LTAT.05.003
Software Engineering

Lecture 01.2:
Introduction to Software Engineering

Dietmar Pfahl
email: dietmar.pfahl@ut.ee

Fall 2018
Schedule of Lectures (Tentative)

**Week 01: Introduction to SE**
- Week 02: Requirements Engineering I
- Week 03: Requirements Engineering II
- Week 04: Analysis

**Week 05: Development Infrastructure I**
- Week 06: Development Infrastructure II
- Week 07: Architecture and Design
- Week 08: Verification and Validation I
- Week 09: Verification and Validation II

**Week 10: Continuous Development and Integration**
- Week 11: Refactoring (and TDD)
- Week 12: Agile/Lean Methods
- Week 13: no lecture
- Week 14: Software Craftsmanship
- Week 15: Course wrap-up, review and exam preparation
Software Engineering

What?
Why?
Software Engineering

Who has practiced SE?
Who knows what SE is?
Software Development – Three Ps

- Software Development

[Diagram showing a cycle with three nodes labeled P? and the text “Project or Iteration”]
Software Development – Three Ps

- Software Development
  - Project or Iteration
  - Products
  - People
  - Processes

Project or Iteration
Software Development – Three Ps

- Software Development

Project or Iteration

Products

People

Processes
Products in Software Development

**Code:**
- Production code:
  - Source code
  - Object code
- Non-production code:
  - Test code

**Non-Code:**
- Requirements
- Specifications
- Architecture/Design docs
- Issue reports
- User manuals
- Plans of all kinds
- ...

**Models**

**Products**

**Properties of Software:**
- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

**Types of Software:**
- Embedded/real-time
- Information System
- Web application
- System software
- ...

**Types of Software:**
- Embedded/real-time
- Information System
- Web application
- System software
- …
Software in a Car

State-of-practice:
- 40-100 ECU
- 5-10 Mio Lines of Code
- > 100 MB Software
- > 3 Bus Systems

ECU = Electronic Control Unit

Source:
Properties of Software

The software should deliver the required functionality and performance to the user and should be maintainable, dependable and acceptable.

Maintainability
Software must evolve to meet changing needs;

Dependability (Reliability)
Software must be trustworthy;

Efficiency
Software should not make wasteful use of system resources;

Usability
Software must be accepted by the users for which it was designed. This means it must be understandable, usable and compatible with other systems.
SW Product Modeling

UML = Unified Modeling Language
Software Development – Three Ps

- Software Development
- Products
- Processes
- Project or Iteration

People
People in Software Development

Roles:
- Project Manager
- Product Manager
- Architect/Analyst
- Programmer
- Tester
- ...

Skills:
- Must match roles

Training:
- Must fill skill-gaps

Education:
- Curricula (ACM/IEEE)

Teams:
- Team building
- Geographically distributed (international/global)
- Mechanisms for collaboration/cooperation
- Motivation, Personality, Values, Culture

User models
Software Development – Three Ps

- **Software Development**
- **Products**
- **People**
- **Processes**

Project or Iteration
Software Development Process

Coding

Deploying

Processes

LTAT.05.003 / Lecture 01.2 / © Dietmar Pfahl 2018
Software Development Process

- Find Requirements
- Analysis / Designing
- Coding
- Testing
- Deploying
Software Development Process

(Royce, 1970)
Processes in SW Development

Process (Model) Elements:
- Activity
- Input/Output Product(s)
- Roles
- Methods/Techniques/Tools

Process Taxonomy:
- Non-engineering processes
  - Business processes
  - Social processes
- Engineering processes
  - Product-engineering proc.
    - Technical prod.-eng. proc.
    - Managerial prod.-eng. proc.
  - Process-engineering proc.

