Iterative, Evolutionary, and Agile

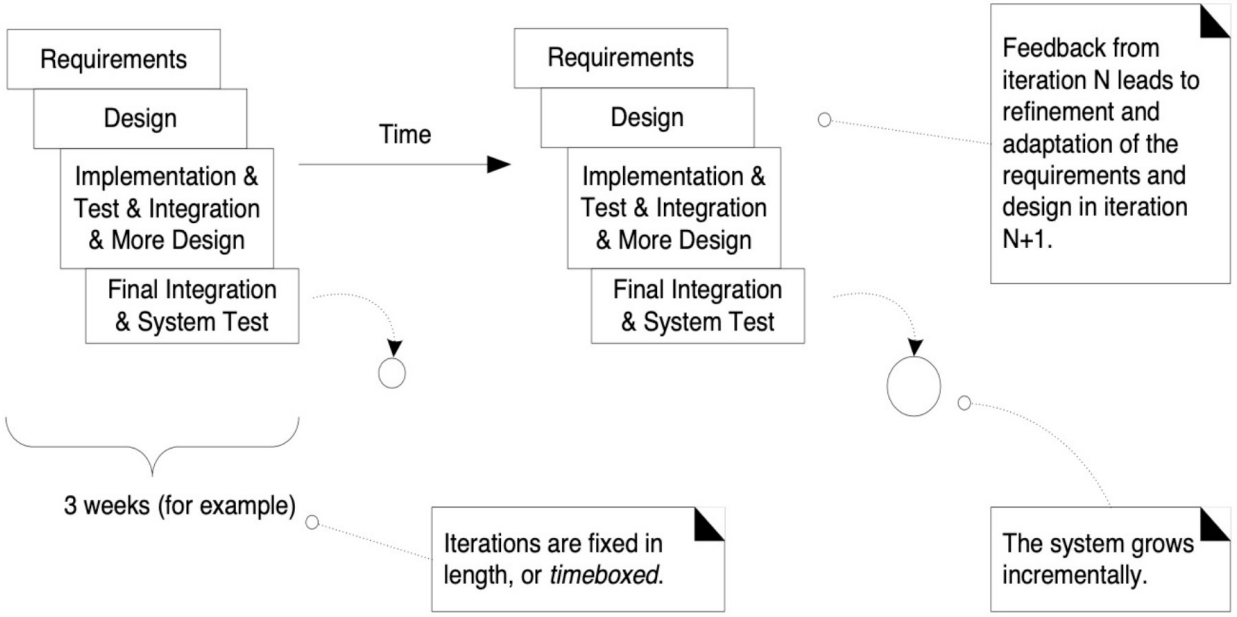
A very brief introduction to Iterative Development

Larman, Ch2

Iterative, Evolutionary, and Agile

You should use iterative development only on projects that you want to succeed. Martin Fowler

A **software development process** describes an approach to building, deploying, and possibly maintaining software.



