UNIT-I

Iterative, Evolutionary and Agile

MADHESWARI.K
AP/CSE
SSNCE

Topics to be covered?

- Agile Modeling
- Unified Process

Iterative and evolutionary development

- Contrasted with a **sequential** or **waterfall lifecycle**
- Involves early programming and testing of partial system in repeating cycles.
- Development starts before all the requirements are defined in detail.
- Feedback is used to clarify and improve the evolving specifications.
- Higher success, productivity rates and lower defect levels.

What is UP? Are other methods Complementary

- A **software development process** describes an approach to building, deploying and maintaining software.
- **Unified Process-** emerged as popular **iterative** software development process for building **object oriented systems.**
- **RUP: Rational Unified process-** detailed refinement of the unified process, has been widely adapted
- Includes skilful practices from Extreme Programming and Scrum
- For example, XP's **test-driven development, refactoring** and **continuous integration** practices can fit within a UP project.
- The UP combines commonly accepted best practices, such as an iterative lifecycle and risk-driven development, into a cohesive and well-documented process description.

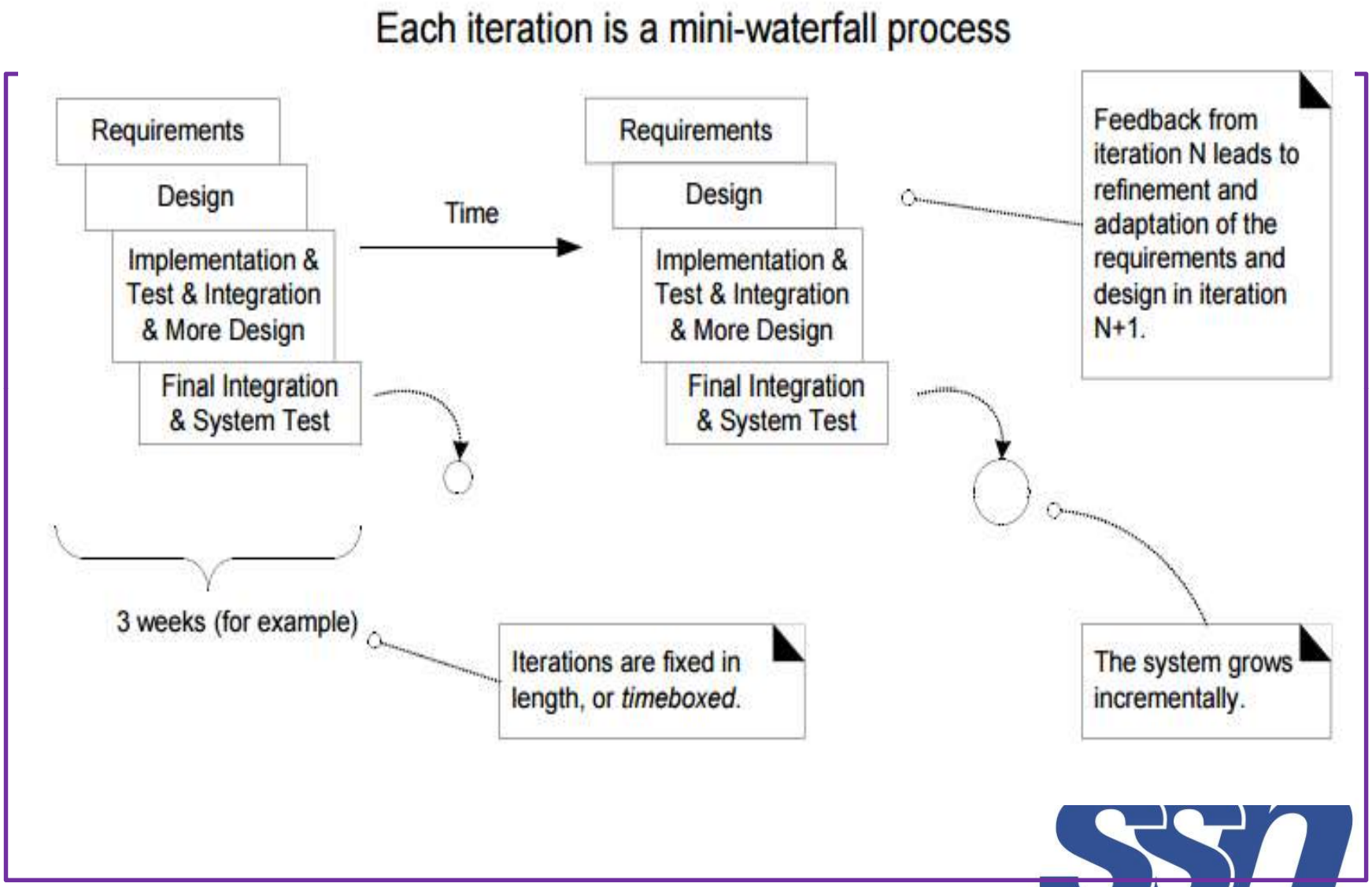
Importance of the Unified Process(UP)

