Software Requirements Analysis, Design and Modeling

Part 1 Introduction

庞雄文
Tel：18620638848
Wechat: augepang
QQ: 443121909

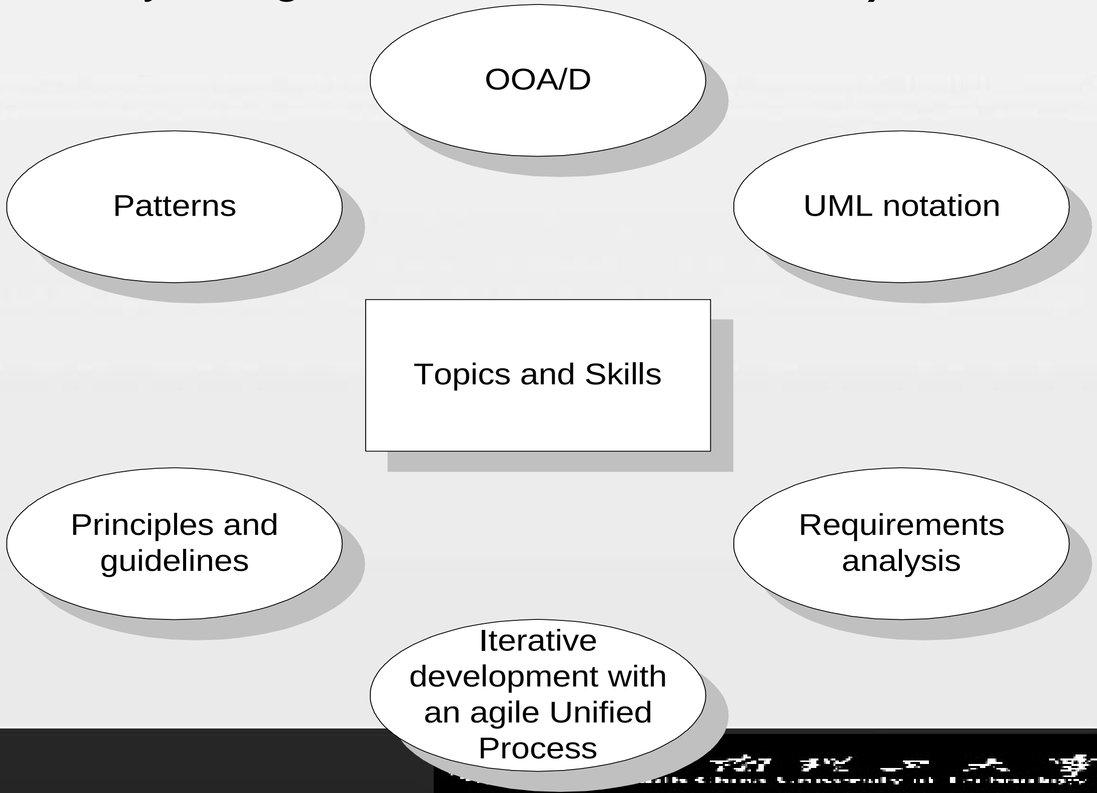
Contents

1 Introduction of Course

- **Prerequisite**
  - Software development experience, software engineering concept
- **About this course**
  - a undergraduate-level introduction to software requirements analysis, design and modeling.
  - a tool to help developers and students learn core skills in use-case based requirements analysis, object-oriented analysis and design (OOA/D).
- **This Course can:**
  - Master use-case based requirements analysis
  - Apply principles and patterns to create better object designs.
  - Iteratively follow a set of common activities in analysis and design, based on an agile approach to the UP as an example
  - Create frequently used diagrams in the UML notation.
  - Agile process (UP)

1 Introduction of Course

- **Contents by Major Topics**
  - **Agile Practices,UP, Iterative Development**
  - **OOA/D**
    - **Domain Model、Requirement Model、Analysis Model、Design Model**
    - **GRASP(**General Responsibility Assignment Software Patterns**)**
    - **GOF Design Patterns**
  - **UML**


OOA/D
Patterns
UML notation
Topics and Skills
Principles and guidelines
Requirements analysis
Iterative development with an agile Unified Process

1 Introduction of Course

- **Benefit:**
  - learn a process roadmap (UP)
  - Design well
  - learn UML for modeling
  - learn design patterns
  - learn from a realistic study、learn from experience
  - design frameworks

- **Schedule**
  - 理论课程 3-14周，English and Chinese, Course Hours:64.
  - 实验课 4次实验
  - 考试---英文试卷考试
  - 成绩组成:考试 60%、实验20%、作业20%
  - 学分：3.5分

1 Introduction of Course

- **Required Textbook**
  - Craig Larman,"Applying UML and Patterns: An Introduction to object-oriented Analysis and Design and iterative development”,Third Edition, Addison-Wesley, 2005
  - Object-Oriented Analysis and Design with Applications

- **Required Textbook(in Chinese)**
  - 邵维忠、杨芙清，面向对象的分析与设计，清华大学出版社，2013.
  - Craig Larman，UML和模式应用(原书第三版)，机械工业出版社,李洋等译

- Related Tools
  - UMLNET
  - IBM RSA (rational rose)
  - Enterprise Architecture

2 Introduction of OOA/D

- What will learn?
  - UML vs. Thinking in Objects
  - OOD: Principles and Patterns
    - responsibility-driven design
  - Use Cases (requirements analysis)
  - Iterative Development, Agile Modeling, and an Agile UP
  - Case Study
    - POS、Monoploy Game

  - emphasizes mastery of the fundamentals
    - how to assign responsibilities to objects,
    - frequently used UML notation,
    - common design patterns
    - framework design and architectural analysis

2 Introduction of OOA/D

- **The Most Important Learning Goal**
  - skillfully assign responsibilities to software objects.
  - Responsibilities? Method?Function?

  - fundamental principles.----GRASP
    - General responsibilities assignment software patterns (9)
    - Such as :
      - Information Expert
      - Creator

背黑锅我来，送死你去，拼全力为众生
……

2 Introduction of OOA/D

- What is Analysis and Design?
  - Analysis
    - emphasizes an investigation of the problem and requirements, rather than a solution
    - Such as :Requirements analysis、Architectural analysis
    - Do the right thing
  - Design
    - emphasizes a conceptual solution (in software and hardware) that fulfills the requirements, rather than its implementation
    - Such as: Object design, Database design, UI design
    - Do the thing right
  - Implement
    - Implement the design solution using programming language
    - programming language(TIOBE index)
    - Java, c, python, c++, c#,VB.NET,JS,PHP,SQL,swift,GO

2 Introduction of OOA/D

- What is Analysis and Design?
  - ?
    - The gap between analysis and design is sometimes very small
    - Analysis and design is in a continuous process
    - To a activity, some people think of it as analytical activity, but others think it is design

More close to analysis ⟷ More close to design

-what?
-requirement
-understanding of the business

-how?
-solution
-understanding and description of solution

2 Introduction of OOA/D

- **What is Object-Oriented Analysis and Design**
  - **Understanding the problem domain and solution from object (thing / concept) perspective**
  - **OOA:**
    - **finding and describing the concepts in (not software object or class) the problem domain.**
    - **OUTPUT:**
      - The concepts in the domain, the responsibility and the relation of concepts
  - **OOD**
    - **defining software objects based on the concepts in OOA**
    - **defining how the software objects collaborate to fulfill the requirements**

2 Introduction of OOA/D

■ **What is Object-Oriented Analysis and Design**

domain concept

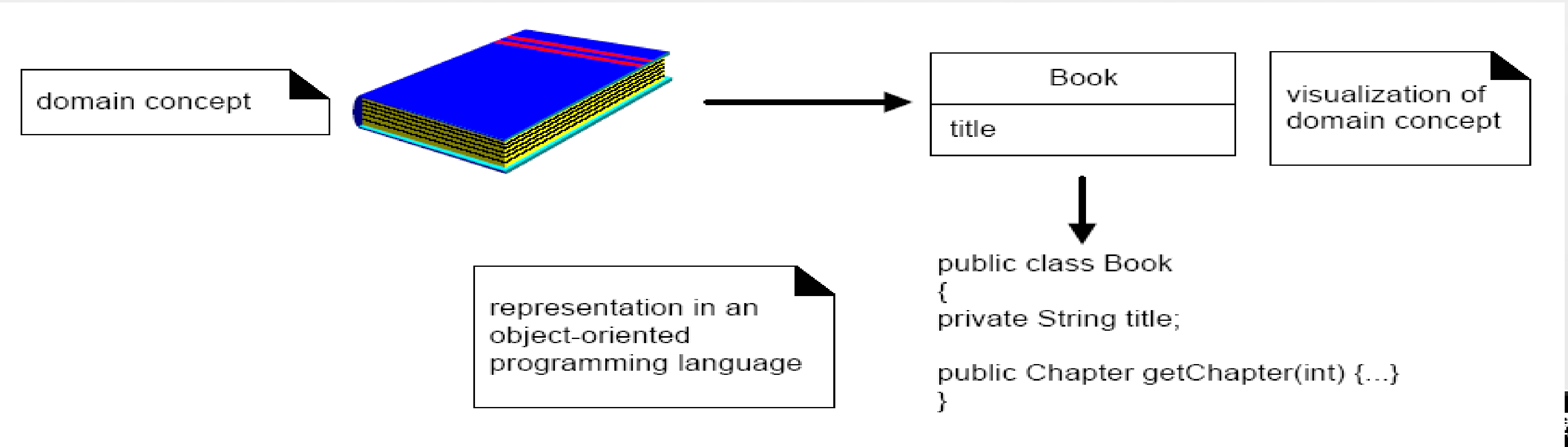
Plane | tailNumber

visualization of domain concept

representation in an object-oriented programming language

```
public class Plane
{
private String tailNumber;

public List getFlightHistory() {...}
}
```

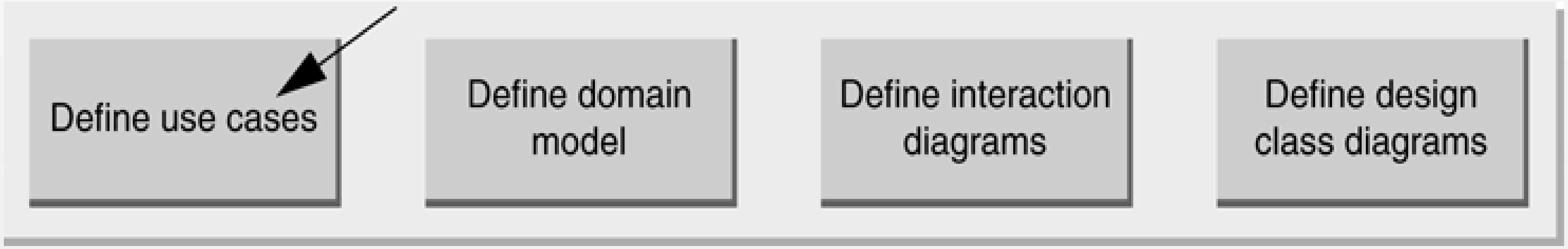

domain concept
Book
title
visualization of domain concept
representation in an object-oriented programming language

```
public class Book
{
private String title;

public Chapter getChapter(int) {...}
}
```

2 Introduction of OOA/D

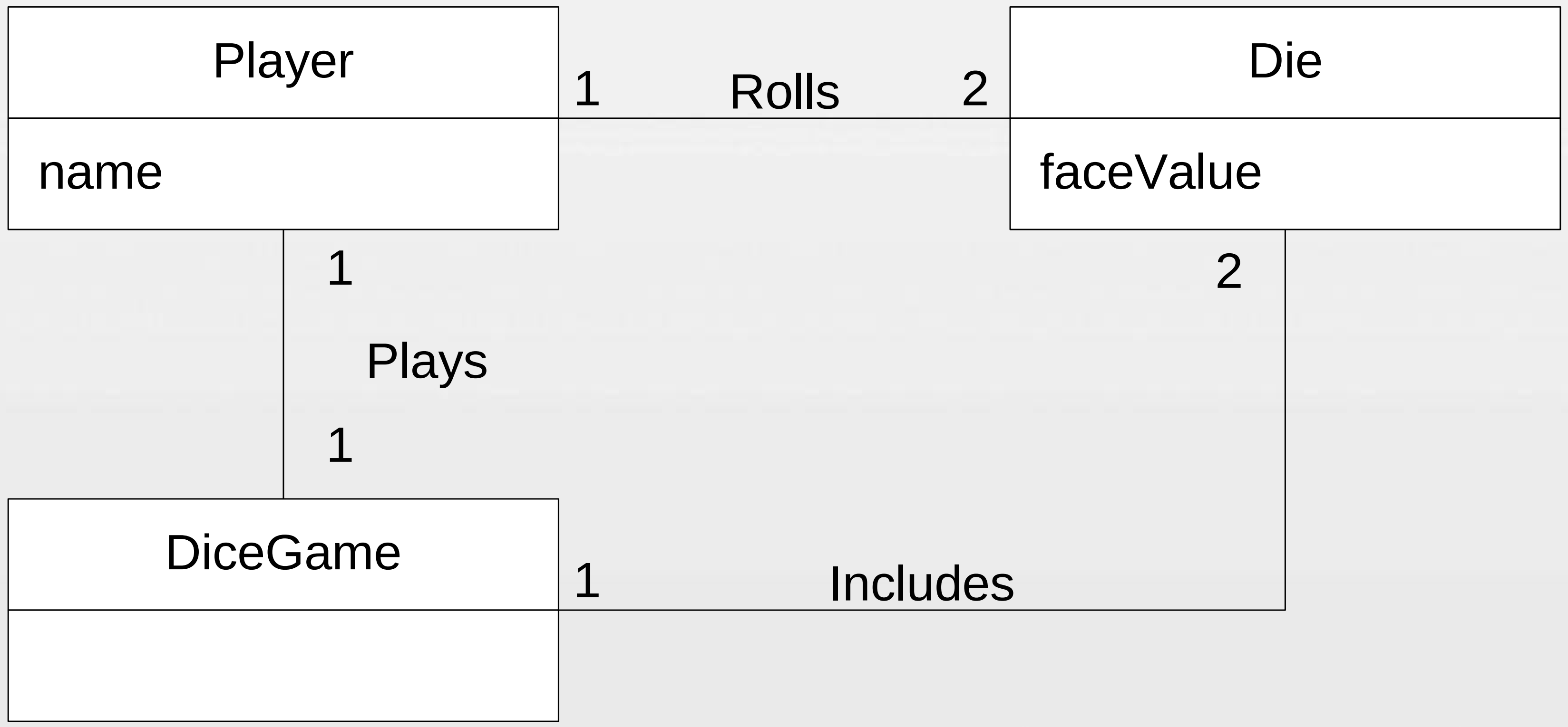
- A Short Example --dice game(骰子游戏,两个骰子的点数加起来为7赢，否则输)