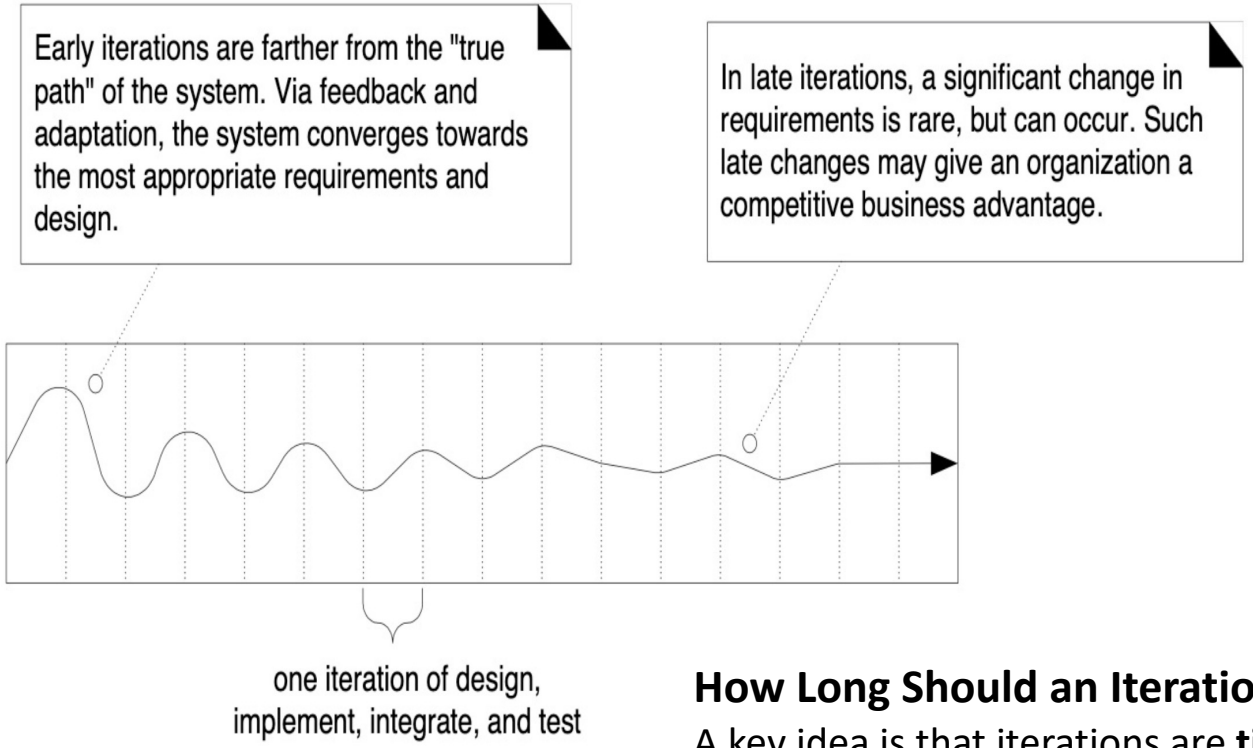
Development is organized into a series of short, fixed-length (for example, three-week) mini-projects called **iterations**;

The outcome of each is a tested, integrated, and executable *partial* system.

Each iteration includes its own requirements analysis, design, implementation, and testing activities.

Iterative feedback and evolution leads towards the desired system.

The requirements and design instability lowers over time.



Benefits to Iterative Development

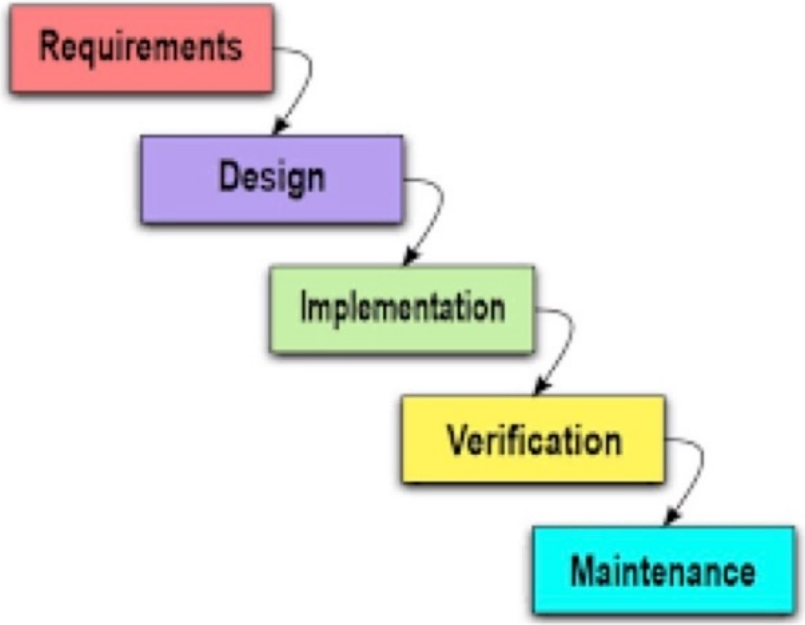
- less project failure, better productivity, and lower defect rates;
- early rather than late mitigation of high risks (technical, requirements, objectives, usability, and so forth)
- early visible progress
- early feedback, user engagement, and adaptation, leading to a refined system, closely meets the real needs of the stakeholders
- managed complexity;
- the team not overwhelmed by "analysis paralysis"