1) The UP is an *iterative* process. Iterative development influences how to introduce OOA/D , and to understand how it is best practiced.

2) UP practices provide an example *structure* for how to do and thus how to explain OOA/D.

3) The UP is flexible, and can be applied in a lightweight and *agile* approach that includes practices from other agile methods (such as XP or Scrum).

Iterative and evolutionary development



Iterative and evolutionary development

- A key practice in both the UP and most other modern methods is **iterative development**.
- In this lifecycle approach,Development is organized into a series of short fixed length mini projects called iterations
- The outcome of each is a tested, integrated, and executable partial system.
- Each iteration includes its own requirement analysis, design, implementation, and testing activities.
- Also known as iterative and incremental development. Grow incrementally over time, iteration by iteration
- Also known as iterative and evolutionary development - feedback and adaptation evolve the specifications and design.
- Early iterative process ideas were known as spiral development and evolutionary development [Boehm]

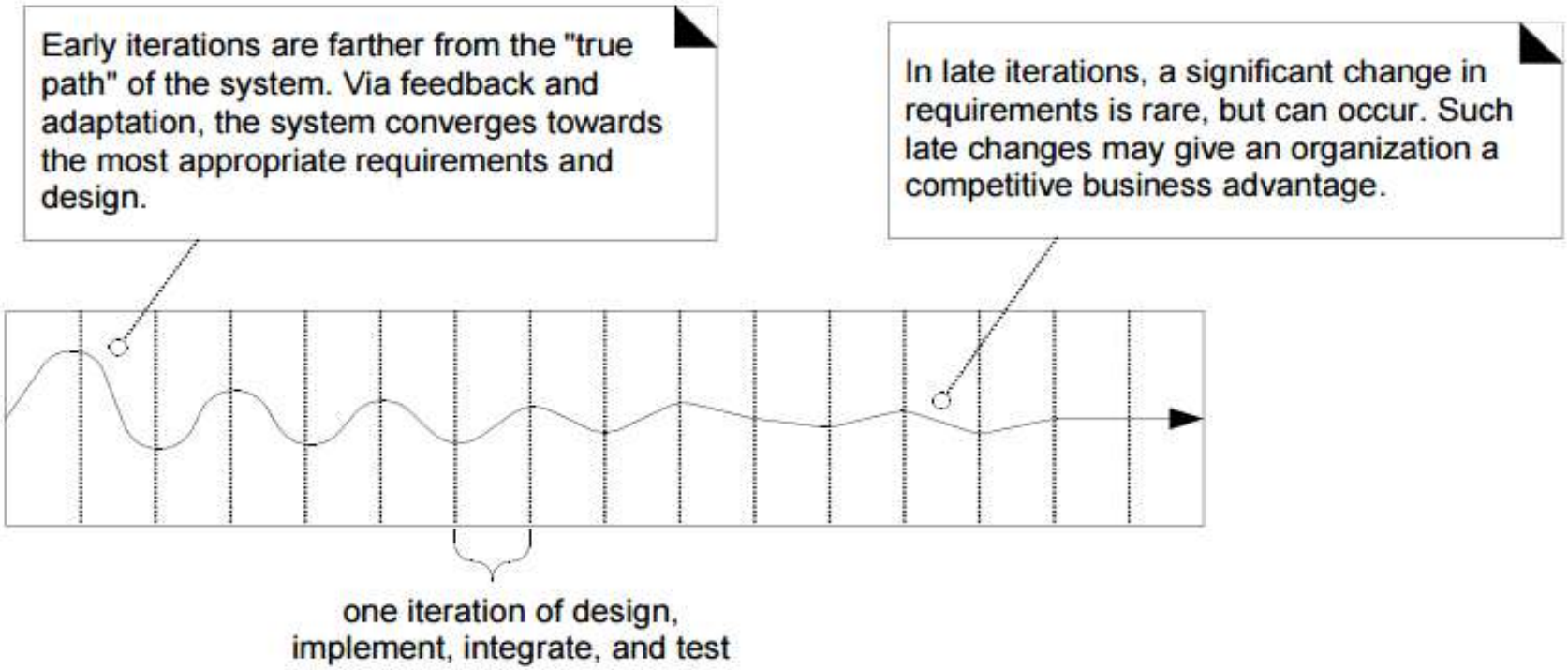
Benefits of iterative development

Benefits include:

- 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

Benefits of iterative development

Iterative development entails the refactoring of previously written code in subsequent iterations. Requires good architecture, automated tools and library of unit tests to be successful

