Software Development Process Models and their Impacts on Requirements Engineering

Organizational Requirements Engineering

Prof. Dr. Armin B. Cremers
Sascha Alda

b-it
Overview

- Phases during Software Development
- Different Software Development Processes
  - Waterfall
  - Spiral Model
  - Rational Unified Process
- Rapid Software Development
  - Agile Software Development and Extreme Programming (XP)
- Impacts on Requirements Engineering
Overview

- Phases during Software Development
- Different Software Development Processes
  - Waterfall
  - Spiral Model
  - Rational Unified Process
- Rapid Software Development
  - Agile Software Development and Extreme Programming (XP)
- Impacts on Requirements Engineering
Software Lifecycle Activities …and their models

- Requirements Elicitation
- Analysis
- System Design
- Object Design
- Implementation
- Testing

Use Case Model
.Application Domain Objects
.Sub-systems
.Solution Domain Objects
.Source Code
.Test Cases
Software Process

- A structured set of activities required to develop a software system
- (Still) Rely on people making decisions and judgements
  - Automate software processes (→ CASE tools) have limited success
- Many processes are proposed. Fundamental activities are:
  - Specification
  - Design and Implementation
  - Validation
  - Evolution
- Processes can be improved by process standardization
  - Improved communication between stakeholders
  - Reduction in training time
Software Lifecycle Activities III

- No ideal process
- Structured plan-based development process
  - Fixed requirements, many project members
  - Critical systems
- Agile development process
  - Rapidly changing requirements
  - Business systems
- Other factors for selecting a process
  - Kind of Software System to be developed
    - completely new vs. re-engineering of existing
    - off-the-shelf vs. customized
  - Team size, project size, project time
  - Team members
    - Experiences, Incentives, Attitudes
  - Budget
Software Lifecycle Activities VII: Different kinds of Development

- Greenfield Engineering
  - Development starts from scratch, no prior system exists, the requirements are extracted from the end users and the client
  - Triggered by user needs
  - Example: Develop a game from scratch: German Toll System (2003)

- Re-Engineering
  - Re-design and/or re-implementation of an existing system using newer technology
  - Triggered by technology enabler
  - Example: New Operating System like Windows Vista

- Interface Engineering
  - Provide the services of an existing system in a new environment
  - Triggered by technology enabler or new market needs
  - Example: SMS-based games, information systems (beginning of 2000)

Lead to different software development processes
Overview

- Phases during Software Development
- Different Software Development Processes
  - Waterfall
    - Spiral Model
    - Rational Unified Process
- Rapid Software Development
  - Agile Software Development and Extreme Programming (XP)
- Impacts on Requirements Engineering
Waterfall Model (Royce, 1970)

- Activity-centered process that prescribes the sequential executions of life cycle activities (Result: documents)
- All requirements are completed before the system design activity starts
  - The following phase should not start before the previous one has finished
- The final product is eventually produced (no intermediary results)
- Goal: Never turn back once an activity is completed
  - Refinement of first model: only few iterations are allowed → costly
  - Existing problems are left for later resolution, programmed around
Waterfall Model II

Requirements Development

• Problem Understanding
• Defining the problem (what is required)
• Representing the problem (formally stating the problem def.)
• Decomposing the problem
• Defining the constraints of the problem (what can't be done)

System and Software Design

• Transform the problem statement into a solution statement
• Decompose problem and reconstruct it as a solution
• Determine the transformation was correct
• Decompose the solution into implementable pieces

Implementation

• Physically construct software
• Detailed design of code
• Write code

Integration, Evaluation, Testing

Test Activities
• Determine correctness of software
• Determine if software meets its intended function

Operation, Maintenance, Evolution

OME Activities
• Put software into use
• Detect program and design errors
Waterfall Model III

- **Advantages:**
  - Fits with other engineering processes
  - Documentation is produced at each phase
  - Simplistic view of software development (good for beginners…)