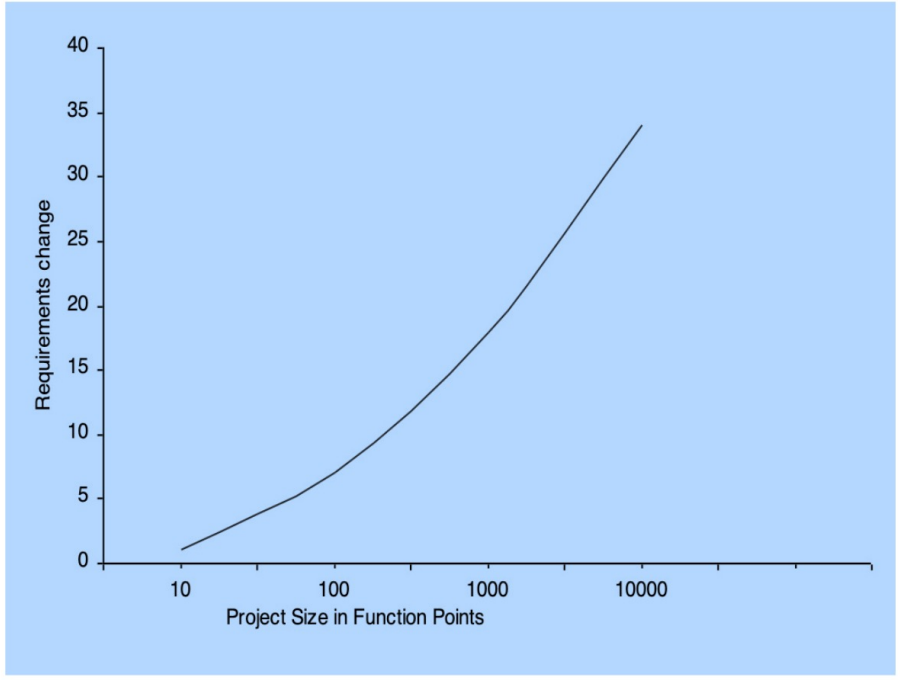
How Long Should an Iteration Be? What is Iteration Timeboxing?

A key idea is that iterations are **timeboxed**, or fixed in length.
Most iterative methods recommend an iteration length between two and six weeks.
Short is good.

What About the Waterfall Lifecycle?

- Sequential) lifecycle process
- Attempt to define (in detail) all or most of the requirements before programming.
- And often, to create a thorough design (or set of smodels) before programming.
- Research shows conclusively that the waterfall waterfall is poor practice for most software projects, rather than a skillful approach.

