SWA – Related Concepts, Techniques & Methodologies

Lecture 01

BIL428 Software Architectures

Asst.Prof.Dr. Mustafa Sert
msert@baskent.edu.tr

Department of Computer Engineering, Başkent University
Ankara 06810 TURKEY

Content

- The development challenge
- Waterfall development
- Iterative and Evolutionary development
- SWA design principles

Why is software development difficult?

- The **problem domain** (also called application domain) is difficult
- The **solution domain** is difficult
- The **development process** is difficult to manage
- Software offers **extreme flexibility**
- Software is a **discrete system**
  - Continuous systems have no hidden surprises
  - Discrete systems can have hidden surprises! (Parnas)

David Lorge Parnas is an early pioneer in software engineering who developed the concepts of modularity and information hiding in systems which are the foundation of object oriented methodologies.

Software Engineering is more than writing Code

- Problem solving
  - Creating a solution
  - Engineering a system based on the solution
- Modeling
- Knowledge acquisition
- Rationale management

Techniques, Methodologies and Tools

- **Techniques:**
  - Formal procedures for producing results using some well-defined notation
- **Methodologies:**
  - Collection of techniques applied across software development and unified by a philosophical approach
- **Tools:**
  - Instruments or automated systems to accomplish a technique
  - CASE = Computer Aided Software Engineering

Software Engineering: A Working Definition

Software Engineering is a collection of techniques, methodologies and tools that help with the production of

A high quality *software* system developed with a given **budget** before a given **deadline** while ***change*** occurs

Challenge: Dealing with complexity and change

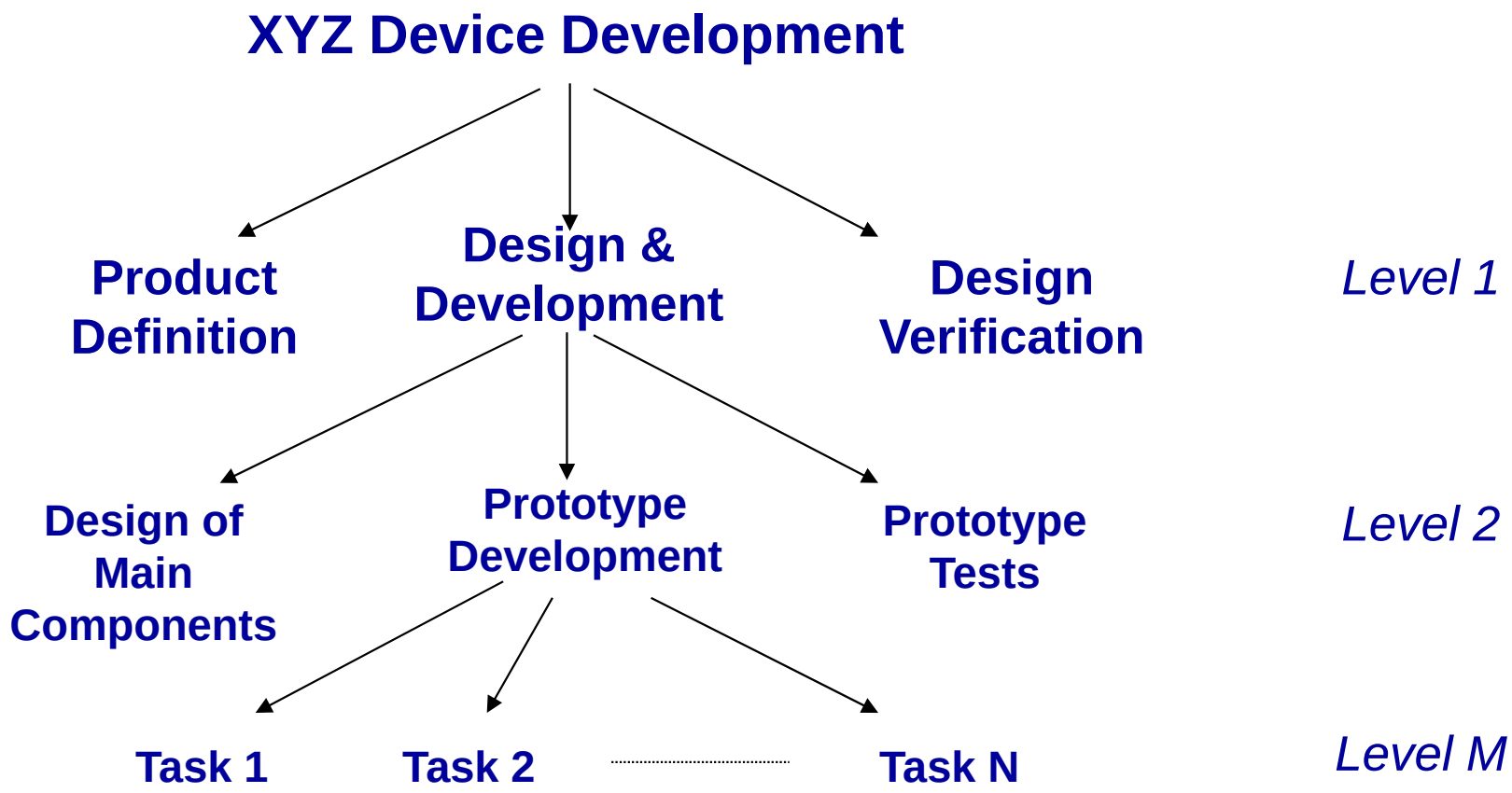
Software Engineering: A Problem Solving Activity

How do we do that?

