CS 619 Introduction to OO Design and Development

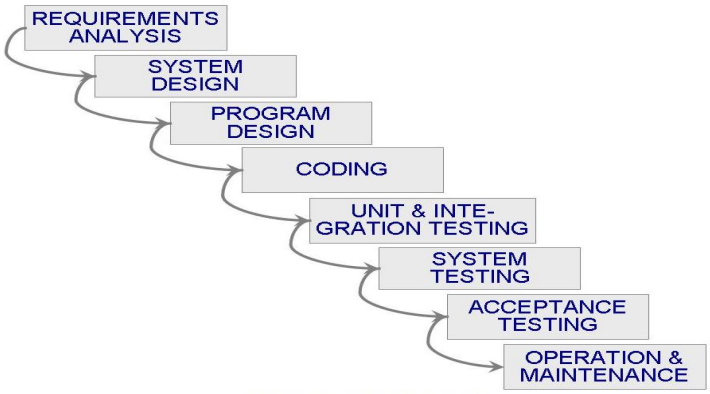
Fall 2012

SE: Processes, Models, and Tools

- **Processes:**
  - Systematic ways of organizing teams and tasks so that there is a clear, traceable path from customer requirements to the final product. (e.g.,Waterfall, Prototyping, Spiral etc.)
  - Processes help organize and co-ordinate teams, prepare documentation, reduce bugs, manage risk, increase productivity, etc.
- **Models:**
  - Well-defined formal or informal languages and techniques for organizing and communicating arguments and decisions about software. e.g:
    - specification languages (Z, etc)
    - design models (UML, etc)
  - Models help stake-holders communicate: customers with developers, designers and developers, developers and testers etc.
  - If they are formal, they also can help support automation
- **Tools:**
  - Programs which automate or otherwise support software development tasks: e.g., Eclipse, Make, CVS, etc.
  - Tools increase productivity, quality and can reduce costs

Waterfall Model

- One of the first process development models proposed
- It presents a very high-level view of the development process and sequence of process activities
- Each major phase is marked by milestones and deliverables (artifacts)



Blaming the Waterfall

- Waterfall model is strongly associated with high rates of failure, lower productivity, and higher defect rates (than iterative projects).
  - On average, 45% of the features in waterfall requirements are never used, and
  - early waterfall schedules and estimates vary up to 400% from the final actuals.
- Can we understand the whole system, in all of its details, before it was built?
- Can we view software development as manufacturing process?

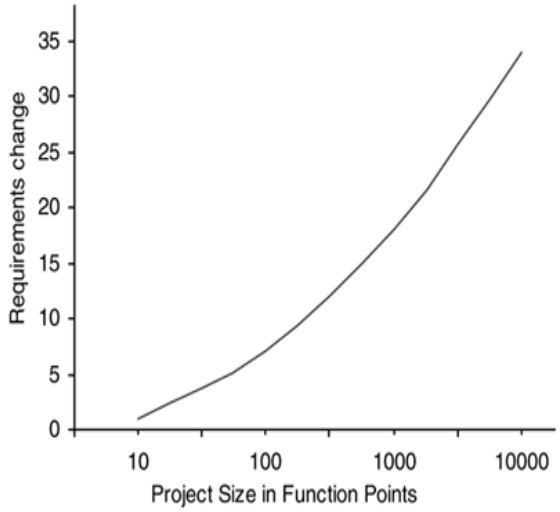
4

Blaming the Waterfall

- Waterfall is strongly related to a key *false assumption* underlying many failed software projects that
  - the specifications are predictable and stable and can be correctly defined at the start, with low change rates.
- This turns out to be far from accurate and a costly misunderstanding.
- A study [by Boehm and Papaccio] showed that **a typical software project experienced a 25% change in requirements.**
- And this trend was corroborated in another major study of thousands of software projects, with **change rates that go even higher 35% to 50% for large projects**. See the figure.

5

Figure 2.3. Percentage of change on software projects of varying sizes.