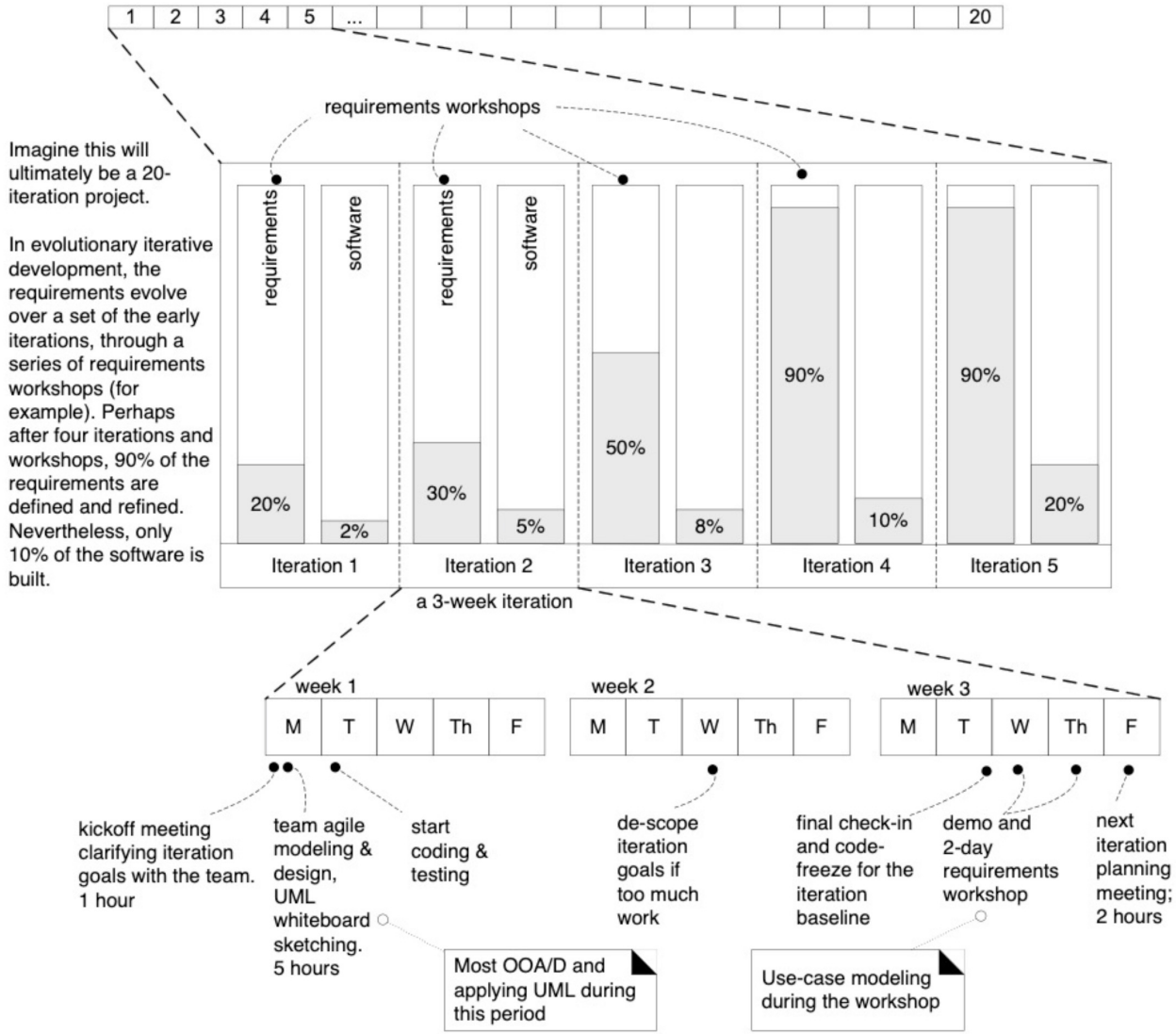




Guideline: Don't Let Waterfall Thinking Invade an Iterative or UP Project

"waterfall thinking" often incorrectly still invades a so-called iterative project.

Ideas such as "let's write all the use cases before starting to program" or "let's do many detailed OO models in UML before starting to program" are examples of unhealthy waterfall thinking


1
2
3
4
5
...
20
requirements workshops
Imagine this will ultimately be a 20-iteration project.
In evolutionary iterative development, the requirements evolve over a set of the early iterations, through a series of requirements workshops (for example). Perhaps after four iterations and workshops, 90% of the requirements are defined and refined. Nevertheless, only 10% of the software is built.
requirements
software
20%
2%
30%
5%
50%
8%
90%
10%
90%
20%
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
a 3-week iteration
week 1
week 2
week 3
M
T
W
Th
F
kickoff meeting clarifying iteration goals with the team. 1 hour
team agile modeling & design, UML whiteboard sketching. 5 hours
start coding & testing
Most OOA/D and applying UML during this period
de-scope iteration goals if too much work
final check-in and code-freeze for the iteration baseline
demo and 2-day requirements workshop
Use-case modeling during the workshop
next iteration planning meeting; 2 hours

Agile Methods and Attitudes

Agile development methods

- apply timeboxed iterative and evolutionary development,
- employ adaptive planning, promote incremental delivery, encourage *agility* rapid and flexible response to change.
- Example practices from the **Scrum** agile method include a ***common project workroom*** and *self-organizing teams* that coordinate through a daily stand-up meeting with four special questions each member answers. Example practices from the **Extreme Programming (XP)** method include *programming in pairs* and **test-driven development**.

The Agile Manifesto and Principles

The Agile Manifesto

Individuals and interactions *over processes and tools*
Working software *over comprehensive documentation*
Customer collaboration *over contract negotiation*
Responding to change *over following a plan*

The Agile Principles

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter time scale.