- **Problems:**
  - Inflexible partitioning of the project into distinct self-contained stages
  - No flexible iteration are intended, every phase is worked out once
    - The delivery of the overall product may retarded
  - Process model is only appropriate when the requirements are well-understood
  - Requirements have to be fixed very early before design begins
  - Problems in designing software for social domains
    - missing iteration hinders user involvement
    - missing flexibility does not allow reacting on changes in requirements
Spiral Model I (Boehm 1988)

- Activity-centered process model as an answer to Waterfall model
- Accommodate frequent changes during software development
- Based on the same activities as the Waterfall model. Adds several (sub-) activities (cycles)
  - Objective setting
  - Risk assessment and reduction
  - Development and validation
  - Planning for next phases
- Main Issues
  - Process is arranged as a spiral
  - Each loop represents a phase of the software process (e.g. concept of operation, requirements, design, code, unit tests)
  - Risks are explicitly assessed and resolved throughout the process
Spiral Model II

1. Determine objectives, alternatives, constraints
2. Evaluate alternatives, identify and resolve risks
3. Plan next phases
4. Develop & verify next level product
To be more detailed each loop includes minimally the following steps:

- Determine product, process objectives
- Specify constraints e.g. functionality, performance
- Evaluate alternative product, process solutions e.g. design alternatives, buy components
- Determine project risks
- Resolve risk: (e.g. prototyping, simulation or bench marking)
- Develop and verify next-level product
- Plan next phase

Final review

- at each crossing of the X-axis a review has to be made.
- ensure that everybody in the project (users, developers, customers) are committed to the developed products and the planning for the next cycle
**Strengths**

- Generic approach which can incorporate other software process models as special cases
- Avoids the difficulties of existing software models through risk-driven approach
  - Tries to eliminate errors in early phases
  - Explicit coverage of risk evaluation and minimization
  - Provides mechanisms for software quality assurance
  - Works good for complex, dynamic, innovative project
  - Reevaluation after each phase allows changes in user perceptsives, technology advances or financial perspectives
Weaknesses

- The Spiral Model isn't really a "cookbook" approach.
  - Management has to decide how to structure the project into phases.
  - Flexibility of this model is high and sometimes more as it is convenient.
  - Lack of explicit process guidance in determining objectives, constraints, and alternatives.

- Risk assessment expertise
  - A lot of experience in software projects is necessary to accomplish that task successfully.

- Problems in Social domains
  - Process focuses on the product, not its context.
  - No concepts for introducing systems.
  - "User-designer communication in social domains (e.g. interactive systems)?"
    - Scenario-Based Design?!
    - Use Cases?!
    - Participation with real users?!
ISO 13407 provides guidance on achieving quality in use by incorporating user-centered design activities throughout the development life cycle of interactive computer-based systems.

Description of user-centered design as a multi-disciplinary activity:
- Incorporates human factors
- Ergonomics knowledge
- Improving human working conditions

There are four user-centered design activities that need to start at the earliest stages of a project:
- understand and specify the context of use
- specify the user and organizational requirements
- produce design solutions
- evaluate designs against requirements.
Human-Centered Design Process (ISO 13407) Design of Interactive Systems

Plan the human centered process (determine exigency)

Meet requirements

Context of use (user characteristics, task, environment)

analyze

Result of evaluation

evaluate

Design solutions (Prototype, Simulations)

develop

interpret

User and organizational requirements
The Usability Maturity Model in ISO TR 18529 consists of seven sets of base practices to describe what has to be done in order to represent and include the users of a system during the lifecycle.

