System Analysis and Design

L03. UP and Agile Methods

Topics

- Gitee sign up:
  - https://gitee.com/cssysu?invite=b810bf1b08efe5891f16a8a78751823ac20f3b2416bcd2898e7cde0b62298f89b1a29de438e524ba439bc1f65eaa027829b3ecda9b20b2e0
- 分组
  - 10个组(一组7个成员,其中一个组长)
  - Cmake, uml, git, linux, c++, …
  - Mesh application

Topics

- Unified Process
- Agile Development Methods

Unified Process

- The **Unified Process** is a popular *iterative* and *Incremental* software development process for building object-oriented systems.
- The process is **scalable**: you need not use the entire framework of the process for every project, only those that are effective.
- The process is **effective**: it has been successfully employed on a large population of projects.
- Improves **productivity** through use of practical methods that you've probably used already (but didn't know it).
- Iterative and incremental approach allows start of work with incomplete, imperfect knowledge.
- UP is **Use Case Driven**
- UP is **Architecture Centric**
- UP is **Risk Focused**

什么是用例驱动？

Use Case Driven

- Use case
  - A <u>prose</u> representation of a sequence of actions 段落描述
  - Actions are performed by one or more *actors* (human or non-human) and the system itself
  - These actions lead to valuable results for one or more of the actors—helping the actors to achieve their goals
- Use cases are expressed from the perspective of the users, in natural language, and should be understandable by all stakeholders 用户的角度
- ***Use-case-driven*** means the development team employs the use cases from requirements gathering through code and test

以系统结构为中心

Architecture Centric

- Software architecture captures decisions about:
  - The **overall structure** of the software system
  - The **structural elements** of the system and their **interfaces**
  - The **collaborations** among these structural elements and their expected behavior
- Architecture-centric: software architecture provides the central point around which all other development evolves
  - Provides a '**big picture**' of the system
  - Provides an **organizational framework** for development, evolving the system by attending to **modifiability** qualities of the system
  - Facilitates **reuse**

Risk-Driven and Client-Driven Planning

- UP encourages a combination of **risk-driven** and **client-driven iterative** planning.
- This means that the goals of the early iterations are chosen to
  - Identify and minimize highest risks
  - Build visible features client wants most.
- Risk-driven iterative development includes more specifically the practice of **architecture-centric** iterative development, meaning that early iterations focus on building, testing, and stabilizing the core architecture.
- Why? Because not having a solid architecture is a common high risk.

主要的(关键的)实践经验与思想

Critical UP Practices

- UP practices provide **an example *structure*** for how to do and thus how to explain OOAD.
- The central idea to appreciate and practice in the UP is short timeboxed iterative, evolutionary, and adaptive development. Some additional best practices and key concepts in the UP:
  - Tackle high-risk and high-value activities early
  - Continuously engage users
  - Build cohesive core architecture early
  - Test early, often, and realistically
  - Apply use cases where appropriate
  - Do visual modeling
  - Carefully manage requirements 实用的变化请求与配置管理
  - Practice change request and configuration management.

Unified Process Phases UP的阶段

- A UP project organizes the work and iterations across four major phases:

四个阶段

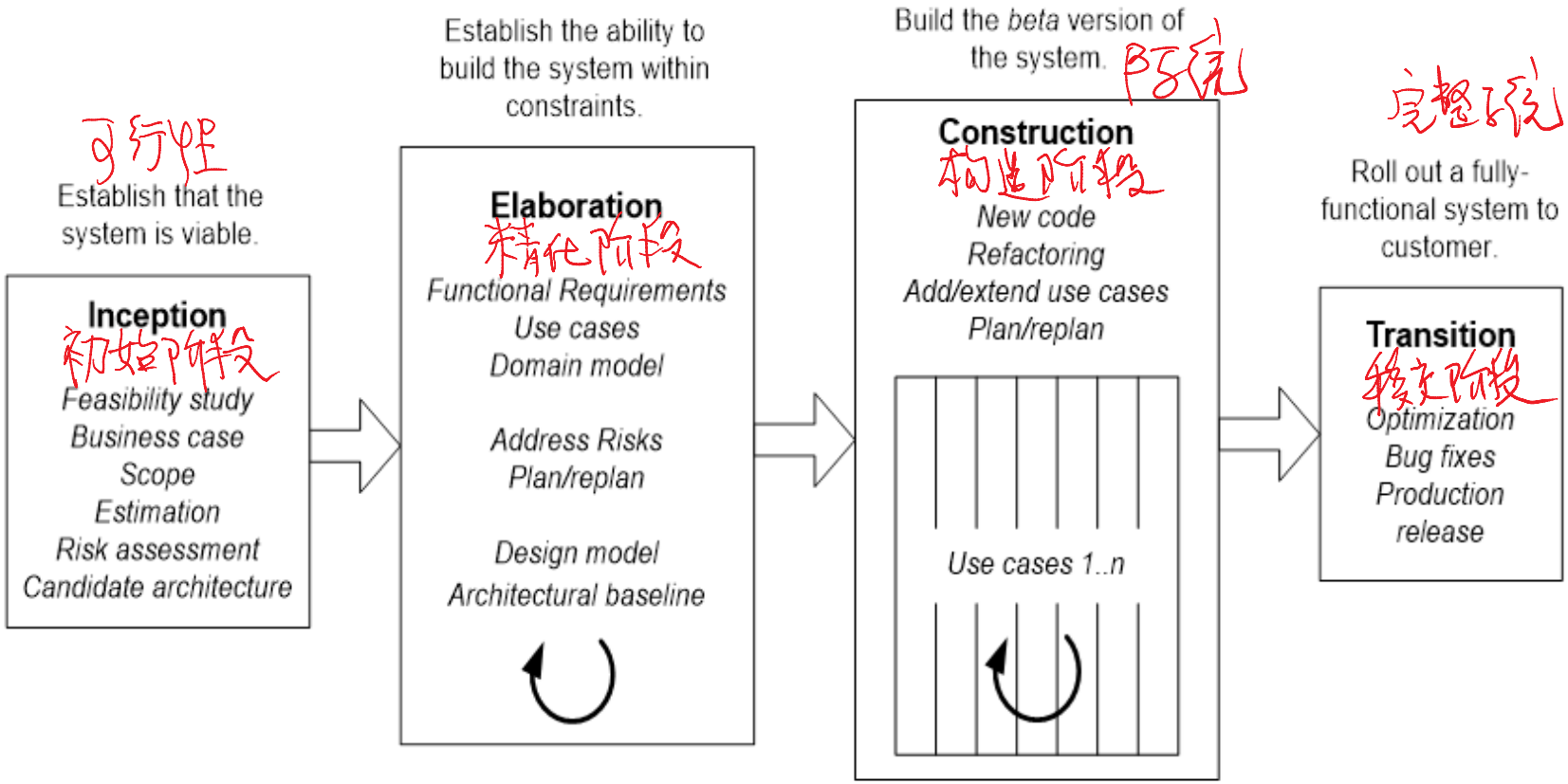
  - **Inception** feasibility phase and approximate vision, business case, scope, vague estimates.
  - **Elaboration** refined vision, iterative implementation of the core architecture, resolution of high risks, identification of most requirements and scope, more realistic estimates.
  - **Construction** iterative implementation of the remaining lower risk and easier elements, and preparation for deployment.
  - **Transition** beta tests, deployment.
- This is *not* the old "waterfall" or sequential lifecycle of first defining all the requirements, and then doing all or most of the design.
- These phases are more fully defined in subsequent chapters.

