Chapter 1

Object-Oriented Analysis and Design

Key Components of Course

- Object-oriented analysis (OOA)
  - Analysis: Investigation of the problem and requirements
    - Example questions: How will the system be used? What functions must it perform?
  - OOA: Find and describe the domain objects
    - Example: Flight information system
    - Objects: Plane, Flight, Pilot
- Object-oriented design (OOD) or object design
  - Design: A conceptual solution that fulfills requirements
    - NOT the implementation
    - Excludes low-level details
  - OOD: Define software objects and how they collaborate.
- Most critical learning goal in this class: Assigning responsibilities to software objects.

Key Components of Course

- UML: Unified Modeling Language
  - A standard diagramming notation
  - How to write down requirements, design intent, etc. in a standardized, graphical way so your teammates can understand it.
  - Software tools for manipulating UML documents
- An iterative development process
  - The Rational Unified Process
  - The “agile” (light, flexible) approach

Fig. 1.1

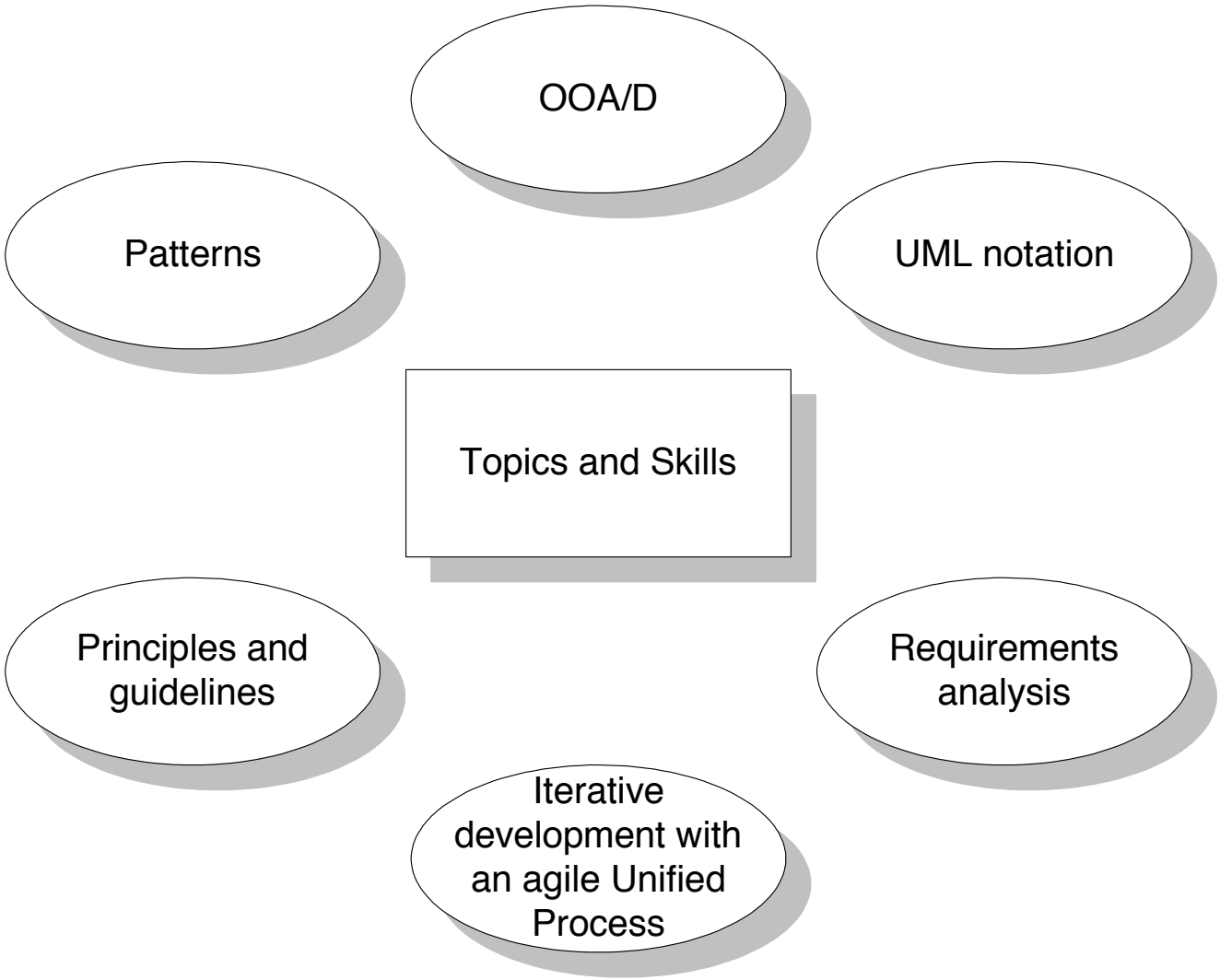
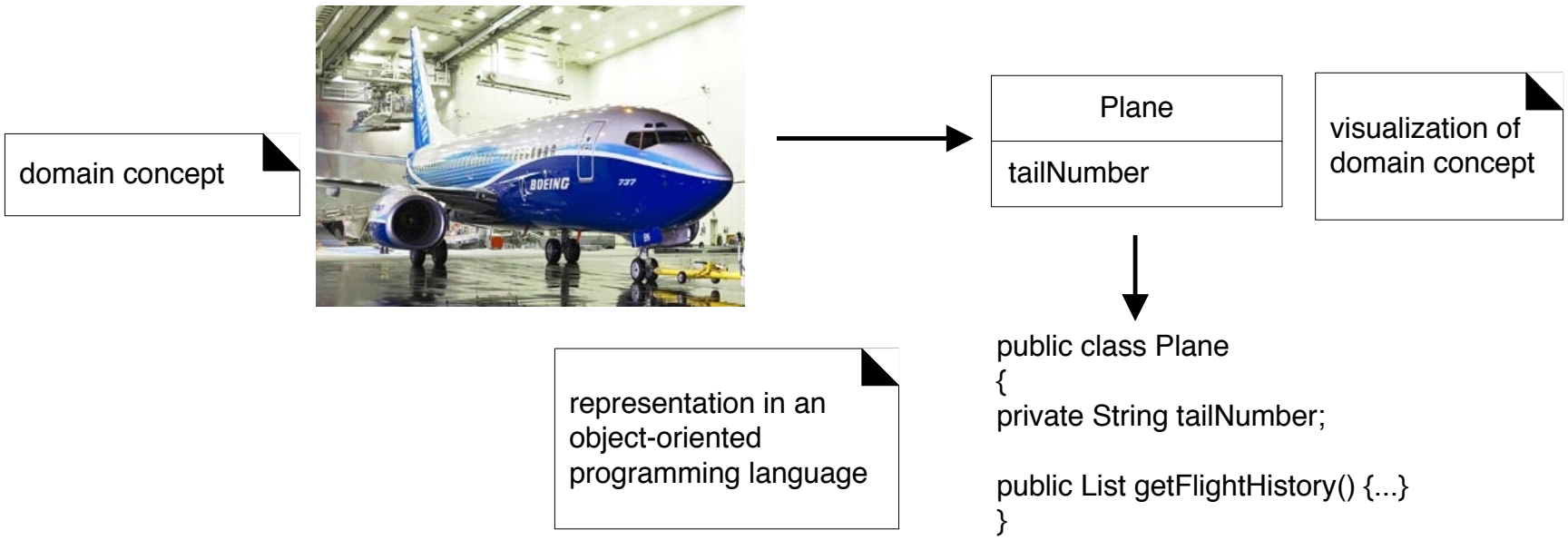