Process Modeling:
- Descriptive PMs
- Prescriptive PMs
  - Standards
  - Families

Process Types:
- Heavy-weight (rich)
- Light-weight
  - Lean
  - Agile
  - Kanban

Processes
Agile Process
Agile Process

Scrum

eXtreme Programming (XP)
Scrum Elements – Process, Artifacts, Roles

http://www.scrumforteamsystem.com/processguidance/v1/Scrum/Scrum.html
13 XP Practices

**Project Cycle**
- Planning Game (Poker)
- Small Releases
- Whole Team
- Customer Tests

**Development Cycle**
- Simple Design
- Pair Programming
- TDD (Unit Test)
- Refactoring

**Supporting Practices**
- Coding Standard
- Sustainable Pace (40-hour week)
- Metaphor (Common Understanding)
- Continuous Integration
- Collective Ownership

Processes
Comparison of Basic Process Types

- Waterfall
- Incremental, e.g. RUP
- Agile - XP

RUP = Rational Unified Process
XP = Extreme Programming
Survey – Software Development Processes and Practices

• 26 Countries

• 500+ Responses (from Industry!)

• URL: https://helenastudy.wordpress.com

RUP = Rational Unified Process
XP = Extreme Programming
### Process Frameworks Used in 26 Countries

#### Frameworks/Methods
Which of the following frameworks and methods do you use?

<table>
<thead>
<tr>
<th>Framework/Method</th>
<th>Often Used</th>
<th>Always Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scrum</td>
<td>11%</td>
<td>8%</td>
</tr>
<tr>
<td>Iterative Development</td>
<td>12%</td>
<td>7%</td>
</tr>
<tr>
<td>Kanban</td>
<td>14%</td>
<td>16%</td>
</tr>
<tr>
<td>DevOps (CI/CD)</td>
<td>23%</td>
<td>10%</td>
</tr>
<tr>
<td>Classic Waterfall Process</td>
<td>10%</td>
<td>12%</td>
</tr>
<tr>
<td>eXtreme Programming (XP)</td>
<td>11%</td>
<td>20%</td>
</tr>
<tr>
<td>Lean Software Development</td>
<td>18%</td>
<td>23%</td>
</tr>
<tr>
<td>Domain-Driven Design</td>
<td>14%</td>
<td>18%</td>
</tr>
<tr>
<td>ScrumBan</td>
<td>17%</td>
<td>11%</td>
</tr>
<tr>
<td>Feature Driven Development (FDD)</td>
<td>11%</td>
<td>12%</td>
</tr>
<tr>
<td>V-shaped Process (V-Model)</td>
<td>13%</td>
<td>12%</td>
</tr>
<tr>
<td>Phase / Stage-gate model</td>
<td>9%</td>
<td>20%</td>
</tr>
<tr>
<td>Model-Driven Architecture (MDA)</td>
<td>9%</td>
<td>20%</td>
</tr>
<tr>
<td>Scaled Agile Framework (SAFe)</td>
<td>10%</td>
<td>15%</td>
</tr>
<tr>
<td>Team Software Process</td>
<td>10%</td>
<td>15%</td>
</tr>
<tr>
<td>Personal Software Process</td>
<td>10%</td>
<td>15%</td>
</tr>
<tr>
<td>Nexus</td>
<td>10%</td>
<td>15%</td>
</tr>
<tr>
<td>Large-Scale Scrum (LESS)</td>
<td>10%</td>
<td>15%</td>
</tr>
<tr>
<td>SSADM</td>
<td>10%</td>
<td>15%</td>
</tr>
<tr>
<td>Spiral Model</td>
<td>10%</td>
<td>15%</td>
</tr>
<tr>
<td>Dynamic Systems Development Method</td>
<td>10%</td>
<td>15%</td>
</tr>
<tr>
<td>Crystal Family</td>
<td>10%</td>
<td>15%</td>
</tr>
<tr>
<td>PRINCE2</td>
<td>10%</td>
<td>15%</td>
</tr>
<tr>
<td>Rational Unified Process</td>
<td>10%</td>
<td>15%</td>
</tr>
</tbody>
</table>

