ОБЕКТНО-ОРИЕНТИРАН АНАЛИЗ И ДИЗАЙН (ПРОЕКТИРАНЕ) - ООАД

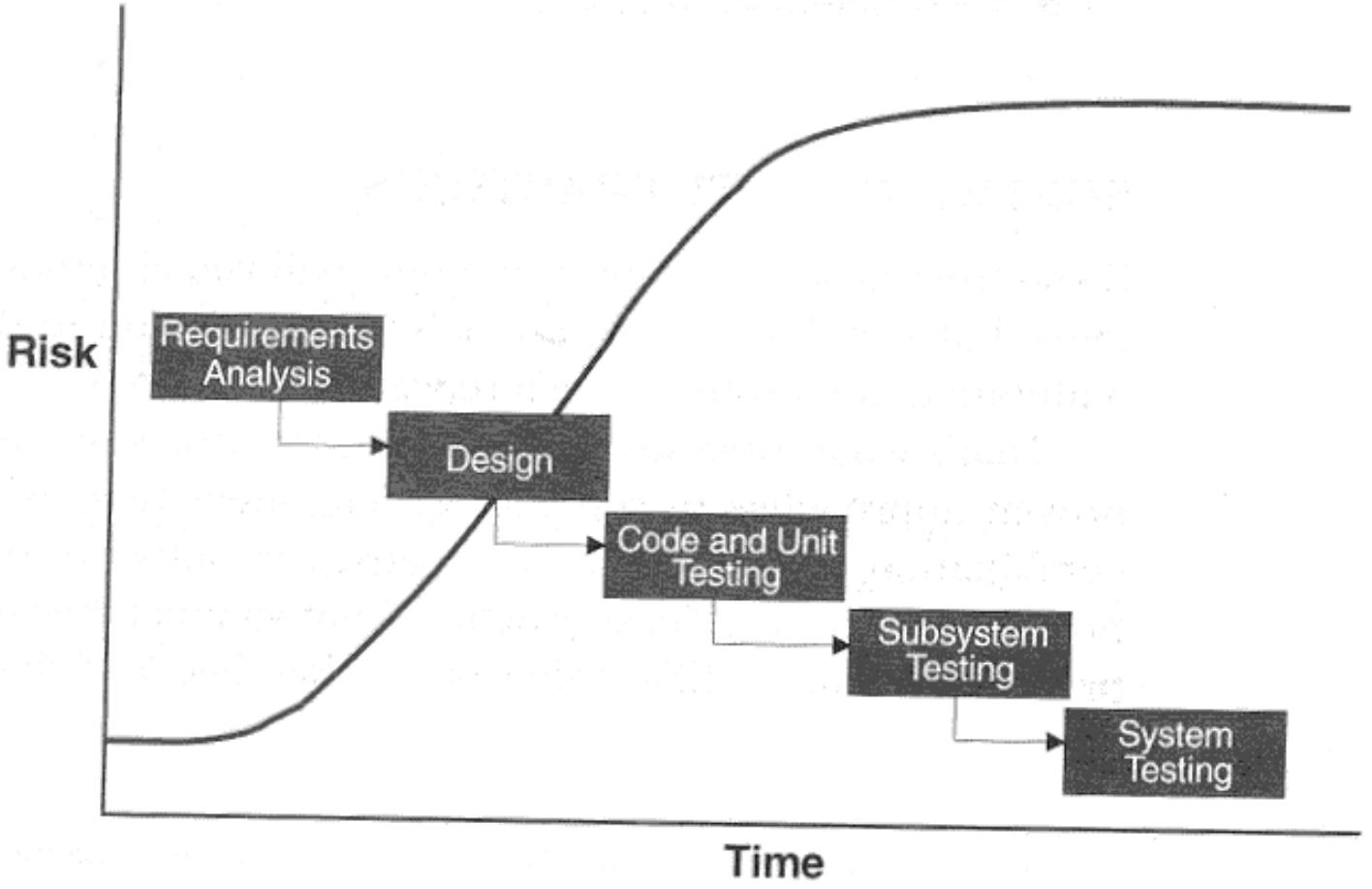
Съдържание

- Модели на жизнения цикъл
- Unified Process (UP)
- Анализ и проектиране
- Обектно-ориентиран анализ и проектиране
- UML

Основни понятия

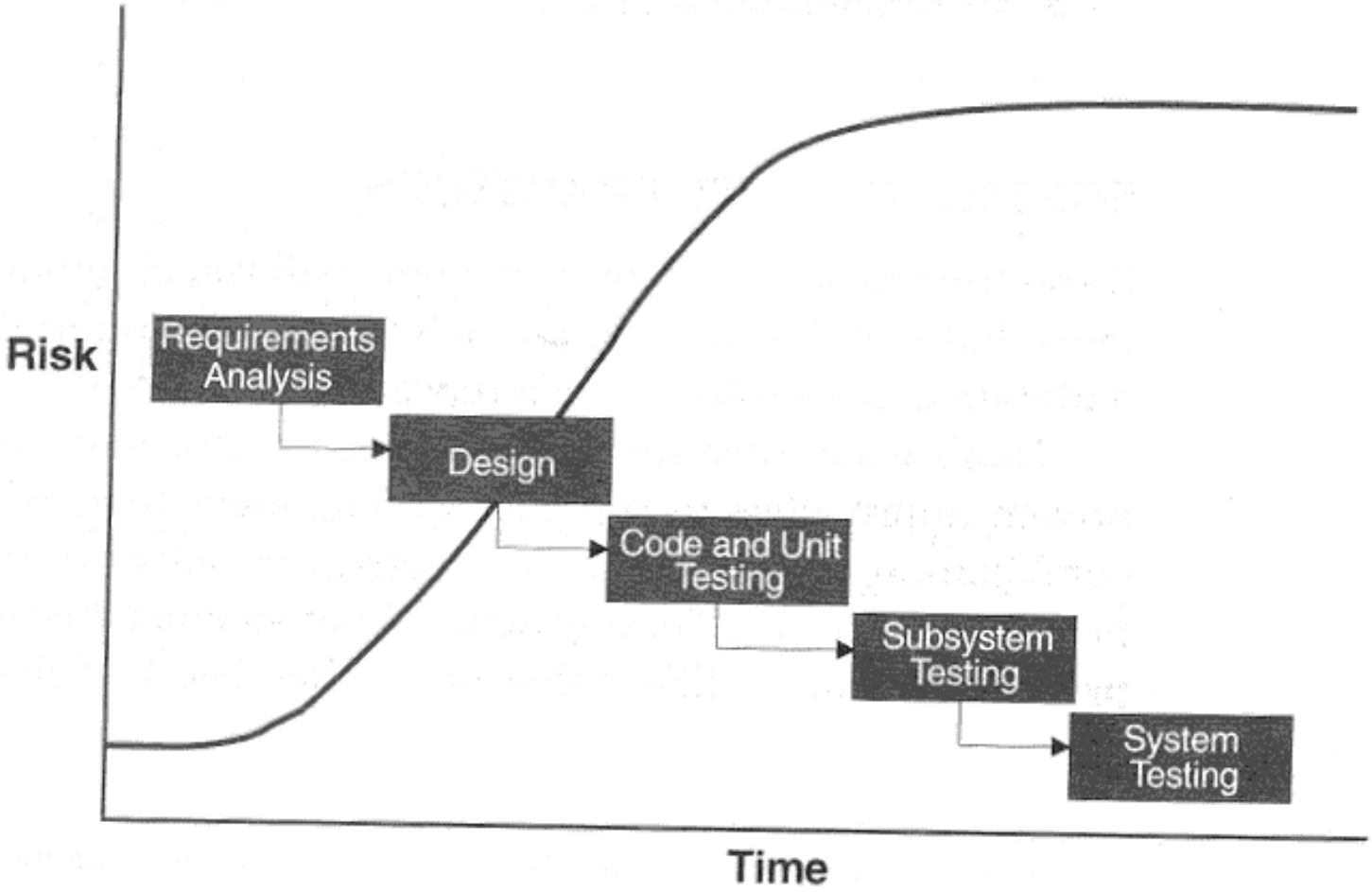
- **Жизнен цикъл** (ЖЦ) на програмния продукт (ПП): период на създаване и използване на ПП.
- **Начало**: възникване на идеята за създаване
- **Край**: преустановяване на използването на последното копие на ПП
- **Софтуерен процес** (СП): набор от дейности, необходими за разработката на ПП.
- **Модел на СП**: абстрактно представяне на софтуерния процес. Дефинира набор от дейности, задачи, ключови резултати и отчетни материали, необходими за изграждането на висококачествен софтуер.
- **Фаза**: отрязък от време, през което се извършват определени дейности по разработвания ПП.