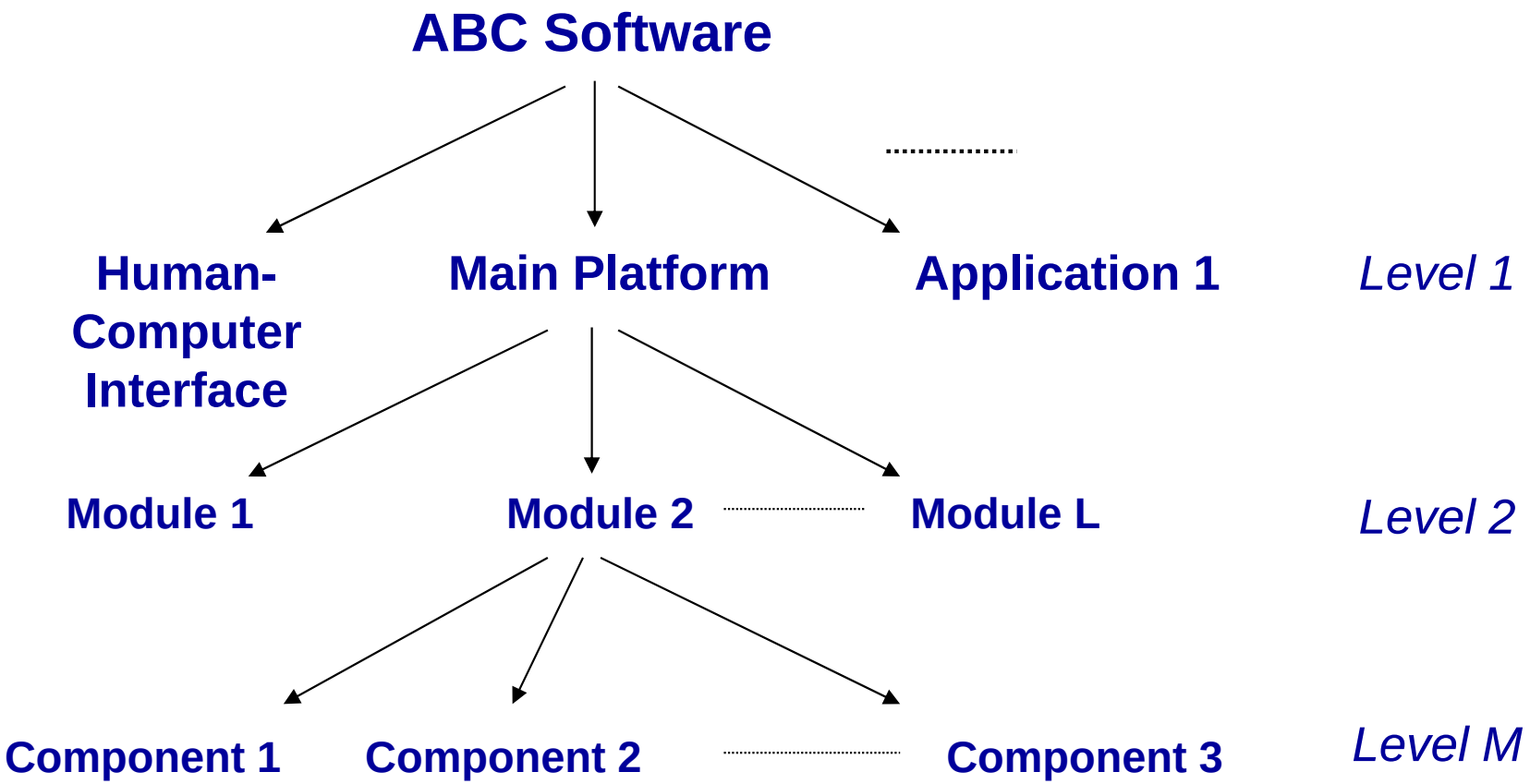
- **Analysis:**
  - **Understand** the nature of the problem and break the problem into pieces
  - This means, that we need to identify the pieces of the puzzle (In object-oriented development, we will call this object identification).
- **Synthesis:**
  - Put the pieces together into a large structure, usually by keeping some type of structure within the structure

For problem solving we use techniques, methodologies and tools.

Activity/Process Oriented Decomposition



Function/Component Oriented Decomposition



SWE Development Activities

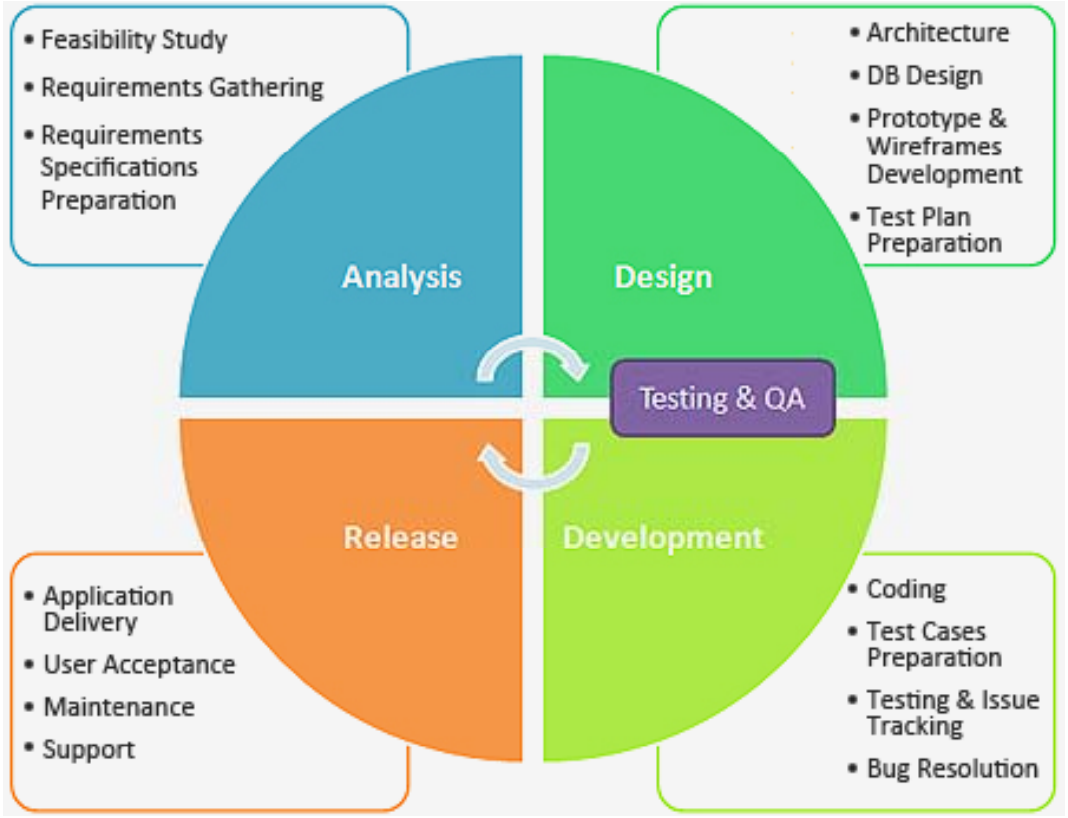
- Development activities deal with the complexity by constructing models of the problem domain or the system:
  - Requirements elicitation
  - Analysis
  - System design
  - Object design
  - Implementation

Managing Software Development

- Management activities focus on planning the project, monitoring its status, tracking changes, and coordinating resources such that a high-quality product is delivered on time and within budget:
  - Communication
  - Rationale management
  - Testing
  - Software configuration management
  - Project management
  - **Software life cycle modeling activities**

Waterfall SW Development Model

- Waterfall model is the most common version of Software Development Life Cycle (SDLC) for software development.
- It is called Waterfall since it follows a linear development method where each phase is completed before the next one is started and there is no loop back.
- It follows the principle of Doing things right the first time & every time



Waterfall – When/How do we use it?