OOA/D
Patterns
UML notation
Topics and Skills
Principles and guidelines
Requirements analysis
Iterative development with an agile Unified Process

Fig. 1.2

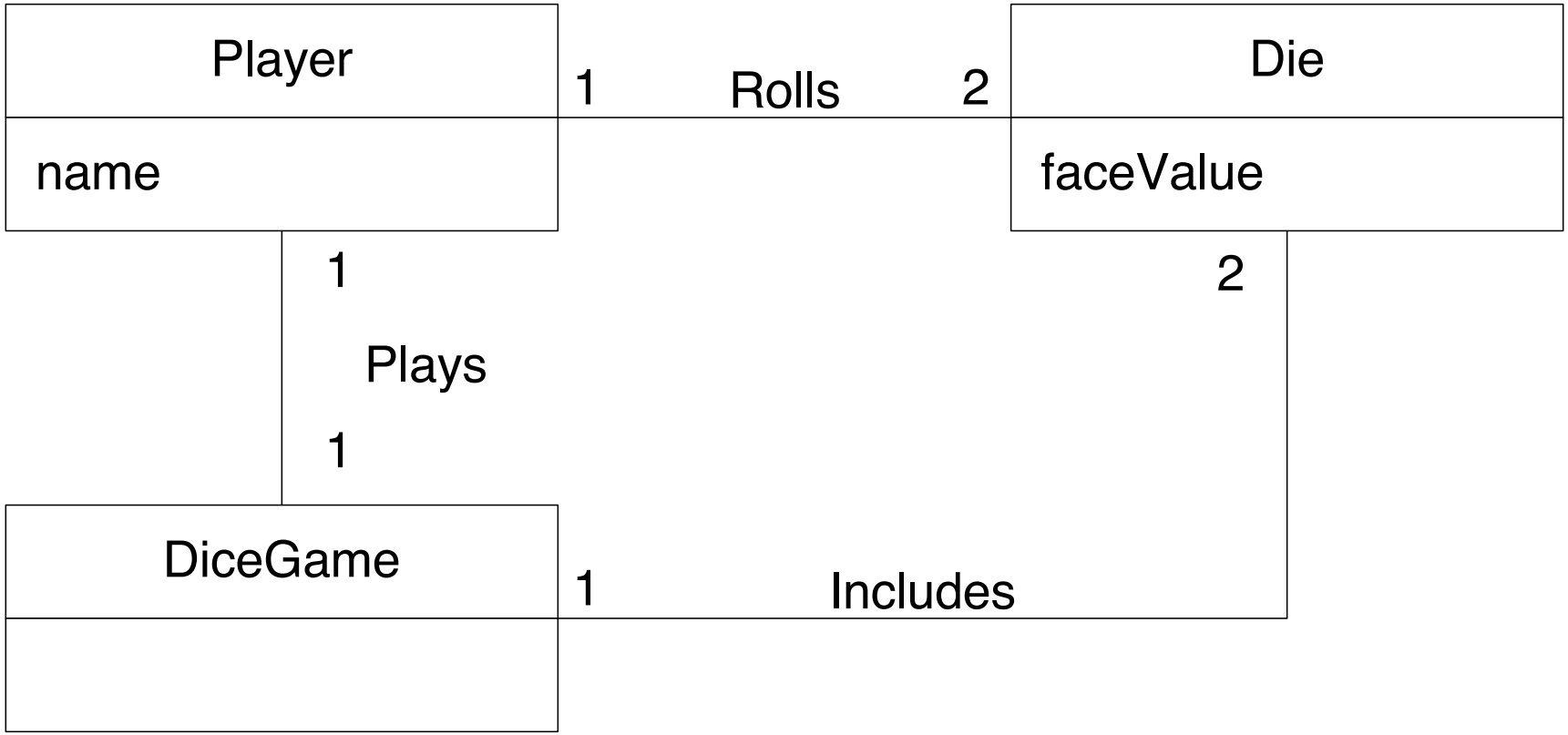

domain concept
BOEING
737
Plane
tailNumber
visualization of domain concept
representation in an object-oriented programming language
public class Plane
{
private String tailNumber;
public List getFlightHistory() {...}
}

Short Example: Dice Game

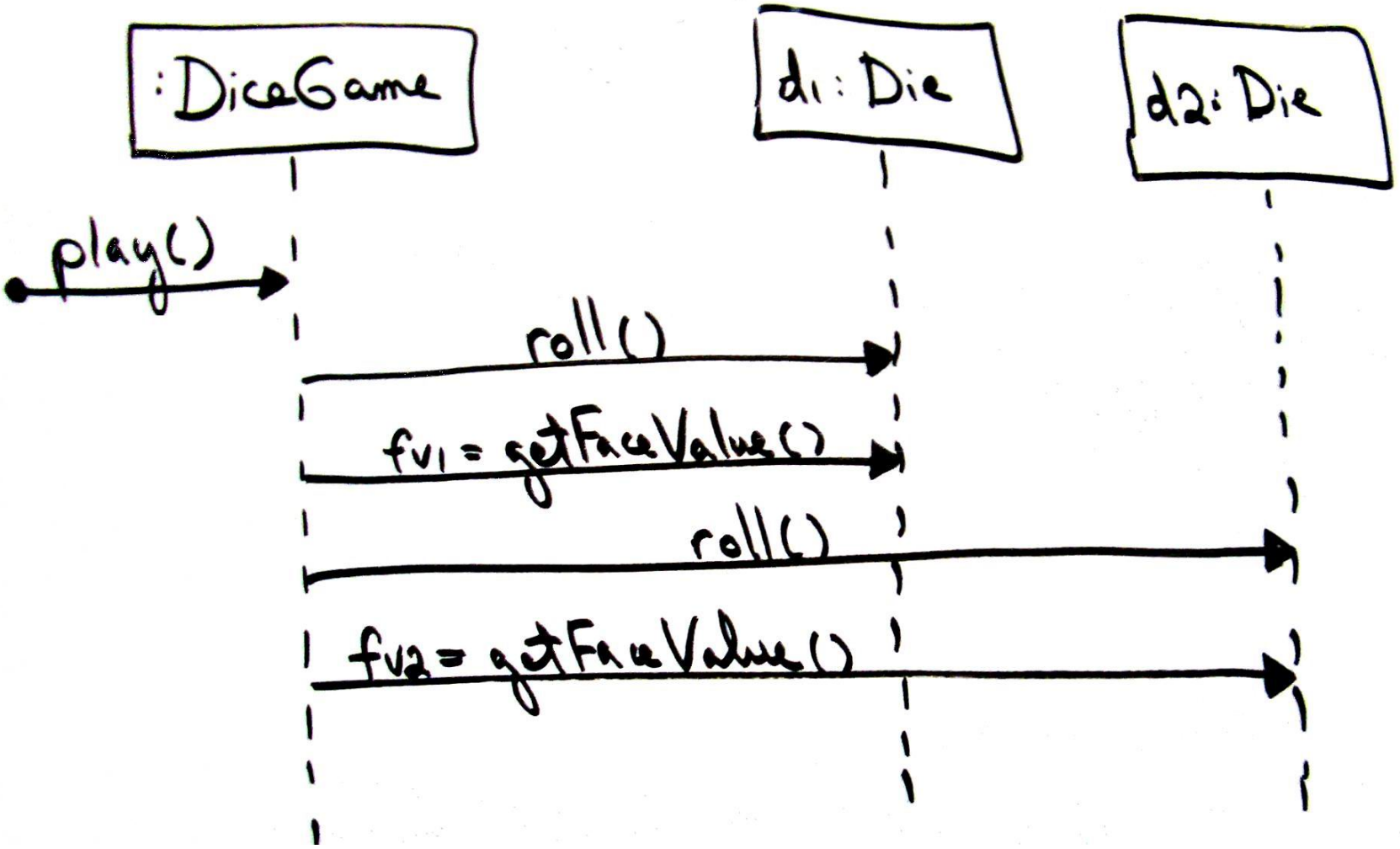
- Software simulates the player rolling two dice
  - If total is seven, they win. Otherwise, they lose.
- A few key steps and diagrams in iterative development
  1. Define use cases
  2. Define domain model
  3. Draw interaction diagrams, assign object responsibilities
  4. Draw design class diagrams