Основни фази – Каскаден модел



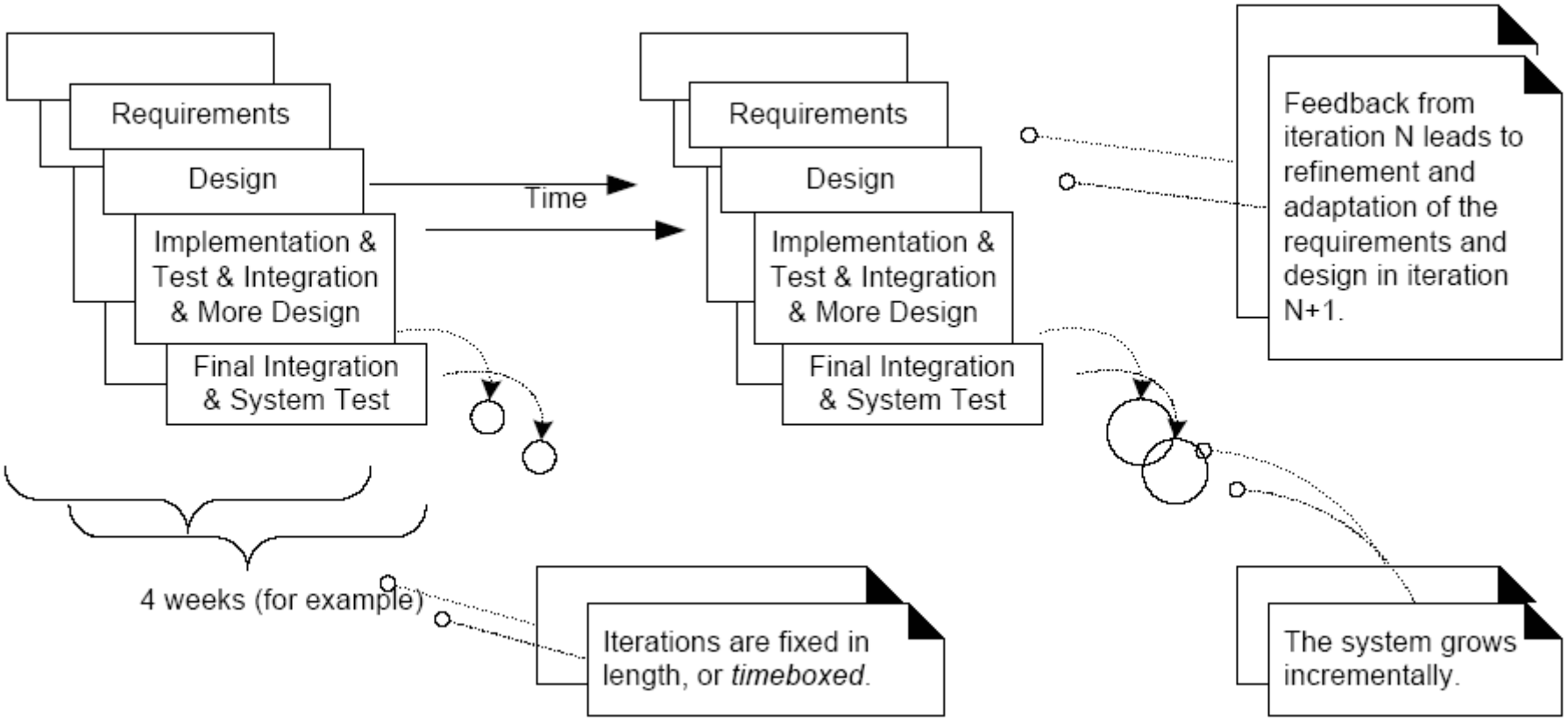
The waterfall lifecycle

Последователните методи увеличават риска с времето

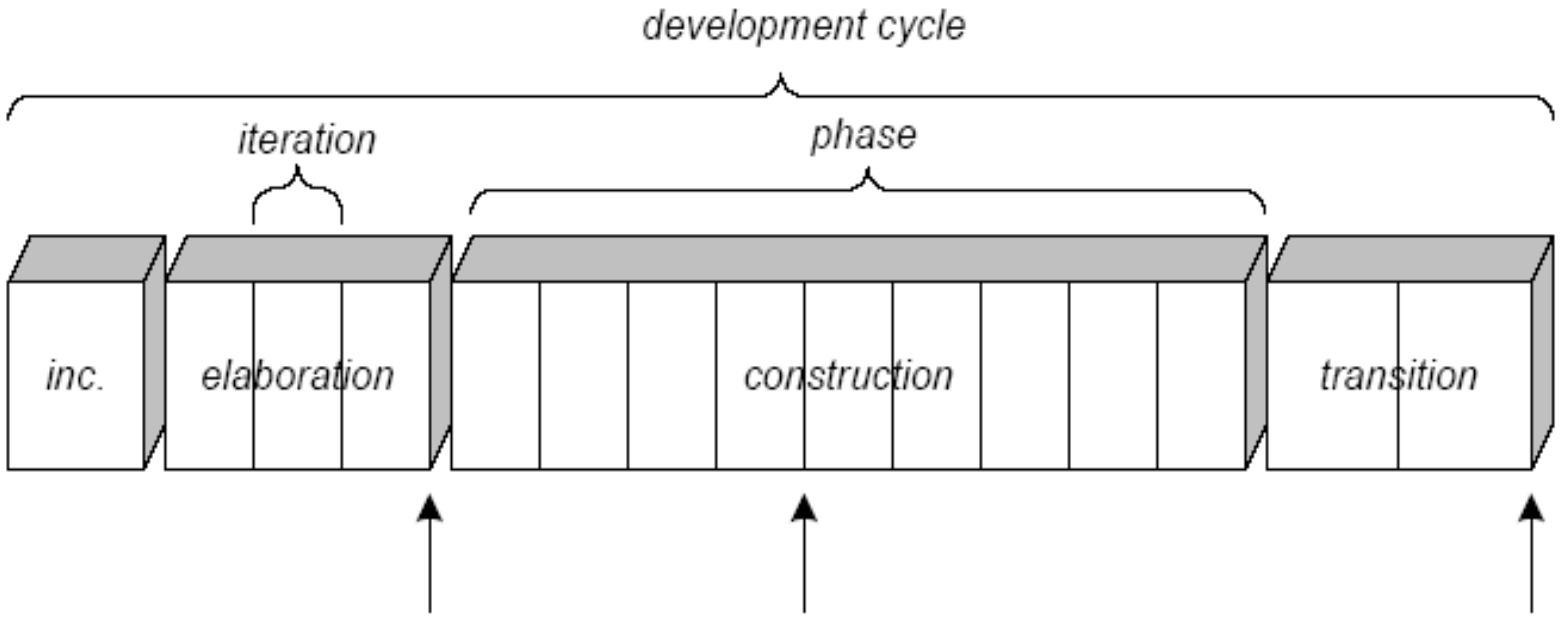


The waterfall lifecycle

Основна идея на Unified Process: итеративна разработка



Фази на UP и общо разписание



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.

Фази на UP

1. **Планиране (Inception)**.
2. **Детайлизиране (Elaboration)**
3. **Изграждане (Construction)**
4. **Предаване (Transition)**

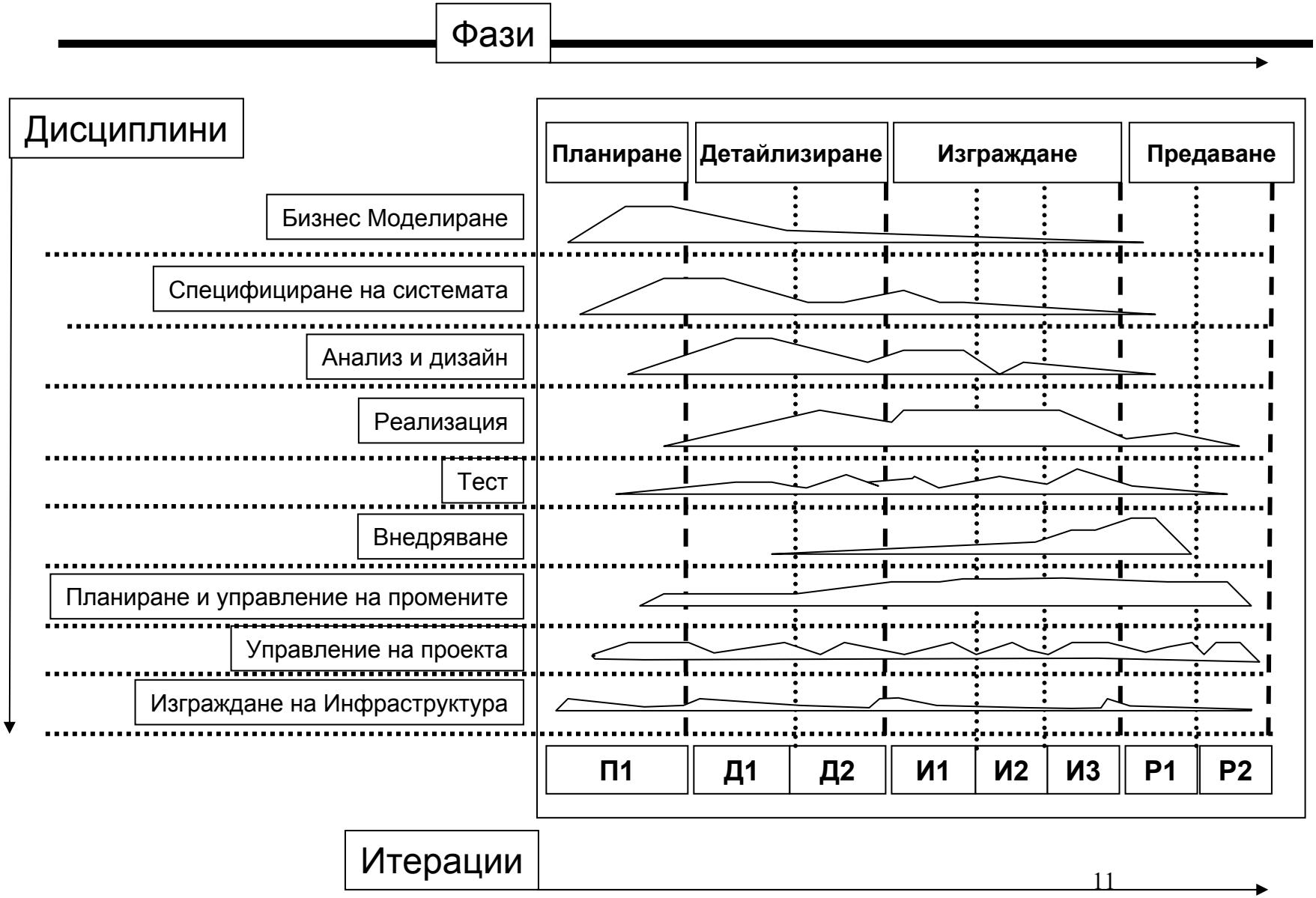
RUP

- **Фази**:
  - **Планиране (Inception)**
    - Дефиниране на обхвата на проекта – основните бизнес изисквания се формулират чрез набор от use-cases
    - Избор на обща архитектура – определяне на подсистеми
    - Планиране – ресурси, рискове, график
  - **Детайлизиране (Elaboration)**
    - Разширяване и детайлизиране на use-cases
    - Анализ и дизайн
    - Изграждане на архитектура

RUP

- **Фази**:
  - **Изграждане (Construction)**
    - Завършване на дейностите по анализ и дизайн
    - Разработване на софтуерни компоненти, които реализират отделните use-cases
    - Системни тестове (напр. unit и интеграционни тестове)
  - **Предаване (Transition)**
    - Обучение на потребители
    - Потребителски тестове
    - Отстраняване на грешки

