Software Engineering
Lecture 02: Processes

Peter Thiemann
University of Freiburg, Germany

SS 2013
Terms

**Software**
- Organized collection of computer data and instructions

**Component**
- Solves isolated task
- Developed by a single person

**SW System**
- Multiple components
- Developed by a team
Programing in the Small

- Development of a system comprised of a small number of “mind-sized” components
- Often clear requirements
- Sometimes algorithmic aspects
- Procedure for a single component:
  - Procedural decomposition, top-down
  - “stepwise refinement” (N. Wirth),
Programming in the Large

- Development of a **software system** comprised of many components
- **requirements** at first **fuzzy**
- Size or complexity dictate ...
  - decomposition in a large number of components
  - development in a team
  - size determines duration, but beware of Brook’s law!
Brook’s law: Adding manpower to a late SW project makes it later
Issues Arising with Programming in the Large

- **Requirements** need to be investigated
  - Communication problem customer ↔ developer
  - Understanding the problem
- **Design** of the system is significant task
  - **Decomposition in components** (interfaces, contracts)
  - **Information hiding** (D.L. Parnas)
  - Design for maintenance
    - Long life span
    - High probability of changes (aging)
  - Promising approach: **object-oriented analysis and design**
- **Construction** of components: programming in the small
- **Testing** required on many levels
Conclusion

- Programming in the large is a structured approach to all activities in the development of a software system
- Unfortunately, ...
Conclusion

- Programming in the large is a structured approach to all activities in the development of a software system
- Unfortunately, ...  
  - there are many overall approaches (process models)
  - there are many techniques with similar goals
Process Models

- **Process Model**: structured network of *activities* and *artifacts*
- An activity transforms a set of artifacts into new artifacts
Phases provide structure of process model

Description of a phase

- goals
- activities
- roles
- required/new artifacts
- patterns, guidelines, and conventions
Desiderata for Process Models

- The fewer phases, artifacts, roles, the better
- Artifacts should cover standard case
- Tool support
- Quality assurance for each artifact
- Traceability
The Classic: Waterfall Model

- Early error correction is cheaper (e.g. after analysis phase 100 times cheaper than after deployment)
- Hence, after every phase: check of previous phases
- Potentially return to previous phase
- Phases may overlap
Requirements Analysis

tractability

cost analysis

result:

decision on continuation of project

documents: (artifacts)

- Requirement specification (Lastenheft)
- Cost estimation
- Project plan
Processes

Definition / Specification

starting point:

vague, incomplete, inconsistent requirements

result:

complete, consistent, unequivocal, accomplishable requirements

documents:

- **System specification** (Pflichtenheft)
- **Product model** (e.g. OOA)
- **GUI model**
- **User manual**
Definition / Specification (cont’d)

- Only **external behavior** of system
- **Analysis of requirements**
  - functional / non-functional requirements
  - prioritization
- **Main outcome: system specification**
  - fixes the scope of the product
  - serves as basis for **contract** between customer and contractor
  - basis for **final acceptance**
  - functionality
  - user interface
  - interfaces to other systems
  - performance (response time, space usage)
  - required hard and software
  - guidelines for documentation
  - time scheduling
  - quality
Design

starting point: system specification / product model
  ▶ Decomposition in components / subsystems
  ▶ Logical interfaces of each component
  ▶ Choice of technologies

result: Software architecture (with specification of components)
Implementation and Testing

starting point: Software architecture

- Coding of component specifications
- Compilation to machine language
- Unit testing up to component level

result: implemented components and testing protocols
Integration, system test, and deployment

- **Integration**
  - stepwise addition of single components
  - tested with data fixed in advance
    (functional requirements only)

- **System test**
  - entire system (incl. hardware)
  - non-functional requirements (performance, GUI)

- **Deployment**
  - transfer of software system to its working environment

*result*: deployed product, protocol of final acceptance
Processes

Maintenance

- Supervision
- Bug fixes
- Changes due to changes in requirements (incl. extensions)

result: maintained product
Concrete Process Models

1. V-Model
2. Prototyping model
3. Phased models (evolutionary, incremental, spiral)
4. Unified Software Process
5. Agile development techniques
V-Model “Entwicklungsstandard für Systeme des Bundes”

- System requirements
- Software requirements
- Preliminary design
- Detailed design
- Code and debug
- Unit test
- Component test
- Software integration
- Acceptance test
- System integration

Processes:

- Analyze and Design
- Test and Integrate

Level of abstraction:

Time:

Peter Thiemann (Univ. Freiburg) Software Engineering

SWT 20 / 41
V-Model

- Builds on waterfall model
- Emphasizes validation connections between late phases and early phases
- Objectives
  - risk minimization
  - quality assurance
  - cost reduction
  - communication between stakeholders
- Current instance: V-Model XT
Prototyping Model

Lifecycle

```
requirements | design prototype | build prototype | test prototype |
```

```
document requirements | design | code | test | integrate
```
Prototyping - Overview

Advantages:

▶ understanding the requirements for the user interface
▶ improves understanding between developer and client
▶ early testing of feasibility, usefulness, performance, etc.

