CS 430 - Lecture 06 - The Unified Process II

Collin Roberts

September 26, 2023

Outline

The Interaction Between Phases and Workflows

As pointed out last lecture,

1. phases have a **time orientation** (questions like "when do we need to do deliver artifact x?" can be answered by naming a phase), and
2. workflows have a **task orientation** ("with what related artifacts should artifact x be grouped?" can be answered by naming a workflow).

Motivation to Separate Workflows and Phases

Why 1-D models like Waterfall break down in practice: they assume that the time and task orientations agree with one another. In reality they do not, because of the **moving target problem**. Iteration and Incrementation leads to the splitting of tasks and time, which in turn leads us to the Unified Process.

Global Remarks

1. What follows is a summary of what artifacts are typically produced for each workflow and phase.
2. Different projects will require different timelines, and hence different phase breakdowns.

Goal

Determine whether it is worthwhile to develop the target software product. Is it economically viable to build it?
Here we explain the interaction between the Inception phase and each workflow (i.e. which workflow artifacts are typically produced during the Inception phase).

Requirements Workflow - Key Steps

1. Understand what is

Definition 1

The **domain** *of a software product is the place (e.g. TV station, hospital, air traffic control tower, etc.) in which it must operate.*

Requirements Workflow - Key Steps

2. Build

Definition 2

A **business model** *is a description of the client's business process, i.e. "how the client operates within the domain".*

3. Determine the project **scope**.
4. The developers make the initial

Requirements Workflow - Key Steps

Definition 3

A **business case** *is a document which answers these questions.*

1. *Is the proposed software cost effective? Will the benefits outweigh the costs? In what timeframe? What are the costs of* **not** *developing the software?*
2. *Can the proposed software be delivered on time? What impacts will be realized if the software is delivered late?*
3. *What risks are involved in developing the software, and how can these risks be mitigated? Similarly to above, what risks are there if we do not build it? There are three major risk categories (next slide).*

Major Risk Categories

1. Technical Risks
2. Bad Requirements
3. Bad Architecture

Analysis Workflow

Goal: Extract the information needed to design the architecture.

Design Workflow

1. Create the design.
2. Answer all questions required to start Implementation.

Implementation Workflow

1. Usually little to no coding is done during the inception phase.
2. Sometimes it will be necessary to build a **proof-of-concept prototype**.

Test Workflow

Goal: Ensure that the requirements artifacts are correct.

Deliverables from the Inception Phase

1. initial version of the domain model
2. initial version of the business model
3. initial version of the requirements artifacts
4. initial version of the analysis artifacts
5. initial version of the architecture
6. initial list of risks
7. initial use cases (from analysis workflow, usually documented in UML)
8. plan for Elaboration phase (we must always plan for the next phase)

Deliverables from the Inception Phase

9. initial version of the business case (**overall aim of Inception phase**). This describes the scope of the software product plus financial details.
    1. If software is to be marketed, then this includes revenue projections, market estimates and initial cost estimates, etc.
    2. If software is to be used in-house, then this includes the initial cost/benefit analysis.

Goals

1. Refine the initial requirements.
2. Refine the architecture.
3. Monitor risks and refine their priorities.
4. Refine the business case.
5. Produce the SPMP.

Deliverables from the Elaboration Phase

1. the completed domain model
2. the completed business model
3. the completed requirements artifacts
4. the completed analysis artifacts
5. updated version of the architecture
6. updated list of risks
7. SPMP
8. the completed business case

Construction Phase

Goal: Produce the first operational-quality version of the software product (the **beta** release).

Deliverables from the Construction Phase

1. initial user manual and other manuals, as appropriate
2. completed version of the architecture
3. updated list of risks
4. SPMP updated
5. if needed, the revised business case

Transition Phase

Goal: Ensure the client's requirements have been met (using, in part, feedback from the users of the beta version).

Deliverables from the Transition Phase

1. final versions of all the artifacts
2. final versions of all manuals / other documentation

Transition Phase

Example: Do the Lecture 06 Example Here.

Transition Phase

In each answer that follows, we state

1. the workflow first, then
2. the phase(s) during which that artifact will likely be delivered.