RUP

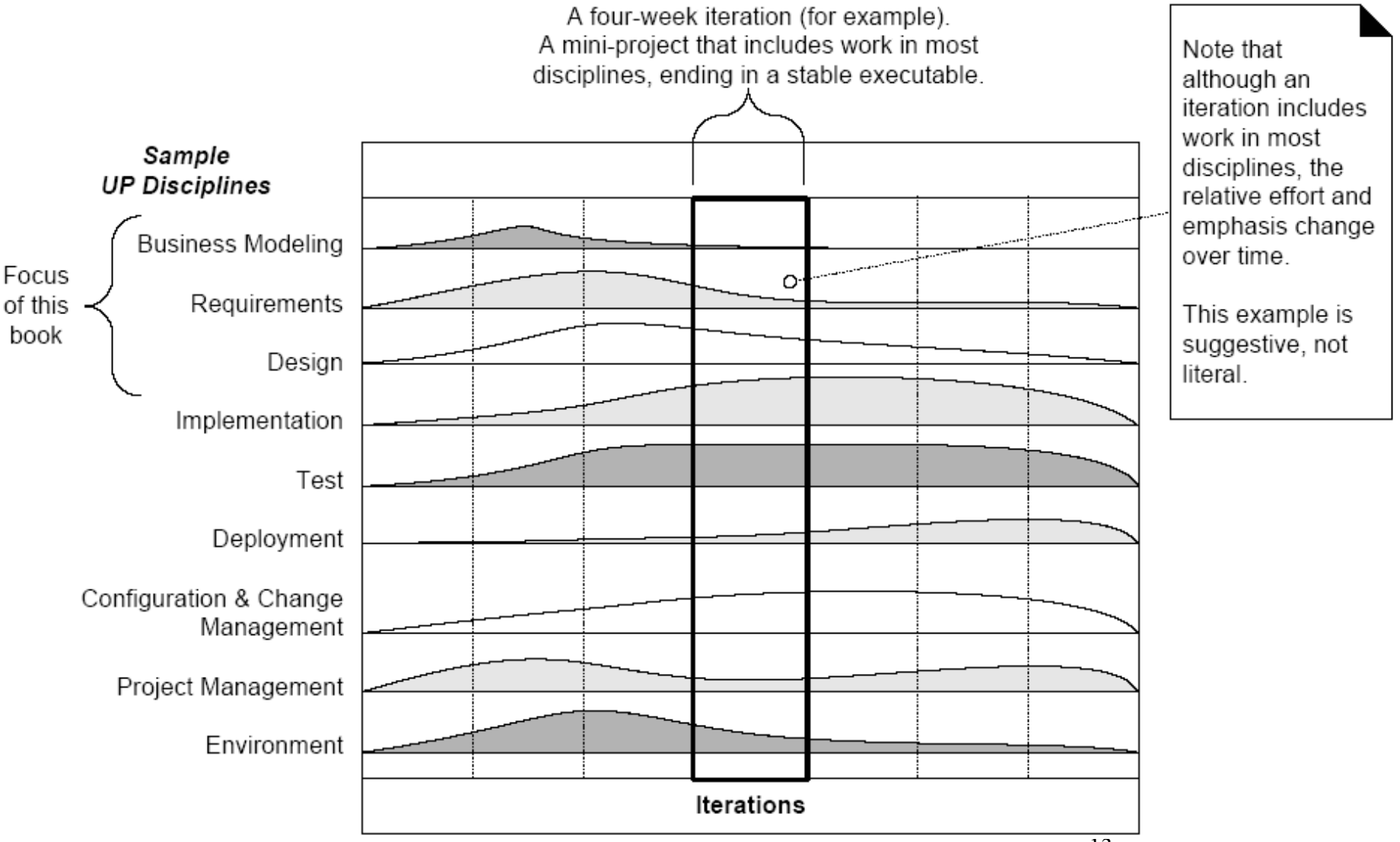

Фази
Дисциплини
Планиране
Детайлизиране
Изграждане
Предаване
Бизнес Моделиране
Специфициране на системата
Анализ и дизайн
Реализация
Тест
Внедряване
Планиране и управление на промените
Управление на проекта
Изграждане на Инфраструктура
П1
Д1
Д2
И1
И2
И3
Р1
Р2
Итерации

Настройка на процеса и разработката

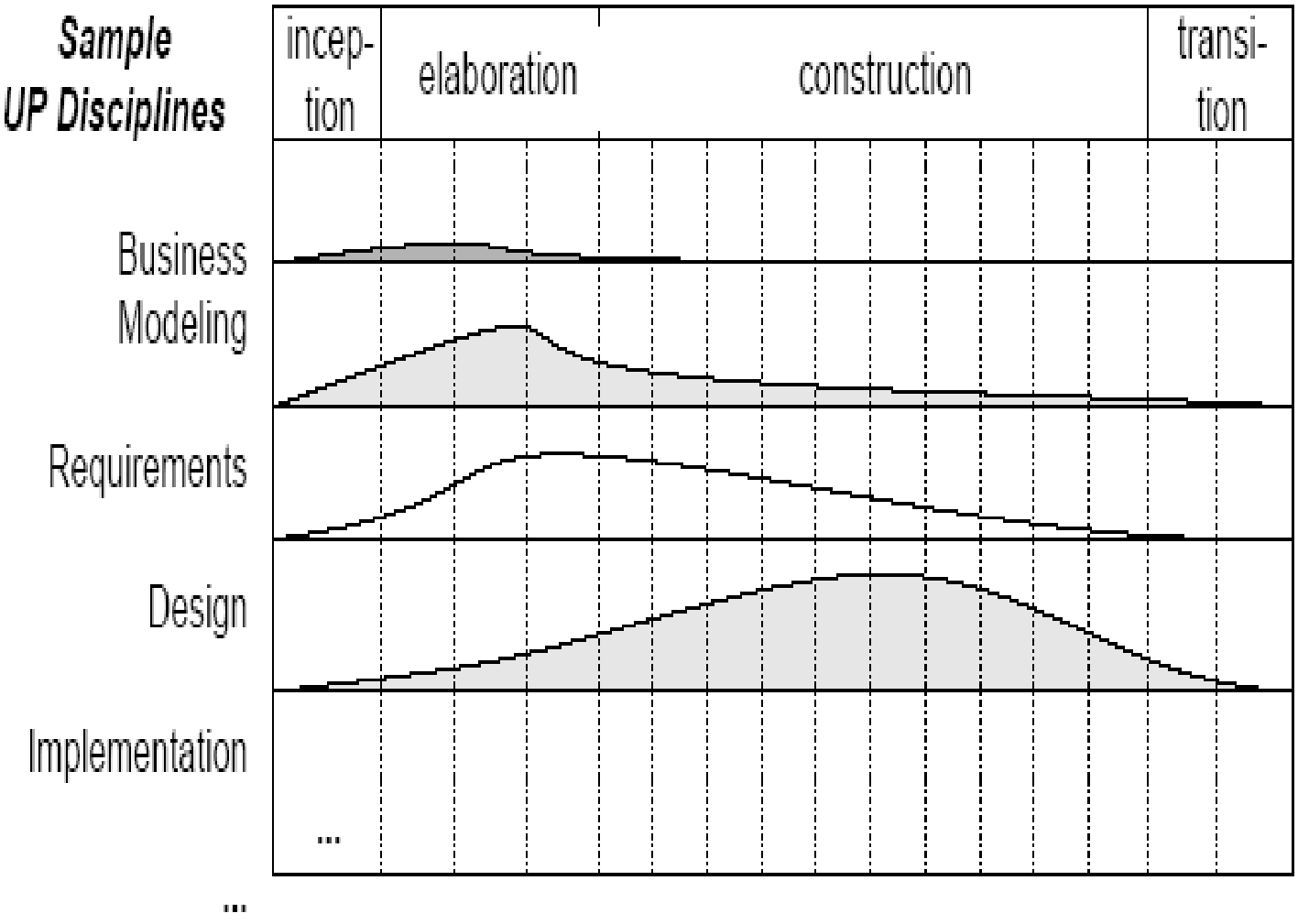
Discipline	Artifact Iteration-*	Incep. 11	Elab. El. .En	Const. CL.Cn	Trans. T1..T2
Business Modeling	Domain Model		s		
Requirements	Use-Case Model	s	r		
	Vision	s	r		
	Supplementary Specification	s	r		
	Glossary	s	r		
Design	Design Model		s	r	
	SW Architecture Document		s		
	Data Model		s	r	
Implementation	Implementation Model		s	r	r
Project Management	SW Development Plan	s	r	r	r
Testing	Test Model		s	r	
Environment	Development Case	s	r		