- Define use cases –requirement
  - Use cases are simply written stories or scenarios of how people use the application
  - brief version of the Play a Dice Game use case
    - Player requests to roll the dice. System presents results: If the dice face value totals seven, player wins; otherwise, player lose

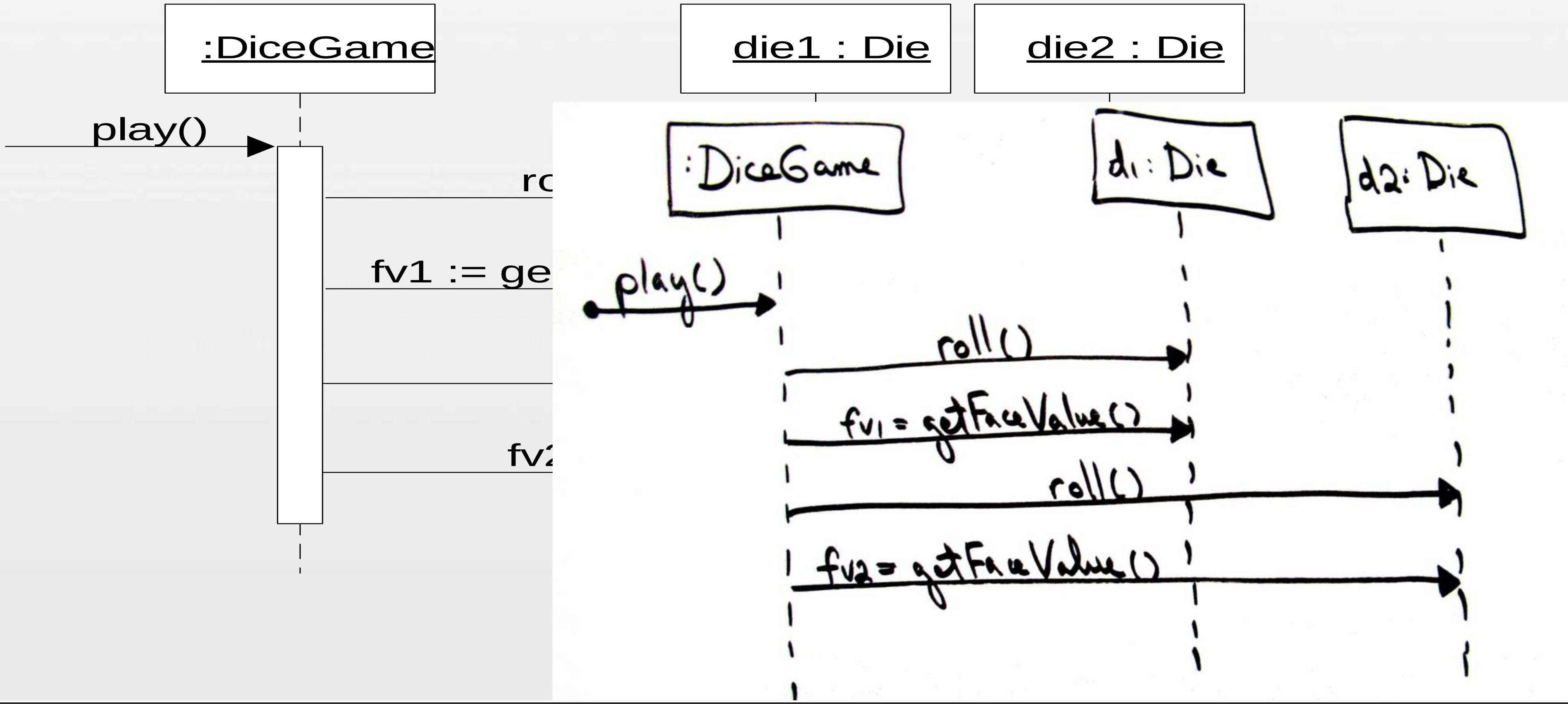
2 Introduction of OOA/D

- A Short Example --dice game
  - Domain model
    - identification of the concepts, attributes, and associations that are considered noteworthy
    - Not software object, conceptual object model



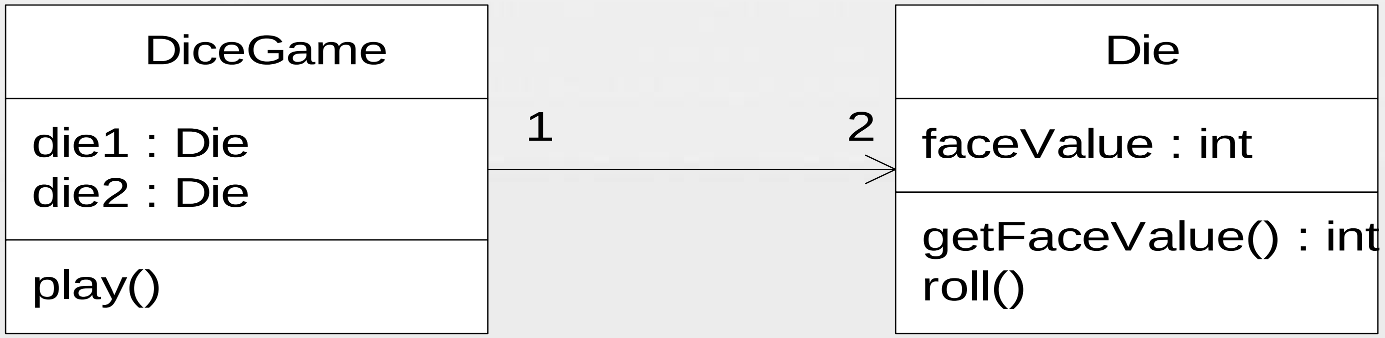
2 Introduction of OOA/D

- A Short Example --dice game
  - Assign Object Responsibilities and Draw Interaction Diagrams
    - OOD concerned with objects' responsibilities and collaborations
    - Use sequence diagram to illustrate these collaborations



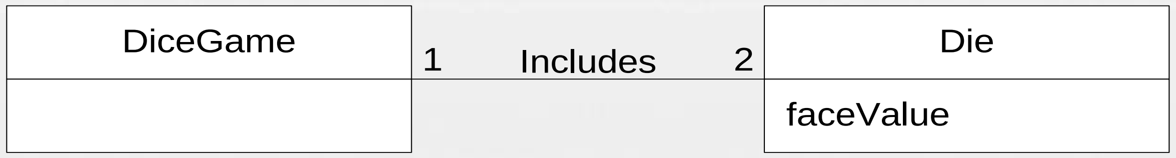
2 Introduction of OOA/D

- A Short Example --dice game
  - Define Design Class Diagrams
    - interaction diagrams is the dynamic view of collaborating objects
    - Use design class diagram to illustrates the attributes and methods of the classes .
    - static view of the class definitions



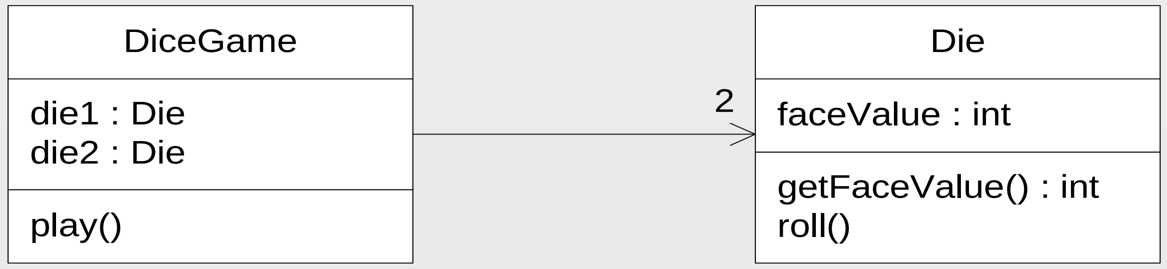
2 Introduction of OOA/D

- A Short Example --dice game
  - Difference between design class diagram and domain model
    - Different abstract level in the OOA/D
    - Domain model is the concept and the relations in the domain
    - Design class diagram is the software object, attributed and relation



Conceptual Perspective (domain model)

Raw UML class diagram notation used to visualize real-world concepts.



Specification or Implementation Perspective (design class diagram)

Raw UML class diagram notation used to visualize software elements.

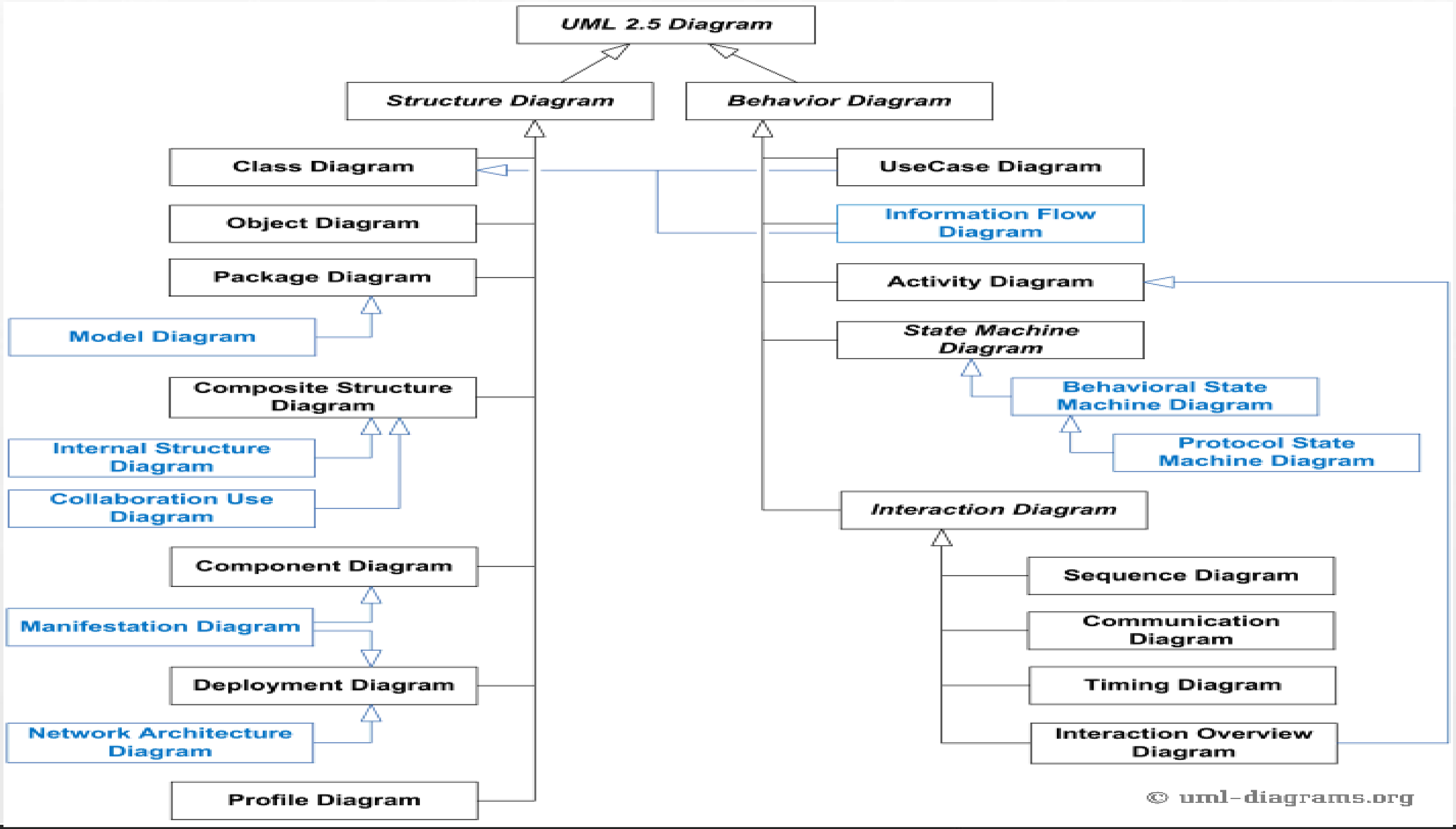
2 Introduction of OOA/D

- **What is UML**
  - is a visual language for specifying, constructing and documenting the artifacts of systems (in book)
  - An industry-standard graphic language to specifying, visualizing, constructing and documenting of software system
  - Model software systems in a number of diagrams(graphical notations to express the OOA/D of software)

  - Tools
    - Commercial :IBM RSA, Enterprise Architecture(EA)
    - Open source: Umlnet, ArgoUML, StarUML,etc