1. Use cases: A description of the user-system interaction
   - Player requests to roll the dice
   - System presents results
     - If the dice face value totals seven, player wins
     - Otherwise, player loses

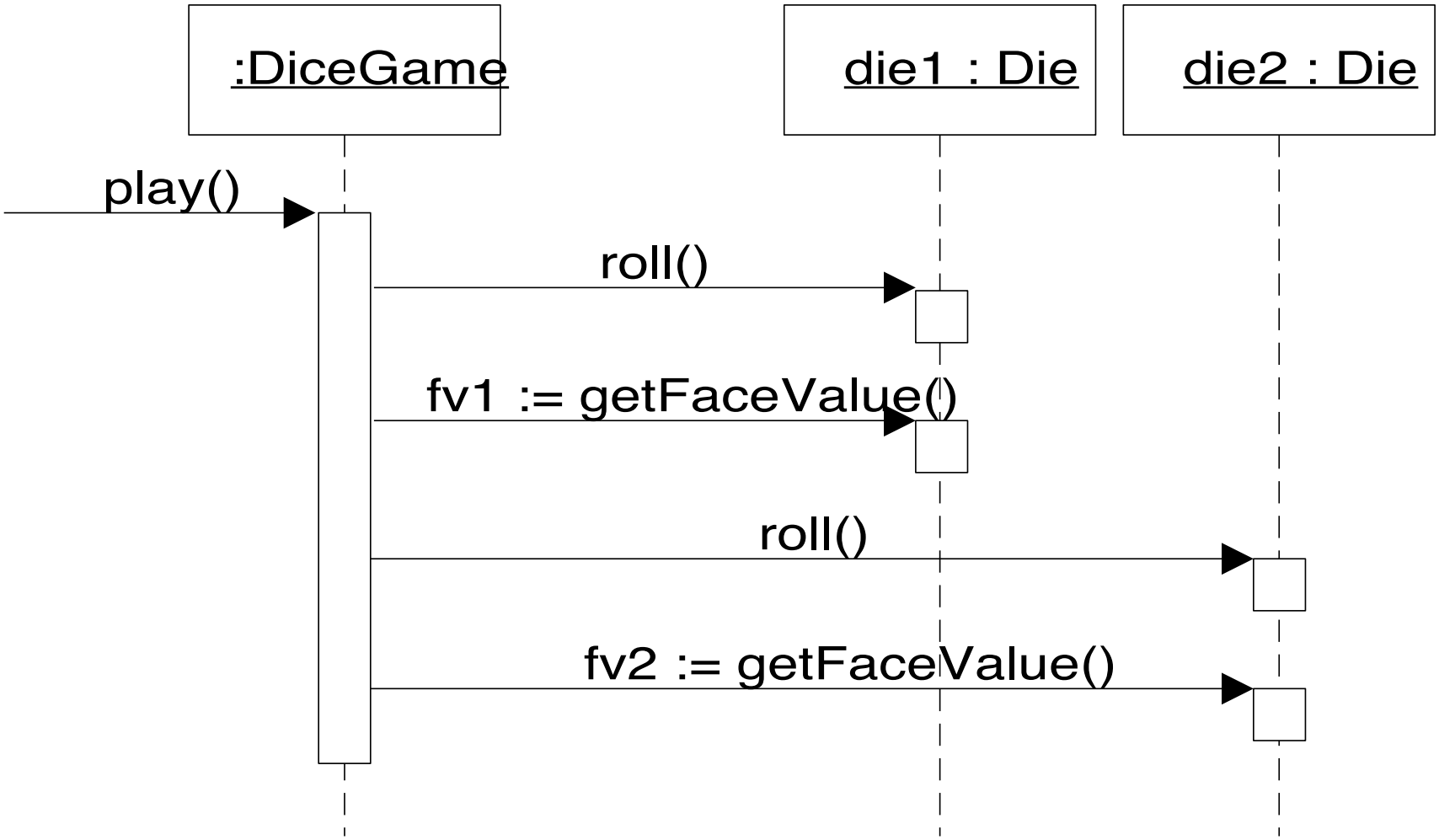
2. Define domain model (real world objects, not software objects)



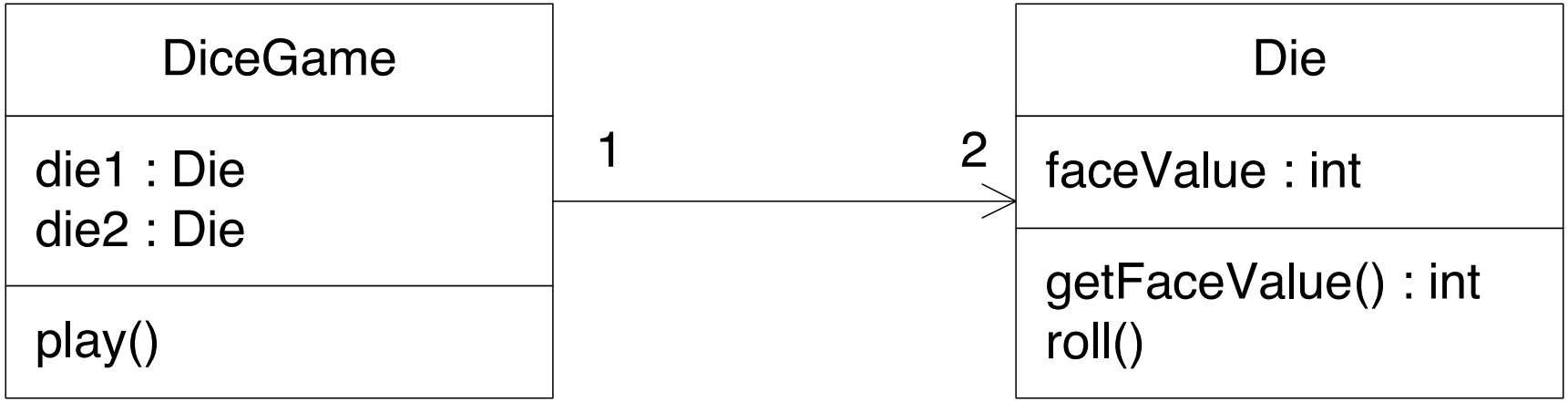
Draw interaction diagrams, assign object responsibilities