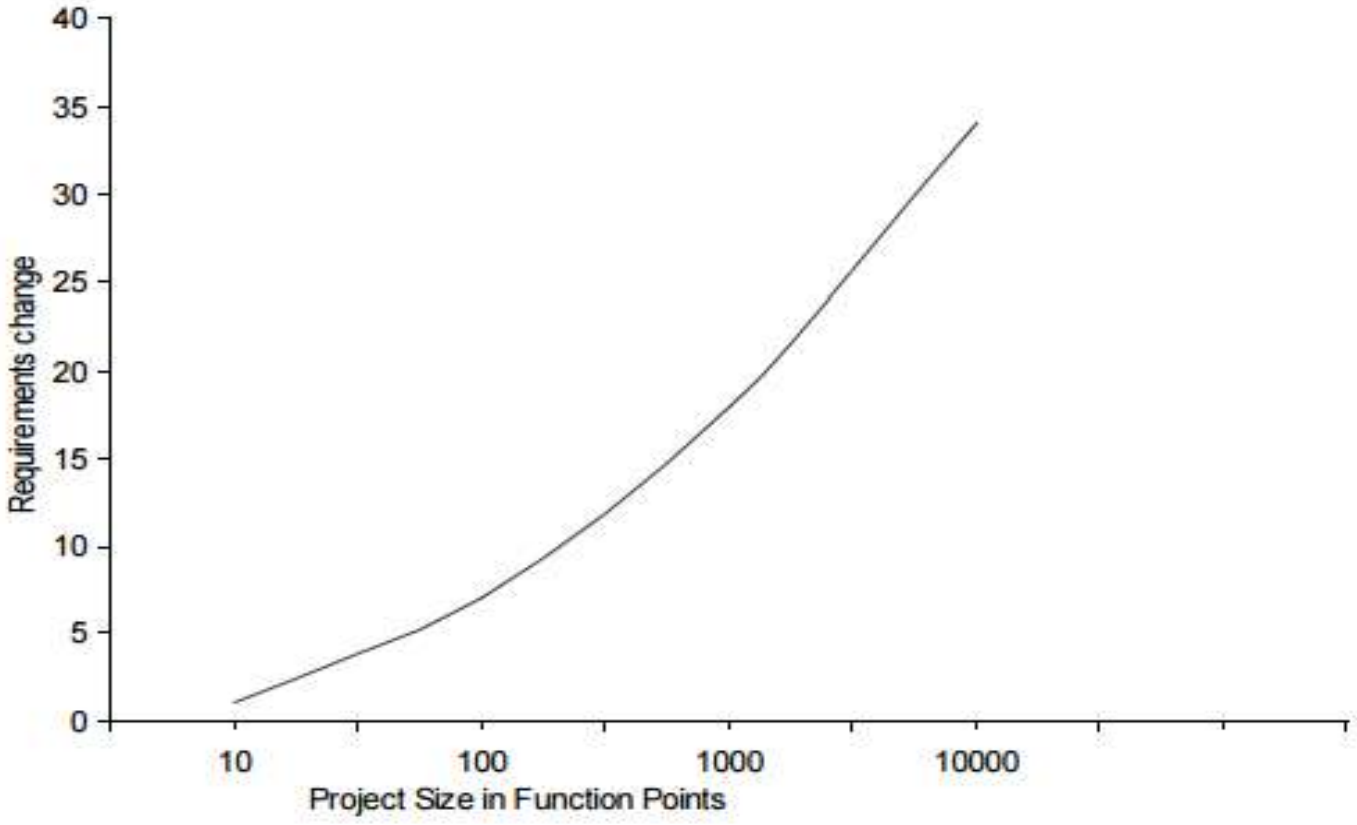


Why is Waterfall is so failure prone ?

- There isn't one simple answer to why the waterfall is so failure-prone, but it 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 higher35% to 50% for large projects as illustrated in Figure 1.5.

Why is Waterfall is so failure prone ?

Requirements change increases with project size
Manufacture of a SW 'product' is not predictable
Waterfall process too rigid



Need for feedback and adaptation

In complex, changing systems (such as most software projects) feedback and adaptation are key ingredients for success.

- Feedback from early development, programmers trying to read specifications, and client demos to refine the requirements.
- Feedback from tests and developers to refine the design or models.
- Feedback from the progress of the team tackling early features to refine the schedule and estimates.
- Feedback from the client and marketplace to re-prioritize the features to tackle in the next iteration

Risk driven and client Driven iterative planning

- UP encourages the combination of risk-driven and client-driven iterative planning
- This means that the goals of the early iterations are chosen to

  1) identify and drive down the highest risk

  2) build visible features that the cares most about
- Risk driven iterative development includes the best practices of architecture-centric iterative development , meaning that early iterations focus on building, testing and stabilizing the core architecture.

agile methods

- **Agile development** methods usually apply timeboxed iterative and evolutionary development, employ adaptive planning, promote incremental delivery, and include other values and practices that encourage *agility,* rapid and flexible response to change.
- **Agile methods** share best practices like evolutionary refinement of plans, requirements, and design
- In addition, they promote practices and principles that reflect an agile sensibility of simplicity, lightness, communication, self-organizing teams, and more.

five agile principles

- Satisfy the customer through early and continuous delivery of valuable software.
- Agile processes harness change for customer's competitive advantage.
- Deliver working software frequently
- Agile software promote sustainable development
- The best, architecture, requirements, and designs emerge from self-organizing teams