Unified Process Phases

- Inception is not a requirements phase;
  - rather, it is a **feasibility phase**, where just enough investigation is done to support a decision to continue or stop.
- Elaboration is not the requirements or design phase;
  - rather, it is a phase where **the core architecture is iteratively implemented,** and
  - high-risk issues are mitigated. 降低高风险

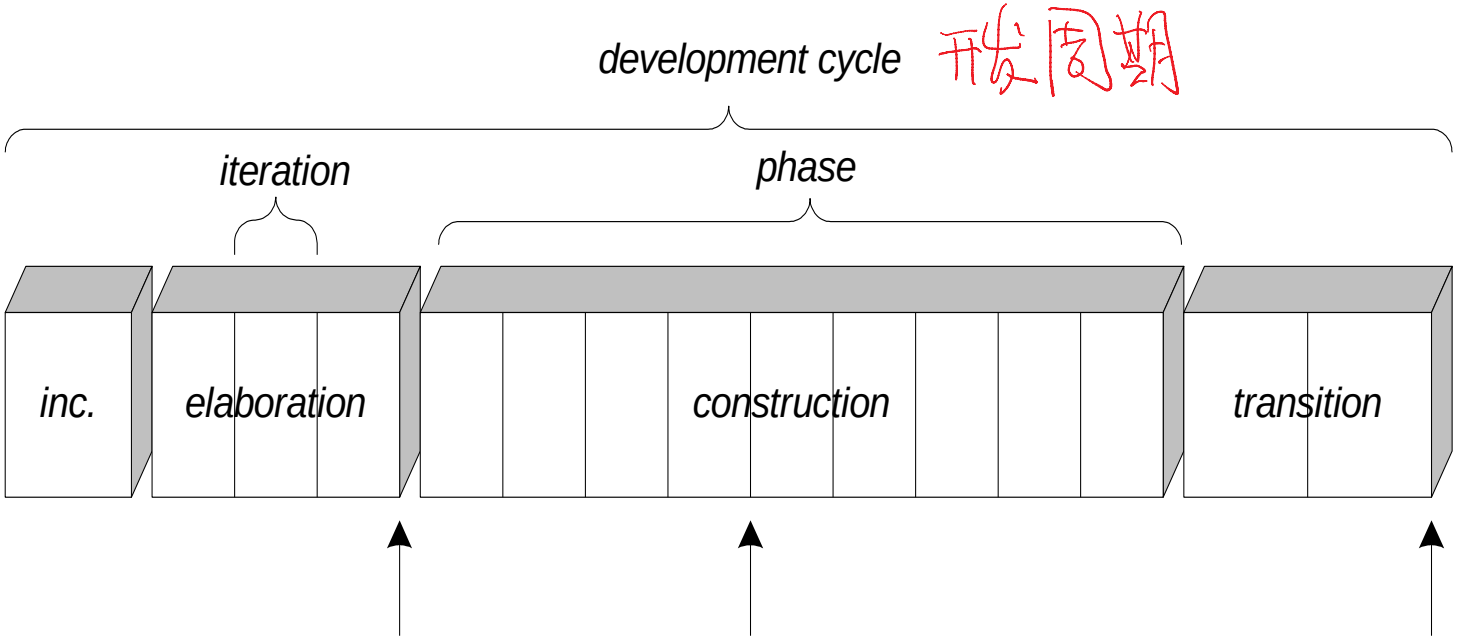
Unified Process Phases

UPrisa四个阶段



- Inception – “Daydream”
- Elaboration – “Design/Details”
- Construction – “Do it”
- Transition – “Deploy it”
- Phases are not the classical requirements/design/coding/implementation processes
- Phases iterate over many cycles