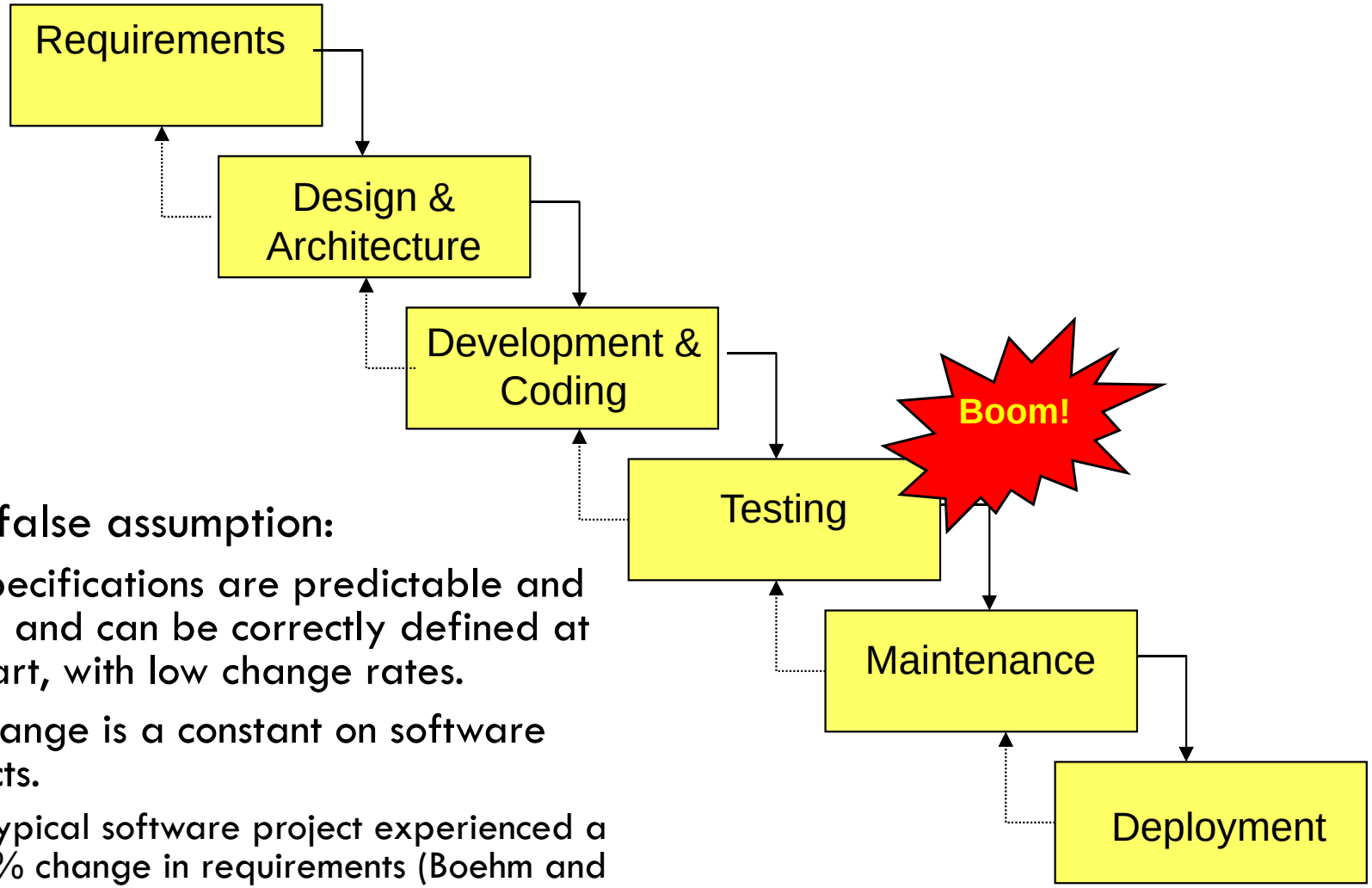
- **When?**
  - The requirements are clearly and un-ambiguously outlined
- **How?**
  - Complete project execution is divided in to well defined stages of analysis, design, development, testing & QA, release & user acceptance followed by maintenance and support
  - A schedule is typically set with deadlines for each stage of development at the start of the project
- **In theory,** this model leads to the project being delivered on time because each phase has been planned in detail

Waterfall – Drawbacks

- **Practically,**
  - This model involves high risk as it does not embrace the inevitable changes and revisions that become necessary with most of the projects.
  - Once an application is in the testing stage, it is very difficult to go back and change something that was not thought of in the analysis stage
- Promotes big up-front "*speculative*" requirements and design steps before programming.
- Was historically promoted due to belief or hearsay rather than statistically significant evidence.***
  - Success/failure studies show that the waterfall is strongly associated with the highest failure rates for software projects.
  - On average, 45% of the features are never used, and early schedules and estimates vary up to 400% from the actuals.

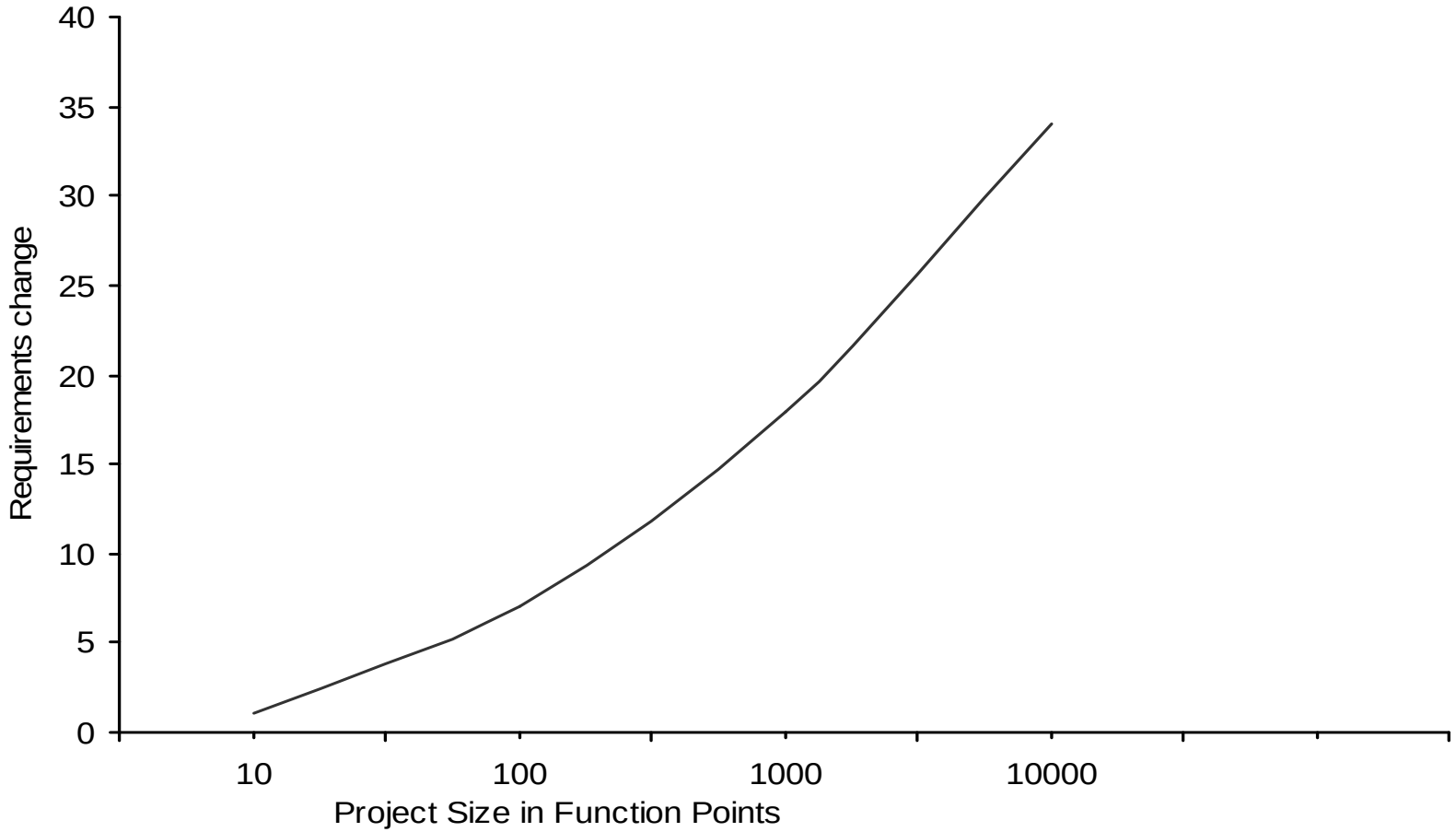
*** *We should approach to the author's such claims very carefully! Waterfall is still the preferred approach in many projects.*

