

Systems Development Methodologies

Chapter 22

In this Lecture you will Learn:

- What a systems development methodology is
- Why methodologies are used
- The need for different methodologies
- The main features of one methodology

Why Methodology?

- Techniques of system development must be organized if they are to work together
 - ◆ Analyst has drawn collaboration diagrams for main use cases
 - ◆ Should she now convert these to sequence diagrams and write operation specifications?
 - ◆ Or concentrate on class diagram, inheritance and composition structures?
- UML itself contains nothing that helps to make this decision

Why Methodology?

- Organization of tasks is not contained within the techniques
- Must be described at a higher level of abstraction
- Method of a project means the particular way that tasks in that project are organized
- Sometimes called **Process of Software Development**

Method or Methodology?

- **Method** – An instantiation of the principles in a given situation, the step-by-step description of the steps involved in doing a job
- **Methodology** – Set of general principles that guide the choice of a method suited to a specific task or project
- No two projects are identical, so method is specific to one project

Task vs. Technique

- A **Task** is something you do in a particular project. Tasks have products.
 - ◆ A task might be ‘Analyze the requirements for a use case’.
- A **Technique** specifies how to carry out a task.
 - ◆ One technique for doing this would be the UML collaboration diagram.

Increasing Level of Abstraction in Defining a Project

Level of Abstraction	Example of Application	Typical Product
Task	Developing a first-cut class diagram	Specific version of a class diagram
Technique	Description of how to carry out a technique, e.g. UML class modelling	Any UML class diagram
Method	Specific techniques used on a particular project that lead to a specific software product	A product costing system
Methodology	General selection and sequence of techniques capable of producing a range of software products	A range of business software applications

What is a Methodology?

- Avison and Fitzgerald (1988): a collection of
 - ◆ Procedures
 - ◆ Techniques
 - ◆ Tools
 - ◆ Documentation aids
- Organized within a lifecycle structure
- Usually also with an underlying philosophy which captures a particular view of the meaning and purpose of Information System development

Elements of a Methodology

- UML class diagram is a **Technique**, and so is operation specification
- Rational Rose (CASE software) is a **Tool**
- Find classes by inspecting use case descriptions is a **Procedure**
- Operation specifications should not be written until class model is stable is an **Aspect of Structure**
- Analysis and design are **Stages** or **Activities**
- OO development promotes software which is robust and resilient to change is a **Philosophy**

Logical Views of a System

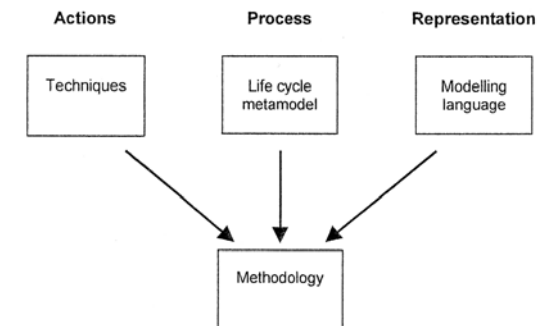
- Three complementary views in software development:
 - ◆ **Data View** – Attributes and associations that must be stored within the software
 - ◆ **Process View** – operations carried out on the data
 - ◆ **Temporal View** – sequence of processes and time constraints (also sequences of events that impinge on the system)
- Methodology needs techniques to discover, analyse and model each of these views

Why Use a Methodology?