UP Phases are Iterative & Incremental



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

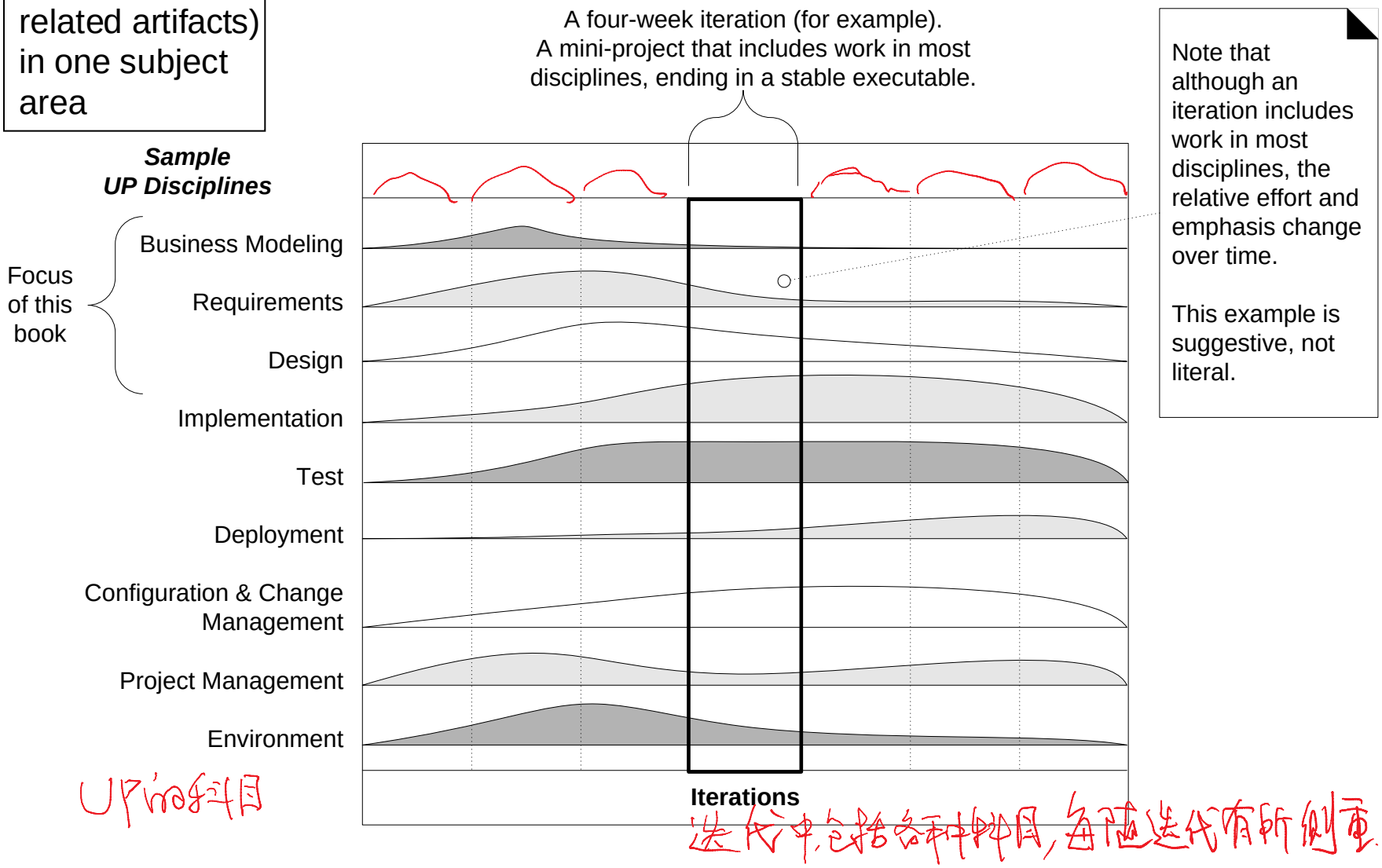
The difference (delta) between the releases of 2 subsequent iterations.