Why the waterfall lifecycle fails?



- The key false assumption:
  - The specifications are predictable and stable and can be correctly defined at the start, with low change rates.
  - But change is a constant on software projects.
    - A typical software project experienced a 25% change in requirements (Boehm and Papaccio 1988)

Some Statistics..



Percentage of change on software projects of varying sizes. (Jones 1997)

Some Statistics..

Dollars at Risk in the Average Organization: $74 million

This data represents 20,821 projects closed in the last 12 months by 134 organizations.

- Average number of projects closed per firm 155
- Average total cost of closed projects per firm..... $200 million
- Average cost per project..... $1.3 million
- Percentage of projects at risk—recovered (25%) or failed (12%)..... 37%
- Average dollars at risk per firm .....$74 million
- Average dollars saved due to successful project recoveries per firm.....$50 million
- Average dollars lost due to project failures per firm.....$24 million

Project Management Solutions, Inc., (PM), 2011

Top 5 Causes of Troubled Projects..

1
REQUIREMENTS
Unclear, lack of agreement, lack of priority, contradictory, ambiguous, imprecise

2
RESOURCES
Lack of resources, resource conflicts, turnover of key resources, poor planning

3
SCHEDULES
Too tight, unrealistic, overly optimistic

4
PLANNING
Based on insufficient data, missing items, insufficient details, poor estimates

5
RISKS
Unidentified or assumed, not managed

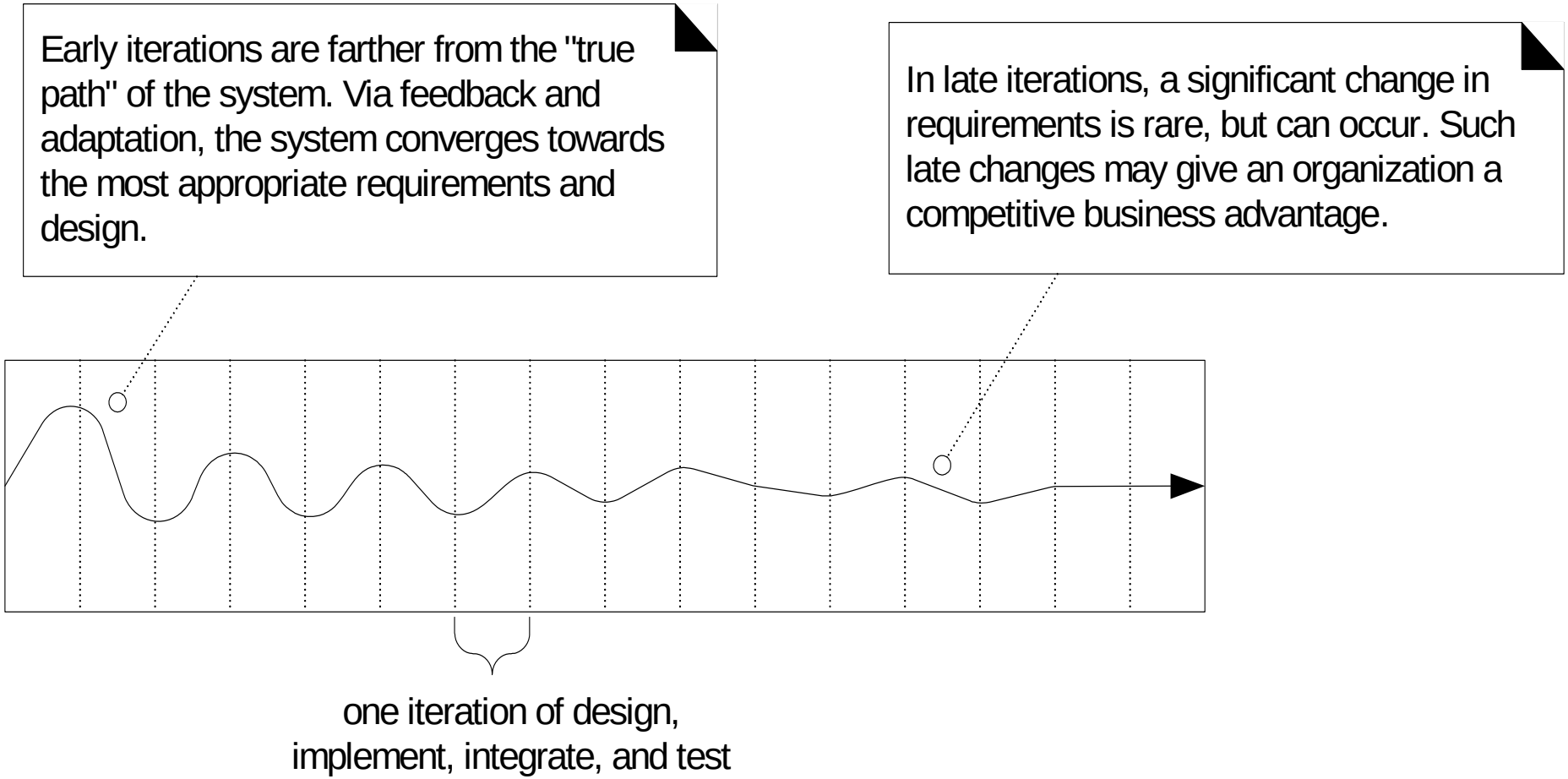
Project Management Solutions, Inc., (PM), 2011

Iterative and Evolutionary Development

- Involves early programming and testing of a partial system, in repeating cycles.
- Relies on short quick development steps (or iterations), feedback and adaptation to clarify the requirements and design so successively enlarging and refining a system.
  - Normally assumes that the development starts before all requirements are defined in detail, feedback is used to clarify and improve the evolving specifications.
- Each iteration will include requirements, analysis, design, implementation, and test.