2 Introduction of OOA/D

■ What is UML



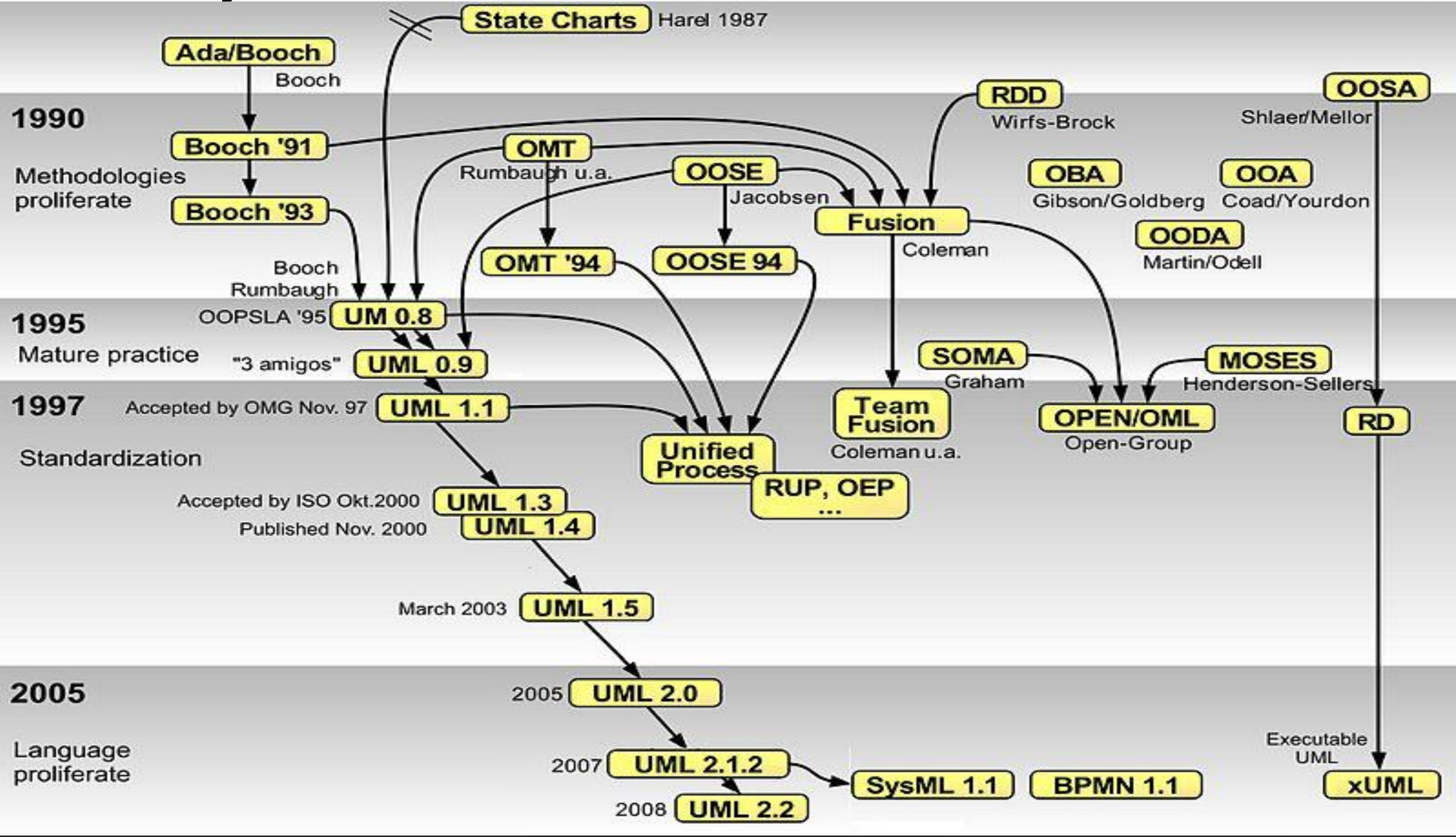
2 Introduction of OOA/D

- **Why UML?**
  - Visual Modeling is a good things. The purpose of modeling is to communicate.
  - Any model that cannot communicate effectively is useless(任何不能有效沟通的模型都是要流氓)

  - Why UML
    - Use graphic notation to communicate more clearly than nature language (imprecise) and code (too detailed)
    - Help acquire an overall views of a system
    - UML is not dependent on any one language or technology
    - UML move us from fragmentation to standardization

2 Introduction of OOA/D

- History of UML

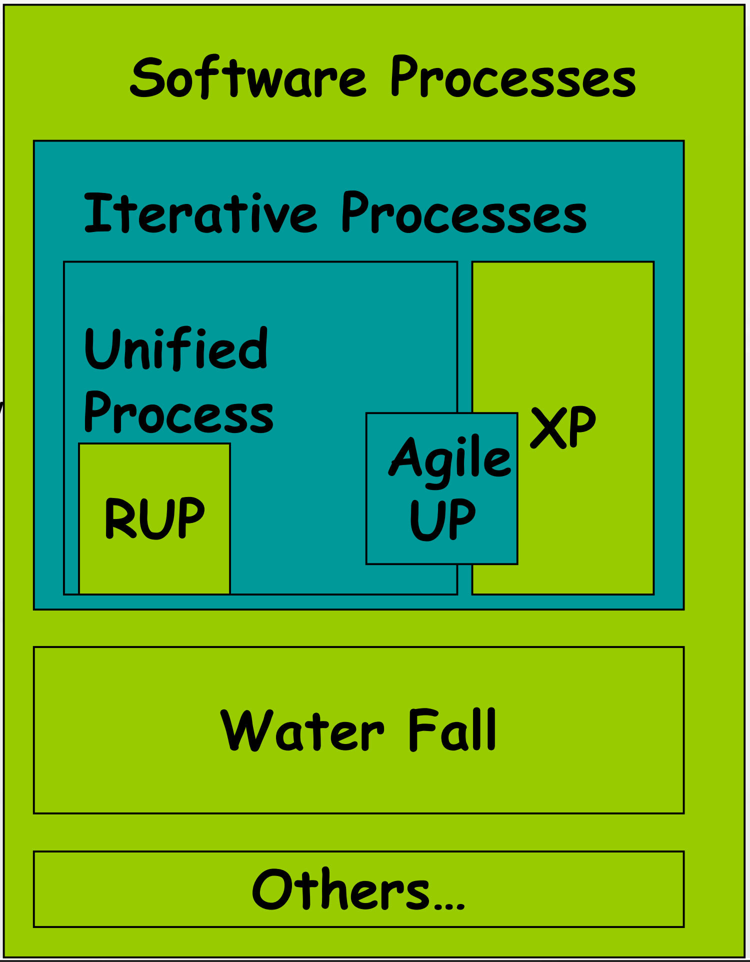


3 Iterative, Evolutionary, and Agile

- What is software development process
  - Definition
    - describes an approach to building, deploying, and possibly maintaining software
  - software process is software? (软件过程本身就是软件)
    - software process is software that is executed by a human-made virtual machine (由人构成的虚拟机执行的软件)
  - Why software process is so important?
    - But why software process is unimportant?

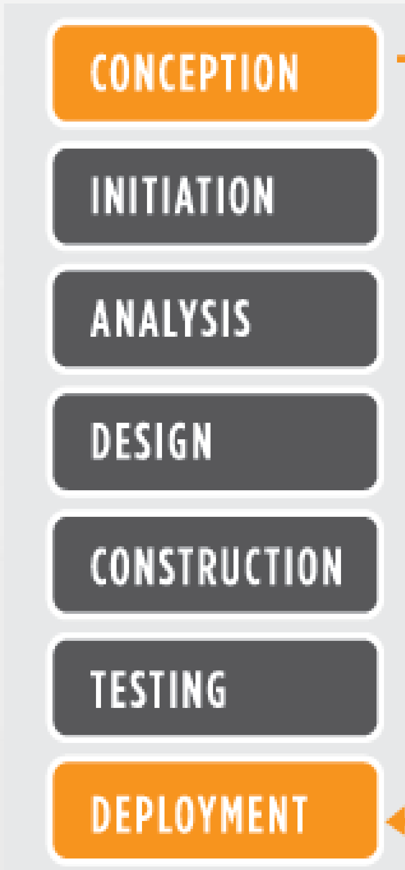
3 Iterative, Evolutionary, and Agile

- What is UP(Rational UP)
  - popular iterative software development process for building object-oriented systems
  - This course use UP because
    - UP is an iterative process.
    - UP practices provide an example structure for how to do and thus how to explain OOA/D
    - UP is flexible, and can be applied in a lightweight and agile approach
      - Extreme Programming (XP)
      - test-driven development, refactoring, continuous integration



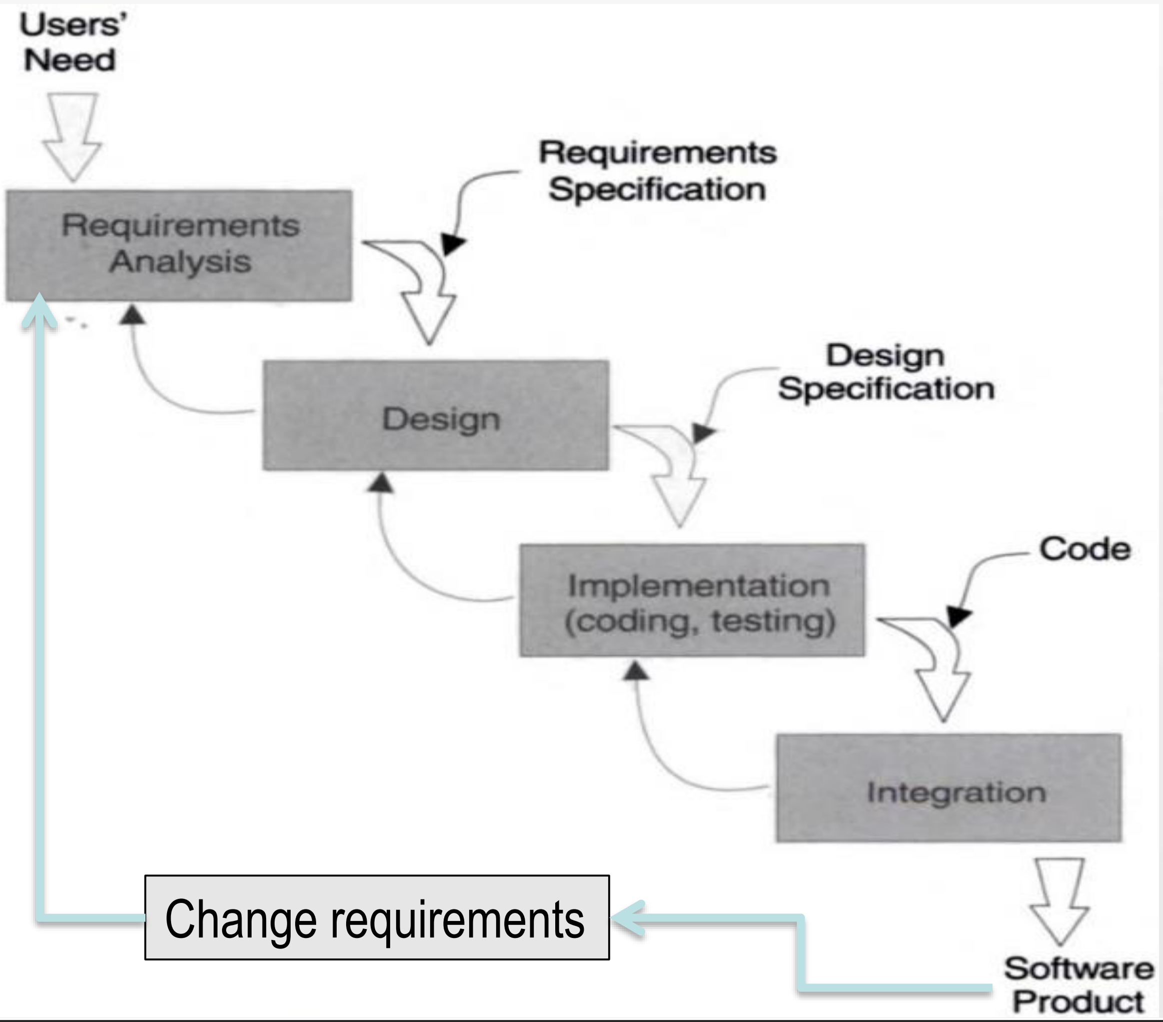
3 Iterative, Evolutionary, and Agile

- **What is Iterative and Evolutionary Development**
  - **Waterfall lifecycle**
    - ***Waterfall*** is a linear approach to software development
    - **attempt to define (in detail) all or most of the requirements before programming**
  - **pros and cons**
    - **45% of the features in waterfall requirements are never used**
    - **early waterfall schedules and estimates vary up to 400% from the final actuals**



3 Iterative, Evolutionary, and Agile

- **What is Iterative and Evolutionary Development**
  - ➢ **Waterfall lifecycle**



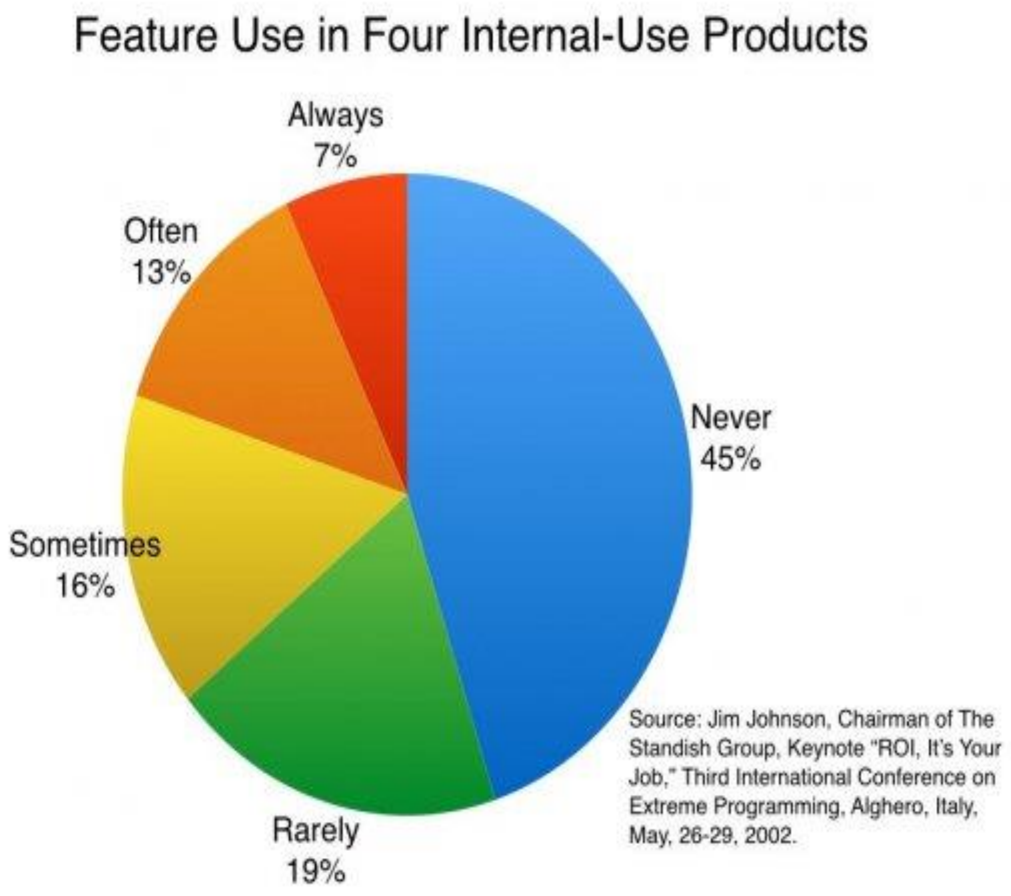
Key features of waterfall lifecycle

1. Many Documents.
2. Frozen requirement via contract, signed requirements are the basis for all the next works
3. Completed and detailed plan and budget
4. Users are the last to see software.

   users don't know what they want, but they know what they don't want when they set it.

3 Iterative, Evolutionary, and Agile

- **What is Iterative and Evolutionary Development**
  - **Waterfall lifecycle**



According to the Standish Group, an amazing 45 percent of features implemented are never used, while another 19 percent are used only rarely.

If features never used were not implemented in the first place, development time would be cut in about half.