Transition Phase

1. Business Model: First Draft
    1. Requirements
    2. Inception
2. Business Model: Completed
    1. Requirements
    2. Inception or Elaboration
3. User Requirements: First Draft
    1. Requirements
    2. Inception
4. User Requirements: Completed
    1. Requirements
    2. Elaboration (or Construction)

Transition Phase

5. Inspection Report: User Requirements
    1. Testing
    2. Elaboration (or Construction)
6. SPMP: First Draft
    1. Analysis
    2. Inception (or Elaboration)
7. SPMP: Completed
    1. Analysis
    2. Transition (or Construction)

Transition Phase

8. System Architecture Design: First Draft
   1. Design
   2. Elaboration (or Inception)
9. Code: First Draft
   1. Implementation
   2. Construction (or Elaboration for “low hanging fruit”)
10. Code: Ready for Deployment
    1. Implementation
    2. Transition (or Construction for “low hanging fruit”)

One- Versus Two-Dimensional Life-Cycle Models

Fundamental Question: Can one's "position" in the Life-Cycle be described along only one axis (in which case the model is 1-D), or does one need two axes (in which case the model is 2-D)?

One- Versus Two-Dimensional Life-Cycle Models

Examples (not exhaustive):

1-D	**2-D**
Classical (Waterfall) (Figure 3.2a)	Unified Process (Figure 3.2b)
Code-And-Fix	Iteration and Incrementation
Open Source?	Spiral
Possibly Others...	Possibly Others...

One- Versus Two-Dimensional Life-Cycle Models

Remarks:

1. Two-dimensional models are more complicated, but for all the reasons from Chapter 2, we cannot avoid working with them, especially the Unified Process.
2. The Unified Process is the best model we have so far, but it is sure to be superseded by a superior methodology in the future.

Improving the Software Process

1. Our **fundamental problem** in Software Engineering is our inability to manage the software process effectively (the text cites a US government report from 1987 to justify this statement).
2. The US DoD responded by creating the Software Engineering Institute (SEI) at Carnegie Mellon University.
3. SEI in turn created the **Capability Maturity Model (CMM)** initiative.

Capability Maturity Models

The CMMs are a related group of strategies for improving the software process, irrespective of the choice of Life Cycle model used. The word "maturity" indicates that an organization matures as it improves its processes.

Capability Maturity Models

1. SW-CMM (software) We will focus on this one.
2. P-CMM (HR, “P” for “people”)
3. SE-CMM (systems engineering)
4. IPD-CMM (integrated product development)
5. SA-CMM (software acquisition)

These are gathered up as **CMM Integration (CMMI)**.

Idea of SW-CMM

Use of new software techniques alone will not result in increased productivity and profitability, because our problems stem from how we manage the software process. Improving our management of the software process should drive improvements in productivity and profitability.

Capability Maturity Models

An organization advances incrementally through five levels of maturity.

1. Initial
    1. No sound software engineering practices are in place: everything is done ad-hoc.
    2. Most projects are late and over budget.
    3. Most activities are responses to crises, rather than preplanned tasks.
    4. Many organizations are at the initial level!

Capability Maturity Models

2. Repeatable
    1. Basic software Project Management practices are in place ("repeatable" because planning & management techniques are based on past experience with similar projects).
    2. Some measurements are taken (e.g tracking costs, schedules).
    3. Managers identify problems as they arise and take immediate corrective action to prevent them from becoming crises.

Capability Maturity Models

3. Defined
    1. The process for software production is fully documented (management / technical).
    2. There is continual process improvement.
    3. **Reviews** are used to achieve software quality goals.
    4. **CASE** environments increase quality / productivity further.

Definition 4

CASE *stands for* **Computer Aided/Assisted Software Engineering***.*

We will discuss CASE in more detail in Chapter 5.

Capability Maturity Models

4. Managed
    1. The organization sets quality/productivity goals for each software project.
    2. Both are measured continually and corrective action is taken when there are **unacceptable** deviations. (Statistical methods are used to distinguish a random deviation from a meaningful violation of standards.)
    3. Typical measure: # faults / 1000 lines of code, in some time interval.

Capability Maturity Models

5. Optimizing
    1. The goal is **continual process improvement**.
    2. Statistical quality / process control techniques are used to guide the organization.
    3. **Positive Feedback Loop:** Knowledge gained from each project is used in future projects. Therefore productivity and quality steadily improve.