Draw interaction diagrams, assign object responsibilities



4. Draw class diagrams (software objects)



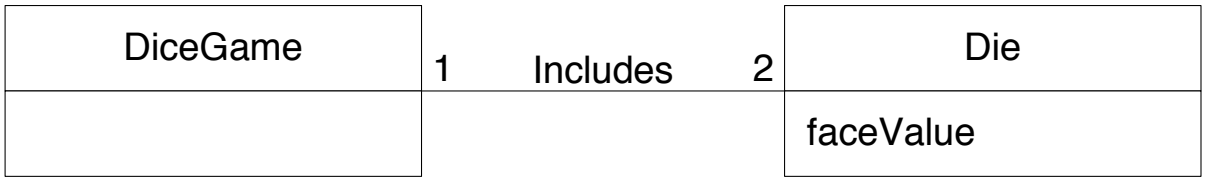
The UML

- A visual language for
  - specifying,
  - constructing, and
  - documenting

systems

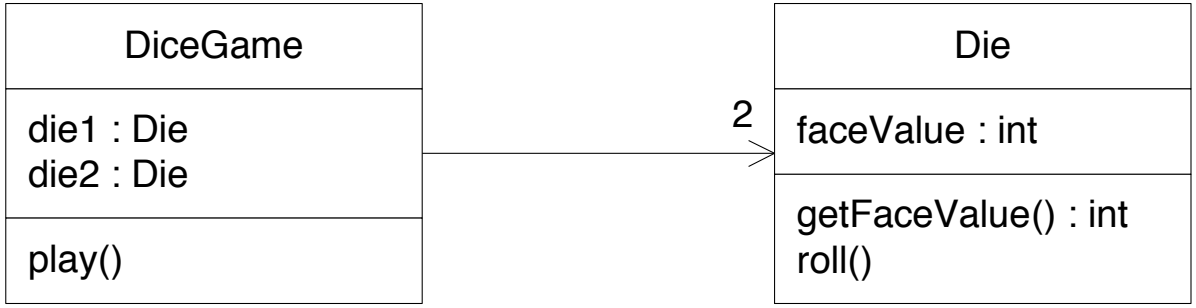
- A standard diagramming notation
- Three perspectives to use UML
  - Conceptual perspective
  - Software specification perspective
  - Software implementation perspective

Fig. 1.6



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.

UML and Silver Bullet Thinking

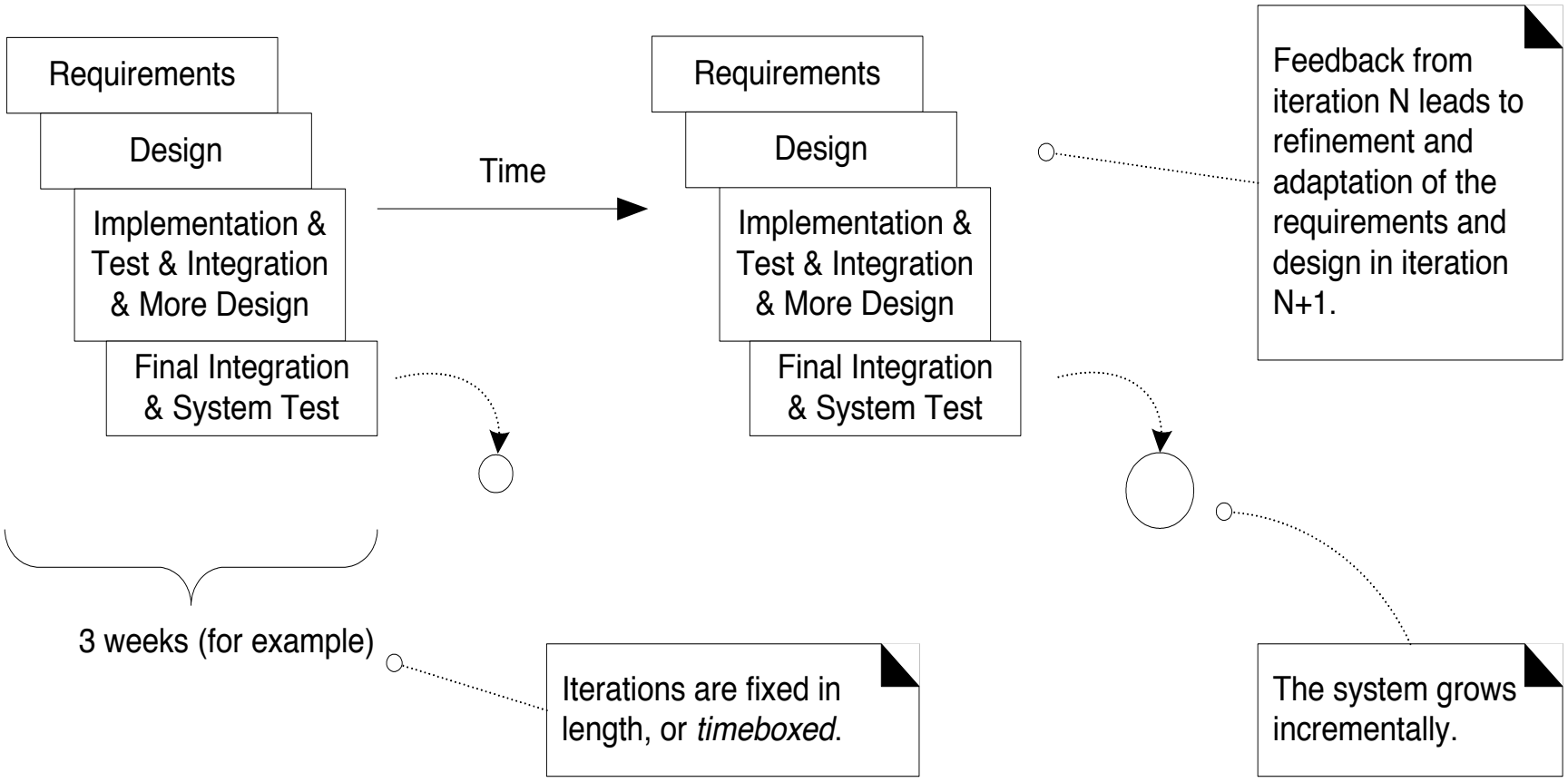
- Fundamental mistake
  - There is a special tool in software that will make a dramatic order-of-magnitude difference in productivity, realiability, or simplicity.
- Tools don't compansate for design ignorance
- **Death by UML Fever [Bell04]**