3 Iterative, Evolutionary, and Agile

■ **What is Iterative and Evolutionary Development**

➢ **Waterfall lifecycle**

SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
2011-2015 Chaos Report				
All Size	Waterfull	11%	60%	29%
Large Size	Waterfull	3%	55%	42%
Medium Size	Waterfull	7%	68%	25%
Small Size	Waterfull	44%	45%	11%
2018 Chaos Report				
All Size	Waterfull	26%	53%	21%
Large Size	Waterfull	9%		
Medium Size	Waterfull	19%		
Small Size	Waterfull	59%		

Source Link

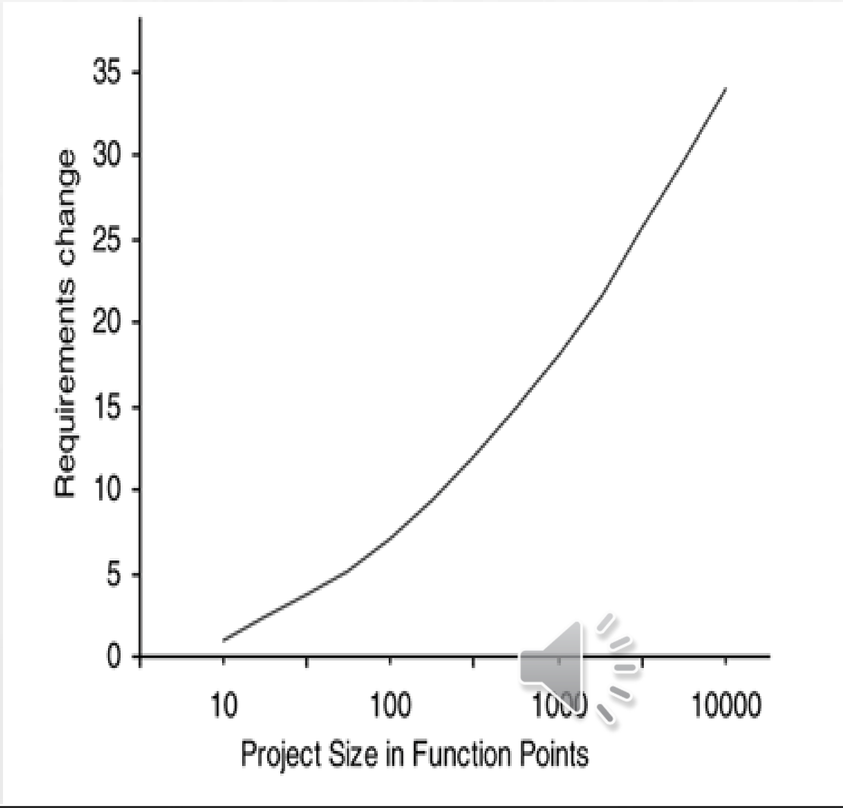
Successful:
The project is completed on-time and on-budget, with all features and functions as initially specified

Challenged
The project is completed and operational but over-budget, over the time estimate, and offers fewer features and functions than originally specified.

Failed
The project is cancelled at some point during the development cycle

3 Iterative, Evolutionary, and Agile

- **What is Iterative and Evolutionary Development**
  - **Why waterfall lifecycle so failure??**
    - **Wrong Assumption 1 : Requirements will be frozen?** **Requirement will change.** typical software experienced a 25% change in requirements , 35%-40% for large projects.



3 Iterative, Evolutionary, and Agile

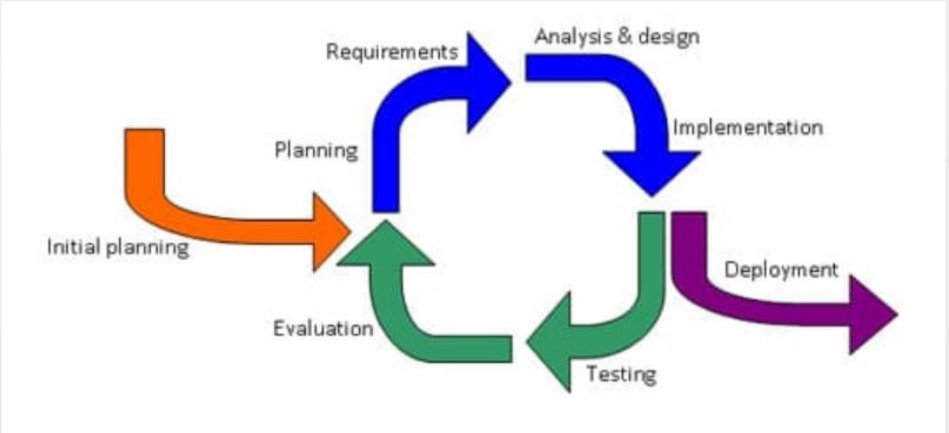
- **What is Iterative and Evolutionary Development**
  - Why waterfall lifecycle so Failure-Prone? **Why requirement change frequently?**
    - **The users change**
      - Users see other system
      - Work environment evolves.
    - **The problem change**
      - users don't know what they want exactly, but they know what they don't want when they set it
      - Efforts to detail, capture and freeze requirement may lead to wrong system at the initial phase
    - **The underlying technology changed**
      - New software, hardware and technology emerge and you will want to adopt them.
    - **The marker change frequently, so requirement should change to match.**

3 Iterative, Evolutionary, and Agile

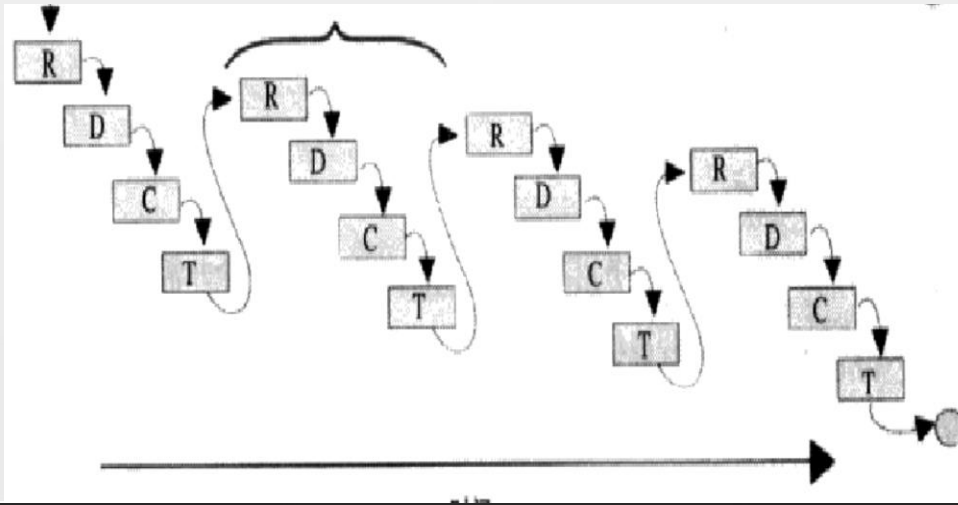
- **What is Iterative and Evolutionary Development**
  - **Why waterfall lifecycle so Failure-Prone?**
    - **Wrong Assumption 2 : We can get the right design on paper before processing**
      - The right design only can achieved by using formal methods in the design phase, but few of formal technologies are readily available.
      - What we have are documents, the design solutions can only be verified by reviewing the documents
    - **Other Problem to large projects**
      - Large project always have long time term.
      - The quality feedback always very long
      - The designer only do each activity once, so they don’t have chance to learn from their mistakes.

3 Iterative, Evolutionary, and Agile

- **What is Iterative and Evolutionary Development**
  - **How to improve success rate for large projects?**
    - waterfall lifecycle was successful for small size projects
    - users don't know what they want exactly, but they know what they don't want when they set it.
  - **Iterate is coming**
    - Break down the lifecycle of large projects into a succession of small waterfall lifecycle.
    - Let users see software early, they exactly know what they don't want

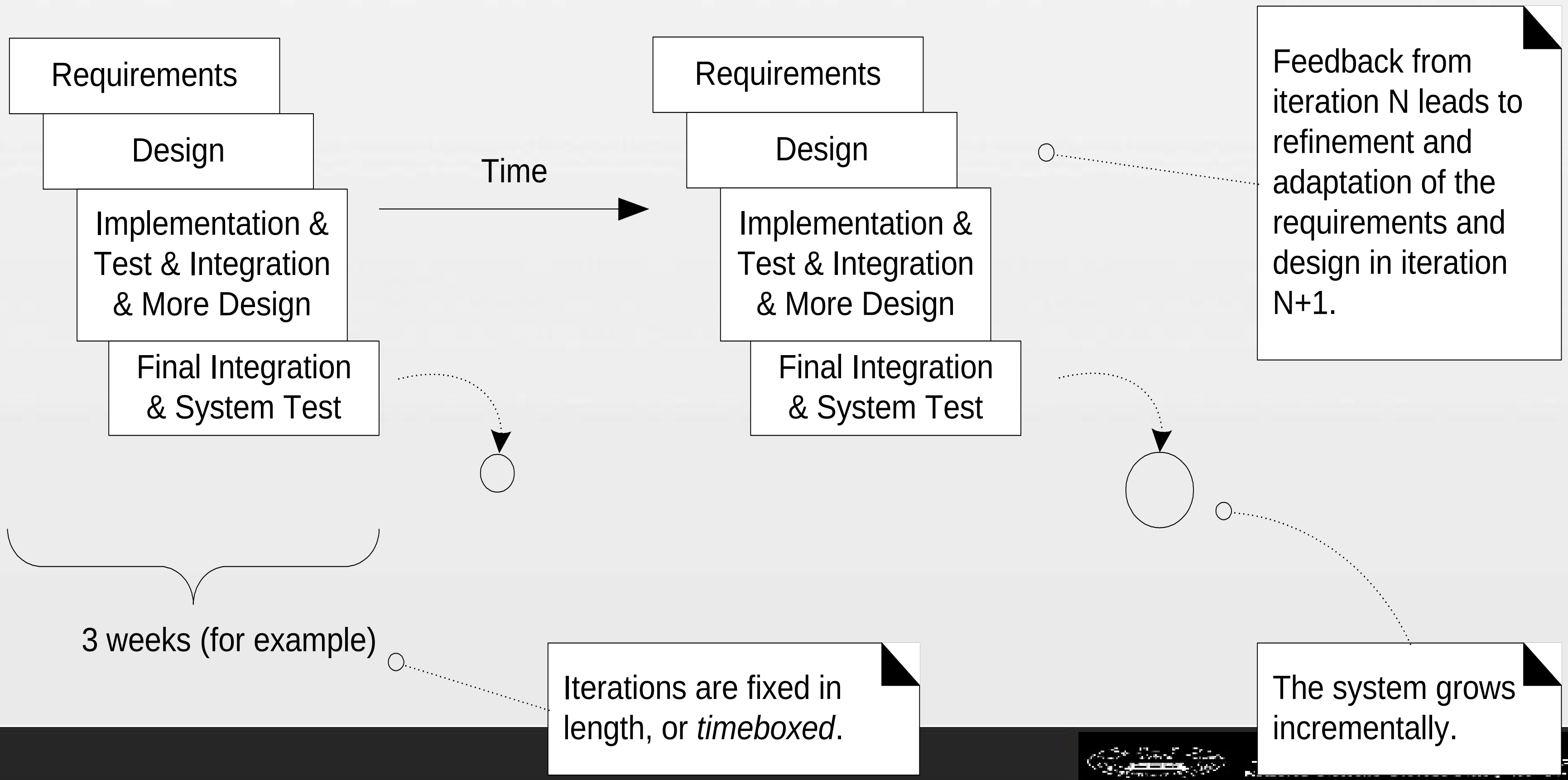


Model 1: Typical iterative development process



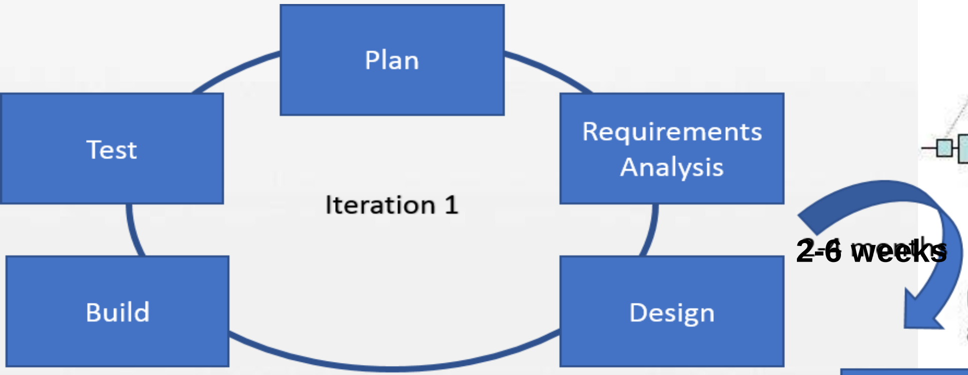
3 Iterative, Evolutionary, and Agile

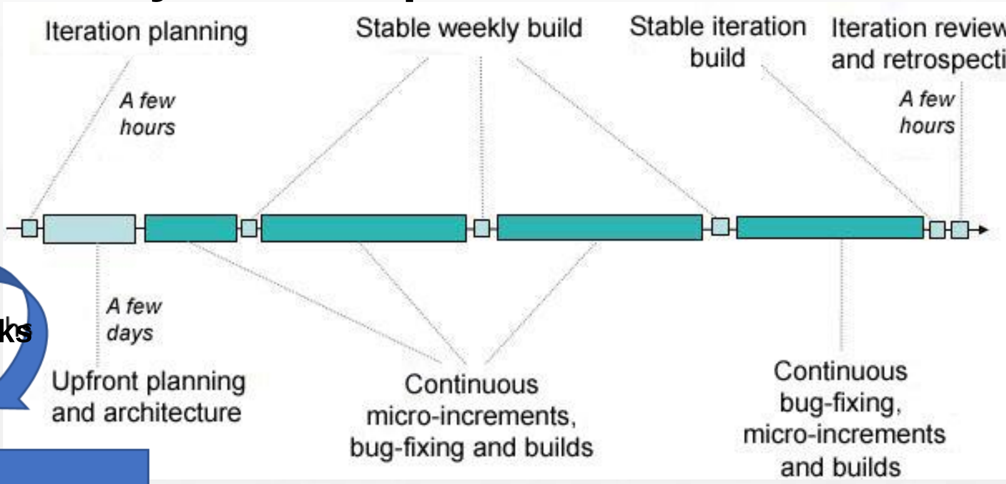
- **What is Iterative and Evolutionary Development**
  - development is organized into a series of short, fixed-length (for example, three-week) mini-projects called iterations
  - successive enlargement and refinement of a system through multiple iterations

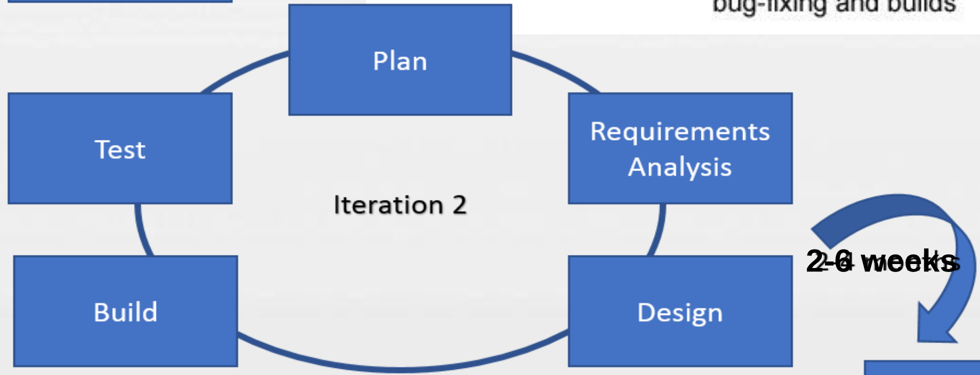

Requirements
Design
Implementation & Test & Integration & More Design
Final Integration & System Test
Time
Requirements
Design
Implementation & Test & Integration & More Design
Final Integration & System Test
Feedback from iteration N leads to refinement and adaptation of the requirements and design in iteration N+1.
3 weeks (for example)
Iterations are fixed in length, or *timeboxed*.
The system grows incrementally.