Table 2.1 Sample Development Case of UP artifacts, s - start; r - refine

UP дисциплини



UP дисциплини



The relative effort in disciplines shifts across the phases.

This example is suggestive, not literal.

UP дисциплини

- **Бизнес моделиране** (**Business Modeling**)— разработката на едно приложение включва моделиране на домейн обектите. При по-мащабен анализ или реинженеринг на бизнес процесите се включва динамично моделиране на бизнес процесите в цялото предприятие.

- **Изисквания** (**Requirements**) — анализ на изискванията, описание на потребителски случаи (use cases) и определяне на нефункционалните (non-functional) изисквания.

- **Дизайн** (**Design)** — всички аспекти на дизайна, включително цялостната архитектура, обектите, базите от данни, мрежите.

RUP

- Добри практики на RUP
  - Софтуерът се разработва итеративно
  - Изискванията се управляват
  - Софтуерът се моделира визуално
  - Верифицира се качеството
  - Контролират се промените в софтуера
  - ...

Допълнителни добри практики и концепции на UP

Някои допълнителни добри практики и ключови идеи в UP включват:

- отбелязване на високо-рисковите и важните следствия в ранните итерации
- непрекъснато взаимодействие с потребителите за оценка, корекции и изисквания
- създаване на свързана, централна архитектура в ранните итерации
- непрекъснато оценяване на качеството
- ранно тестване
- прилагане на потребителски случаи
- визуално моделиране на софтуера (UML)
- внимателно управление на изискванията
- практики на промяна на изискванията и управление на конфигурациите

Итеративна разработка и Unified Process (UP)

- Итеративната разработка е в центъра на OOA/D.
- Неформално процесът по разработка на софтуер описва начин за създаване, внедряване и възможна поддръжка на софтуера.
- **Unified Process (UP)** - процес за разработката на софтуер при реализация на обектно-ориентирани системи.
- **Rational Unified Process (RUP)** - детайлизирано приложение на UP

Unified Modeling Language

- UML (Unified Modeling Language) - език за определяне, визуализация, създаване и документиране на документи на софтуерни системи, както и за бизнес моделиране и други не-софтуерни системи
  - Grady Booch и Jim Rumbaugh комбинират двата най-популярни метода за създаване на диаграми – Booch и OMT (Object Modeling Technique) - 1994
  - По-късно към тях се присъединява Ivar Jacobson, създател на метода Objectory
    - (т.нар. група - The Three Amigos).
  - OMG (Object Management Group, група за индустриални стандарти) приема UML като стандарт – 1997

Използване на UML и шаблони в ООА/Д

- Какво означава да имаме добър обектен дизайн?
  - Основни умения в ООАД
- Приложение на UML
  - UML не е OOAD
  - UML - стандартна нотация за диаграми
  - Унифициране
- Приложение на UP
  - Итеративен и инкрементален
- Case Study
  - Използване на добър пример за изясняване на концепциите

Потребителски случаи

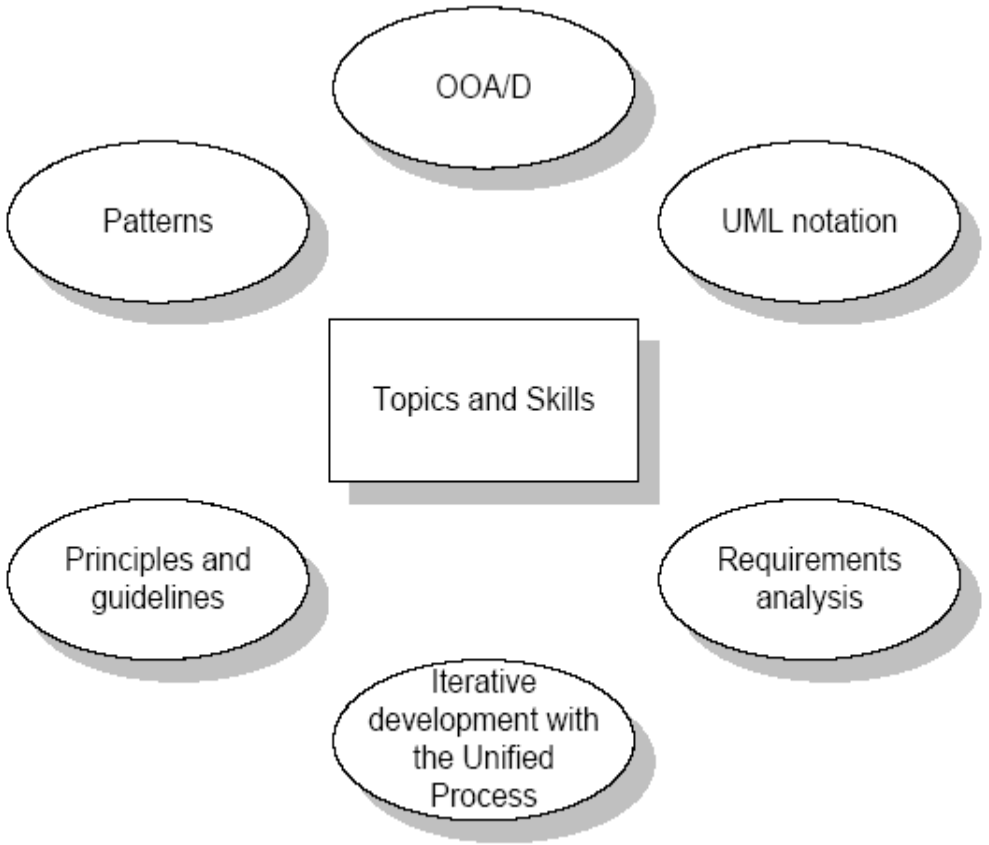
- Потребителски случаи (Use cases) и анализ на изискванията
  - OOA/D (и всеки софтуерен дизайн) е строго свързан с необходимостта от анализ на изискванията
  - Използване на **use cases**.
- Пример за повтарящ се процес - the Unified Process (UP)
  - Анализът на изискванията и OOA/D трябва да бъдат представени в контекста на някакъв процес на разработка.
  - В този случай Unified Process се използва като примерен повтарящ се процес на разработка, в който се представят тези теми.