Current research demonstrates that iterative methods are associated with higher success and productivity rates, and lower defect levels.

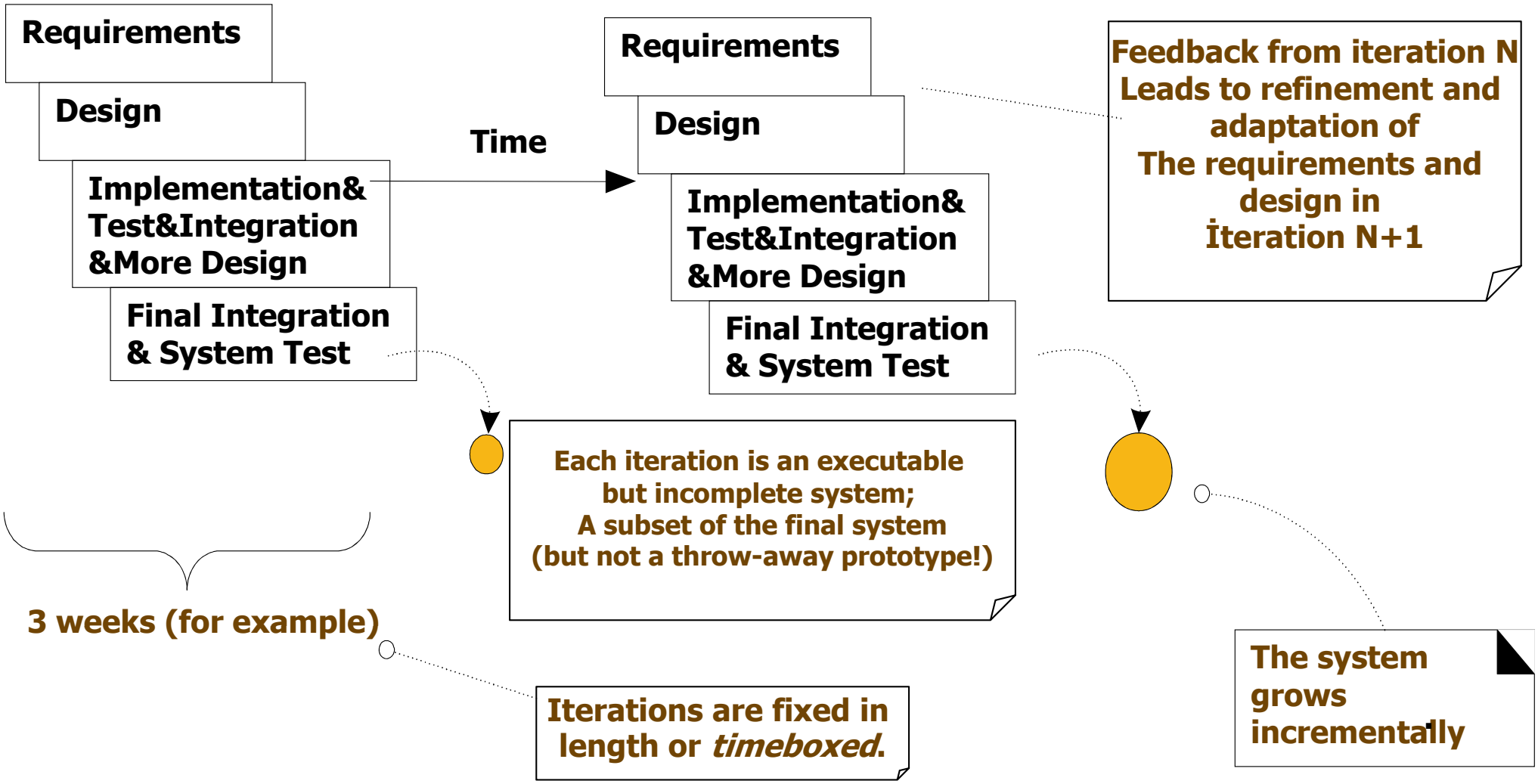
Fig. Iterative feedback and evolution leads towards the desired system. The requirements and design instability lowers over time.



Timeboxing

- A key idea is that iterations are timeboxed, or fixed in length.
  - Most iterative methods recommend in iteration length bw 2 – 6 weeks.
  - If it seems that it will be difficult to meet the deadline, the recommended response is to de-scope
    - De-scoping: removing tasks or requirements from the iteration, and including them in a future iteration, rather than slipping the completion date.

Fig. Iterative and Evolutionary Development (also known as iterative and incremental development; spiral development and evolutionary development)



ITERATIVE DEVELOPMENT PHASES..