Chapter 2

Software Development Process

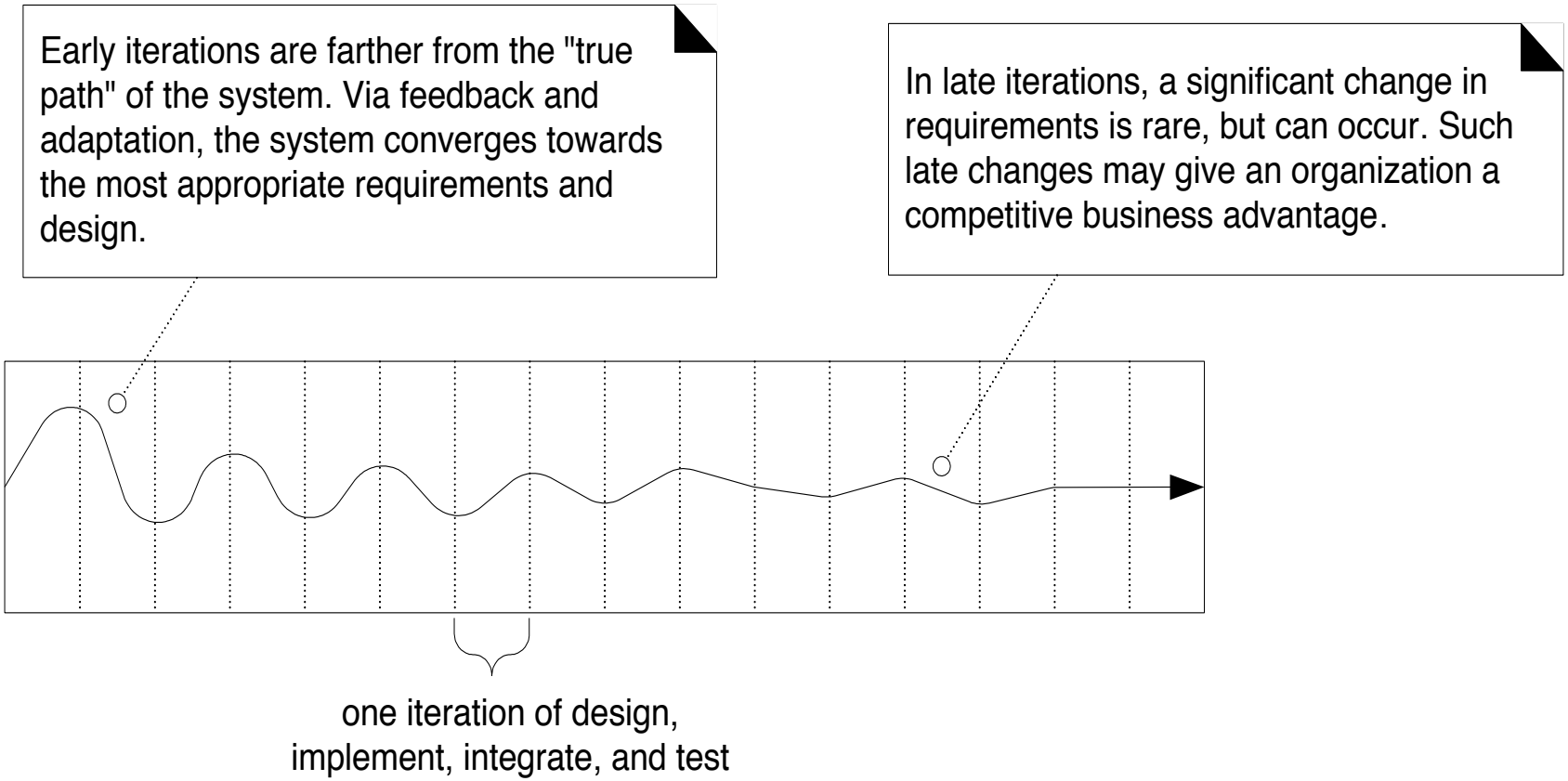
- A software development process describes an approach to building, deploying, and possibly maintaining software
- **Unified Process**
  - popular iterative software development process
  - for object-oriented systems
- **Iterative and Evolutionary Development**
  - series of short, fixed-length mini projects called **iterations**
  - outcome of each iteration is an executable partial system

Iterative and evolutionary development

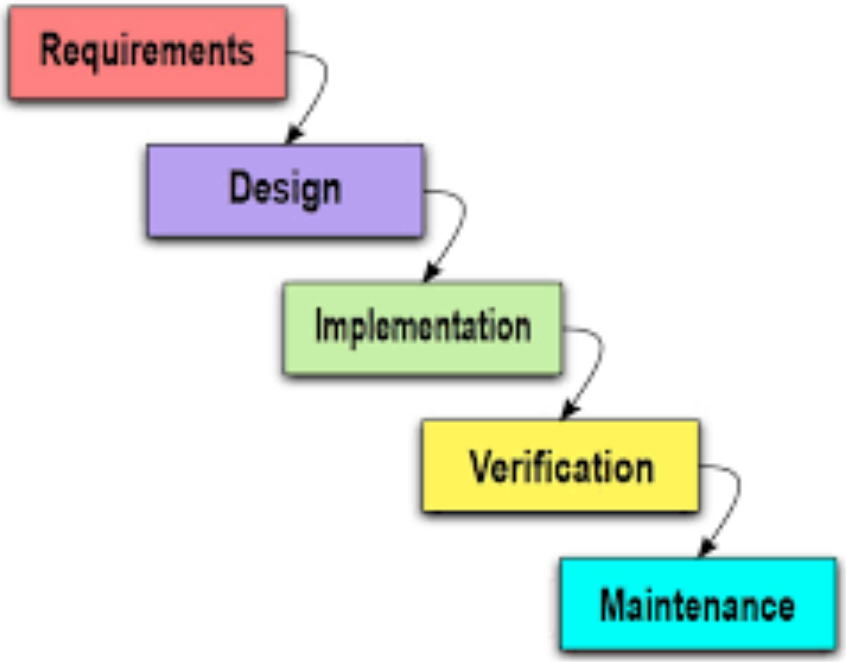


Result of each iteration: An executable but incomplete system

Fig. 2.2


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

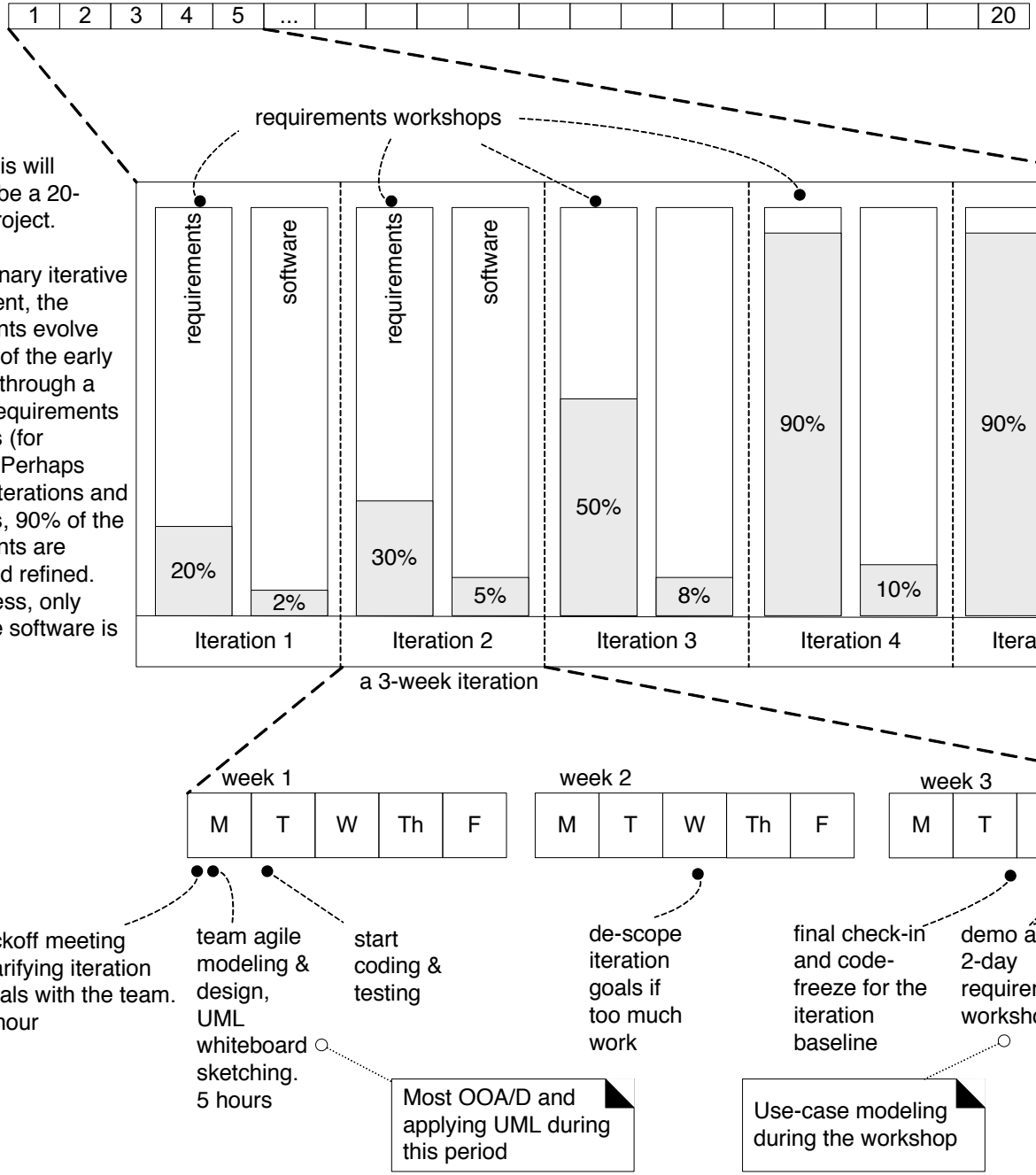
The Waterfall Lifecycle



- sequential
- define all requirements (in detail) before programming
- research shows (conclusively) that waterfall is a poor practice for software development
- Why
  - False assumption that the specifications are predictable and stable and can be correctly defined at the beginning

