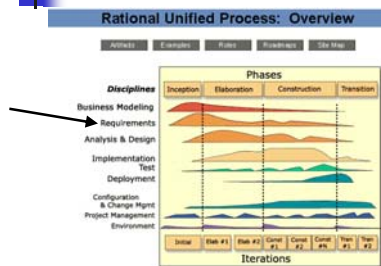
Pick up old HW

Grades are too low – everyone gets 10 free points!

Introduction to Elaboration

- Reference: Chapters 8-9
- As context, recall key UP ideas:
 - The phases
 - The iterations
 - The disciplines
- (See Figure 2.3, and one like 2.4 next)

Disciplines in the RUP (not quite fig. 2.4)



Elaboration Vs. Inception

- Inception lays the groundwork for the project
- 1st inception iteration: somewhat unique because
 - typically there is only 1 iteration in inception phase
 - some people don't even call it an iteration
 - we are starting from zero
 - (there is no preexisting project state to refine)
 - hence the concept of incremental change doesn't apply yet
- Even in software loops, the first iteration is different!
 - If all iterations were equal, the 11th iteration would make what 10% change in project status?
 - ...the 1st iteration would make what % change?

Elaboration Vs. Inception II

- Iterations in the elaboration phase
 - Gradually start getting more routine
 - Why, mathematically speaking?
 - Exercise: given routineness is from 0 to 1, define a routineness function and sketch its qualitative shape (no calculating points!)
 - But when they finally get routine it's no longer *elaboration* (it's *construction* – see Figure)
- Product implementation begins in the elaboration phase
 - Rapid prototypes could be written in inception
 - ...but they are not part of the final product
- Key UP idea: after each elaboration iteration
 - ...there is a working system

More on Elaboration Phase, First Iteration

- Start programming
- Start with building risky parts of the system (why?)
- Build core architecture
 - This involves implementing various hooks (stubs & drivers – what are they?)
 - Note the connection to the top-down vs. bottom-up implementation issue
 - Which are among those most thoroughly tested by use during iterative development?
 - Upper-level, mid-level, lower-level, utilities (at what level are utilities?)
- Continue to develop requirements?
- Plan the next iteration (see next slide)

Inception: Plan the First Elaboration Iteration *First Elaboration Iteration: Plan the Second*

- Each iteration involves planning the next one
 - Does it involve anything else?
 - A key part of a plan is what to build
- How to figure out what to build next:
 - Three factors to weigh
 - Risk – riskiest things should be done early (why?)
 - Criticality – most important things should be done early (why?)
 - Coverage – build entire core architecture in early iterations (why?)
 - Recall temperate fruit committee example
 - Better to do one kind of fruit at a time, or gradually add functionality to all at once?
- See Figure p. 111
- Should the list be revised in future iterations?
- Exercise: do a similar table for the Use Case Diagram Assistant

Planning For the First Elaboration Iteration: POS Example

- Build “basic, key scenario”
 - Ring up items, process payment in cash
 - How does this rate in terms of risk, criticality and coverage?
- This is not “fancy or complex” – it is a “happy path scenario”
 - Why?
 - Does each iteration complete a use case?

Some Elaboration Questions

1. How changeable are the artifacts produced by Inception later in elaboration?
2. Does programming begin in elaboration?
3. What parts of the system should be designed and implemented in early elaboration iterations?
(risky parts? complex parts? Exception handling? main sequences of events? Least important parts? Most interrelated parts? Full use cases?)
4. Should testing be done during early elaboration iterations?
5. Is the data structure for recording the set of temperate fruits part of the core architecture?
6. Can software interfaces (method calls, argument lists, object identities, etc.) be written without making working functionalities?

Elaboration Questions II

7. Should each iteration produce a more functional system?
8. Should use case boundaries and iteration boundaries coincide?
9. Is planning done during each iteration? If so, for what?
10. Should risky things be built early in elaboration, or later?
11. What is a risky aspect of the Use Case Assistant system?
12. Should mission-critical functions be built early, or later?
13. Is building alternate scenarios, or different main sequences, more urgent for early implementation?
14. Should feature priority ranks be reviewed during each iteration?
15. What is one high and one low ranking feature for the Use Case Assistant system? Why?

Elaboration Iteration 1 (and up)

- Specific skills are useful
- For example: **System Sequence Diagrams (SSDs)**
- The language for making SSDs is the **UML sequence diagram**
 - UML sequence diagrams can also be used for other things
 - Hence the two terms are not quite interchangeable
- See Figure 9.1
- Make a SSD for each main success scenario
- Make a SSD for each important or complicated alternative scenario
- Maybe we should try something as an exercise

More on SSDs

- Does a SSD include things from multiple use cases?
- Does a use case require multiple SSDs?
- Is there a one-to-one mapping between use cases and SSDs?
- Does an SSD express flow of time?

SSDs III

- An SSD expresses a use case scenario
- To build an SSD, look at the steps of the scenario
- (See Figure 9.2 in book; at this moment, see Figure 9.1)
- Where is the system boundary in Figure 9.1?
- Note the black box nature of the SSD:
 - Events cross system boundaries
 - Events within the system are invisible

SSDs – Event names

- “enterItem” is better than “scanBarcode”
 - Why?
- Verbs are good for starting event names
 - Why?
- What’s better, enterItem or itemEntry?
 - Why?

Why Discuss SSDs Now?

- “Most SSDs are created during elaboration”
 - Why? See Figure again

Disciplines in the RUP (not quite fig. 2.4)