- Helps produce better quality product
 - ◆ Better documented software
 - ◆ More acceptable to user
 - ◆ More maintainable software
 - ◆ More consistent software
- Helps ensure user requirements are met
- Helps project manager control the project
- Reduces development costs
- Promotes communication among all parties

Object-Oriented Development: OPEN

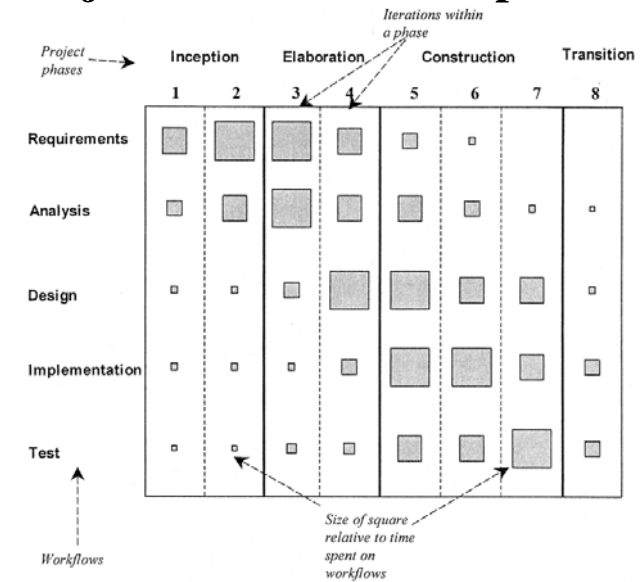
- OPEN Stands for **Object-oriented Process, Environment and Notation** (Graham et al., 1998)
- The methodology includes specifications of:
 - ◆ Activities
 - ◆ Tasks
 - ◆ Techniques
 - ◆ Deliverables
 - ◆ Notation



OPEN

- Basic principles include iterative and incremental development, and
- In OPEN, activities
 - ◆ Have both pre- and post-conditions
 - ◆ Are carried out within timeboxes (as in DSDM)
- OPEN has its own notation, COMN, that is a rival to UML

Object-Oriented Development: USDP

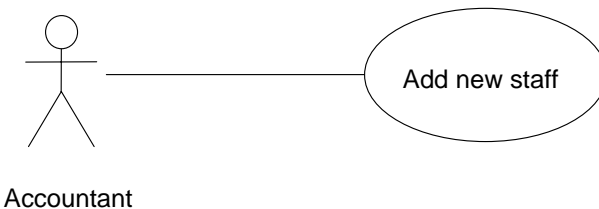


Unified Software Development Process (USDP)

- Public domain methodology for Object-Oriented software development
- Produced by Rational.com team
- The key elements in the philosophy of the USDP.
 - ◆ Use-case Driven
 - ◆ Architecture-centric
 - ◆ Iterative Development
 - ◆ Incremental Development

Key Elements: Use-case Driven

- Each use case is a thread that links a series of models from requirements through to implementation; it is a constant reminder to the systems developers that users' requirements are what matters.

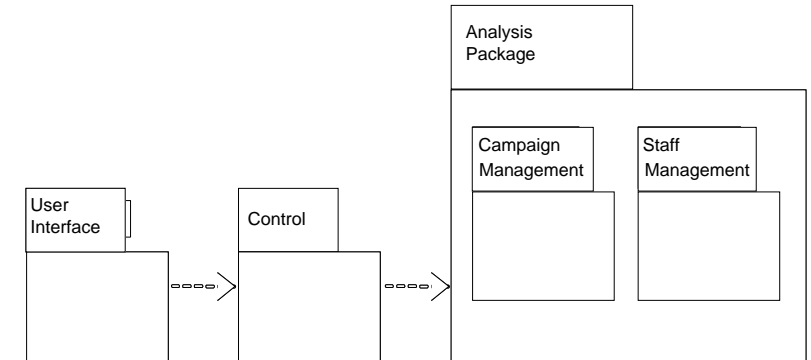


Key Elements: Use-case Driven

- In USDP the basic element is a single interaction between user and system
- Use cases are the start of modeling
- Unit from which later models are derived
- Each use case is a thread that links requirements to implementation
- Has practical significance for users
- Constant reminder to systems developers that only users' requirements really matter

Key Elements: Architecture-Centric

- Software architecture is an essential theme in modeling from the earliest stages of a project.