Don’t Let Waterfall Thinking Invade and Iterative or UP Project

How to do Iterative and Evolutionary Analysis and Design



What are Agile Methods

- Usually apply timeboxed iterative and evolutionary development
- Employ adaptive planning
- Promote incremental delivery
- Encourages agility – rapid and flexible response to change

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.

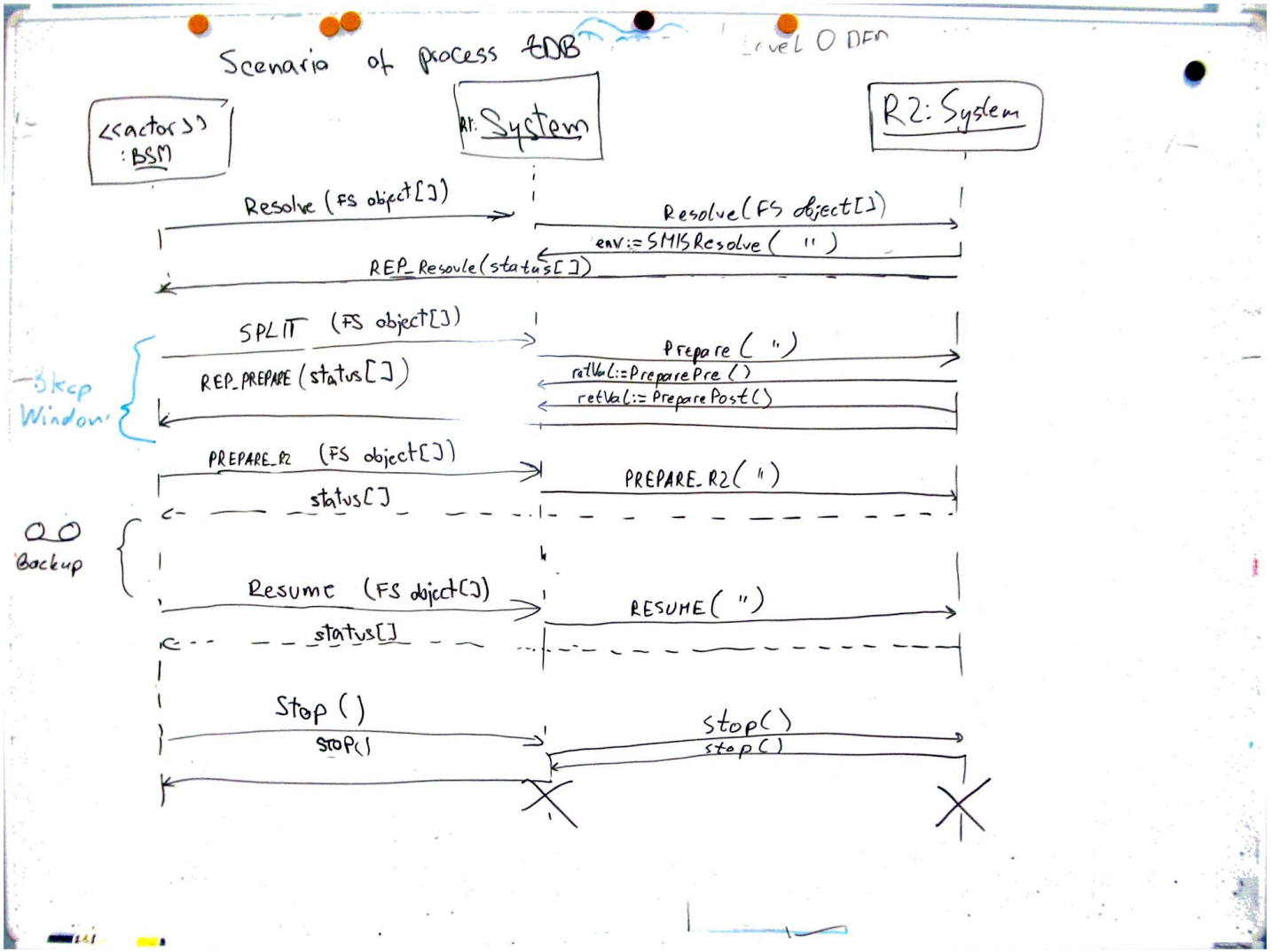
12. The best architectures, requirements, and designs emerge from self-organizing teams.

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

What is Agile Modeling

- Adopting Agile Method does not mean avoiding any modeling
- The purpose of modeling (sketching UML etc) is primarily **to understand , not do document**
- Don’t model or apply UML to all software design. Defer simple cases until programming. Model unusual difficult, tricky parts.
- Use simplest tools, choose tools large visual spaces (for example, sketching UML on whiteboards and capturing the diagrams with camera)
- Don’t model alone, model in pairs (or more)
- Create models in parallel (for example, dynamic view and static views)
- Use good-enough simple notation while sketching
- Know all models are inaccurate – will change – only tested code will demonstrate the true design
- Developers themselves should do OO design modeling for themselves, (not create diagrams for other programmers to implement - an un-agile practice)

Fig. 2.5


Scenario of process tDB
Level 0 DFD
<<actor>>
:BSM
R1: System
R2: System
Resolve (FS object[])
Resolve(FS object[])
env:=SMISResolve(")
REP_Resolve(status[])
SPLIT (FS object[])
Prepare (")
REP_PREPARE (status[])
retVal:=PreparePre()
retVal:=PreparePost()
Bkcp Window
PREPARE_R2 (FS object[])
PREPARE_R2(")
status[]
Backup
Resume (FS object[])
RESUME(")
status[]
Stop ()
stop()
stop()
stop()