❖

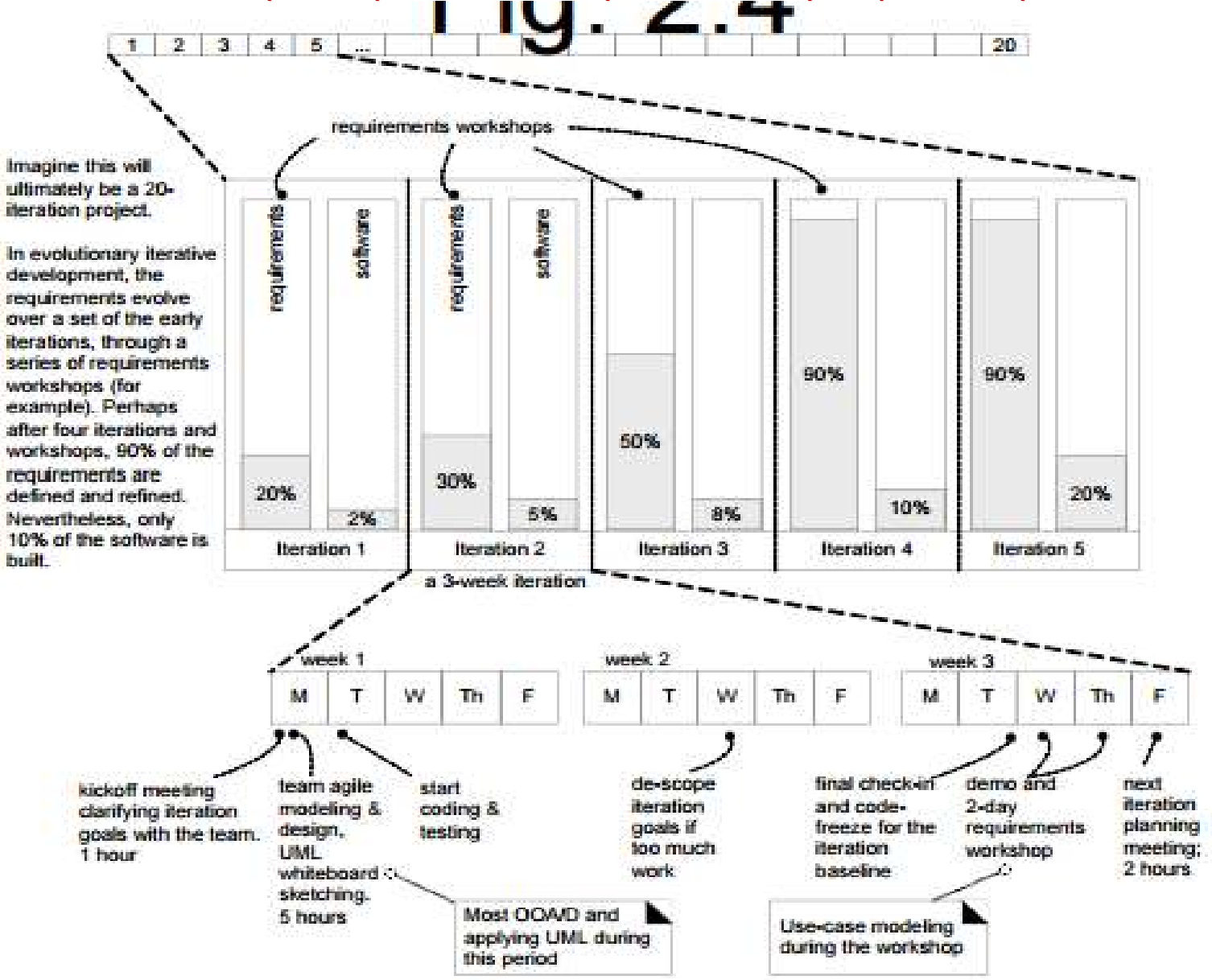
Agile Modeling

- The very act of modeling can and should provide a way to better understand the problem or solution space.
- The purpose of doing UML is to quickly explore(more quickly than with code) alternatives and the path to a good OO design.
- Many agile methods,such as feature-driven development, DSDM,and Scrum include significant modeling sessions, The purpose of modeling is primarily support understanding and communication ,not documentation.
- 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.
- Prefer sketching UML on white boards,and capturing the diagrams with a digital camera.

Agile Modeling

- Model in pairs(or traids) at the whiteboard – The purpose of modeling is to discover, understand and share the understanding.
- Create models in parallel. For example, start sketching in one whiteboard, Dynamic View UML Interaction diagram, and in another whiteboard, the static view, the UML Class Diagram.
- All prior diagrams are incomplete hints – throw-away explorations;only tested code demonstrates the true code.