Теми и умения

- Прилагане на правила и шаблони, за създаване по-добър дизайн на обектите.

- Следване на множеството от общите дейности в анализа и дизайна, на базата на Unified Process.

- Създаване на често използвани диаграми в нотацията на UML



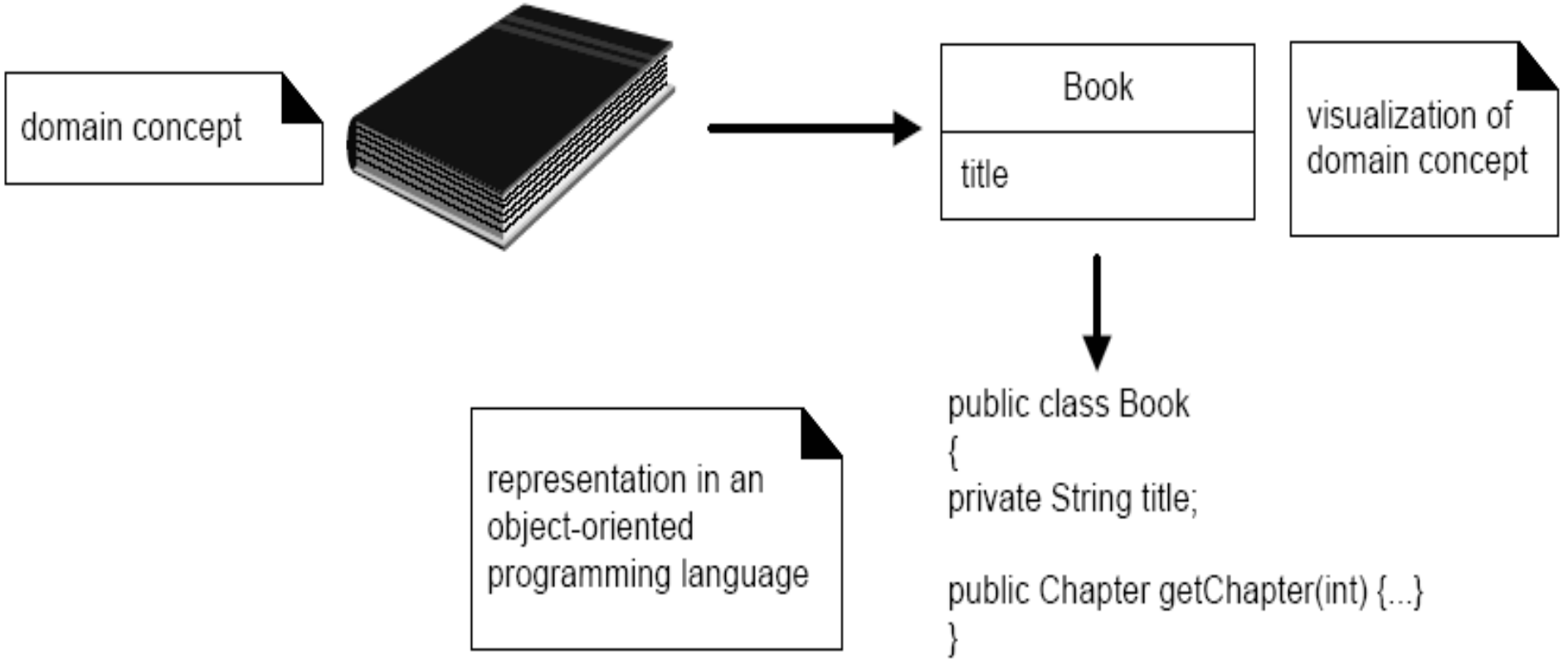
Анализ и проектиране (дизайн)

- **Анализът** набляга повече на изследването на проблема и изискванията, отколкото на решението.
  - “Анализ” е широко понятие, най-добре определено в анализа на изискванията (изследване на изискванията) или анализа на обектите (изследване на домейн обектите).
- **Дизайнът** набляга повече на концептуалното решение, което изпълнява изискванията, отколкото на имплементацията.
  - Пример - описание на схемата на базата от данни и софтуерните обекти
- **Анализът** и **дизайнът** могат да се опишат чрез фразите:
  - направи правилното нещо (анализ) – ***do the right thing (analysis),***
  - направи нещата правилно (дизайн) – ***do the thing right (design)***.

Обектно-ориентиран анализ и дизайн

- По време на **обектно-ориентирания анализ** се набляга на откриване и описване на обектите и/или понятията в разглеждания домейн (предметна област).
  - Пример: В ИС на библиотека, някои от понятията са *Book*, *Library* и *Patron*.
- По време на **обектно-ориентирания дизайн** се набляга на дефиниране на софтуерните обекти и как те си взаимодействат за изпълнение на изискванията.
  - Пример: В библиотечната система, софтуерният обект *Book* може да има атрибут *title* и метод *getChapter*. Накрая, по време на имплементацията или обектно-ориентираното програмиране, обектите се имплементират, например клас *Book* в Java.

Обектно -ориентиран анализ и дизайн



Основни стъпки

- Дефиниране на потребителски случаи
- Определяне на домейн модел
- Създаване на диаграми на взаимодействието
- Създаване на диаграми на класовете
- Създаване на ...