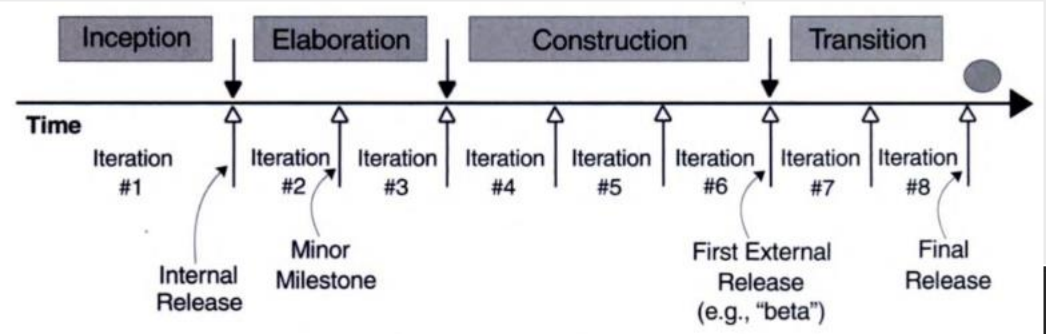
3 Iterative, Evolutionary, and Agile

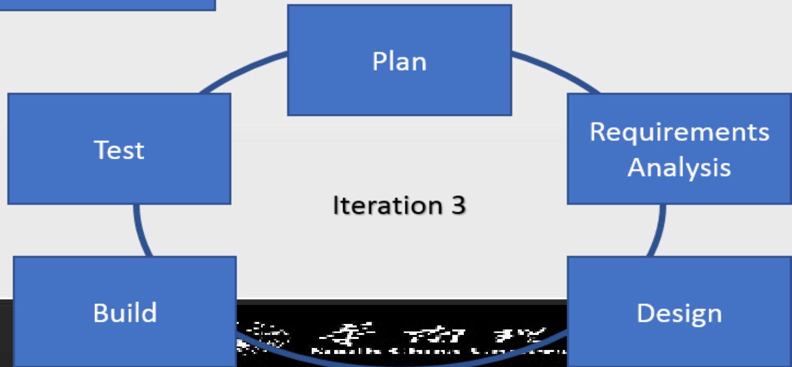
- **What is Iterative and Evolutionary Development**
  - ➢ Iterative lifecycle




Iteration planning
Stable weekly build
Stable iteration build
Iteration review and retrospecti
A few hours
A few hours
A few days
Upfront planning and architecture
Continuous micro-increments, bug-fixing and builds
Continuous bug-fixing, micro-increments and builds

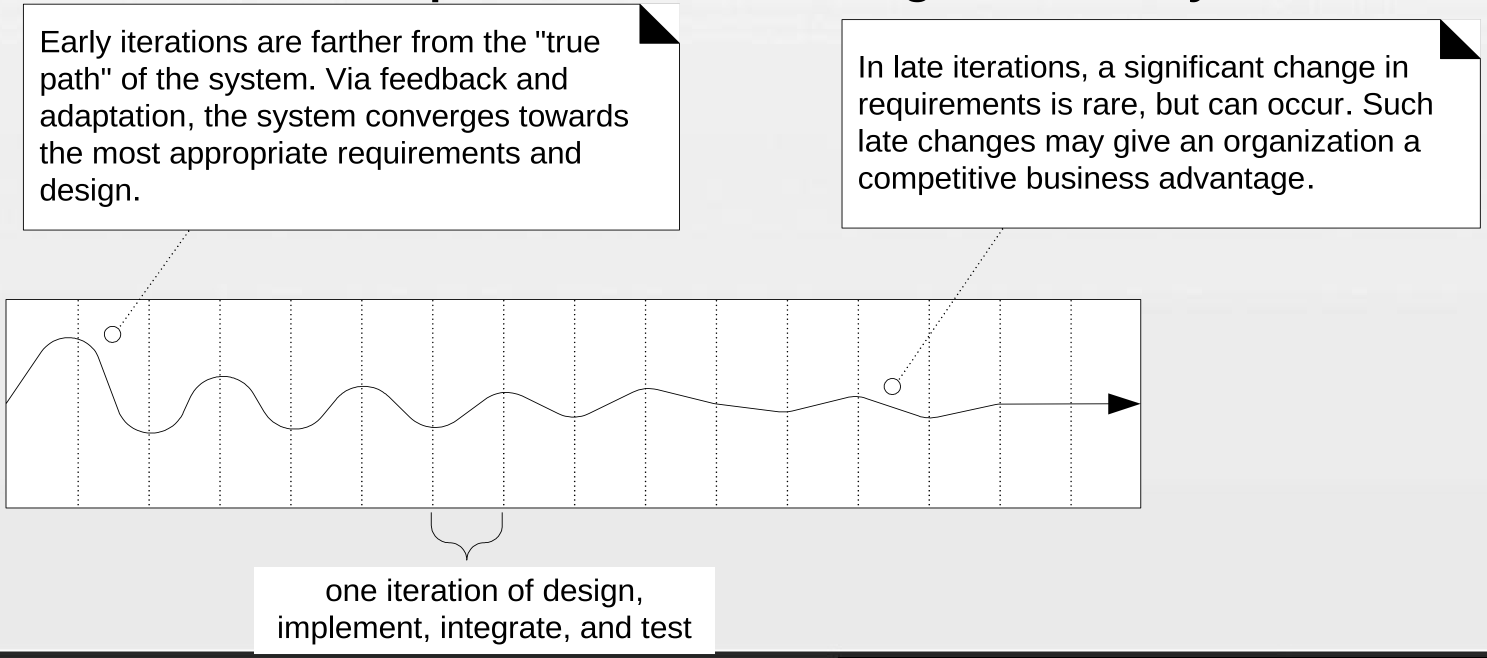






3 Iterative, Evolutionary, and Agile

- **What is Iterative and Evolutionary Development**
  - **Each iteration**
    - **choosing a small subset of the requirements**
    - **quickly designing, implementing, and testing**
    - **feedback from the users, developers, and tests**
  - **early iterations requirements and design may not be exactly, but after many iterations, the requirements and design instability lowers over time**


Early iterations are farther from the "true path" of the system. Via feedback and adaptation, the system converges towards the most appropriate requirements and design.
In late iterations, a significant change in requirements is rare, but can occur. Such late changes may give an organization a competitive business advantage.
one iteration of design, implement, integrate, and test

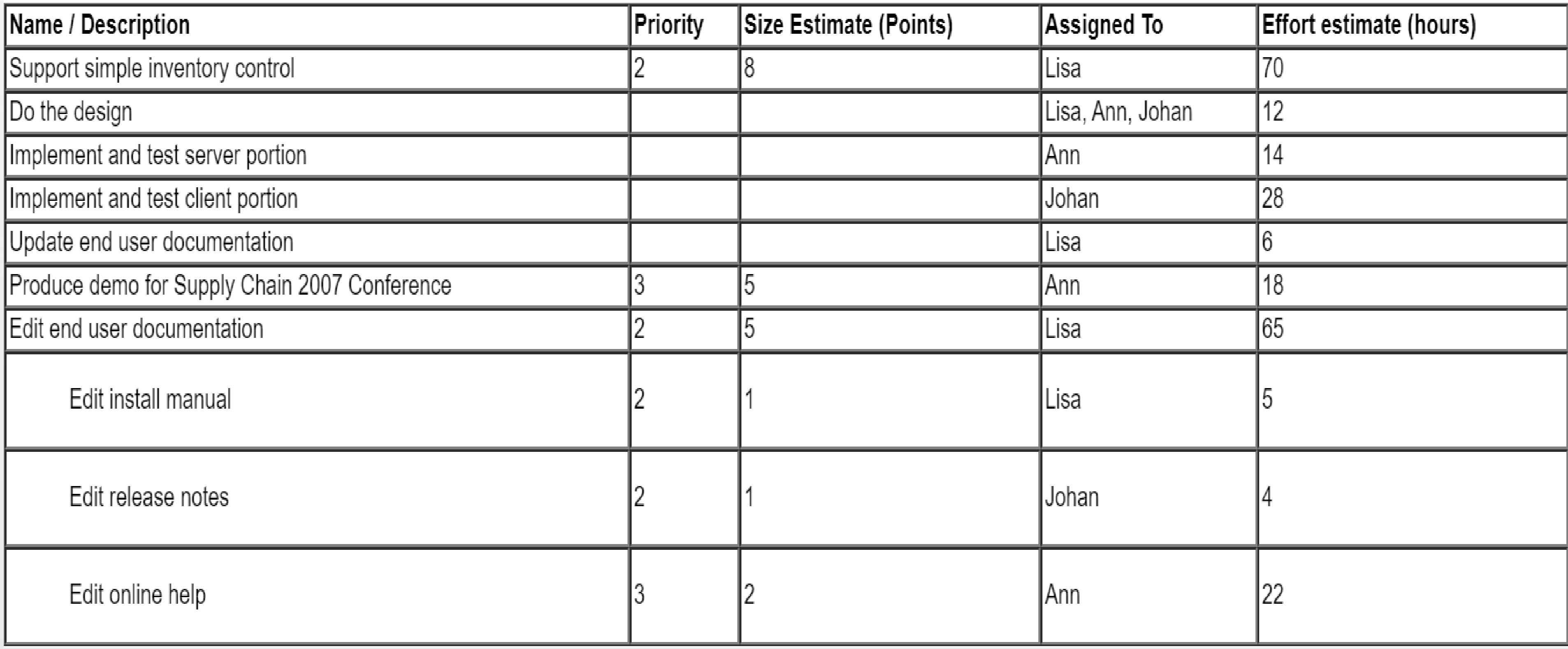
3 Iterative, Evolutionary, and Agile

- **What is Iterative and Evolutionary Development –Example**
  - **a three-week iteration**
    - **Monday morning -- one hour kickoff meeting**
      - clarifying the tasks and goals of the iteration
      - One person, reverse-engineers the last iteration's code into UML diagrams, and prints and displays noteworthy diagrams
    - **remainder of Monday (working in pairs)**
      - sketching rough UML diagrams, and writing some pseudocode and design notes
    - **remaining days**
      - implementation, testing (unit, acceptance, usability, …), further design, integration, and daily builds of the partial system.

  - **The result of each iteration is an executable but incomplete system**

3 Iterative, Evolutionary, and Agile

- **What is Iterative and Evolutionary Development**
  - **Each iteration**
    - **Iterative plan**



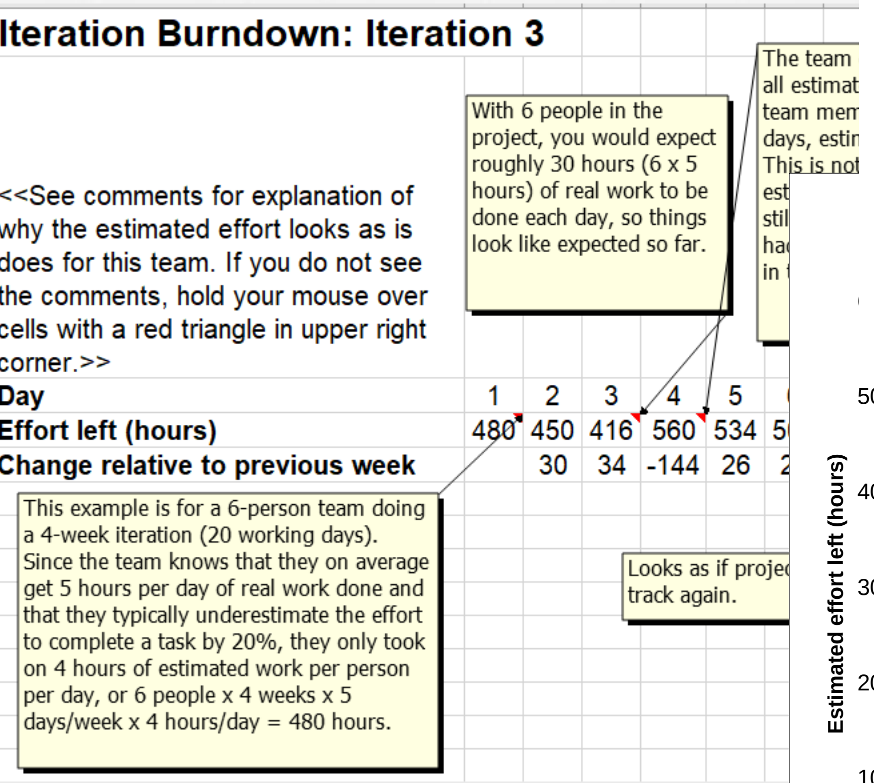
Name / Description	Priority	Size Estimate (Points)	Assigned To	Effort estimate (hours)
Support simple inventory control	2	8	Lisa	70
Do the design			Lisa, Ann, Johan	12
Implement and test server portion			Ann	14
Implement and test client portion			Johan	28
Update end user documentation			Lisa	6
Produce demo for Supply Chain 2007 Conference	3	5	Ann	18
Edit end user documentation	2	5	Lisa	65
Edit install manual	2	1	Lisa	5
Edit release notes	2	1	Johan	4
Edit online help	3	2	Ann	22

– What is point?