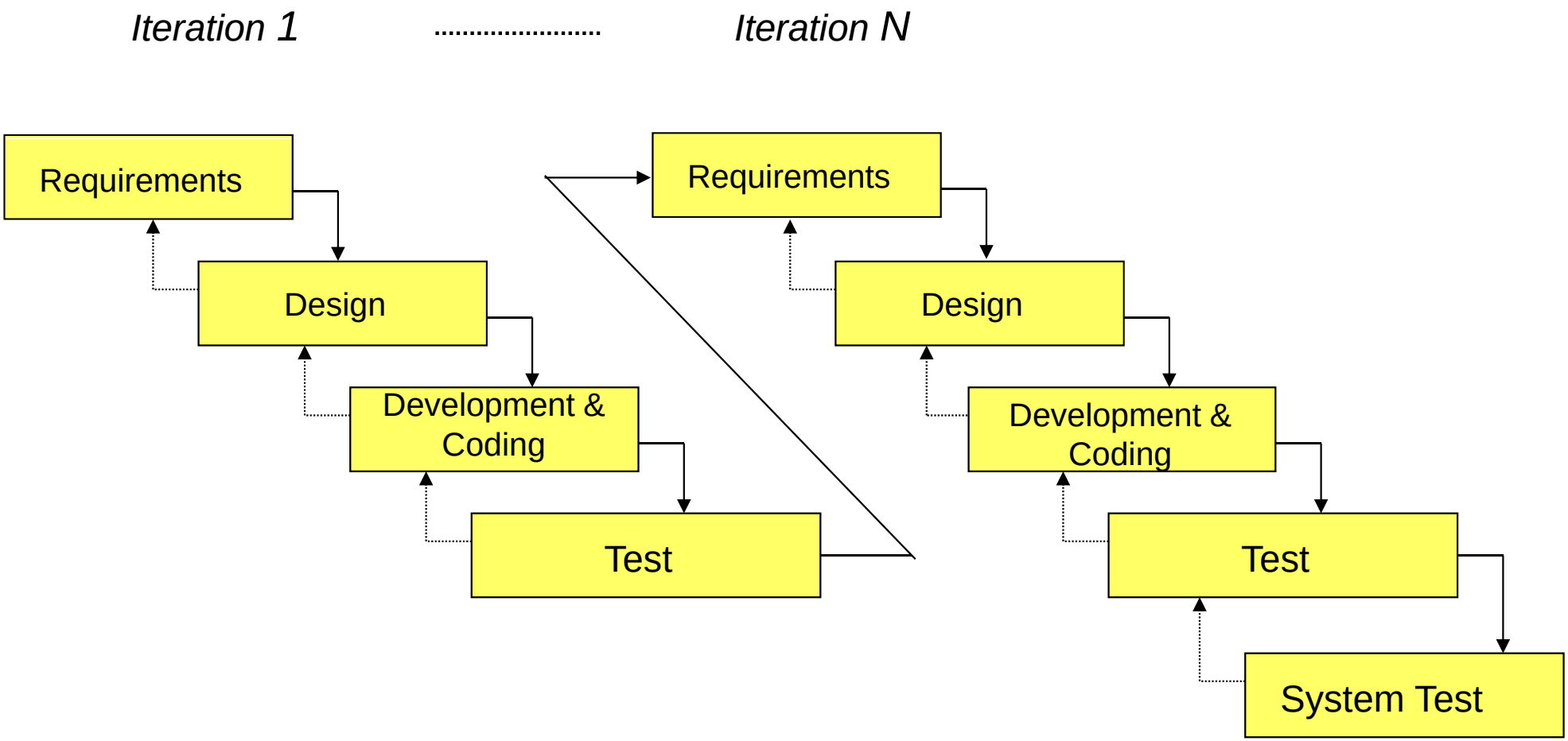
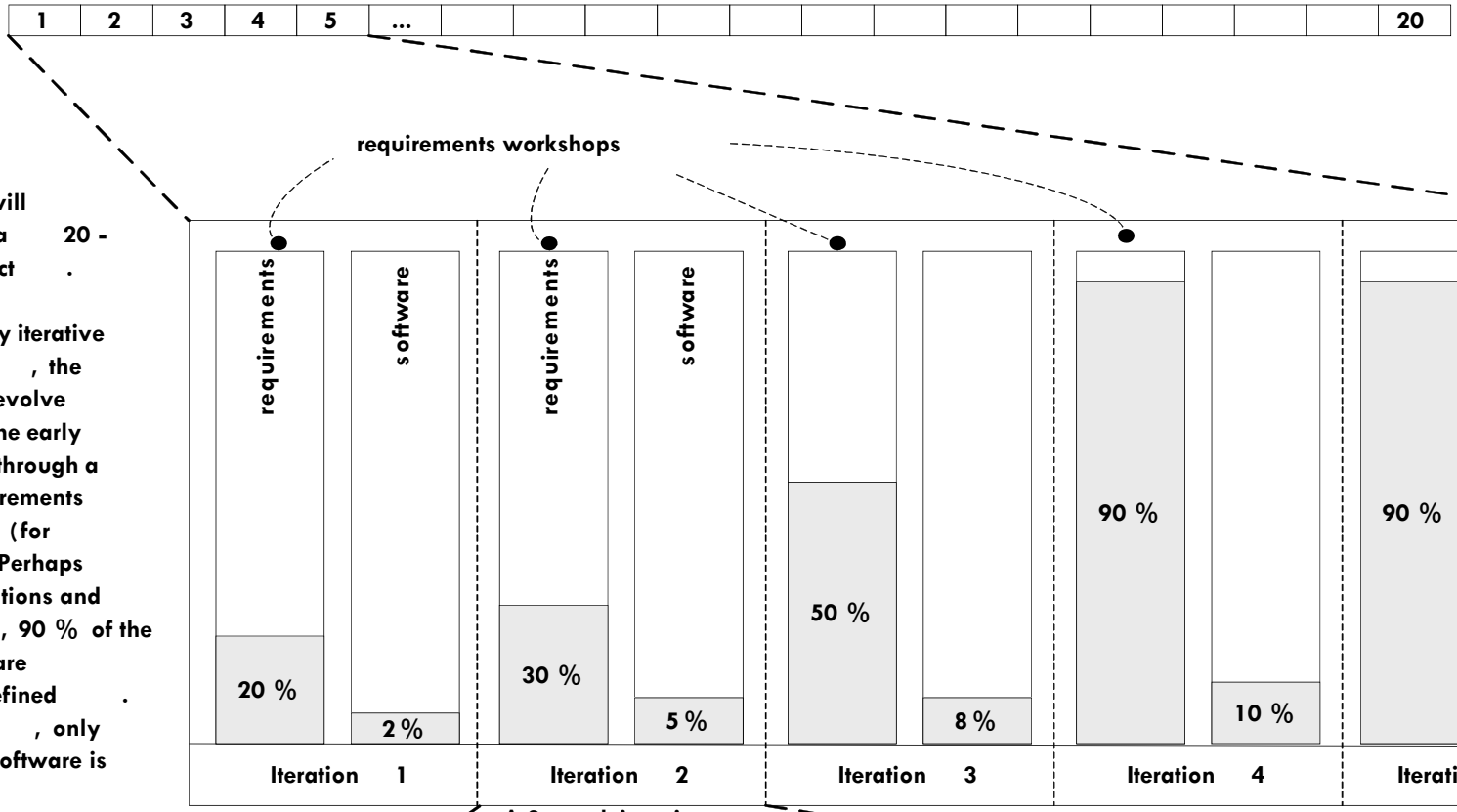
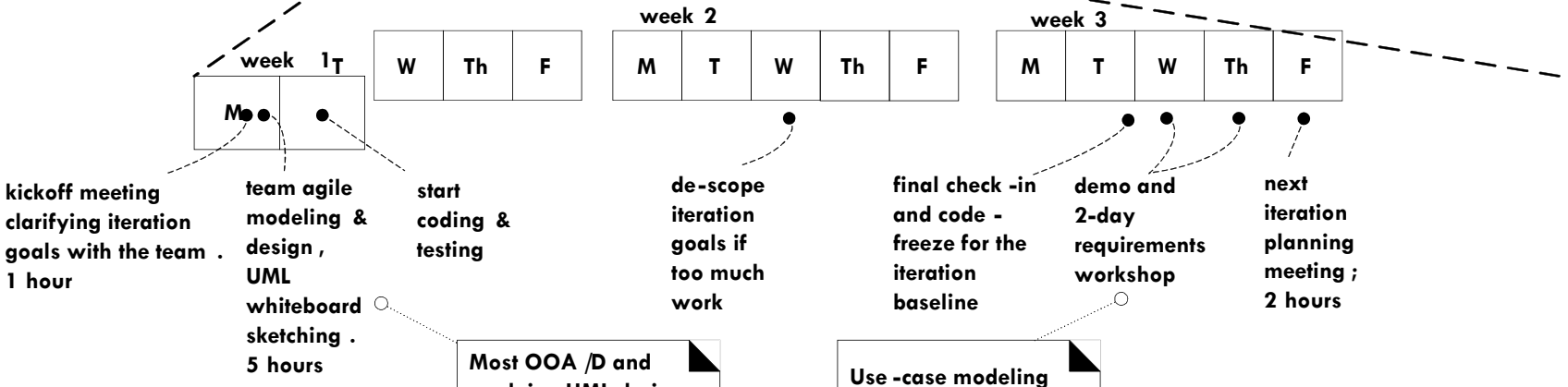


Fig. Evolutionary analysis and design – the majority in early iterations.