final production release

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

UP Disciplines and Phases

A discipline is a set of activities (and related artifacts) in one subject area



Disciplines look traditional, but they iterate in four phases

UP Activities UP活动

- Activities are the tasks performed within a workflow (discipline)
- Activities can describe a wide range of abstraction levels, from high-level ('construct domain model') to low-level ('implement class'). Examples include:
  - Plan iteration
  - Find use cases and actors
  - Execute integration test
  - Review test results

UP Artifacts UP工件

- The UP describes work activities,
  which result in *work products* called *artifacts*
- Examples of artifacts:
  - Vision, scope and business case descriptions
  - Use cases (describe scenarios for user-system interactions)
  - UML diagrams for domain modeling, system modeling
  - Source code (and source code documentation)
  - Web graphics
  - Database schema

Workers

- Workers define the behavior and responsibilities of an individual or a team
  - Examples: Architect, use-case engineer, component engineer, system integrator
- Some important distinctions:
  - Workers participate in the development of the system
  - Actors are outside the system and have usage relationships with the system
  - Stakeholders encompass both actors and workers, as well as others involved with the project

Customizing the UP

- Some UP practices and principles are invariant, such as iterative and risk-driven development, and continuous verification of quality.
- Most activities and artifacts are optional
- Choose the ones that make sense for your project

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.
- Short timeboxed iterations with evolutionary refinement of plans, requirements, and design is a basic practice.
- They promote practices and principles that reflect an agile sensibility of **simplicity, lightness, communication, self-organizing teams, and more**.
- Examples: UP, Extreme Programming (XP), Scrum

The Agile Manifesto 宣言

We Value	
Individuals and interactions	over processes and tools
Working software	over comprehensive documentation
Customer collaboration	over contract negotiation
Responding to change	over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

The Agile Principles (1) 原則

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.

The Agile Principles (2)

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**.

The Agile Principles (3)

10. Continuous attention to technical excellence and good design **enhances agility**.
11. **Simplicity** the art of maximizing the amount of work not done is 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**.

Agile Modeling (1)

- The purpose of modeling is **to understand**, not to document
- Purpose is not for designer to create UML diagrams that are then given to a programmer.
- Don’t model or apply the UML to all or most of the design. **Use it for unusual, difficult, or tricky parts**.
- Use simplest tools possible
- Don’t model alone, model in pairs (or triads), share understanding

Agile Modeling (2)

- Create models in parallel. a dynamic-view with the complementary static-view
- Use “good enough” simple notation 而不是总用精确的UML.
- Know that all models are inaccurate; the final arbiter of the design is the code
- Developers should do the OO design modeling for themselves, not other programmers 否则就是灌布思想

What is an Agile UP?

The UP can be adopted and applied in the **spirit of adaptability and lightness**, that was an agile UP. 在UP中践行可适应性和轻量级的思想，就是敏捷UP，例如：

- Prefer a *small* set of UP activities and artifacts.
- Since the UP is iterative and evolutionary, requirements and designs are not completed before implementation. They adaptively emerge through a series of iterations, based on feedback.
- Apply the UML with agile modeling practices.
- There isn't a *detailed* plan for the entire project. There is a high-level plan (called the **Phase Plan**) that estimates the project end date and other major milestones, but it does not detail the fine-grained steps to those milestones.
- A detailed plan (called the **Iteration Plan**) only plans with greater detail one iteration in advance. Detailed planning is done adaptively from iteration to iteration.

What is the Development Case?

- The choice of practices and UP artifacts for a project may be written up in a short document called the **Development Case** (an artifact in the Environment discipline).
- Subsequent chapters describe the creation of some of these artifacts, including the Domain Model, Use-Case Model, and Design Model.

定制开发案例（一个例子）

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		start		
Requirements	req. workshop vision box exercise dot voting	Use-Case Model	s	refine		
		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	…				
…						

Summary

- Unified Process
- Unified Process Phases
- Unified Process Disciplines, Activities, Artifacts
- Agile Development Methods
  - The Agile Manifesto
  - The Agile Principles
  - Agile Modeling
  - Agile UP