4. Business people and developers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

7. Working software is the primary measure of progress.

8. Agile processes promote sustainable development.

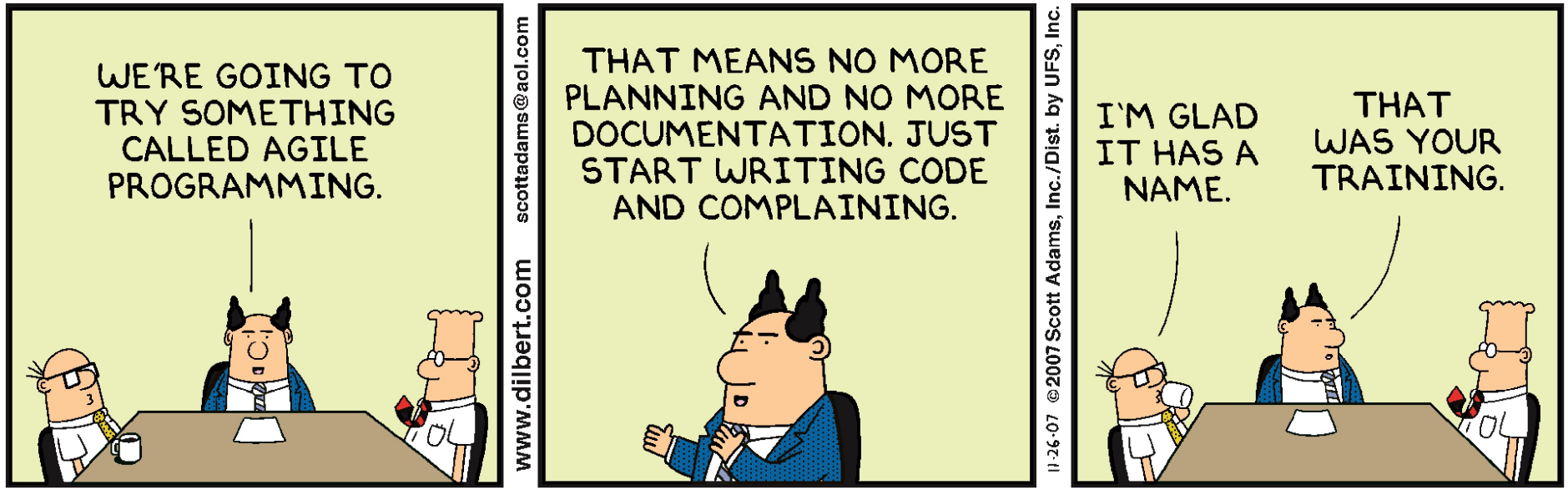
9. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

10. Continuous attention to technical excellence and good design enhances agility.

11. Simplicitythe art of maximizing the amount of work not doneis essential.

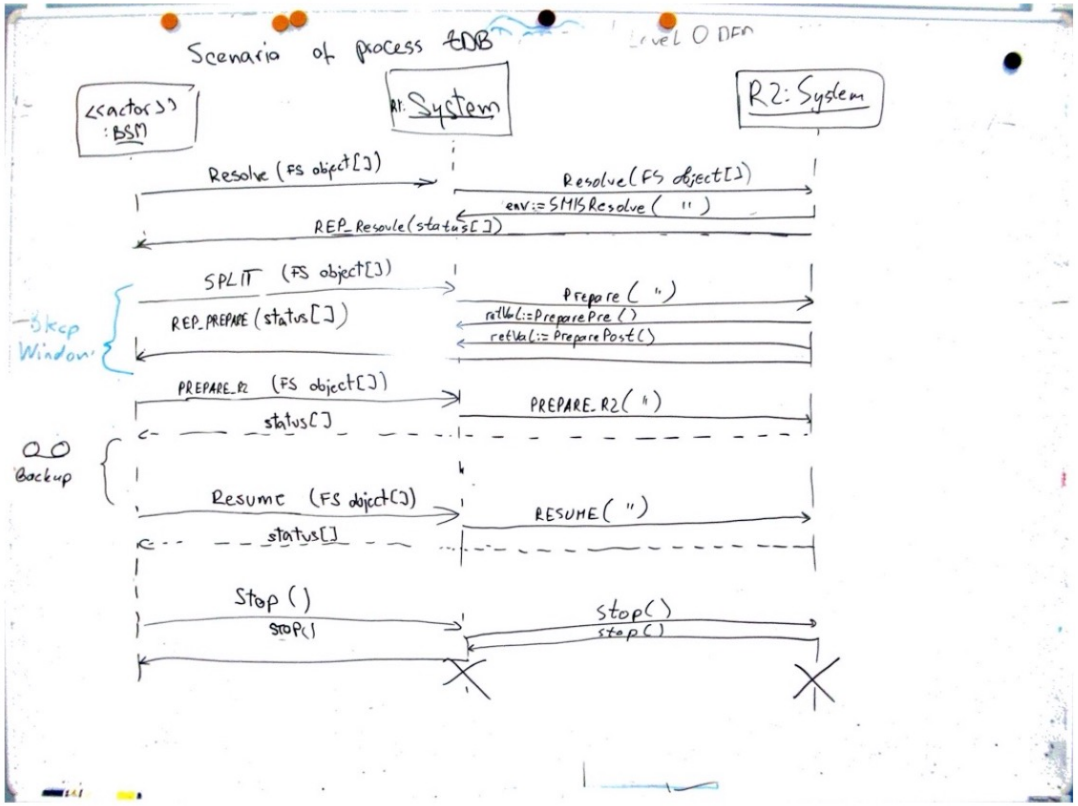
12. The best architectures, requirements, and designs emerge from self-organizing teams.

13. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.


WE'RE GOING TO TRY SOMETHING CALLED AGILE PROGRAMMING.
THAT MEANS NO MORE PLANNING AND NO MORE DOCUMENTATION. JUST START WRITING CODE AND COMPLAINING.
I'M GLAD IT HAS A NAME.
THAT WAS YOUR TRAINING.
www.dilbert.com scottadams@aol.com
11-26-07 ©2007 Scott Adams, Inc./Dist. by UFS, Inc.

Agile Modeling

The purpose of modeling (sketching UML, ...) is primarily to *understand*, not to document.



- **Don't model or apply the UML to all software design**. Defer simple or straightforward design problems until programming solve them while programming and testing. Model and apply the UML for the smaller percentage **of unusual, difficult, tricky parts of the design space**.
- **Use the simplest tool possible**. For example, prefer sketching UML on whiteboards, and capturing the diagrams with a digital camera.[2]
- **Don't model alone**, model in pairs (or triads) at the whiteboard, in the awareness that the purpose of modeling is to discover, understand, and share that understanding. Rotate the pen sketching across the members so that all participate.
- **Create models in parallel.** For example, on one whiteboard start sketching a dynamic-view UML interaction diagram, and on another whiteboard, start sketching the complementary static-view UML class diagram. Develop the two models (two views) together, switching back and forth.
- Use "good enough" simple notation while sketching with a pen on whiteboards. Exact UML details aren't important, as long as the modelers understand each other. Stick to simple, frequently used UML elements.
- **Know that all models will be inaccurate**, and the final code or design differentsometimes dramatically differentthan the model. Only tested code demonstrates the true design; all prior diagrams are incomplete hints, best treated lightly as throw-away explorations.
- Developers themselves should do the OO design modeling, for themselves, not to create diagrams that are given to other programmers to implement an example of un-agile waterfall-oriented practices.

UP Phases

A UP project organizes the work and iterations across four major phases:

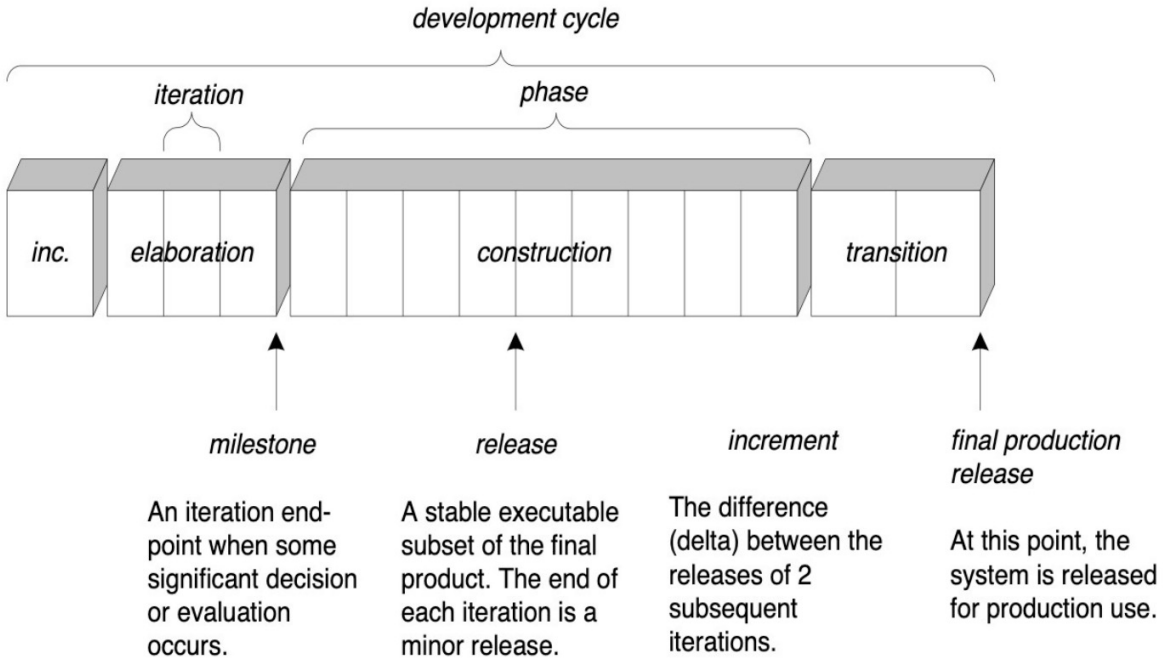
1 Inception approximate vision, business case, scope, vague estimates.

2 Elaboration refined vision, iterative implementation of the core architecture, resolution of high risks, identification of most requirements and scope, more realistic estimates.

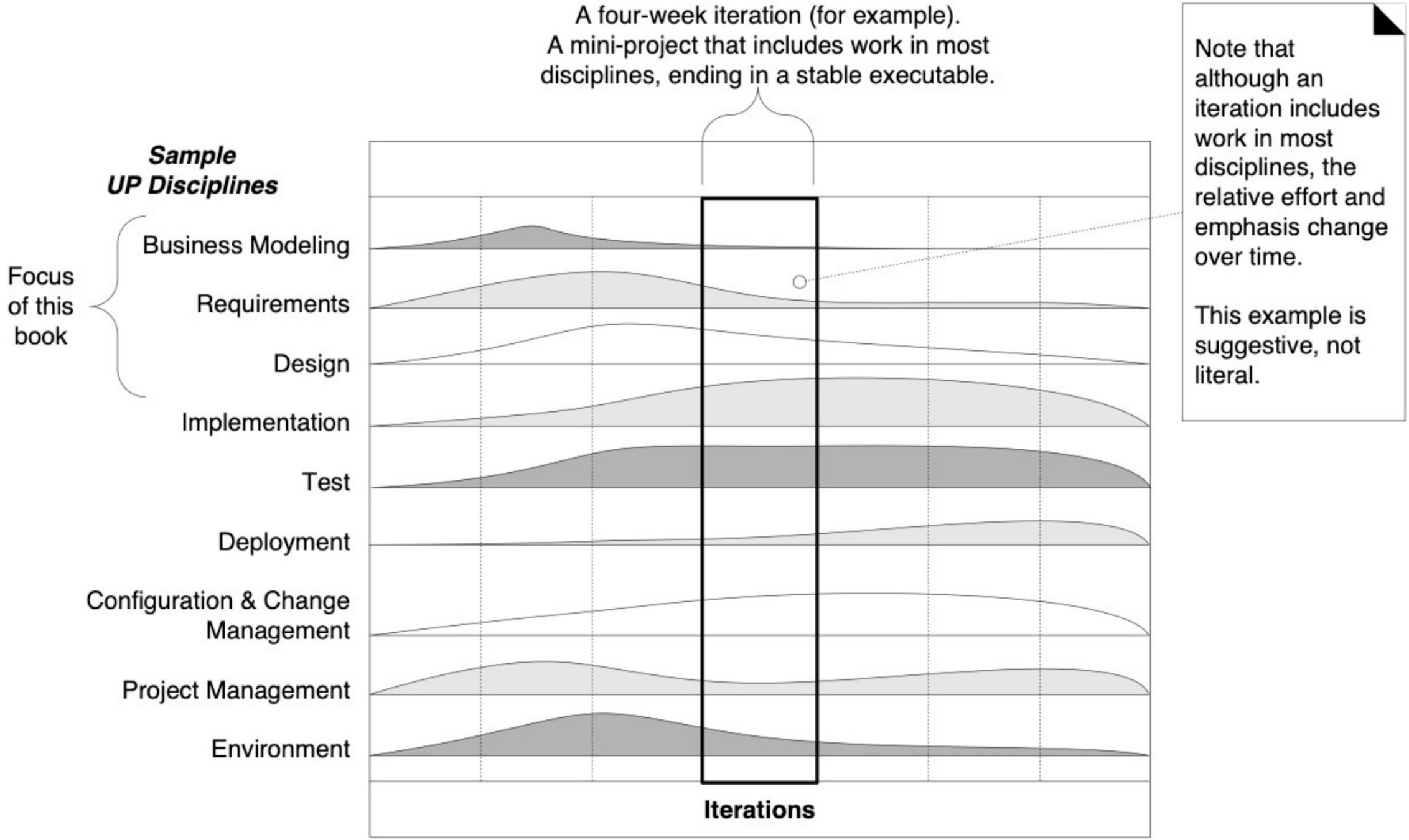
3 Construction iterative implementation of the remaining lower risk and easier elements, and preparation for deployment.