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 .



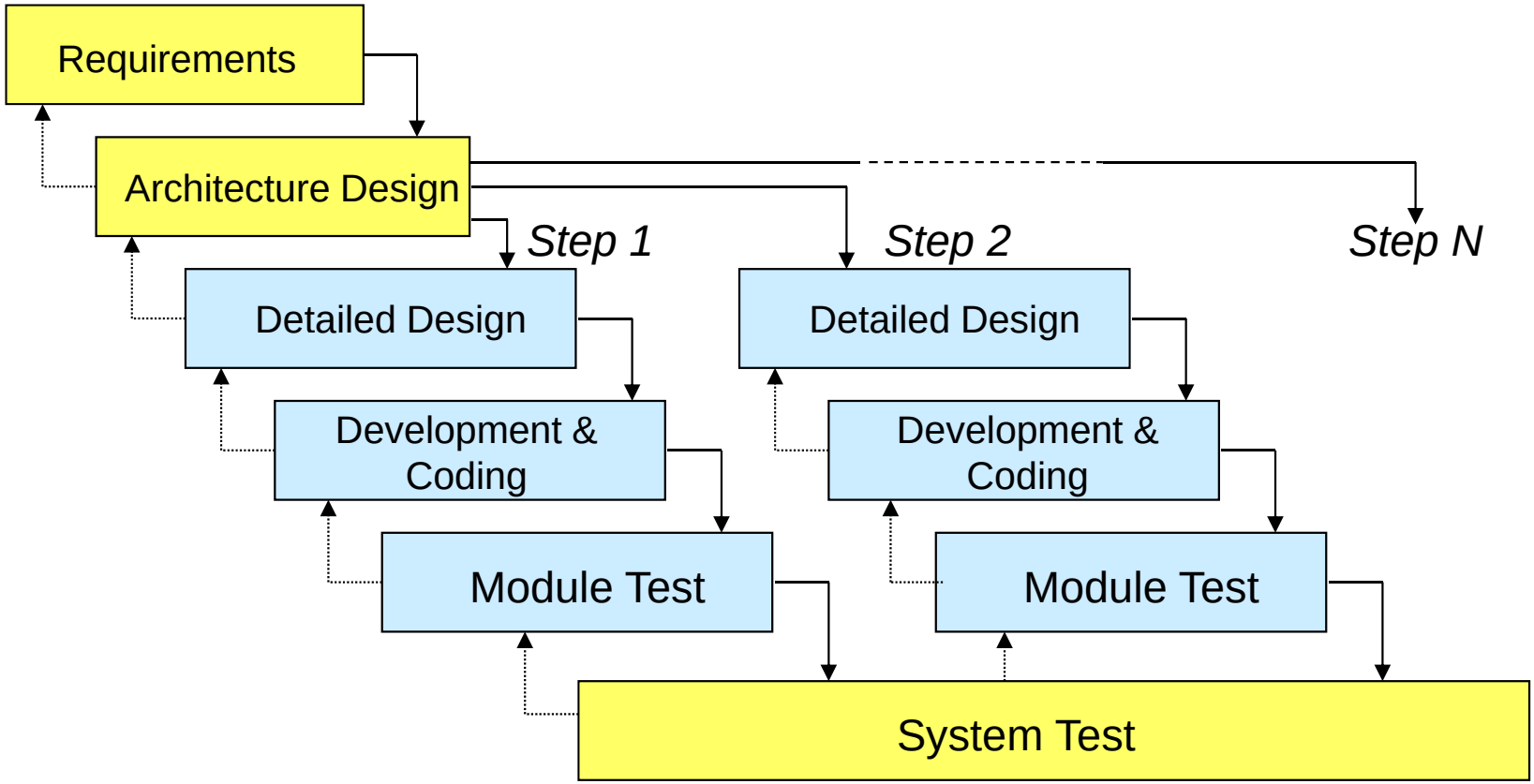
Build-Feedback-Adapt Cycles

- In complex changing systems, 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 and 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.

Benefits to Iterative Development

- Less project failure, better productivity, and lower defect rates
- Early rather than late mitigation of high risks
- Early visible progress
- Early feedback, user engagement, and adaptation
- 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.

Incremental Development



Agile Methods and Attitudes

- 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.
- Existing Agile Methods:
  - Rational Unified Process (RUP), Extreme Programming (XP), Feature Driven Development (FDD), ...

Agile Modeling

- The purpose of modeling (sketching UML, …) is primarily to *understand*, not to document.
  - The very act of modeling can and should provide a way to better understand the problem or solution space.
- From this viewpoint, the purpose of "doing UML or OOA/D" is not for a designer to create many detailed UML diagrams that are handed off to a programmer,
  - but rather to quickly explore alternatives (more quickly than with code) and the path to a good OO design.

SCRUM Development

- **Scrum** is an **iterative and incremental agile software development method** for managing software projects and product or application development