Evolutionary Analysis and design – the majority in early iterations



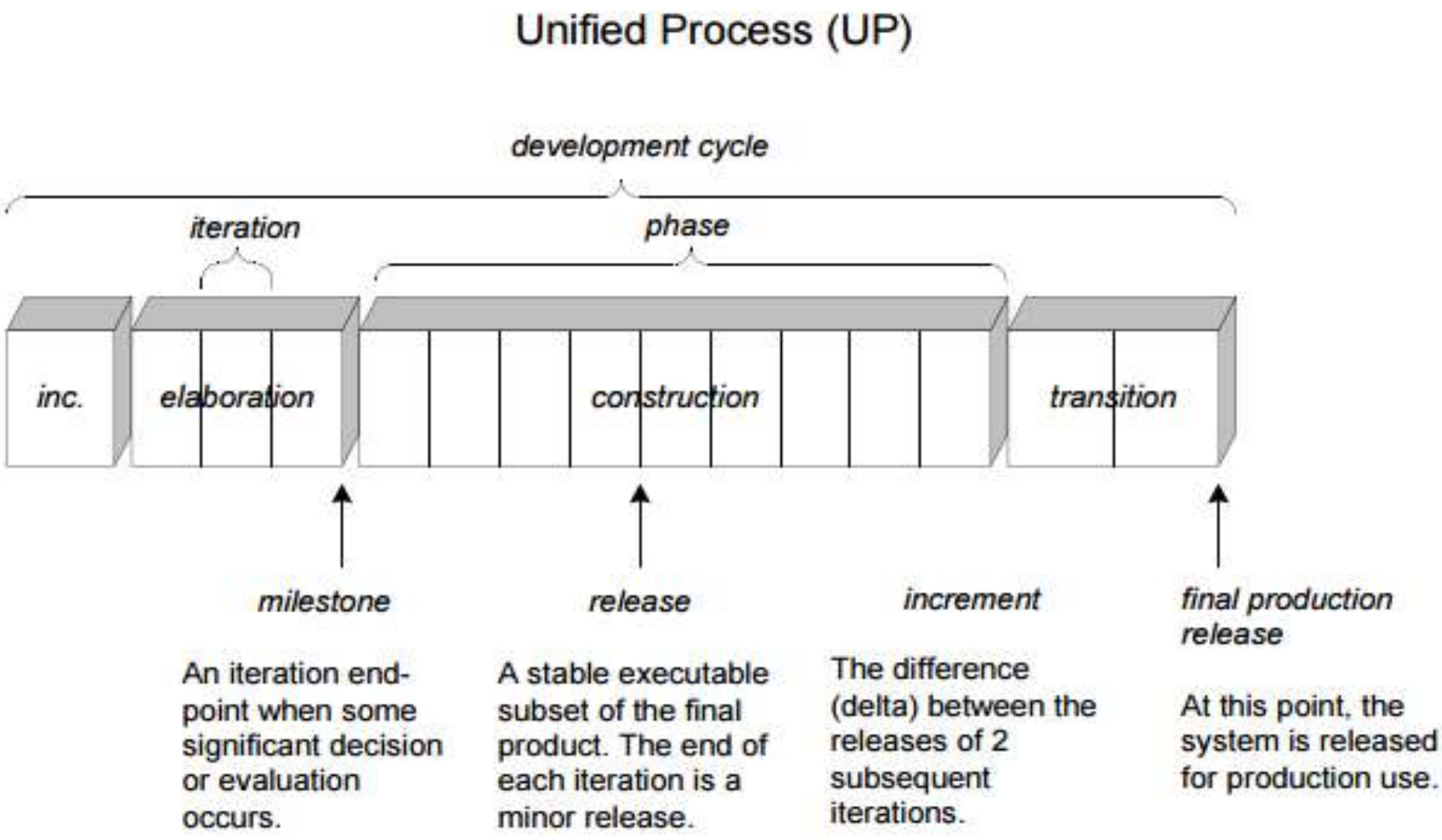
Critical UP practices (or) important UP practices

- The idea for UP practice is short timeboxed, ,iterative, evolutionary, and adaptive development.
- Some additional best practices and key ideas in UP are
  - Tackle high-risk and high-value issue in eary iterations
  - Continual evaluation,feedback and requirements from users
  - Build cohesive,core architecture in ealy iterations
  - Continuously verify quality;test early,often and realistically
  - Practice Change Request and Configuration Management