– What is efforts

3 Iterative, Evoluti

- What is Iterative and Evol
  - Each iteration
    - Iteration Burn-down Re



Iteration Burndown: Iteration 3

<<See comments for explanation of why the estimated effort looks as is does for this team. If you do not see the comments, hold your mouse over cells with a red triangle in upper right corner.>>

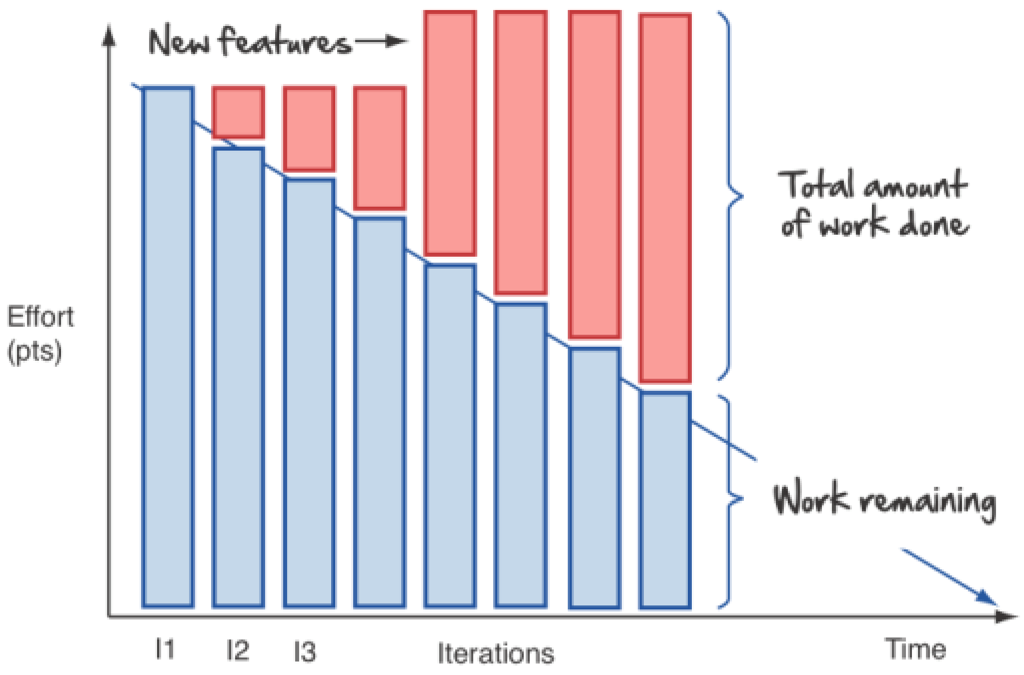
Day	1	2	3	4	5	(
Effort left (hours)	480	450	416	560	534	5
Change relative to previous week		30	34	-144	26	2

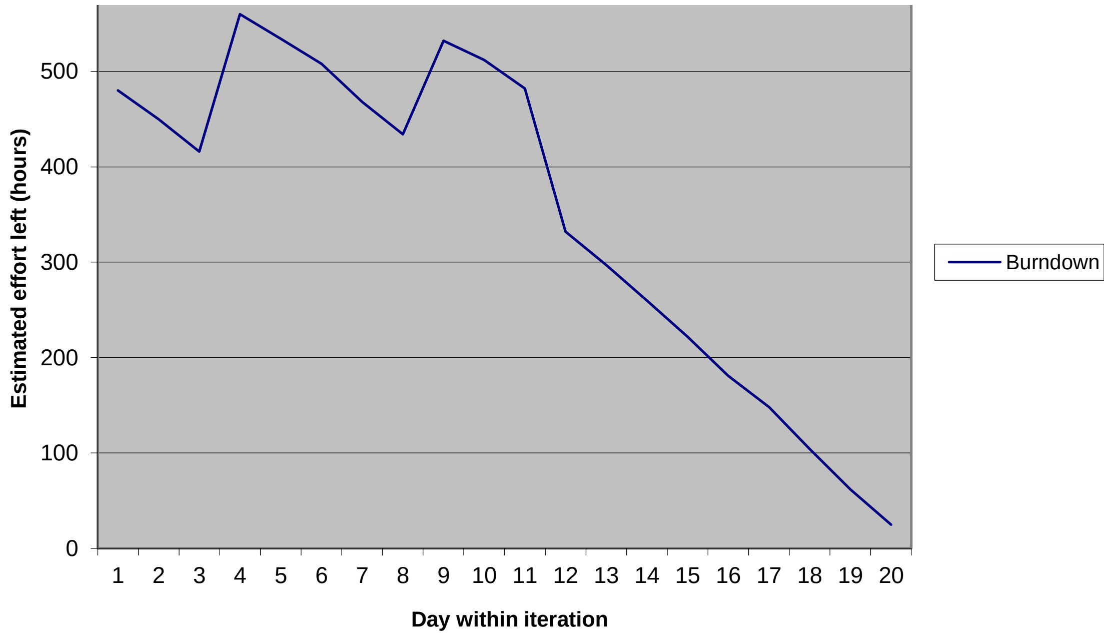
With 6 people in the project, you would expect roughly 30 hours (6 x 5 hours) of real work to be done each day, so things look like expected so far.

The team
all estimat
team mem
days, estir
This is not
est
stil
hac
in t

This example is for a 6-person team doing a 4-week iteration (20 working days). Since the team knows that they on average get 5 hours per day of real work done and that they typically underestimate the effort to complete a task by 20%, they only took on 4 hours of estimated work per person per day, or 6 people x 4 weeks x 5 days/week x 4 hours/day = 480 hours.

Looks as if projec
track again.





3 Iterative, Evolutionary, and Agile

- **What is Iterative and Evolutionary Development**
  - **Iteration Length**
    - typically 4 weeks long, although some teams will work with iterations as short as a week or as long as six weeks.
  - factors driving iteration length,

Factors leading to reduced iteration length	Factors leading to increased iteration length
Small teams	Large teams
Co-located teams	Distributed teams
Strong configuration management system	Poor configuration management system
Dedicated, full-time resources	Matrixed or part-time resources
Automated testing	Lack of automated testing
Integrated tool environment	Absence of good automation and tool integration
Team experienced with iterative development	Team inexperienced with iterative development
Fast decision making	Policies and bureaucracy preventing fast decision ma
Unclear requirements	Well-understood requirements
Unclear or brittle architecture	Well-defined and stable architecture
New and poorly understood technology	Well-understood technology

3 Iterative, Evolutionary, and Agile

- **What is Iterative and Evolutionary Development**
  - How to estimate the efforts(size) of project? Not in this course
    - Code Lines, KLOC
    - Waterfall lifecycle
      - Structured Method
        - **function point method**
          - The function point is a "unit of measurement" to express the amount of business functionality an information system (as a product) provides to a user
      - OO Method
        - Use case point
          - to forecast the software size when the UML are used
    - Iterative lifecycle
      - Use **story points** to estimate the size of **user story.**

3 Iterative, Evolutionary, and Agile

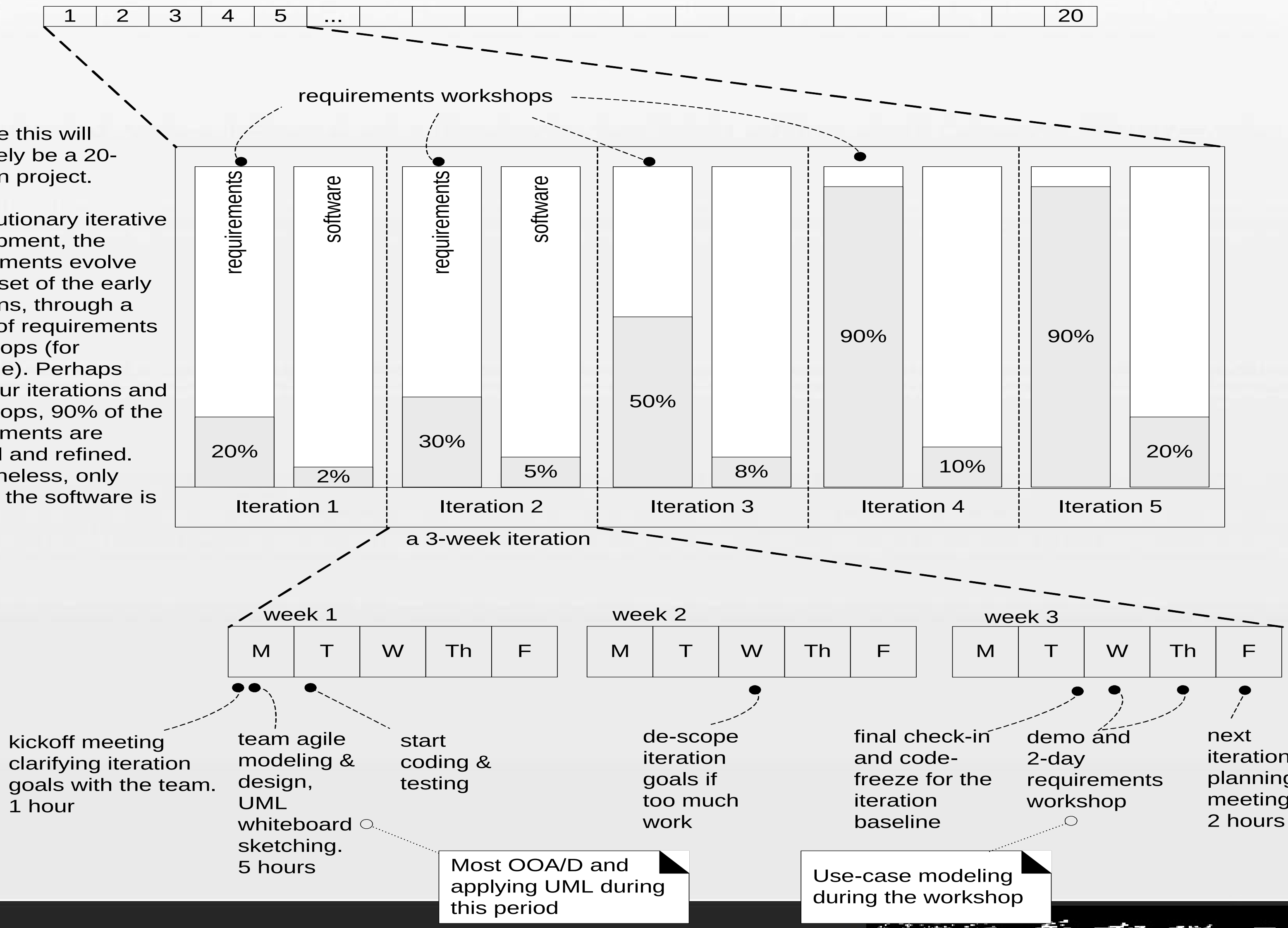
- **How to do Iterative and Evolutionary Analysis and Design**
  - Before iteration-1, hold the first time-boxed requirements workshop
    - pick 10% requirement from this high-level list (architecturally significant, high business value , high risk),do intensive detailed analysis of the requirements
    - Before iteration-1, hold an iteration planning meeting
  - Do iteration-1 over three or four weeks (pick the timebox, and stick to it).
    - Do the second requirements workshop near the end of iteration-1, pick another 10%-15% requirement
    - On Friday morning, hold another iteration planning meeting for the next iteration
  - Do iteration-2; similar steps
  - Repeat, 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
  - …..

3 Iterative, Evolutionary, and Agile

■ How to do Iterative and Evolutionary Analysis and Design

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.



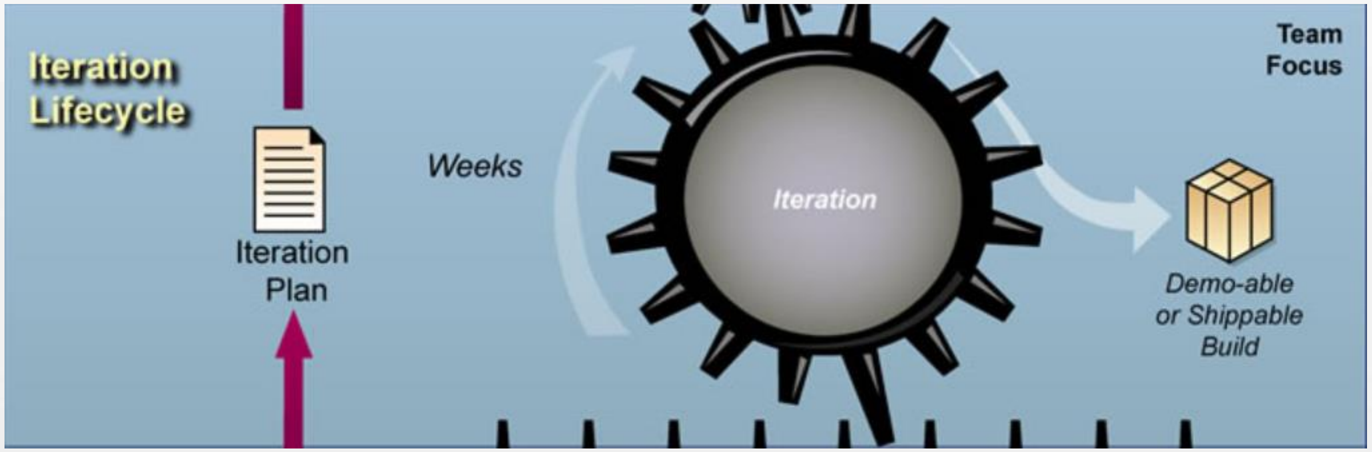
1
2
3
4
5
...
20
requirements workshops
requirements
software
20%
2%
30%
5%
50%
8%
90%
10%
90%
20%
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
a 3-week iteration
week 1
week 2
week 3
M
T
W
Th
F
kickoff meeting clarifying iteration goals with the team. 1 hour
team agile modeling & design, UML whiteboard sketching. 5 hours
start coding & testing
de-scope iteration goals if too much work
final check-in and code-freeze for the iteration baseline
demo and 2-day requirements workshop
next iteration planning meeting; 2 hours
Most OOA/D and applying UML during this period
Use-case modeling during the workshop

3 Iterative, Evolutionary, and Agile

- What is Iterative and Evolutionary Development
  - Benefits of 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 leading to a refined system that more closely meets the real needs of the stakeholders
    - managed complexity
    - the learning within an iteration can be methodically used to improve the development process itself, iteration by iteration
  - key idea
    - timeboxed, or fixed in length.
    - recommend an iteration length between two and six weeks

3 Iterative, Evolutionary, and Agile

- What is Iterative and Evolutionary Development
  - Summary of Iterative lifecycle



- ● Time-boxed, or fixed in length, time unit is weeks
- ● Demo-able or shippable build
- ● Evolutionary or (micro-)incremental development

Iterative development is the core of agile methods.