Essential practice (no.7): Introduce and operate the system:
- 7.1 Management of change
- 7.2 Determine impact on organization and stakeholders
- 7.3 Customization and local design
- 7.4 Deliver user training
- 7.5 Support users in planned activities
- 7.6 Ensure conformance to workplace ergonomic legislation
Rational Unified Process Overview

- Life Cycle model proposed by Booch, Jacobson, and Rumbaugh (“The three Amigos”) derived from the work on UML
- Rational Unified Process (RUP) uses Unified Modelling Language (UML) as core notation
- Described from 3 perspectives
  - A dynamic perspective that shows phases over time;
  - A static perspective that shows process activities;
  - A practice perspective that suggests good practice.
- Unified Process is distinguished by being
  - Use-case driven
  - Architecture-centric
  - Iterative and incremental
RUP proposes a phase model that identifies four discrete phases in the software process

- Inception
  - Establish the business case for the system
  - Decide to cancel or continue the project

- Elaboration
  - Develop an understanding of the problem domain and the system architecture.

- Construction
  - System design, programming and testing.

- Transition
  - Deploy the system in its operating environment.
Each phase may be enacted in an iterative way with the results developed as increments.

The whole set of phases may also be enacted incrementally:
- Whole set = cycle (later on..)

An iteration represents a set of activities for which there is a milestone (“well-defined intermediate event”)

The scope and results of the iteration are captured via discrete work products called *artefacts*. 
To make the development of complex systems manageable, the Unified Process organizes work products produced during the development into *artifacts sets*.

**Artifact set:**
- Related work products (“artifacts”) that are persistent and in a uniform representation format (natural language, Java, UML,…).
- Every artifact in the set is developed and reviewed as a single entity.

**The Unified Process distinguishes the following five sets**
- Management set
- Requirements set
- Design set
- Implementation set
- Deployment set
Artifact sets

- Each artifact set has a different intention and uses different notations to capture the relevant artifacts.
  - Management Set:
    - Notation: Ad hoc text, graphics, textual use cases
    - Goal: Capture plans, processes, objectives, acceptance criteria.
  - Requirements set:
    - Notation: Structured text, models in UML (Use Case, Class, Sequence)
    - Goal: Capture the problem in the language of the problem domain
  - Design set:
    - Notation: Structured text, models in UML
    - Goal: Capture the engineering blueprints
  - Implementation set:
    - Notation: Programming language
    - Goal: Capture the building blocks of the solution domain in human-readable format.
  - Deployment set:
    - Form: Machine language
    - Goal: Capture the solution in machine-readable format.
Effective Deployment of 6 best practices

- Develop software iteratively
  - Plan increments of the system based on customer priorities and develop the highest priority system features early in the process

- Manage requirements
  - Document customers requirements and keep track to changes
  - Analyse the impact of changes on the system before accepting these

- Use component-based architectures
  - Structure the system architecture into components → reuse

- Visually model software
  - Use graphical UML models to present static and dynamic views of the system
Effective Deployment of 6 best practices

- Verify software quality
  - Ensure that the software meets requirements
- Control changes to software
  - Manage changes to the software using a change management system and configuration management procedures and tools
Unified Process: Use Case Driven

- An use case represents a class of functionality provided by the system as a sequence of interaction with some actors.
- A piece of functionality that gives a user a result of value
- *Use Case driven = Use cases* are used as the primary artifacts for deriving the architectural abstractions
  - use cases are specified, designed and are the source for test cases
  - they drive system architecture
  - both mature as the development lifecycle continues
Unified Process: Architecture-Centric

- Software architecture shows different views of the system being built and embodies the static & dynamic aspects of the system (design framework)
- Also influenced by the computer architecture, operating system, DBMS, network protocols etc.
- The form must allow the system to evolve from initial development through future requirements (i.e. the design needs to be flexible)
- Key use cases influence the design of the architecture which may in turn influence development of other use cases
Systems development is frequently a large undertaking - may be divided into several “mini-projects” each of which is an iteration resulting in incremental development of the system.

Incremental Development:
- An approach to software development where the software is delivered and deployed in increments
- First increment satisfies the

Iterative Development
- An approach to software development where the processes of specification, design, programming and testing are interleaved

Concepts of use case driven, architecture centric and iterative & incremental are of equal importance
Unified Process repeats over a series of cycles each concluding with a product release (increment) to the users.