Different UP Phases

- An UP Project organizes work and iterations across four major phases :
- **Inception** – approximate Vision, Business case, Scope, vague estimates
- **Elaboration** – Refined BVision, 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

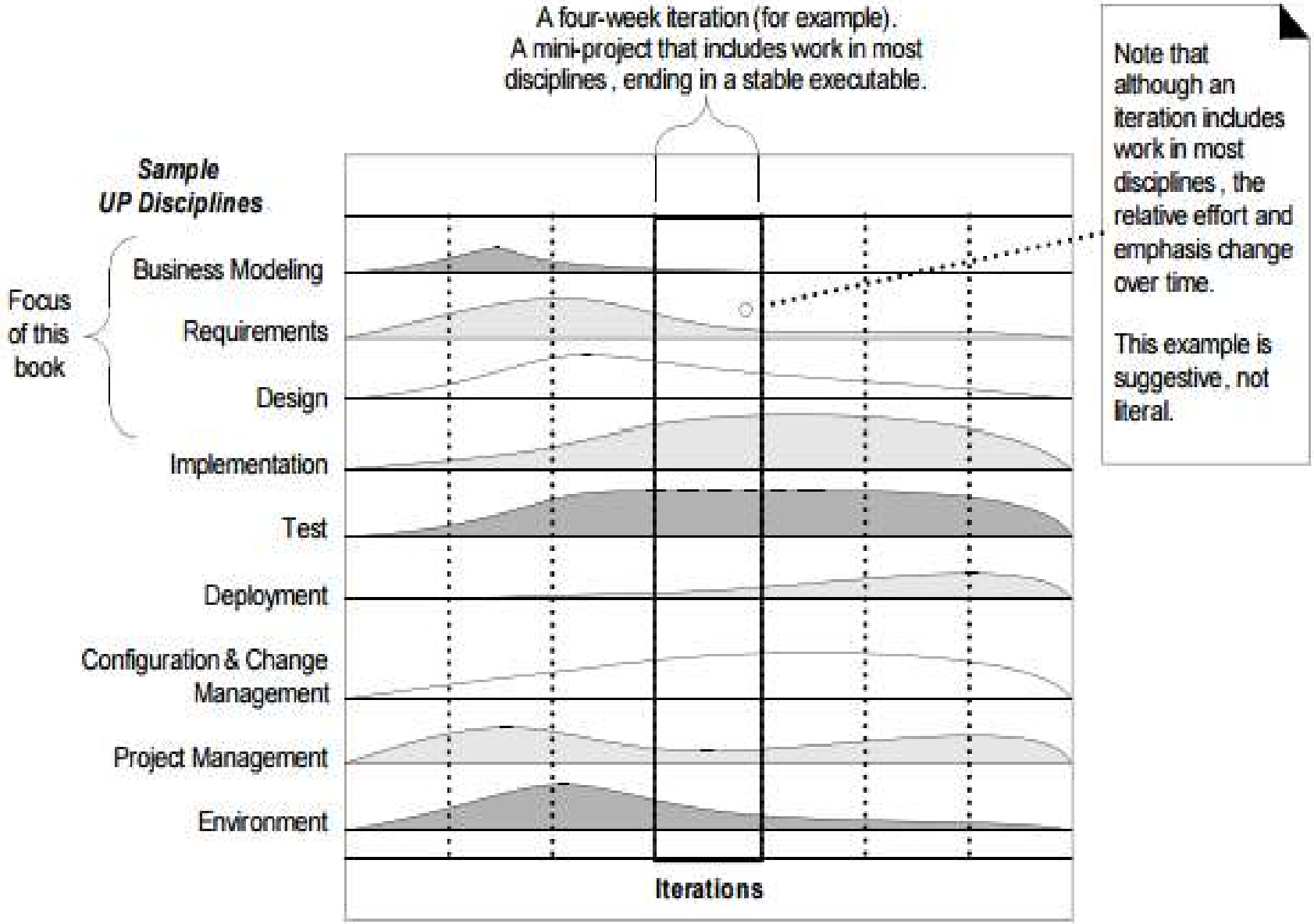
Schedule oriented terms in UP



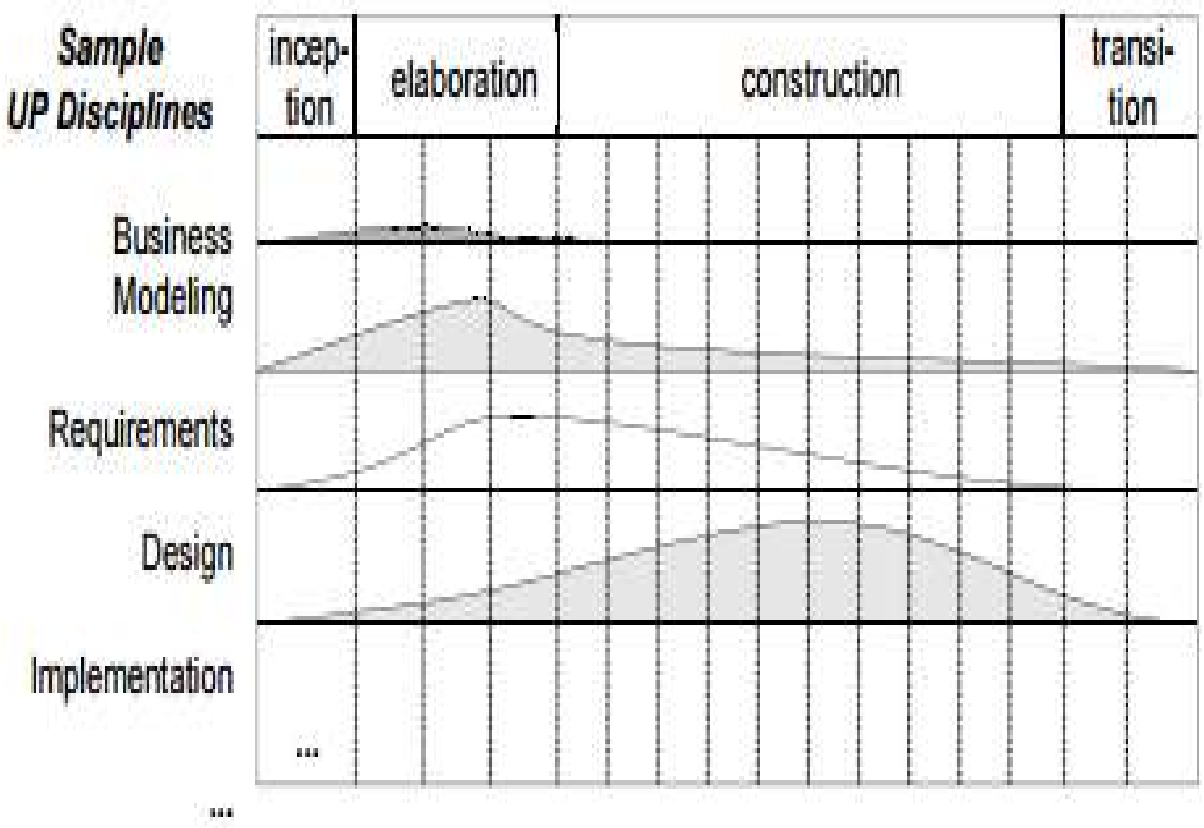
UP disciplines

- In the UP, an artifact is the general term for any work product : Code,Web Graphics,Schema,Test Documents,diagrams,model and so on.
- Some of the artifacts in the following Disciplines are :
  - 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 artifacts
  - Implementation – Programming and building the system, not deploying it

UP disciplines



Unified Process – Disciplines and Phases



The relative effort in disciplines shifts across the phases.

This example is suggestive, not literal.

Unified Process

In this course, we're primarily in the
Inception & early Elaboration phases