Key Elements: Architecture-Centric

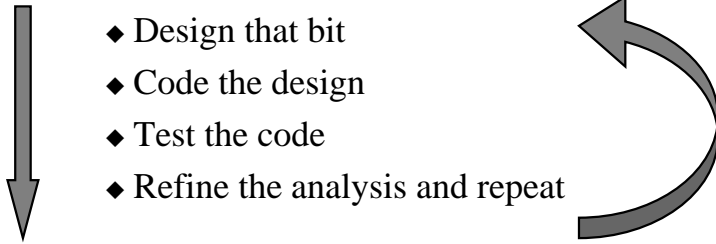
- In USDP, software architecture is a theme from the earliest stages of a project
- Reflected most obviously in:
 - ◆ Stereotyping of boundary, control and entity classes
 - ◆ Use of packages to organize both models and development activity

Key Elements: Iterative

- At each stage of the project, the activities are essentially the same, but they are carried out in an iterative manner.
- A unit of software may pass through several cycles of analysis, design, implementation and testing before it meets users' approval.

Key Elements: Iterative

- The USDP lifecycle is cyclic:
 - ◆ Analyse a bit
 - ◆ Design that bit
 - ◆ Code the design
 - ◆ Test the code
 - ◆ Refine the analysis and repeat



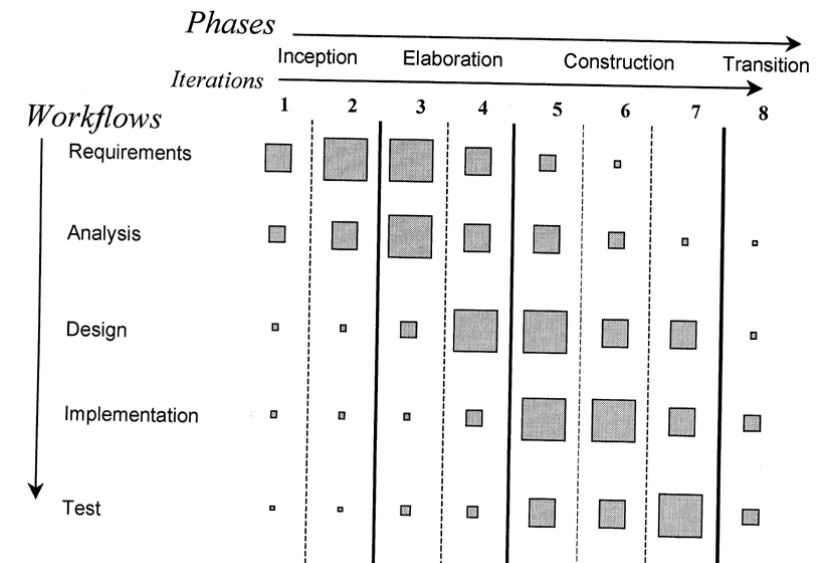
Key Elements: Incremental

- The system is built as a series of increments.
- In the simplest terms, each use case is an increment or unit of delivery that has practical significance to users.
- USDP aims to deliver working, free-standing, useful 'chunks' of software, one at a time
- In its simplest form, each use case may represent one increment of delivered software

Activities and Phases for USDP

- **Activity** has meaning for developers
- **Phase** matters to project manager
- Phases are sequential, delineated by milestones
- Each milestone is a decision point:
 - ◆ Begin next phase or stop now?
- Manager's focus shifts from one phase to the next
- Within each phase, activities iterate

Activities and Phases for USDP



Phases and Iterations

- No set rule for number of iterations
- Within a phase, workflows are the same
- All 4 phases run from requirements to testing, but emphasis changes
- At first, main effort is on capture, modeling, analysis of requirements
- Later phases emphasize implementation and testing

Inception Phase

- Essentially a feasibility stage: do potential risks outweigh potential benefits
- Decision based partly on CBA
- Viability of project judged on delivery of a small subset of requirements as working software

Inception Phase

- Main activities:
 - ◆ Requirements capture
 - ◆ Analysis
 - ◆ Small amount of design, implementation and testing
- Even at this early stage, iteration is likely
- OO approach makes this possible