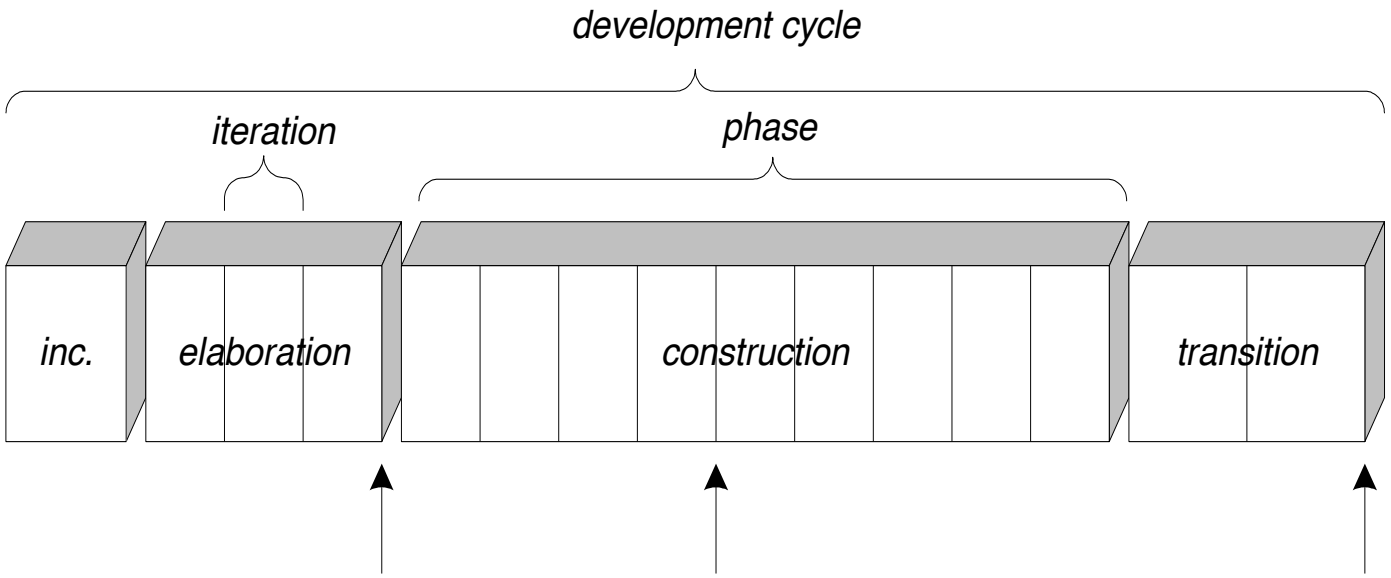
UP Phases

A UP project organizes the iterations across four major phases

1. Inception:
   - Approximate vision
   - Business case
2. Elaboration:
   - Refined vision,
   - Implementation of core architecture
   - Identification of most requirements and scope
3. Construction
   - Iterative implementation of rest
   - 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



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

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

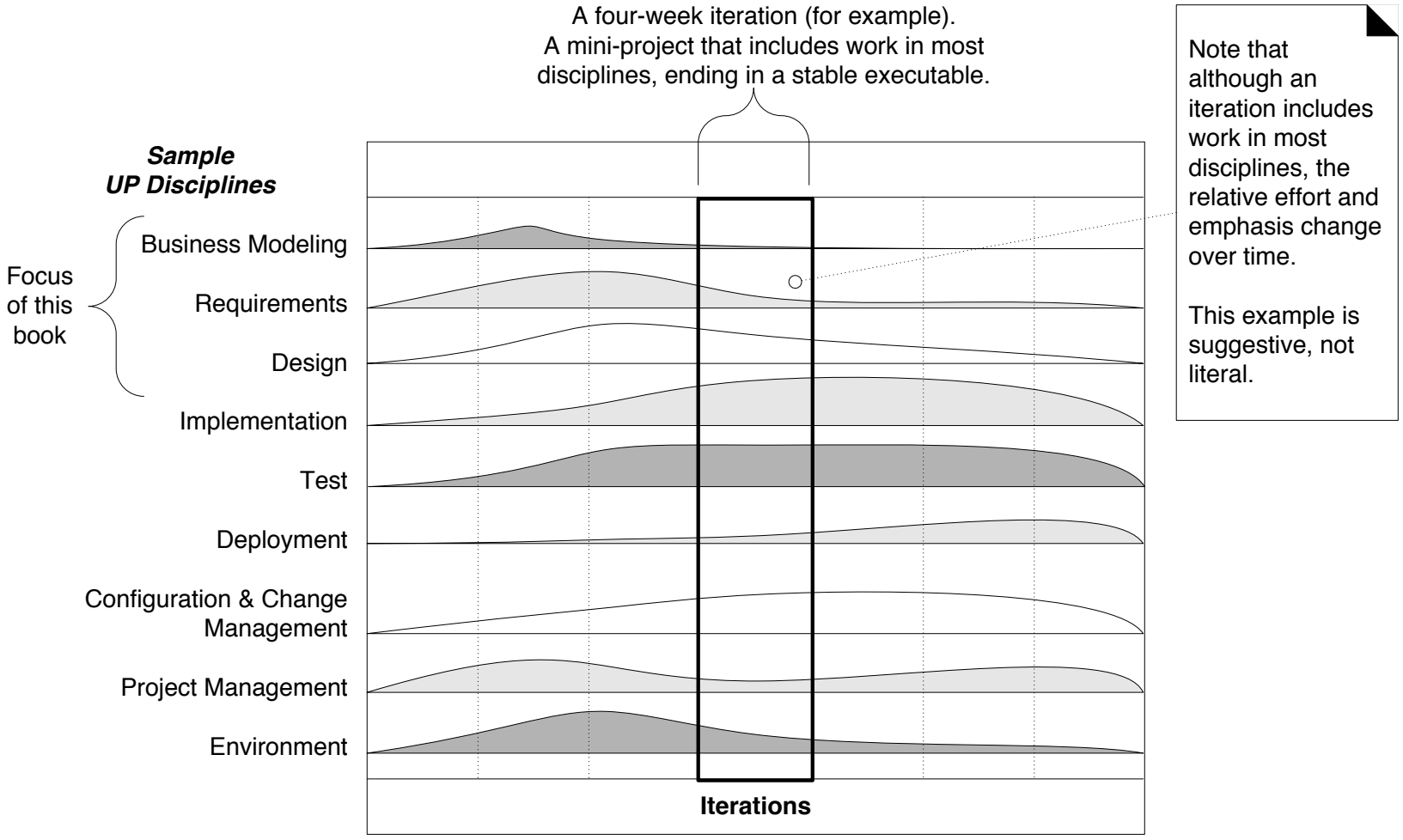
The difference (delta) between the releases of 2 subsequent iterations.

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

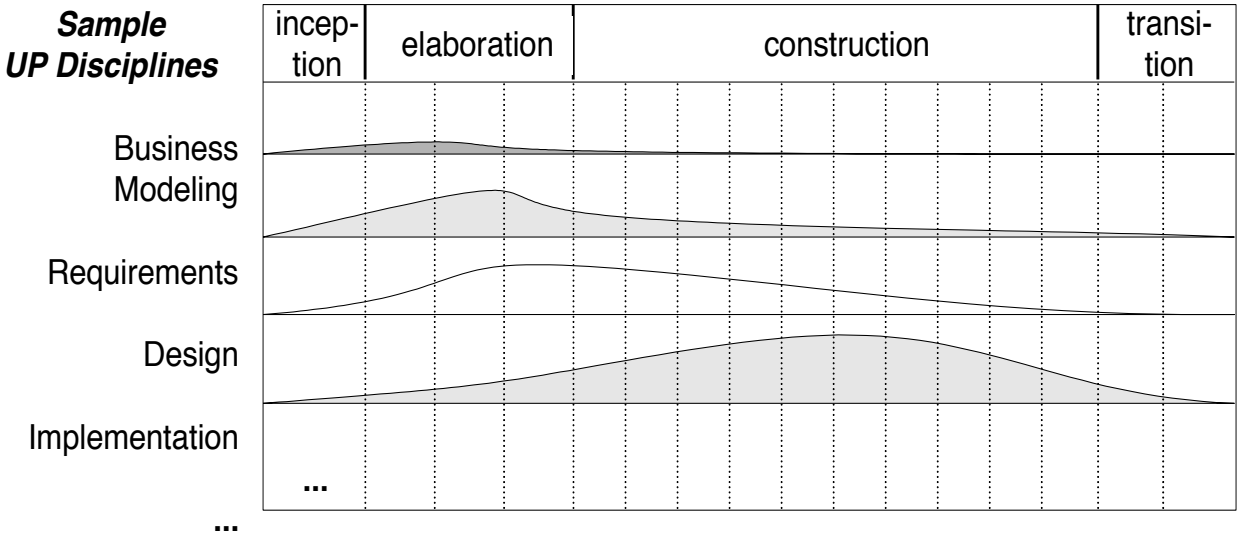
UP Disciplines

- UP describe work activities, such as writing a use case, within **disciplines**
- A discipline is a set of activities (and related **artifacts**)
- An artifact is the general term for any work product: code, web graphics, database schema, text documents, diagrams, models,….
- We will focus on the following 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
  - **Implementation** programming, building system and deploying