3 Iterative, Evolutionary, and Agile

■ **What are Agile Methods**

- **not possible to exactly define agile methods**
- **short timeboxed iterations with evolutionary refinement of plans, requirements, and design is a basic practice the methods**

- **The Agile Manifesto(宣言)**

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**
  - **Our highest priority is to satisfy the customer through early and continuous delivery of valuable software**
  - **Welcome changing requirements, even late in development**

3 Iterative, Evolutionary, and Agile

- What are Agile Methods
  - The Agile Principles
    - Deliver working software frequently
    - Business people and developers must work together daily throughout the project
    - The most efficient and effective method is face-to-face conversation.
    - Working software is the primary measure of progress.
    - Continuous attention to technical excellence and good design
    - Simplicity---the art of maximizing the amount of work not done is essential
    - The best architectures, requirements, and designs emerge from self-organizing teams.
    - ….

3 Iterative, Evolutionary, and Agile

- What is Agile Modeling
  - The purpose of modeling is to understand, not to document
  - Agile modeling implies a number of practices and values, including
    - The purpose of modeling and models is to support understanding and communication, not documentation
    - Don‘t model all or most of the software design， only to unusual, difficult, tricky parts of the design space
    - Use the simplest tool possible (prefer sketching UML on whiteboards, and capturing the diagrams with a digital camera?)
    - Don't model alone, model in pairs (or triads) at the whiteboard
    - Know that all models will be inaccurate
    - Developers themselves should do the OO design modeling
    - ….

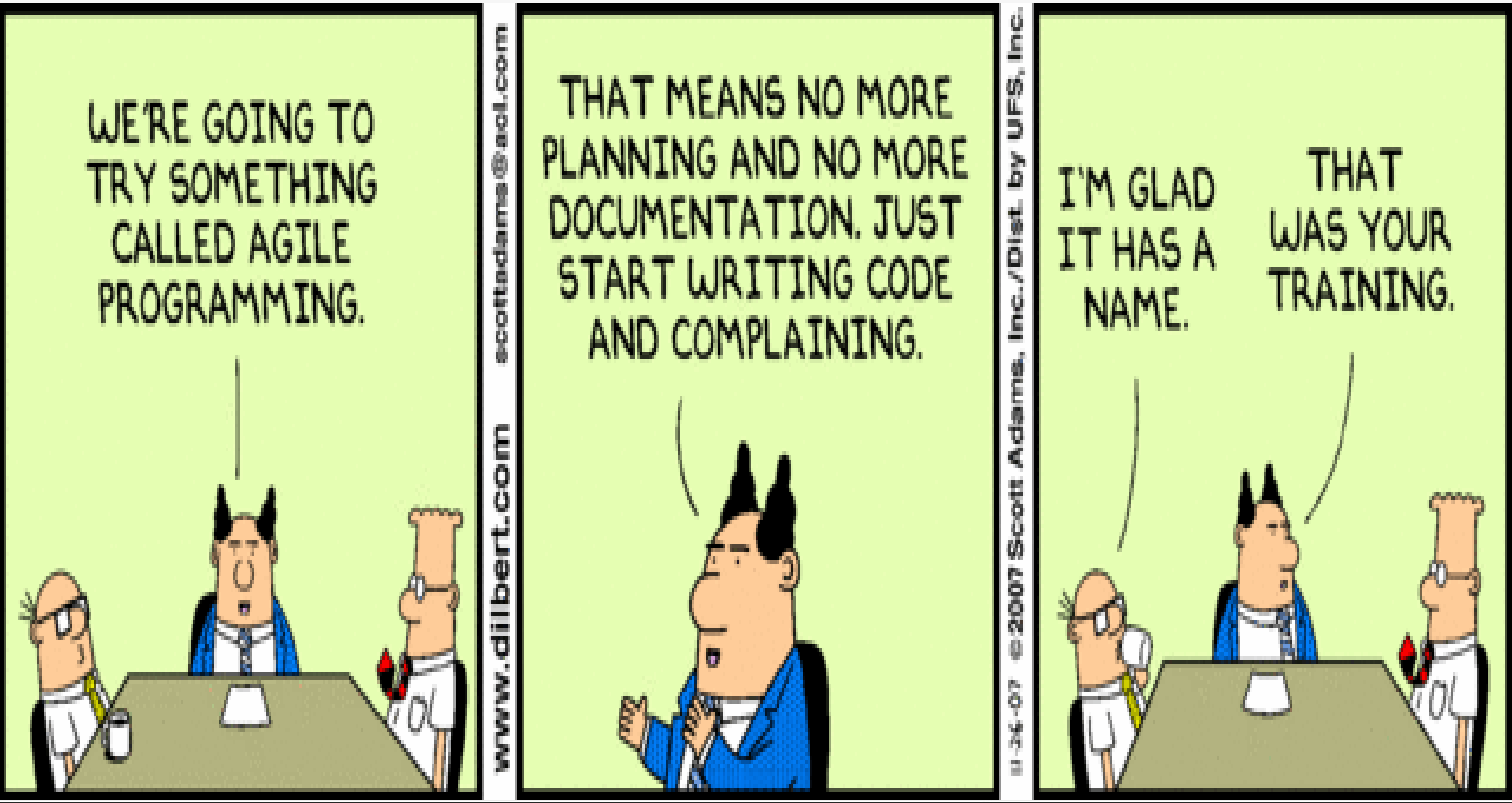
3 Iterative, Evolutionary, and Agile

- **What are Agile Methods?**
  - Difference of agile methods and traditional methods

Feature	Agile Methods	Traditional methods
Project Size	Large	Small
Plan	Short term and coarse	Long Term and detailed
Documents	Light, little	Heavy, many
Design	Simple	Complexed and detailed
Delivery	Many delivery	Seldom
Emphasis	People , more interactions	Process, more tools
Conform to	Users needs	Requirements
Relation to Users	Collaboration	contract negotiation

3 Iterative, Evolutionary, and Agile

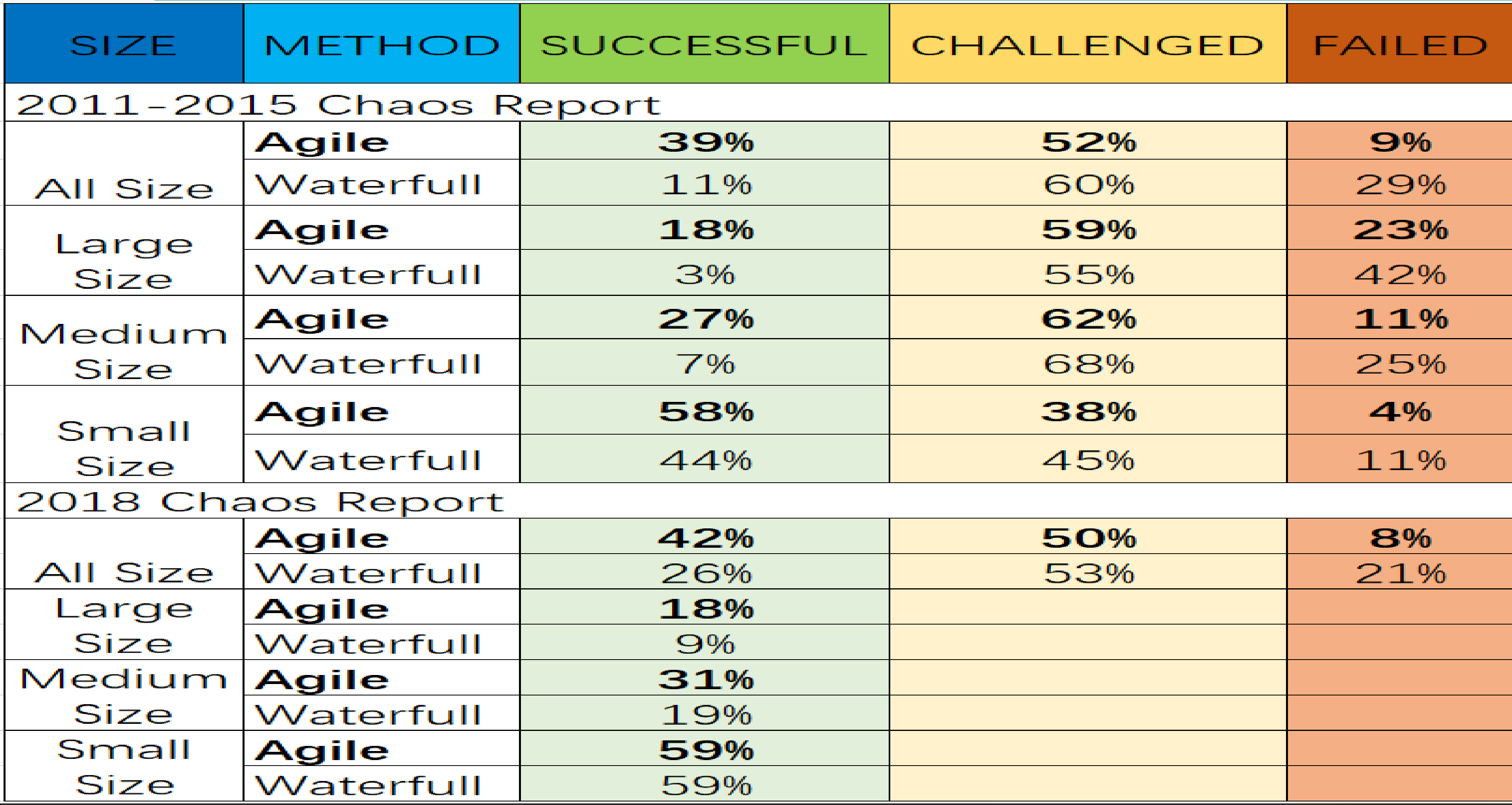
- What are Agile Methods
  - no more documents, no more plan, no more design, just coding?


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.

3 Iterative, Evolutionary, and Agile

What is Agile Modeling

➢ Agile Project Success Rates are 2X Higher than Traditional Projects 201



SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
2011-2015 Chaos Report				
All Size	**Agile**	**39%**	**52%**	**9%**
	Waterfull	11%	60%	29%
Large Size	**Agile**	**18%**	**59%**	**23%**
	Waterfull	3%	55%	42%
Medium Size	**Agile**	**27%**	**62%**	**11%**
	Waterfull	7%	68%	25%
Small Size	**Agile**	**58%**	**38%**	**4%**
	Waterfull	44%	45%	11%
2018 Chaos Report				
All Size	**Agile**	**42%**	**50%**	**8%**
	Waterfull	26%	53%	21%
Large Size	**Agile**	**18%**		
	Waterfull	9%		
Medium Size	**Agile**	**31%**		
	Waterfull	19%		
Small Size	**Agile**	**59%**		
	Waterfull	59%		

3 Iterative, Evolutionary, and Agile

- What is an UP(**Unified Process**)
  - **UP is one of the Iterative and Evolutionary development**
  - **Sample structure to apply OOA/D and UML**
  - **central idea of UP**
    - short time-boxed iterative, evolutionary, and adaptive development
  - additional best practices and key concepts in the UP
    - tackle high-risk and high-value issues in early iterations
    - continuously engage users for evaluation, feedback, and requirements
    - build a cohesive, core architecture in early iterations
    - continuously verify quality; test early, often, and realistically
    - do some visual modeling (with the UML)
    - …..

3 Iterative, Evolutionary, and Agile

- **UP 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

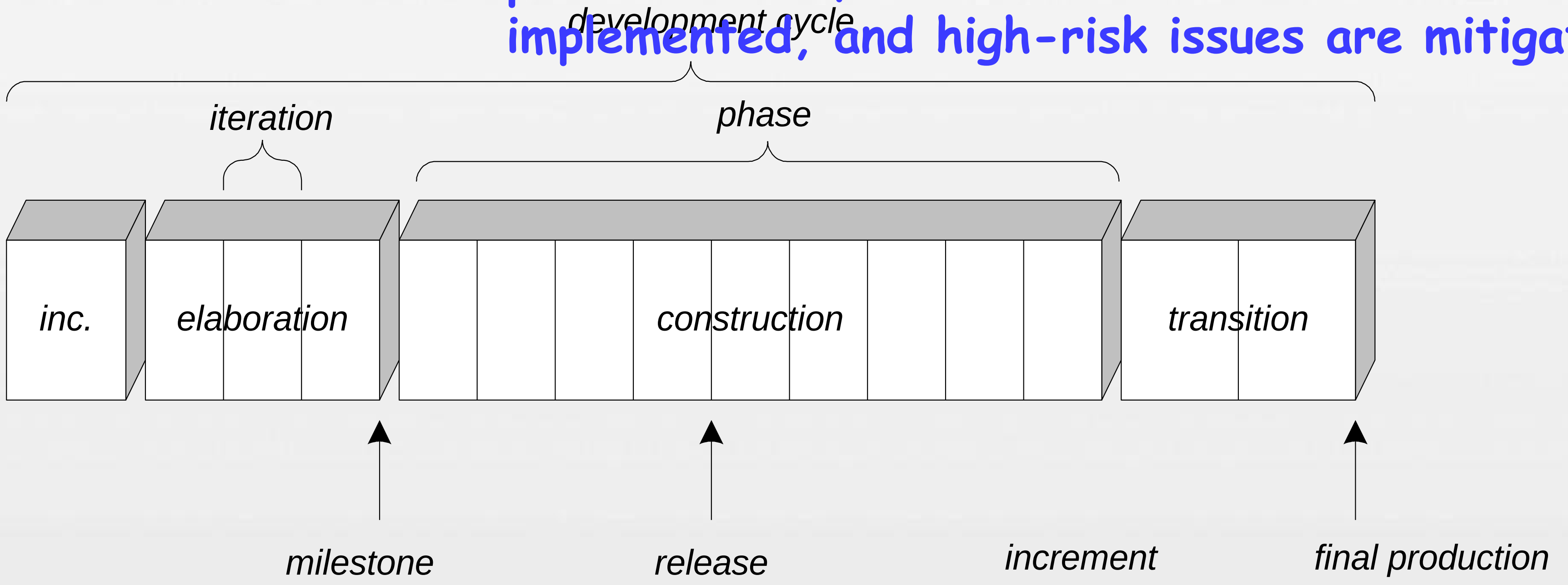
  - **UP is not the old "waterfall" or sequential lifecycle of first defining all the requirements, and then doing all or most of the design.**

3 Iterative, Evolutionary, and Agile

- **UP Phases**

Inception is not a requirements phase; rather, it is a feasibility phase

elaboration is not the requirements or design phase; rather, core architecture is iteratively implemented, and high-risk issues are mitigated



milestone

An iteration end-point when some significant decision or evaluation occurs.

release

A stable executable subset of the final product. The end of each iteration is a minor release.

increment

The difference (delta) between the releases of 2 subsequent iterations.