Elaboration Phase

- Produce design for a suitable system
- Aim is to reduce cost uncertainties
- Demonstrate how system can be built within acceptable timescale and budget
- Proportion of time spent on design activities increases
- Small increase in time on implementation and testing (still small in relation to analysis and design)

Construction Phase

- Build, through several iterations, a system capable of satisfactory operation in target environment
- Implementation and testing rapidly become core activities
- Each iteration moves away from design and towards testing

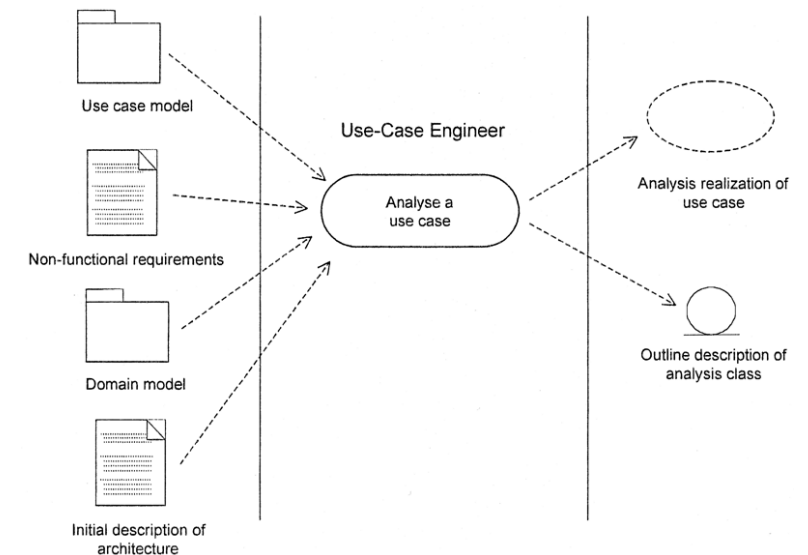
Transition Phase

- Achieves the intended full capability of the system
- Deals with any defects or problems that have emerged
- Includes system conversion, if older system is being replaced

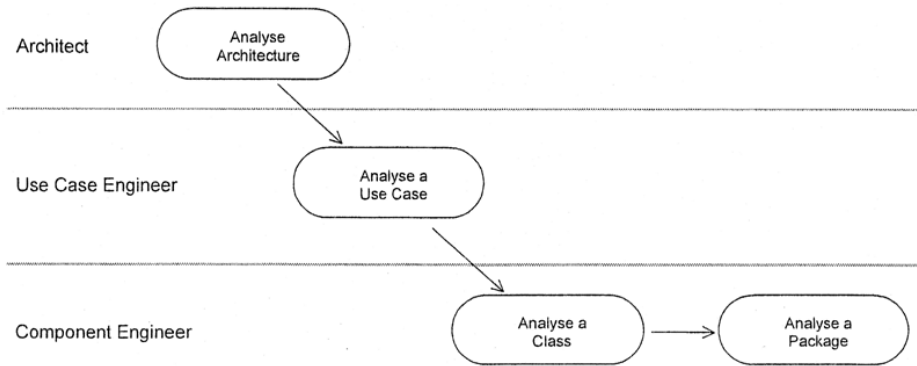
Workers and Activities

- USDP differentiates between real people and the more abstract worker
- A Worker is a project role, e.g.
 - ◆ Use-case specifier
 - ◆ System architect
 - ◆ Component engineer
 - ◆ Integration tester
- No direct one-to-one mapping between people and workers

Inputs and Outputs of an Activity



The Analysis Workflow in USDP



USDP: Summary

- The most mature OO methodology yet
- Has roots in:
 - ◆ Booch method (good at design and implementation)
 - ◆ OMT (good at analysis)
 - ◆ Objectory (strong at requirements engineering and system architecture)
- USDP strives to bring these together
- Also large and complex: significant learning curve involved, or tailor to fit