- **Do not know the framework**: 24%
- **We never use it**: 10%
- **We rarely use it**: 30%
- **We sometimes use it**: 11%
- **We often use it**: 28%
- **We always use it**: 20%
Process Frameworks Used in Estonia

1. Scrum
2. DevOps (CI/CD)
3. Iterative Development
4. Kanban
5. XP
# Process Frameworks Used in Sweden

## Frameworks/Methods
Which of the following frameworks and methods do you use?

<table>
<thead>
<tr>
<th>Framework/Method</th>
<th>Always Used</th>
<th>Often Used</th>
<th>Rarely Used</th>
<th>Never Use</th>
<th>Do not know if use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scrum</td>
<td>15%</td>
<td>38%</td>
<td>31%</td>
<td>8%</td>
<td>15%</td>
</tr>
<tr>
<td>Iterative</td>
<td>8%</td>
<td>23%</td>
<td>15%</td>
<td>38%</td>
<td>31%</td>
</tr>
<tr>
<td>Development</td>
<td>3%</td>
<td>3%</td>
<td>3%</td>
<td>8%</td>
<td>23%</td>
</tr>
<tr>
<td>Classic Waterfall</td>
<td>23%</td>
<td>23%</td>
<td>23%</td>
<td>23%</td>
<td>15%</td>
</tr>
<tr>
<td>Kanban</td>
<td>23%</td>
<td>23%</td>
<td>23%</td>
<td>23%</td>
<td>15%</td>
</tr>
<tr>
<td>DevOps</td>
<td>23%</td>
<td>23%</td>
<td>23%</td>
<td>23%</td>
<td>15%</td>
</tr>
<tr>
<td>ScrumBan</td>
<td>31%</td>
<td>38%</td>
<td>31%</td>
<td>8%</td>
<td>15%</td>
</tr>
<tr>
<td>Feature Driven Development</td>
<td>31%</td>
<td>38%</td>
<td>31%</td>
<td>8%</td>
<td>15%</td>
</tr>
<tr>
<td>V-shaped Process</td>
<td>31%</td>
<td>38%</td>
<td>31%</td>
<td>8%</td>
<td>15%</td>
</tr>
<tr>
<td>Phased Development</td>
<td>31%</td>
<td>38%</td>
<td>31%</td>
<td>8%</td>
<td>15%</td>
</tr>
<tr>
<td>Model-Driven Architecture</td>
<td>31%</td>
<td>38%</td>
<td>31%</td>
<td>8%</td>
<td>15%</td>
</tr>
<tr>
<td>Scaled Agile Framework</td>
<td>31%</td>
<td>38%</td>
<td>31%</td>
<td>8%</td>
<td>15%</td>
</tr>
<tr>
<td>Team Software Process</td>
<td>31%</td>
<td>38%</td>
<td>31%</td>
<td>8%</td>
<td>15%</td>
</tr>
<tr>
<td>Personal Software Process</td>
<td>31%</td>
<td>38%</td>
<td>31%</td>
<td>8%</td>
<td>15%</td>
</tr>
<tr>
<td>Nexus</td>
<td>31%</td>
<td>38%</td>
<td>31%</td>
<td>8%</td>
<td>15%</td>
</tr>
<tr>
<td>Large-Scale Scrum</td>
<td>31%</td>
<td>38%</td>
<td>31%</td>
<td>8%</td>
<td>15%</td>
</tr>
<tr>
<td>SSADM</td>
<td>31%</td>
<td>38%</td>
<td>31%</td>
<td>8%</td>
<td>15%</td>
</tr>
<tr>
<td>Spiral Model</td>
<td>31%</td>
<td>38%</td>
<td>31%</td>
<td>8%</td>
<td>15%</td>
</tr>
<tr>
<td>Dynamic Systems Development Method</td>
<td>31%</td>
<td>38%</td>
<td>31%</td>
<td>8%</td>
<td>15%</td>
</tr>
<tr>
<td>Crystal Family</td>
<td>31%</td>
<td>38%</td>
<td>31%</td>
<td>8%</td>
<td>15%</td>
</tr>
<tr>
<td>PRINCE2</td>
<td>31%</td>
<td>38%</td>
<td>31%</td>
<td>8%</td>
<td>15%</td>
</tr>
<tr>
<td>Rational Unified Process</td>
<td>31%</td>
<td>38%</td>
<td>31%</td>
<td>8%</td>
<td>15%</td>
</tr>
</tbody>
</table>
### Dev. Practives Used in 26 Countries