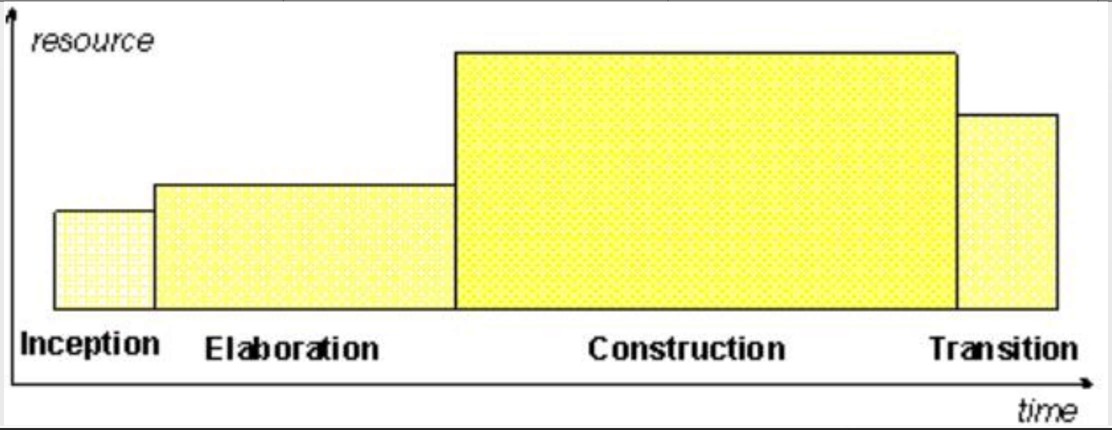
final production release

At this point, the system is released for production use.

3 Iterative, Evolutionary, and Agile

- UP Phases
  - Planning Phases
    - Inception: 5%-10%
    - Elaboration: 20%-30%
    - Construction:50-65%
    - Transition:10%

	Inception	Elaboration	Construction	Transition
Effort	~5 %	20 %	65 %	10%
Schedule	10 %	30 %	50 %	10%

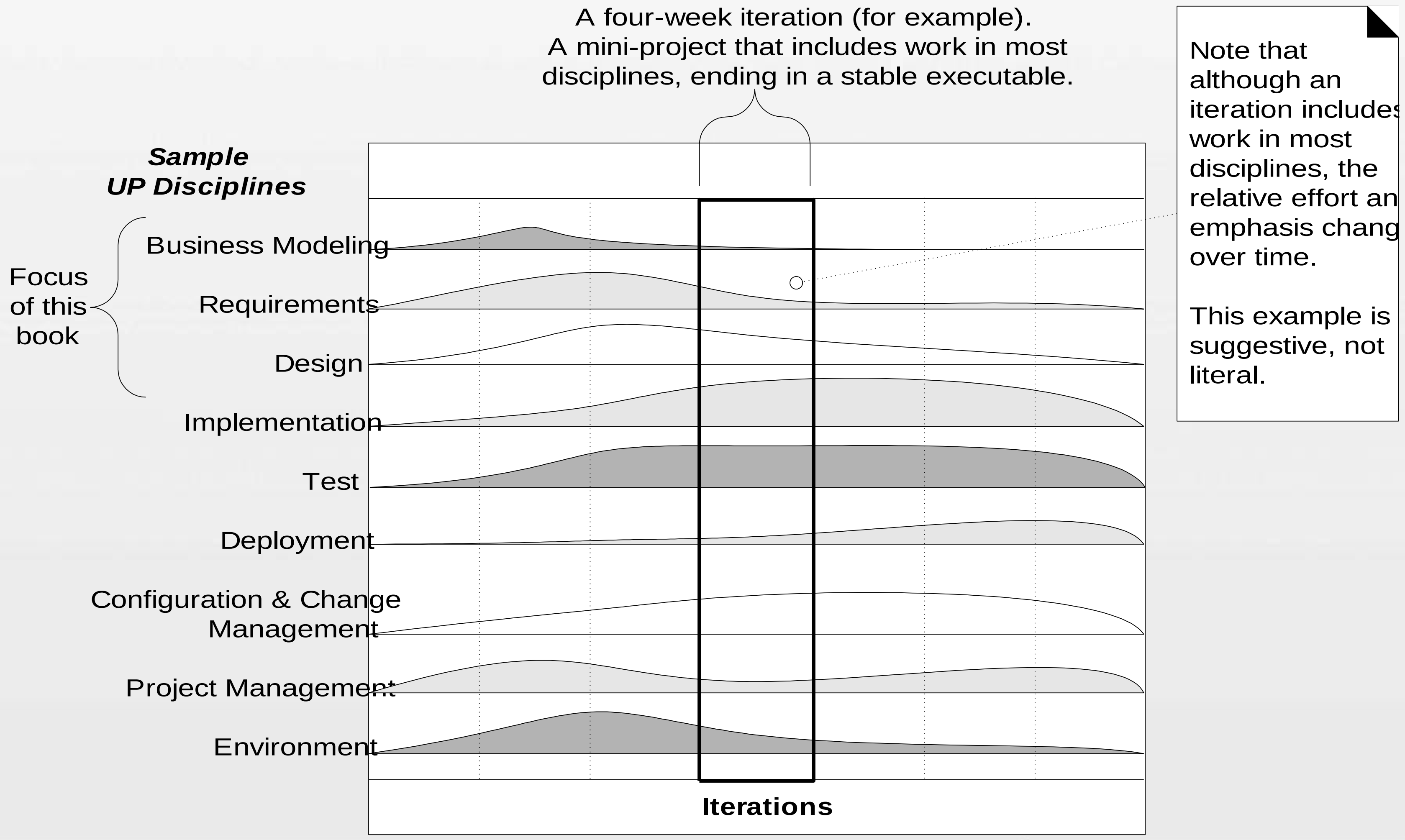


3 Iterative, Evolutionary, and Agile

- UP Disciplines
  - Disciplines, a set of activities (and related artifacts) in one subject area, such as the activities within requirements analysis.
  - Including –nine
    - Business modeling
    - Requirement
    - Design
    - Implement
    - Test
    - deployment
    - Configure and change management
    - Project management and Environment
  - focuses on three artifacts
    - Business modeling, requirement and design

3 Iterative, Evolutionary, and Agile

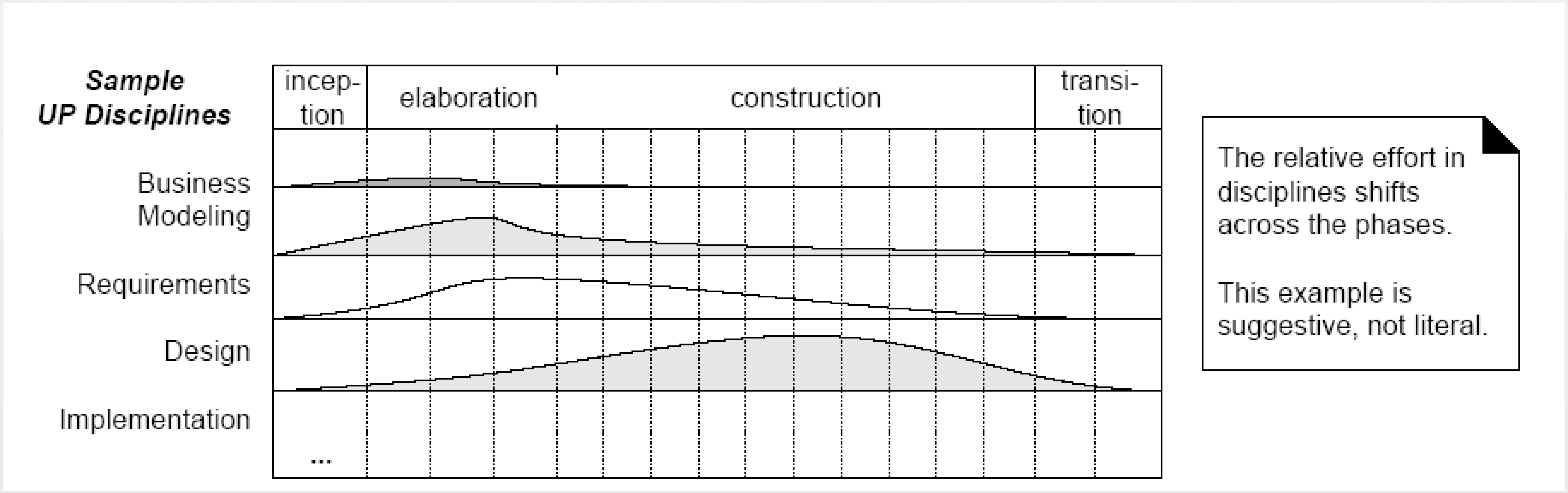
UP Disciplines



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 an emphasis chang 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

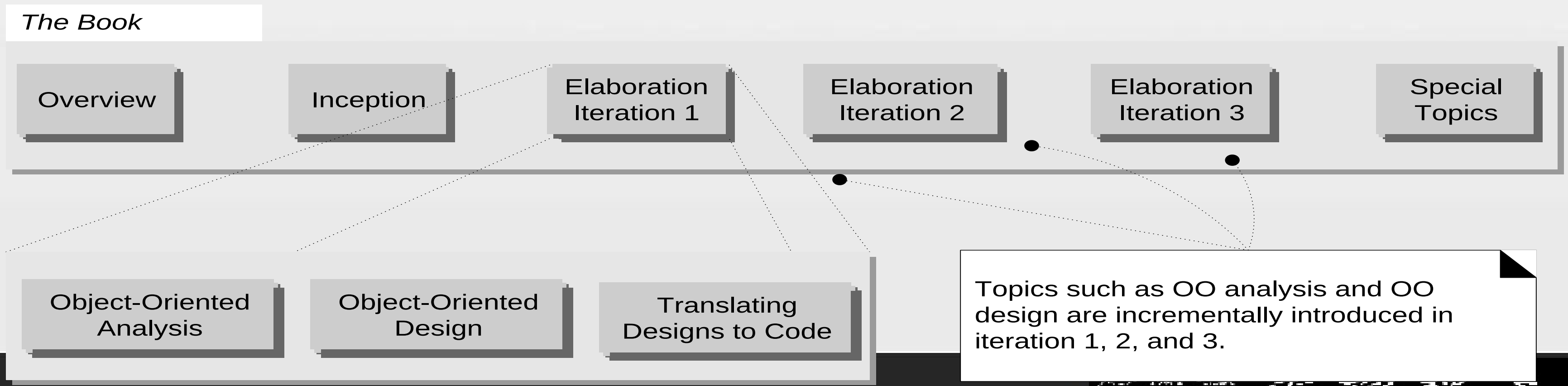
3 Iterative, Evolutionary, and Agile

- **Disciplines and Phases**
  - one iteration work goes on in most or all disciplines. However, the relative effort across these disciplines changes over time.
  - Early iterations apply greater relative emphasis to requirements and design, and later ones less so,



3 Iterative, Evolutionary, and Agile

- **Book Structure Influenced by UP Phases and Disciplines**
  - **The earlier chapters introduce activities in inception; later chapters explore several iterations in elaboration**
    - The inception phase chapters introduce the basics of requirements analysis.
    - Iteration 1 introduces fundamental OOA/D and assignment of responsibilities to objects.
    - Iteration 2 focuses on object design, especially on introducing some high-use "design patterns."
    - Iteration 3 introduces a variety of subjects, such as architectural analysis and framework design



3 Iterative, Evolutionary, and Agile

- Do you Understand Iterative Development or the UP ?
  - try to define most of the requirements before starting design or implementation?
  - spend days or weeks in UML modeling before programming
  - inception = requirements, elaboration = design, and construction = implementation ?
  - think that the purpose of elaboration is to fully and carefully define models and then translated into code during construction
  - adopting the UP means to do many of the possible activities and create many documents
  - plan a project in detail from start to finish

3 Iterative, Evolutionary, and Agile

- Other related Concept
  - Risk-Driven and Client-Driven
    - early iterations are chosen to 1) identify and drive down the highest risks, and 2) build visible features that the client cares most about
    - early iterations focus on building, testing, and stabilizing the core architecture
  - Test drive development (TDD)
    - Key idea of agile method
    - Write test code first ,then functional code

3 Iterative, Evolutionary, and Agile

- Test-Driven Development in C

```
#define FAIL() printf("\nfailure in %s() line %d\n", __func__, __LINE__)
#define _assert(test) do { if (!(test)) { FAIL(); return 1; } } while(0)
 #define _verify(test) do { int r=test(); tests_run++; if(r) return r; } while(0)


int square_01()
 { int x=5; _assert(square(x) == 25); return 0; }

int all_tests()
 { _verify(square_01); return 0; }

int main(int argc, char **argv) {
        int result = all_tests();
        if (result == 0) printf("PASSED\n");
        printf("Tests run: %d\n", tests_run);
        return result != 0;
        }
```

3 Iterative, Evolutionary, and Agile

- **Relations between iterative development, UP,RUP and Agile**
  - **iterative development and UP**
    - UP is one of the iterative lifecycle and the core of UP is iterative development.
  - **UP and RUP**
    - **Rational RUP is the most popular UP lifecycle**
  - **Other UP**
    - **RUP is** a division of IBM since 2003 and is not a single concrete prescriptive process, but rather an adaptable process framework
    - Open Unified Process (OpenUP), the Eclipse Process Framework software development process
    - Enterprise Unified Process (EUP), an extension of the Rational Unified Process
  - **UP and Agile**
    - **Agile method can used in UP lifecycle**

Summary of this week

- Introduce the course and homework
- What is analysis and design?
  - What is OO analysis and OO Design?
- What is iterative development?
- What is (R)UP?
- What is agile method?

Homework

- **What is object-oriented analysis and design?**
- **What is Unified Process?**

- **Object technology is . . .?**
  - A、 A set of principles guiding software construction.
  - B、 A new theory striving to gain acceptance.
  - C、 A dynamic new language by Grady Booch.
  - D、 Based on the principles of abstraction and modularity.

- Does UP agile?
  - A Yes
  - B No

Reference

- https://www.**uml-diagrams.org**
- An Introduction to Agile Modeling