The Scrum Process

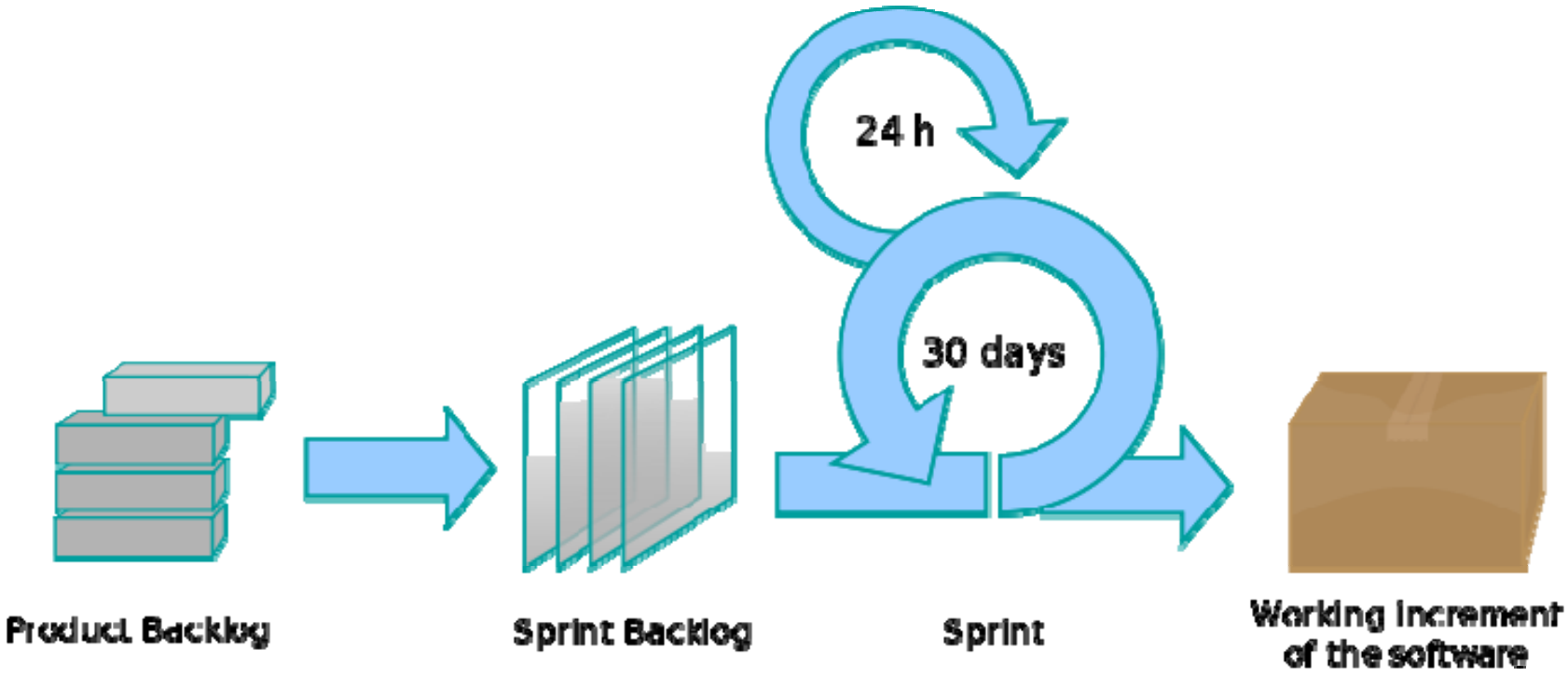


Fig.

http://en.wikipedia.org/w/index.php?title=File:Scrum_process.svg&page=1

Planning Poker..

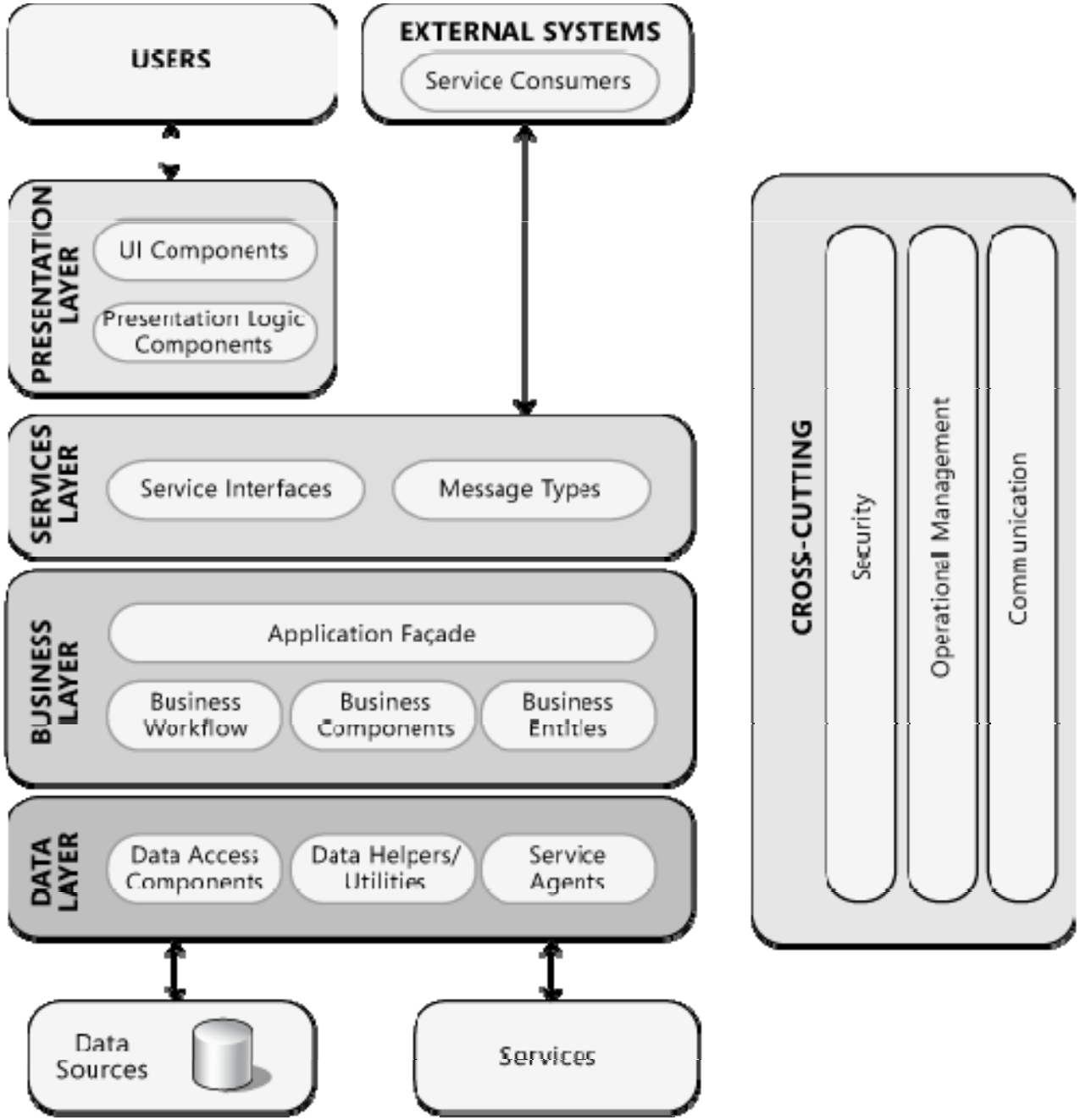
- Also called **Scrum poker** is a consensus-based technique for estimating, mostly used to estimate effort or relative size of user stories in software development
- Most commonly used in **agile software development,** in particular the **Extreme Programming methodology**

Planning Poker

Fig. http://en.wikipedia.org/wiki/File:CrispPlanningPokerDeck.jpg

Key Design Principles of SWA

- Organization of functionality is often referred to as grouping components into "**areas of concern.** "
- The figure illustrates common application architecture with components grouped by different areas of concern (Microsoft)



Key Design Principles

- Separation of concerns
- Single Responsibility principle
- Principle of Least Knowledge
- Don't repeat yourself (DRY)
- Minimize upfront design

Key Design Considerations

- Determine the Application Type
- Determine the Deployment Strategy
- Determine the Appropriate Technologies
- Determine the Quality Attributes
- Determine the Crosscutting Concerns