System Analysis and Design

L03T. How to do Iterative and Evolutionary Analysis and Design?

定到开发实例（一个例子）

Sample Development Case

Discipline	Practice	Artifact	Incep.	Elab.	Const.	Trans.
		Iteration	**I1**	**E1..En**	**C1..Cn**	**T1..T2**
Business Modeling	**agile modeling req. workshop**	**Domain Model**		start		
Requirements	**req. workshop vision box exercise dot voting**	**Use-Case Model**	s	refine		
		Vision	s	r		
		Supplementary Specification	s	r		
		Glossary	s	r		
Design	**agile modeling test-driven dev.**	**Design Model**		s	r	
		SW Architecture Document		s		
		Data Model		s	r	
Implementation	**test-driven dev. pair programming continuous integration coding standards**	**…**				
Project Management	**agile PM daily Scrum meeting**	**…**				
…						

How to do Iterative and Evolutionary Analysis and Design?

- Here's a short *example* of how it can work on a well-run UP project.
- This assumes there will ultimately be 20 iterations on the project before delivery.

Two day requirements workshop

(UP/Rio Inception phase)

- First half day (before iteration-1):
  - do high-level requirements analysis, such as identifying just the names of the use cases and features, and key non-functional requirements.
  - pick 10% from this high-level list (such as 10% of the 30 use case names) that have a blending of these three qualities:
    1. architecturally significant (if implemented, we are forced to design, build, and test the core architecture)
    2. high business value (features business really cares about)
    3. high risk (such as "be able to handle 500 concurrent transactions").
  - Perhaps three use cases are thus identified: UC2, UC11, UC14.

Two day requirements workshop

- Remaining 1.5 days (before iteration-1)
  - do intensive detailed analysis of the functional and non-functional requirements for these three use cases.
  - When finished, 10% are deeply analyzed, and 90% are only high-level.
- An iteration planning meeting (before iteration-1)
  - a subset from UC2, UC11, and UC14 are chosen to design, build, and test within a specified time
  - After choosing the specific subset goals, break them down into a set of more detailed iteration tasks.

Iteration-1
over three or four weeks

- First two days
  - do modeling and design work in pairs, sketching UML diagrams, guided by the chief architect. 建模与设计
- The remaining weeks
  - start programming, testing, and integrating continuously, using the modeling sketches as a starting point of inspiration, knowing that the models are partial and often vague. 编码,测试,集成
- Testing: unit, acceptance, load, usability, and so forth. 可能有各种测试

The last week of Iteration-1

- One week before the end, ask the team if the original iteration goals can be met; if not, de-scope the iteration, putting secondary goals back on the "to do" list.
- On Tuesday of the last week there's a code freeze; all code must be checked in, integrated, and tested to create the iteration baseline.
- On Wednesday morning, demo the partial system to external stakeholders, to show early visible progress. Feedback is requested.

The last week of Iteration-1

- **Second requirements workshop** on the last Wednesday and Thursday
  - Review and refine all the material from the last workshop.
  - Then pick another 10% or 15% of the use cases that are architecturally significant and of high business value, and analyze them in detail for one or two days.
  - When finished, perhaps 25% of the use cases and non-functional requirements will be written in detail. They won't be perfect.

The last week of Iteration-1

- On Friday morning, hold another **iteration planning meeting for the next iteration**.

iteration-2

- Do similar steps.

Repeat for four iterations

- Repeat for four iterations and five requirements workshops
- so that at the end of iteration-4, perhaps 80% or 90% of the requirements have been written in detail, but only 10% of the system has been implemented.
- Note that this large, detailed set of requirements is based on feedback and evolution, and is thus of much higher quality than purely speculative waterfall specifications.

The end of the **elaboration phase**

(UP 的第2个 phase)

- We are perhaps only 20% into the duration of the overall project.
- In UP terms, this is the end of the **elaboration phase**.
- At this point, estimate in detail the effort and time for the refined, high-quality requirements.
- Because of the significant realistic investigation, feedback, and early programming and testing, the estimates of what can be done and how long it will take are much more reliable.
- After this point, requirements workshops are unlikely; the requirements are stabilized though never completely frozen.