6

Blaming the Waterfall

The conclusion:

- Any analysis, modeling, development, or management practice based on the assumption that things are long-term stable (i.e., the waterfall) is fundamentally flawed.
- Change is the **constant** on software projects. Iterative and evolutionary methods assume and embrace change and adaptation of partial and evolving specifications, models, and plans based on feedback.

7

Chapter 2. Iterative, Evolutionary, and Agile

- OOAD is best practiced with an iterative development process.
- Agile practices such as agile modelling are key to applying the UML in an effective way in OOAD.
- Among many iterative development processes available, the book chooses the Unified Process (UP) as a sample iterative method under which topics are presented.
- Iterative and Evolutionary development vs. waterfall lifecycle?

8

Agile Methods

- Emphasis on flexibility in producing software quickly and capably
- Agile manifesto
  - Value **individuals and interactions** over **process and tools**
  - Prefer to **invest time in producing working software** rather than in **producing comprehensive documentation**
  - Focus on **customer collaboration** rather than **contract negotiation**
  - Concentrate on **responding to change** rather than on **creating a plan and then following it**

What's UP?

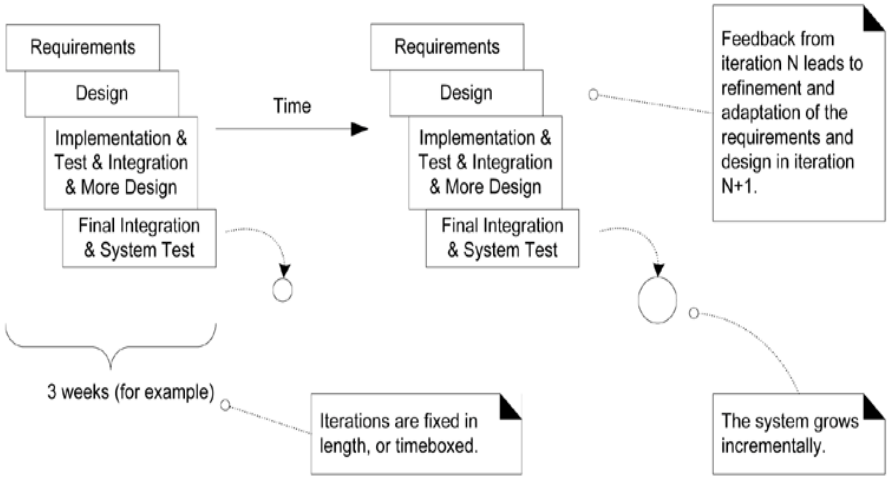
- UP is an iterative development process.
- In an iterative development process, software development is organized into a series of short, fixed-length (for example, three-week) mini-projects called **iterations.**
- Each iteration involves choosing a small subset of the requirements, and quickly designing, implementing, and testing.
- The outcome of each is a tested, integrated, and executable partial system.

10

UP

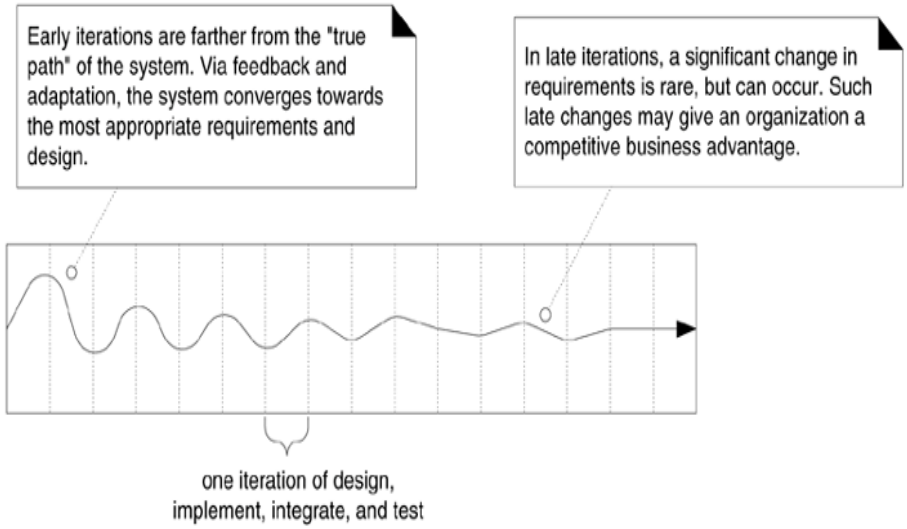
- UP is Iterative and incremental
  - The system increments and evolves incrementally over time, iteration by iteration,
  - **iterative and incremental development, or, iterative and evolutionary development.**
- UP is use case driven
  - A use case is a piece of functionality in the system that gives a user a result of value
  - Use cases capture functional requirements
  - Use case answers the question: *What is the system supposed to do for the user?*
- UP is architecture-centric