Problems:

▶ customers treat the prototype as the product
▶ a prototype is not a specification
▶ significant user involvement
Phased Models

Evolutionary Development
1. model core requirements
2. design and implement
3. deploy
4. feedback from customer
5. revise/extend requirements
6. revise/extend design
7. revise/extend implementation
8. iterate from 3 until all requirements met

Incremental Development
1. model all requirements
2. design and implement only core requirements
3. deploy
4. feedback from customer
5. revise requirements
6. design further requirements
7. implement further requirements
8. iterate from 3 until all requirements met
Incremental Development

(each iteration adds more functionality)
Evolutionary Development

(each iteration incorporates new requirements)

version 1

version 2

version 3
Spiral Model (Barry Boehm 1988)

Determine Goals, Alternatives, Constraints

Evaluate Alternatives and Risks

Plan

Develop and Test
Comments on Phased Models

- Incremental development
  - avoids 'big bang' implementation
  - but assumes all requirements known up-front

- Evolutionary development
  - allows for lessons from each version to be incorporated into the next
  - but: hard to plan for versions beyond the first; lessons may be learned too late

- Spiral model
  - primarily targeted at very large projects
  - iterative model that incorporates prototyping and risk analysis
  - but: cannot cope with unforeseen changes
  - not clear how to analyze risk
The Unified Software Process

Use-Case Driven

- Which user-visible processes are implemented by the system?
- Analysis, design, implementation, and testing driven by use-cases

Architecture centric

- Architecture developed in parallel to use cases (mutual dependency)

Iterative and Incremental

- eliminate risks first
- checkpoint after each iteration
- on failure of an iteration step, only current extension needs to be reconsidered
- small steps speed up project
- easy stepwise identification of the requirements
Structure of the Unified Software Process

- sequence of cycles
- after each cycle: product release with code, manuals, UML models, and test cases

Cycle consists of 4 phases:
  - Inception, Elaboration, Construction, Transition
- each phase consists of iterations
Processes

Cycle

<table>
<thead>
<tr>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration #1</td>
<td>Iteration #2</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

new product versions
Main-Workflows and Phases

- Each phase ends with a milestone.
- Each phase processes all workflows (with varying intensity).

![Diagram showing the main workflows and phases in software engineering.](attachment:diagram.png)
Inception Phase

- functionality of system from users’ perspective
  most important use cases (stakeholder needs)
- preliminary sketch of suitable architecture
- project plan and cost
- identify most important risks (with priorities)
- plan elaboration phase
- **GOAL:** rough vision of the product
Elaboration Phase

- specify (most) use cases in detail
- design architecture
- implement most important use cases
- result: initial architecture
- plan activities and resources for remaining project
- use cases and architecture stable?
- risk management?
- **GOAL:** prototype (proof-of-concept for architecture)
Construction Phase

- implement system
- high resource needs
- small architectural changes
- **GOAL:** system ready for customer (small errors acceptable)
Transition Phase

- deliver beta-version to customer
- address problems (immediately or in next release)
- train customer
Agile Development Techniques

Extreme Programming (XP, Kent Beck 1999)

- frequent releases
- short development cycles
- pair programming
- unit testing with tests developed before the code
- features specified by tests
- implement features when needed
- clear progress marks
- don’t spend much time on design
- stakeholder involvement
Agile Development Techniques
Scrum (Hirotaka Takeuchi and Ikujiro Nonaka 1986)

- Flexible approach to development; incremental process
- Adaptability to changing requirements

Core roles  Product owner, Scrum master, Team
Ancillary roles  Stakeholders, Managers

Sprint  2-4 weeks of intense development; goal: working increment that implements the sprint backlog; sprint backlog frozen during a sprint; self organization; burn down chart

Sprint Backlog requirements chosen for a sprint
Product Backlog as yet unimplemented requirements
The Pig and the Chicken

Roles in Scrum

Last Word

On Agile Development


We're all set to start using a jail in our development process.

I never said that Scrum was "A JAIL." I said it was "AGILE."

You're free to go. Apparently, Scrum isn't as great as I originally thought.

By Clark & Vizdos

© 2006 implementingscrum.com

Summary

- Software has unique problems with far-reaching consequences
- Creating software systems requires structured process models
- Classic process phases: waterfall model
- Commonly used process models: V-model, prototyping, evolutionary, incremental, spiral, unified SW process, agile development