Continue in a series of three-week iterations

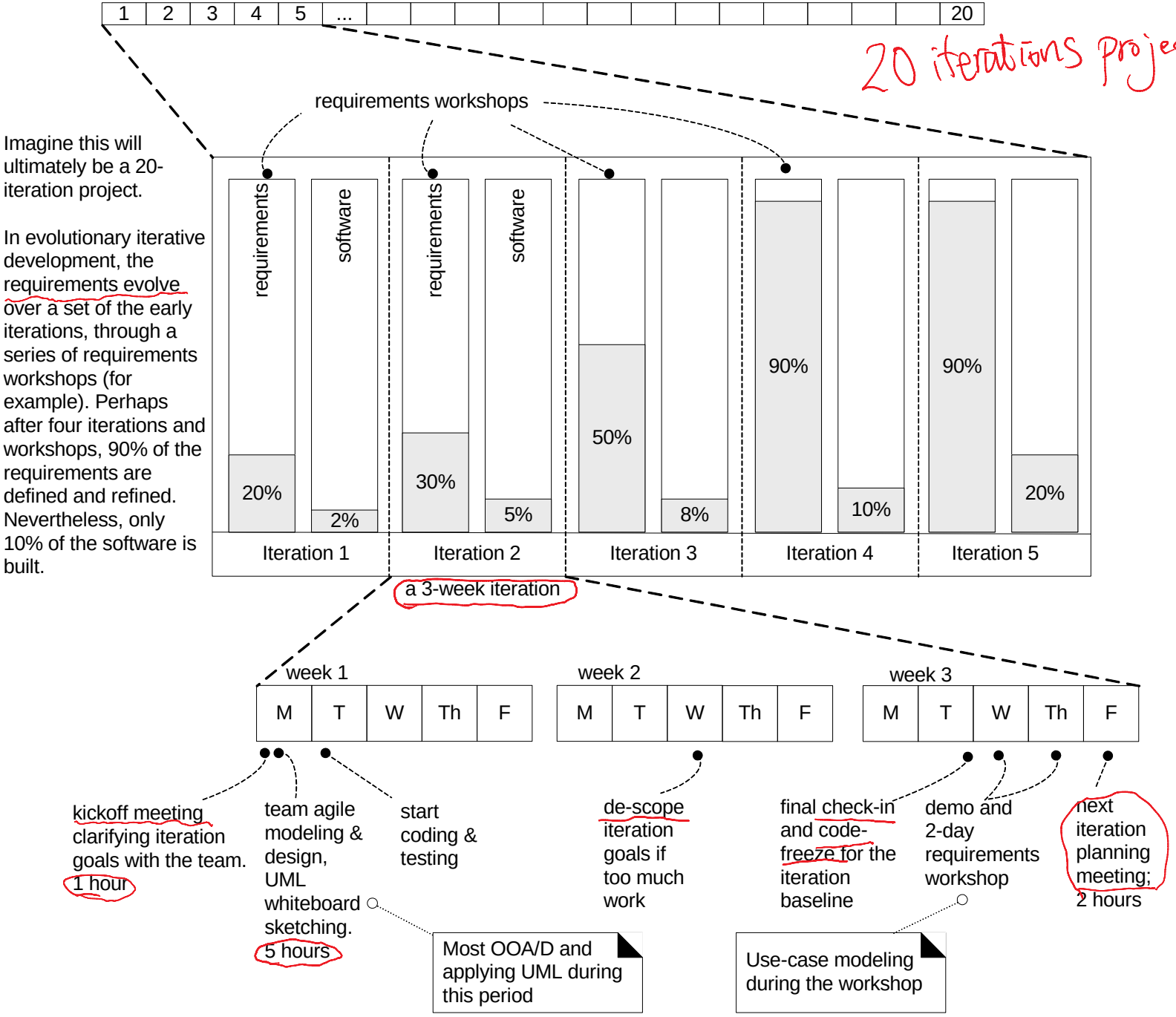
- Continue in a series of three-week iterations
- choosing the next step of work adaptively in each iteration planning meeting on the final Friday
- re-asking the question each iteration, "Given what we know today, what are the most critical technical and business features we should do in the next three weeks?“

The benefit of such an iterative and evolutionary development

- In this way, after a few iterations of early exploratory development, there comes a point when the *team can more reliably answer "what, how much, when."*

Evolutionary analysis and design

the majority in early iterations



Development Teams

- Development Environment: Coding (Implementation)
- UML Modeling:
- Cmake
- Git

Mesh

- 1D,2D,3D
- Different Elements Selection:
  - 2D**: Tri**angle, **quad**rilateral
  - 3D: **Tet**rahedron, **hex**ahedron, **Pyramid**, **Wedge**
- structural, unstructured
- Partitioned or not

Mesh Storage

- Storage in Text, binary File ,
- Single file or multiple files
- plain file or structural file like hdf5
- storage in Memory
- Different File Storage formats
- Different memory storage formats

Mesh processing

- sequential or parallel processing
- Editing
- Load in and Output
- Coarsening
- sketching
- partitioning
- Viewing (text)
- Visualization

Computing Over Mesh

- Kernel functions on mesh elements
- Algorithm over all mesh
- sequential or parallel Computing
- Partition Topology matches with architectural topology