11



12

What's UP?



13

How Long Should an Iteration Be?

- Most iterative methods recommend an iteration length between two and six weeks. ***Small steps, rapid feedback, and adaptation*** are central ideas in iterative development.
- A key idea is that iterations are *timeboxed, or fixed in length*.

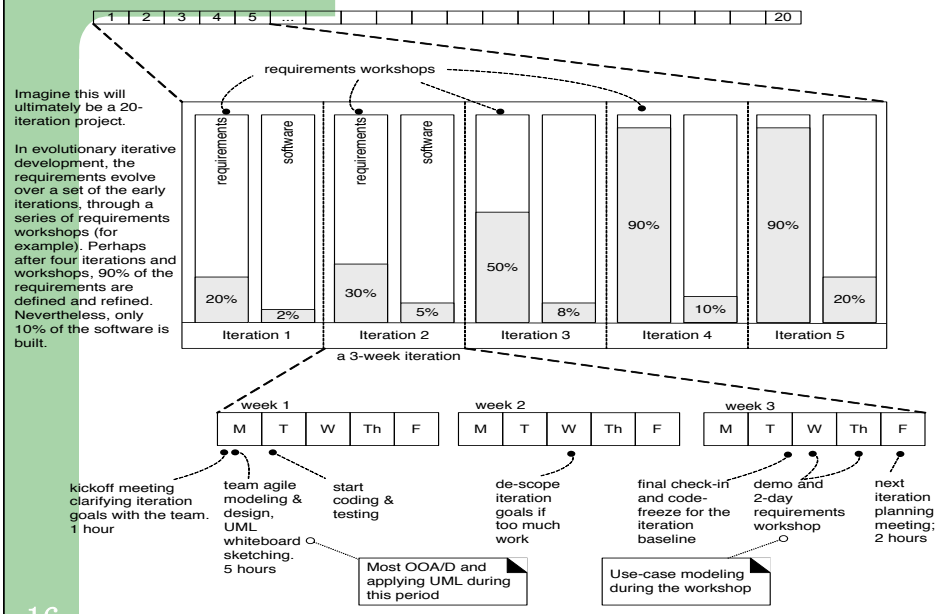
14

Benefits of an iterative development process

- less project failure, better productivity, and lower defect rates; shown by research into iterative and evolutionary methods
- 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 that more closely meets the real needs of the stakeholders
- managed complexity; the team is not overwhelmed by "analysis paralysis" or very long and complex steps
- the learning within an iteration can be methodically used to improve the development process itself, iteration by iteration

15

How to do Iterative and Evolutionary Analysis and Design?



16

The UP phases

- A UP project organizes the work and iterations across four major phases:
  - **Inception** -- approximate vision, business case, scope, vague estimates.
  - **Elaboration** – refined vision, iterative implementation of the core architecture, resolution of high risks, identification of most requirements and scope, more realistic estimates.
  - **Construction --** iterative implementation of the remaining lower risk and easier elements, and preparation for deployment.
  - **Transition --** beta tests, deployment.