UP Disciplines vs. Project Iterations



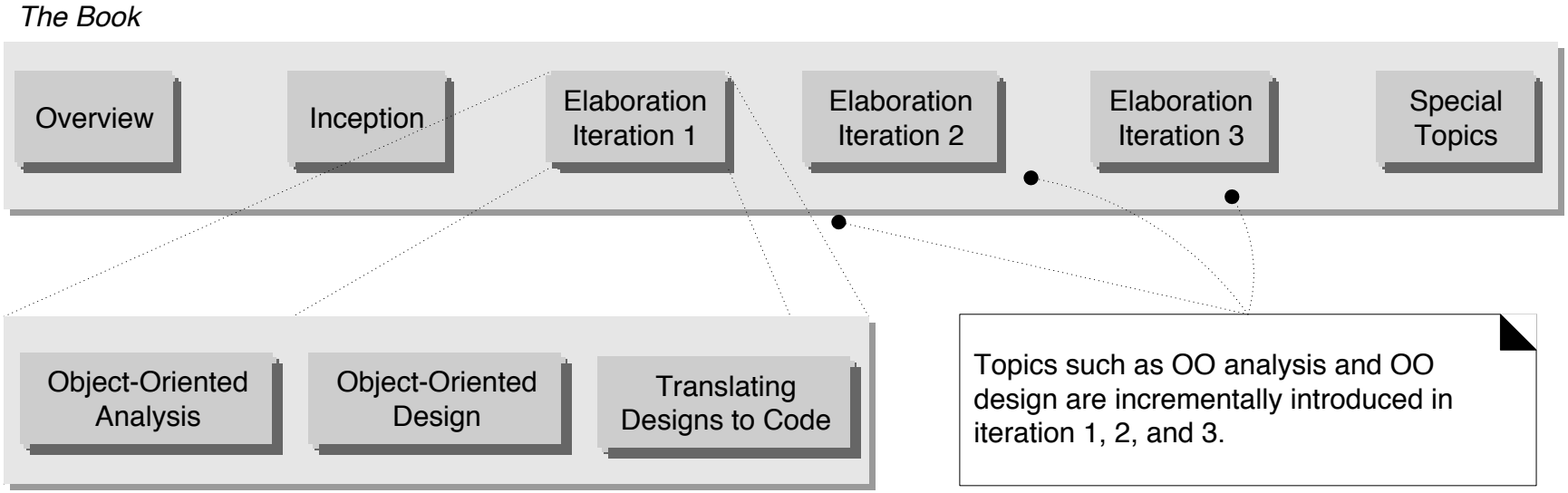
Disciplines and Phases



The relative effort in disciplines shifts across the phases.

This example is suggestive, not literal.

Book Organization – related IP phases and interations



Sample Development Case

Discipline	Practice	Artifact	Incep.	Elab.	Const.	Trans.
		Iteration	I1	E1..En	C1..Cn	T1..T2
Business Modeling	agile modeling req. workshop	Domain Model		s		
Requirements	req. workshop vision box exercise dot voting	Use-Case Model	s	r		
		Vision	s	r		
		Supplementary Specification	s	r		
		Glossary	s	r		
Design	agile modeling test-driven dev.	Design Model		s	r	
		SW Architecture Document		s		
		Data Model		s	r	
Implementation	test-driven dev. pair programming continuous integration coding standards	...				
Project Management	agile PM daily Scrum meeting	...				
...						

Some key rules of thumb

- The purpose of modeling (sketching UML) especially early in the project, is to understand and communicate, NOT to document
- Do modeling in groups
- Know that all models will be inaccurate
  - Final code and design may be very different from initial model
- Developers themselves should do the modeling

Case Studies

- The NextGen POS System
  - A POS system is used to record sales, handle payments. It is typically used in retail stores. It includes hardware components such as bar code scanner, terminal…
  - interfaces third party systems (tax, inventory control systems).

- Monopoly Game Systems
  - Although not as complicated as NextPos, it is rich enough for domain modeling, design, applying patterns…

A POS system, layers (tiers)

PART 2

The Inception Phase

- Inception: Short initial phase
  - What is the vision for this project?
  - Is it feasible?
- Do some preliminary requirements and use cases work
  - It will include analysis of perhaps 10% of the requirements (most requirement analysis occurs during the elaboration phase)
- In your class project, we will assume that this phase has been already carried out
  - Earlier work tells you that the project is feasible
  - Preliminary requirements have been explored

What Artifacts may start in Inception

Artifact[†]	Comment
Vision and Business Case	Describes the high-level goals and constraints, the business case, and provides an executive summary.
Use-Case Model	Describes the functional requirements. During inception, the names of most use cases will be identified, and perhaps 10% of the use cases will be analyzed in detail.
Supplementary Specification	Describes other requirements, mostly non-functional. During inception, it is useful to have some idea of the key non-functional requirements that have will have a major impact on the architecture.
Glossary	Key domain terminology, and data dictionary.
Risk List & Risk Management Plan	Describes the risks (business, technical, resource, schedule) and ideas for their mitigation or response.
Prototypes and proof-of-concepts	To clarify the vision, and validate technical ideas.
Iteration Plan	Describes what to do in the first elaboration iteration.
Phase Plan & Software Development Plan	Low-precision guess for elaboration phase duration and effort. Tools, people, education, and other resources.
Development Case	A description of the customized UP steps and artifacts for this project. In the UP, one always customizes it for the project.

Some artifacts will be partially completed in this phase and iteratively refined in subsequent iterations

The purpose of inception is to collect just enough information to establish a common vision, decide if moving forward is feasible

Most UML diagramming will occur in elaboration phase

Evolutionary Requirements

- Definition: Requirements are the capabilities that the system must support and the conditions to which it must conform.
- RUP approach to requirements
  - A systematic approach to
    - finding
    - documenting
    - organizing, and
    - tracking

  the changing requirements of a system
- In UP and other evolutionary methods (Scrum, XP…), start production-quality programming and testing long before most of the requirements have been analyzed.

Categories of requirements: FURPS+

- FURPS
  - Functional: Features, capabilities, security
  - Usability: Human factors, help, documentation
  - Reliability: Frequency of failure, recoverability, predictability
  - Performance: Response times, throughput, accuracy, availability, resource usage
  - Supportability: Adaptability, maintainability, internationalization, configurability
- +
  - Implementation: Resource limitations, languages and tools, hardware
  - Interface: Interfacing with external systems
  - Operations: System management where it is going to be used
  - Packaging
  - Legal

How are requirements organized and stored?

- UP artifacts
  - Use-case model: A set of typical scenarios of using a system
  - Supplementary specification: All non-functional requirements, features
  - Glossary
    - Key terms
    - Data dictionary
  - Vision
    - Big ideas of project
    - The business case
  - Business rules: Rules of the domain you have to operate in
    - Tax laws
    - Aviation laws
    - Laws of physics