Потребителски случаи (use cases)

Дефиниране на Use Cases

Анализът на изискванията може да включва описание на свързаните домейн процеси - те могат да се опишат като потребителски случаи (**use cases).**

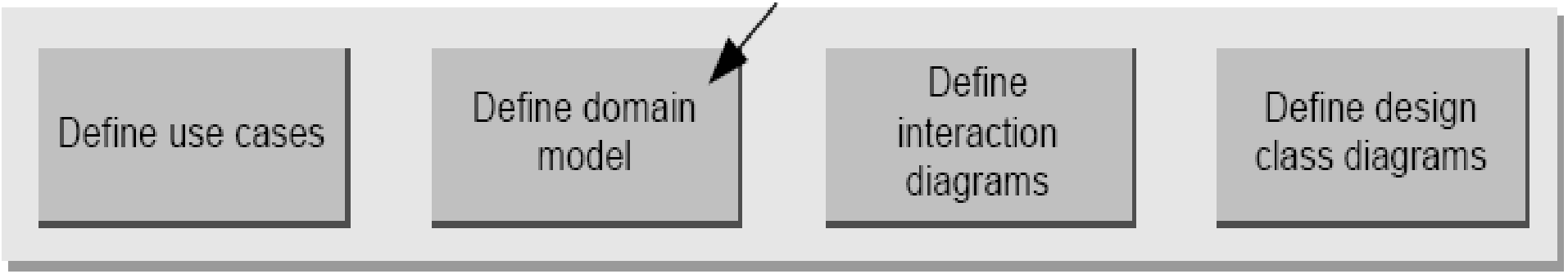
Потребителски случаи - пример

- Потребителските случаи (use cases):
  - са просто написани сюжети;
  - често се използват в анализа на изискванията;
  - не са обектно-ориентиран документ;
  - представляват важна част от унифицирания процес (Unified Process).
- Потребителски случай “Игра на зарове” – *кратко описание*:
  - Играчът хвърля 2 зарчета. Ако резултатът е общо седем, печели, иначе - губи.

Домейн модел

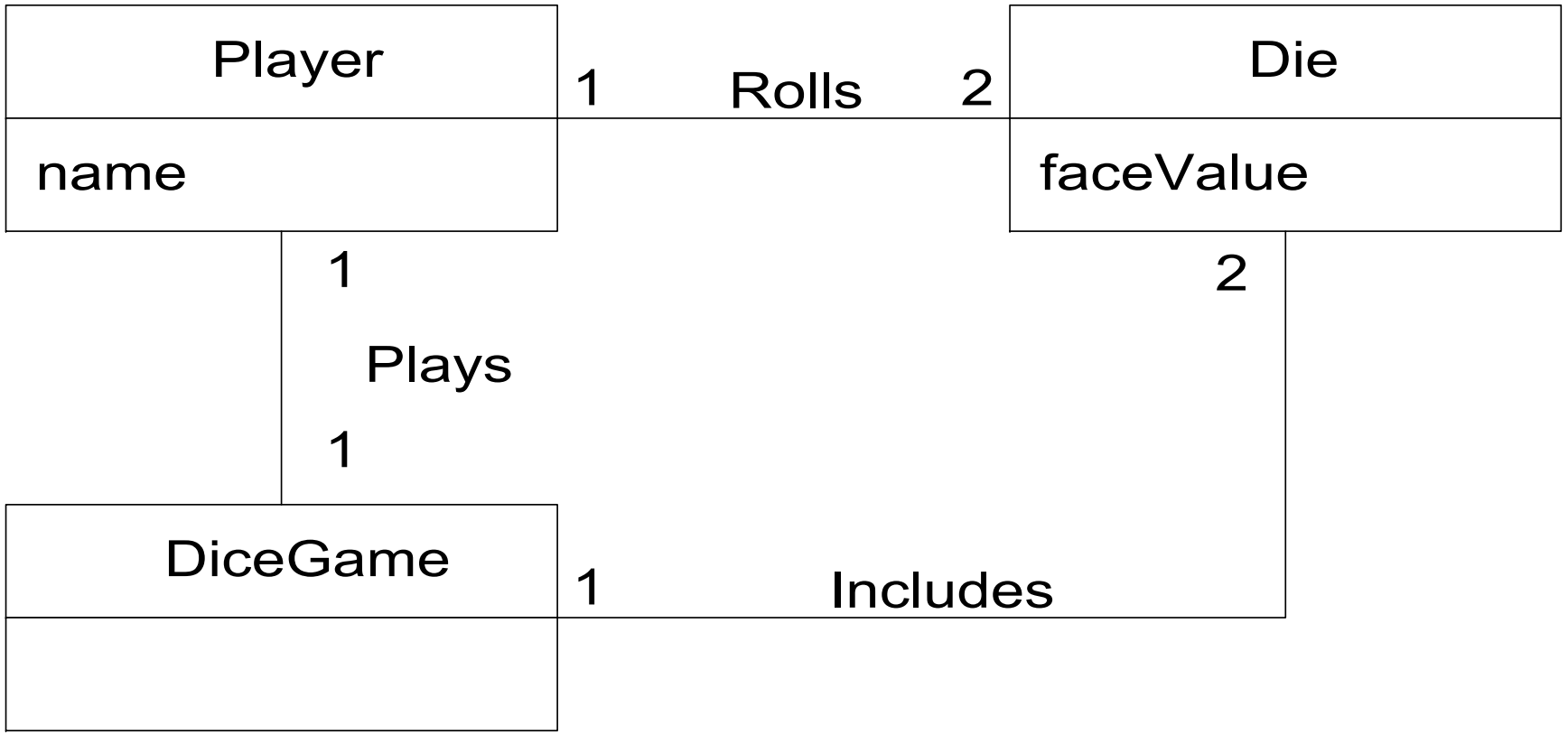
Дефиниране на домейн модела

ООД се занимава със създаване на описания на домейните от перспективата на класификация по обекти. Декомпозицията на домейна включва идентифициране на идеите, атрибутите и асоциациите, които се разглеждат изцяло. Резултатът може да се изрази в модел на домейна, който се обозначава с множество от диаграми, които показват идеите или обектите на домейна.



Домейн модел- пример

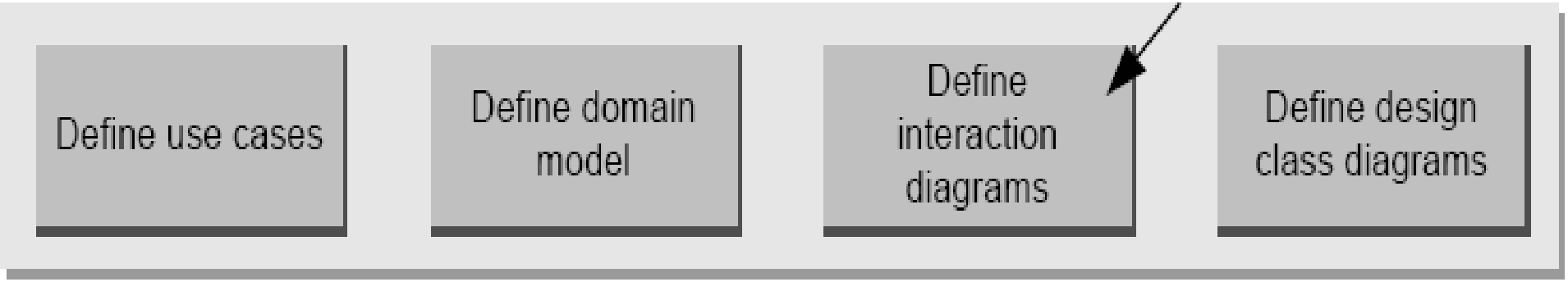
Елемент от домейн модела при игра на зарчета:



Диаграми на взаимодействието

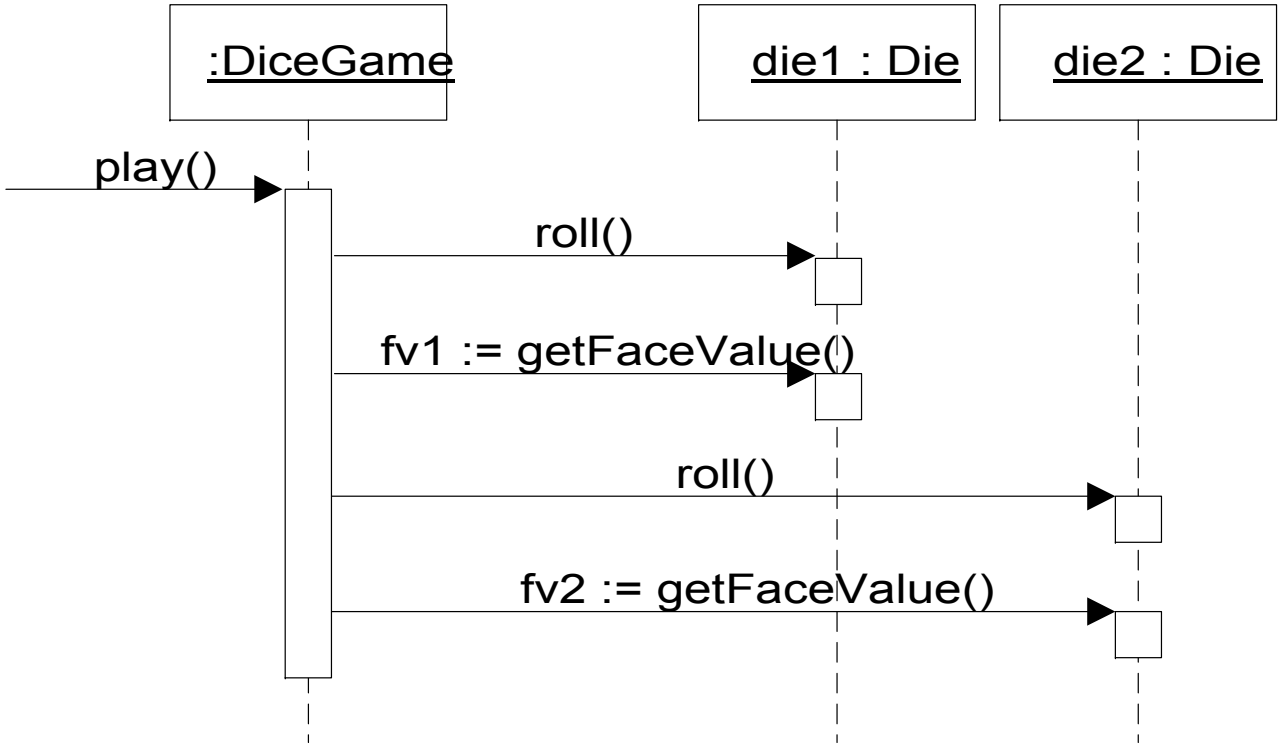
Дефиниране на диаграмите на взаимодействието (Interaction Diagrams)

ООД се занимава с дефиниране на софтуерни обекти и тяхното взаимодействие. Най-често, за да се илюстрира това взаимодействие се използват диаграмите на взаимодействието. Те показват потока съобщения между софтуерните обекти и обръщенията към методите.



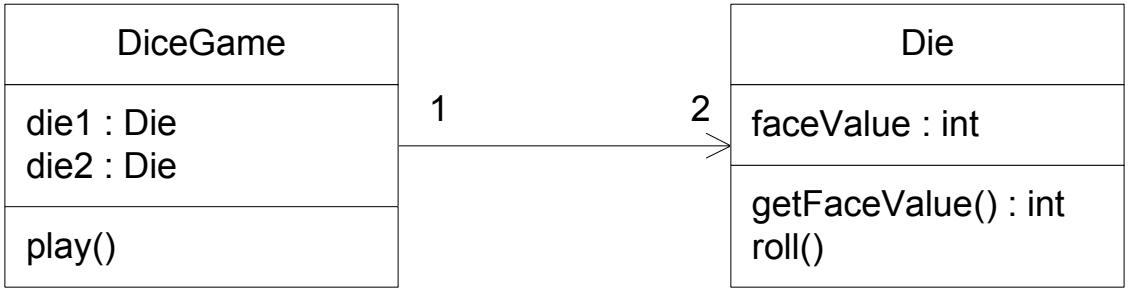
Диаграми на взаимодействието - пример

Диаграмата на взаимодействието илюстрира важна стъпка от играта като изпраща съобщения към наследниците на класовете *DiceGame u Die*.

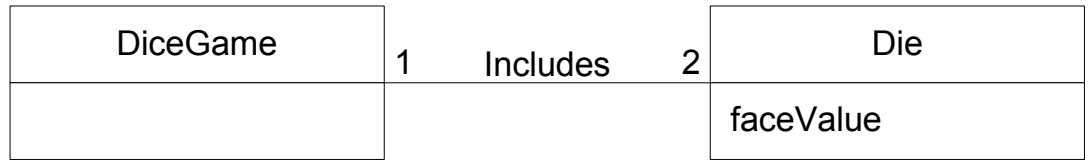


Диаграми на класовете - пример

Част от клас диаграма на игра със зарчета:

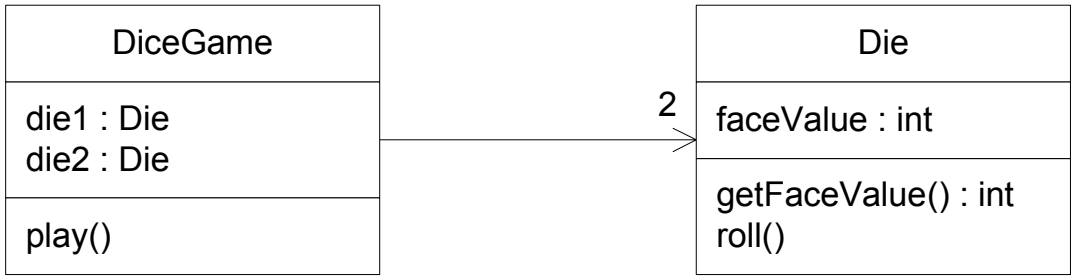


Домейн модел и диаграми на класовете



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.