4 Transition beta tests, deployment.

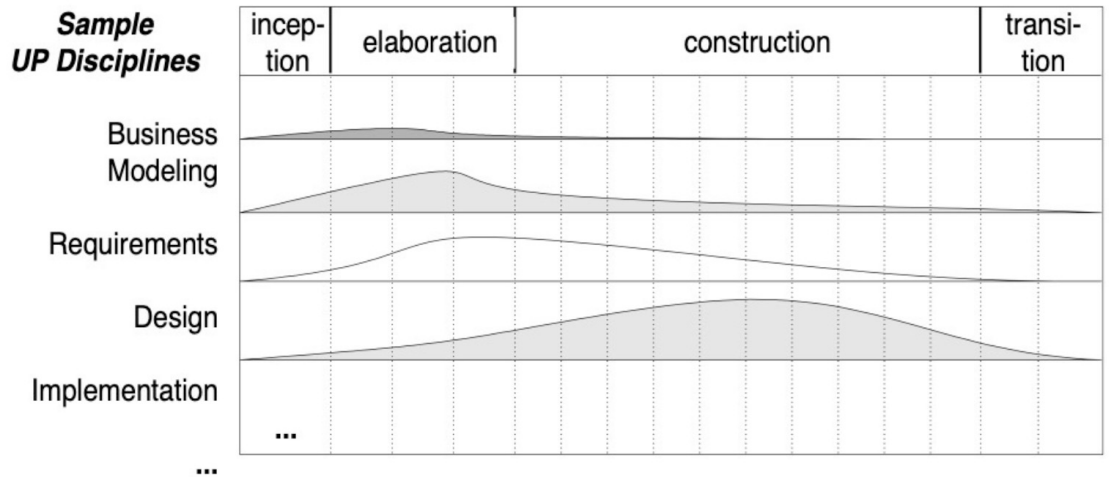
- This is NOT waterfall
- Inception is not requirements (rather a feasibility phase)
- Elaboration is not the requirements or design (rather core-architecture implemented and high-risk issues resolved



Unified Process Phases


A four-week iteration (for example).
A mini-project that includes work in most disciplines, ending in a stable executable.
Note that although an iteration includes work in most disciplines, the relative effort and emphasis change over time.
This example is suggestive, not literal.
Sample UP Disciplines
Focus of this book
Business Modeling
Requirements
Design
Implementation
Test
Deployment
Configuration & Change Management
Project Management
Environment
Iterations

Relationship Between the Disciplines and Phases



The relative effort in disciplines shifts across the phases.

This example is suggestive, not literal.

How we organize UP phases and iterations in this course (book)