Cycles have no specific name but characterize the stage of maturity of the software system (like “birth” → “death”).

Each cycle has four phases (each with a number of iterations):
- Inception, Elaboration, Construction & Transition
- Phases have goals (→ result in artefacts or models)

Delivered products will be described by related models each with “trace” dependencies which chain backwards and forwards:
- Use Case Model
- Analysis Model
- Design Model
- Deployment Model
- Implementation Model
- Test Model
Cycles concluded with a release

Birth

---

Death

Time

Inception | Elaboration | Construction | Transition

i1 | i2 | --- | --- | --- | --- | in-1 | in

i = iteration
Releases

A cycle with its phases and its iterations
Workflow Activities

- All activities that are place during a phase are called workflows.
- During a single phase several workflows may run in parallel.
- Workflows are the static part of process and are not associated with a single phase.
- Phase: have concrete goals (artifact).
- Workflow: Technical activities to achieve the goals of each phase.

- Development-oriented Workflows:
- Cross-functional activities:
  - Management, Environment, Assessment and Deployment.
Workflow Activities

- **Management workflow**
  - Planning the project (Problem statement, SPMP, SCMP, Test plan)

- **Environment workflow**
  - Automation of process and maintenance environment. Setup of infrastructure (Communication, Configuration management, ...).

- **Requirements workflow**
  - Analysis of application domain and creation of requirements artifacts (analysis model).

- **Design workflow**
  - Creation of solution and design artifacts (system design model, object design model).

- **Implementation workflow**
  - Implementation of solution, source code testing, maintenance of implementation and deployment artifacts (source code).

- **Assessment workflow**
  - Assess process and products (reviews, walkthroughs, inspections, testing...)

- **Deployment workflow**
  - Transition the software system to the end user
Core Workflows & Phases

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management Workflow</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Environment Workflow</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirements Workflow</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design Workflow</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementation Workflow</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assessment Workflow</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deployment Workflow</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The RUP is not a suitable process for all types of development but it does represent a new generation of generic processes.

Most important innovation:
- Combination of many views
- Deployment of software is part of the process (almost ignored in other process models)

Based on standards:
- Object-oriented Modeling
- Unified Modeling Language
Overview

- Phases during Software Development
- Different Software Development Processes
  - Waterfall
  - Spiral Model
  - Rational Unified Process
- Rapid Software Development
  Agile Software Development and Extreme Programming (XP)
- Impacts on Requirements Engineering
Rapid Development

Problem

- Because of rapidly changing business environments, businesses have to respond to new opportunities, requirements and competition.
- Because of the changing environment, it is often impossible to arrive at a stable, consistent set of system requirements.

Facts

- Rapid development and delivery is not often the most critical requirement for software systems → Reduce time-to-market
- Businesses may be willing to accept lower quality software if rapid delivery of essential functionality is possible.
Rapid Development

Solution:
- **Rapid software development processes** are designed to produce useful software quickly.
- The **processes** of specification, design and implementation are **concurrent**. There is no detailed specification and design documentation is minimised.
- The system is developed in a **series of increments**. End users evaluate each increment and make proposals for later increments.
- System user interfaces are usually developed using an interactive development system (CASE tools).
- **Accelerated delivery of customer services**. Each increment delivers the highest priority functionality to the customer.
- **User engagement with the system**. Users have to be involved in the development which means the system is more likely to meet their requirements and the users are more committed to the system.
The objective of incremental development is to deliver a working system to end-users. The development starts with those requirements which are best understood.

The objective of throw-away prototyping is to validate or derive the system requirements. The prototyping process starts with those requirements which are poorly understood.
Dissatisfaction with the overheads involved in design methods led to the creation of agile methods. Core issues:

- Focus on the code rather than the design;
- Are based on an iterative approach to software development;
- Are intended to deliver working software quickly and evolve this quickly to meet changing requirements.