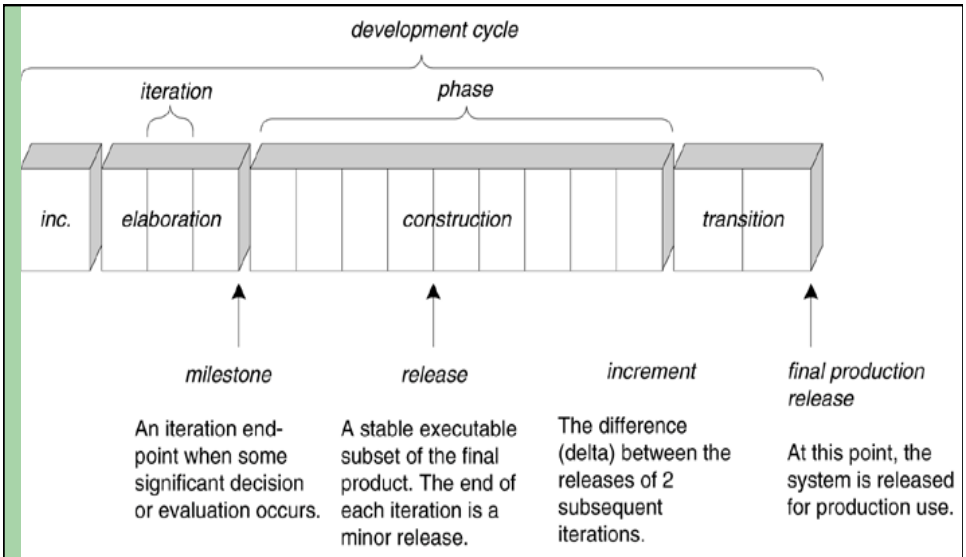
17

The UP phases

- Inception is **not** a requirements phase
- Inception is a feasibility phase, where just enough investigation is done to support a decision to continue or stop

- Similarly, elaboration is **not** the requirements or design phase
- Elaboration it is a phase where the core architecture is iteratively implemented, and high-risk issues are mitigated.