Agile methods are probably best suited to small/medium-sized business systems or PC products.
Agile Software Development: The Agile Software Development Manifesto

- **Permanent Customer Involvement**
  - Role: Provide and prioritize new system requirements

- **Incremental delivery**
  - Software is developed in increments

- **People not process**
  - Skills of team (members) should be appreciated
  - Should be left to work with their own methods, tools etc.

- **Embrace change**
  - Expect system requirements to change, so design the system to accommodate these

- **Maintain simplicity**
  - Focus on simplicity in both software and development process
  - Work together in eliminate complexity

→ http://www.agilemanifesto.org
Problems with agile methods

- It can be difficult to keep the interest of customers who are involved in the process.
- Team members may be unsuited to the intense involvement that characterizes agile methods.
  - In particular shy and reserved people
- Prioritising requirements can be difficult when there are multiple stakeholders.
- Maintaining simplicity requires extra work.
Extreme Programming (Beck, 2000)

- Perhaps the best-known and most widely used agile method.
- Extreme Programming (XP) takes an ‘extreme’ approach to iterative development.
  - New versions may be built several times per day;
  - Increments are delivered to customers every 2 weeks;
  - All tests must be run for every build and the build is only accepted if tests run successfully.

- Programming in pairs
- Continuously re-prioritizing of requirements (client, customer, users)
- Client is part of development team and in charge
In XP, user requirements are expressed as scenarios or user stories.

These are written on cards and the development team break them down into implementation tasks. These tasks are the basis of schedule and cost estimates.

The customer chooses the stories for inclusion in the next release based on their priorities and the schedule estimates.
How does XP work? (1)

- The planning game
  - long-term plans only diffuse
  - Details in short-terms (some days)
  - customer is involved in every development cycle (some weeks)
  - tasks are handed out according to skills

- Short development cycles
  - enforce decomposition into small tasks
  - minimize risk
  - better integration of customer

- Simple Design
  - Design takes into account only short-term goals
  - complexity is low

- Sustainable amount of time
  - Large amounts of overtime are not considered acceptable (reduce quality)
How does XP work? (2)

- **Tests**
  - Unit tests: For implementation details, are developed by the developers themselves, are designed before development
  - “Test-first development”
  - enhances trust in system

- **Refactoring**
  - allow for continuous changes in design
  - Keep code simple and maintainable

- **Programming in pairs**
  - always pair-programming
  - changing pairs (daily)
  - distributes knowledge within the team
How does XP work? (3)

- Collective ownership: Source code belongs to the team
  - everybody can make changes (consultation)
  - no fixed responsibilities
  - pairs work on all areas, no islands of expertise

- Continuous integration
  - integration several times a day
  - system is error-free every evening

- Standards (Coding)
  - is commonly accepted

- Customer agent is part of the team
  - available (in case of questions)
  - works out feature tests
Impacts in Requirements Engineering

- Requirements
  - Change over time (considered in process)
  - XP means “design with/for change”
  - Simplicity encourages fast changes

- User participation
  - More cycles
    - More user involvement
  - Explicit testing during development
    - User is involved by basic XP principles
    - (Feature) Tests are developed with/by users
Conclusions (XP)

- Advantages
  - Flexible
  - Quick
  - Customer-centered
  - Team-oriented

- Disadvantages
  - Little documentation (only tests and code)
  - Scalability problems (only small and medium projects)
  - Depends strongly on team members
Conclusions

- Requirements phases have been regarded in almost all process models

- Traditional Development processes:
  - Waterfall model, RUP
  - Requirement phases have become more and more interleaved with other phases
  - Models like RUP are good for complex projects with relatively fixed requirements

- Agile Processes
  - Higher flexibility (for development of projects with rapidly changing requirements)
  - Customer participation during the whole process (smaller cycles lead to steady involvement)