18



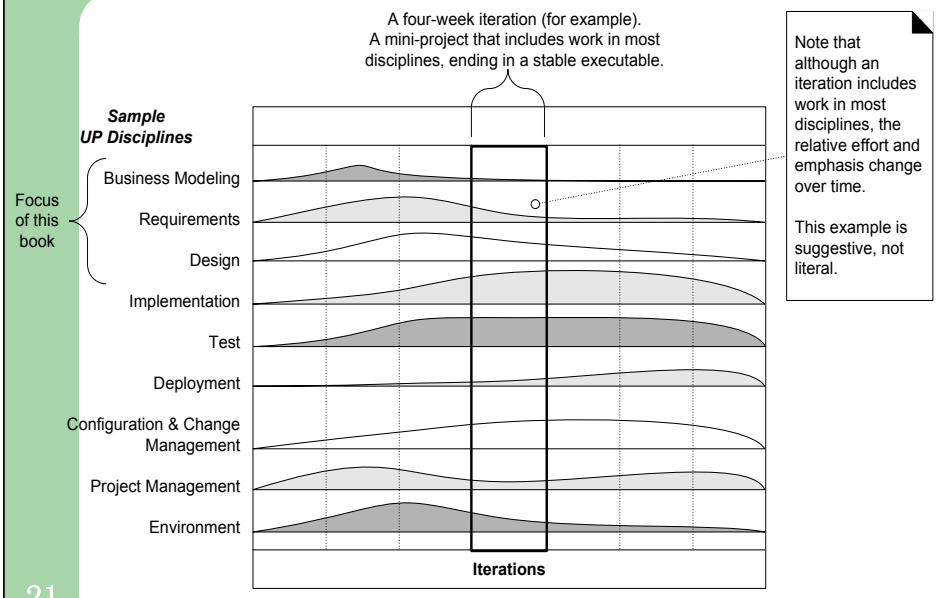
19

The UP Disciplines

- Discipline: A set of activities in a subject area,
- Artifact: general term for any work product: code, Web graphics, database schema, text documents, diagrams, models, and so on.
- We focuses on some artifacts in the following three disciplines:
  - **Business Modeling** - The *Domain Model* artifact, to visualize noteworthy concepts in the application domain.
  - **Requirements** - The *Use-Case Model* and *Supplementary Specification* artifacts to capture functional and non-functional requirements.
  - **Design** - The *Design Model* artifact, to design the software objects.

20

The UP Disciplines



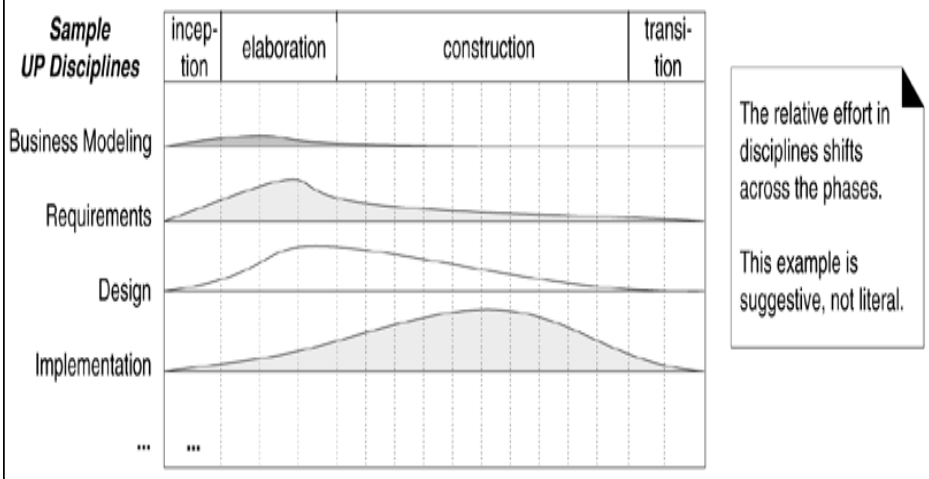
21

UP Disciplines and UP phases

- As illustrated in the figure, during one iteration, work goes on in most or all disciplines.
- However, the relative effort across these disciplines changes over time.
- The following figure shows the changing relative effort with respect to the phases

22

UP Disciplines and phases



23

What is Agile Modeling?

- Adopting an agile method does not mean avoiding any modeling; that's a misunderstanding. Many agile methods normally include significant modeling sessions.
- The purpose of modeling and models is primarily to support understanding and communication, not documentation.
- Don't model or apply the UML to all or most of the software design. Model and apply the UML for the smaller percentage of unusual, difficult, tricky parts of the design space.
- Use the simplest tool possible.

24

What is Agile Modeling?

- 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.*

25

What is Agile Modeling?

- Know that all models will be inaccurate, and the final code or design different - sometimes dramatically different than the model.
- 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.

26

What is Agile UP?

- The UP was not meant by its creators to be heavy or un-agile, although its large optional set of activities and artifacts have understandably led some to that impression.
- Rather, it was meant to be adopted and applied in the spirit of adaptability and lightness - an agile UP.
- Some examples of how this applies:

27

What is Agile UP?

- Prefer a small set of UP activities and artifacts.
  - Since the UP is iterative and evolutionary, requirements and designs are not completed before implementation. They adaptively emerge through a series of iterations, based on feedback.
- Apply the UML with agile modeling practices.
- There isn't a detailed plan for the entire project. There is a high-level plan (called the Phase Plan) that estimates the project end date and other major milestones, but it does not detail the fine-grained steps to those milestones.
- A detailed plan (called the Iteration Plan) only plans with greater detail one iteration in advance. Detailed planning is done adaptively from iteration to iteration.

28