<table>
<thead>
<tr>
<th>Practice</th>
<th>Usage Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding standards</td>
<td>69%</td>
</tr>
<tr>
<td>Release Planning</td>
<td>65%</td>
</tr>
<tr>
<td>Code Review</td>
<td>60%</td>
</tr>
<tr>
<td>Backlog Mnmt.</td>
<td>60%</td>
</tr>
<tr>
<td>Continuous Integration</td>
<td>58%</td>
</tr>
<tr>
<td>Limit WiP</td>
<td>57%</td>
</tr>
<tr>
<td>Daily Stand-Up</td>
<td>55%</td>
</tr>
<tr>
<td>Iter./Sprint Review</td>
<td>54%</td>
</tr>
<tr>
<td>Automated Unit Test</td>
<td>53%</td>
</tr>
<tr>
<td>User Stories</td>
<td>52%</td>
</tr>
<tr>
<td>Detailed Designs</td>
<td>51%</td>
</tr>
<tr>
<td>Definition of done / ready</td>
<td>50%</td>
</tr>
<tr>
<td>Formal estimation</td>
<td>49%</td>
</tr>
<tr>
<td>Security Testing</td>
<td>49%</td>
</tr>
<tr>
<td>Burn-Down Charts</td>
<td>48%</td>
</tr>
<tr>
<td>Pair Programming</td>
<td>47%</td>
</tr>
<tr>
<td>End-to-End (System) Testing</td>
<td>47%</td>
</tr>
<tr>
<td>Collective code ownership</td>
<td>46%</td>
</tr>
<tr>
<td>Test-driven Development (TDD)</td>
<td>45%</td>
</tr>
<tr>
<td>Use Case Modeling</td>
<td>45%</td>
</tr>
<tr>
<td>Automated Code Generation</td>
<td>44%</td>
</tr>
<tr>
<td>On-Site Customer</td>
<td>43%</td>
</tr>
<tr>
<td>Velocity-based planning</td>
<td>43%</td>
</tr>
<tr>
<td>Iteration Planning</td>
<td>42%</td>
</tr>
<tr>
<td>Destructive Testing</td>
<td>41%</td>
</tr>
<tr>
<td>Scrum-of-Scrums</td>
<td>40%</td>
</tr>
<tr>
<td>Model Checking</td>
<td>39%</td>
</tr>
<tr>
<td>Formal Specification</td>
<td>38%</td>
</tr>
</tbody>
</table>

### Processes

1. Coding Standards (69%)
2. Release Planning (65%)
3. Code Review (60%)
4. Backlog Mnmt. (60%)
5. Continuous Int. (58%)
6. Limit WiP (57%)
7. Daily Stand-Up (55%)
8. Iter./Sprint Review (54%)
9. Automated Unit Test (53%)
10. User Stories (52%)

---

**LTAT.05.003 / Lecture 01.2 / © Dietmar Pfahl 2018**
Dev. Practices Used in Estonia

Processes

1-Release Planning (92%)
2-Refactoring (84%)
2-Daily Stand-Up (84%)
4-Coding Standards (83%)
4-Continuous Int. (83%)
6-Collective Code Ownership (67%)
7-Code Review (75%)
8-Iter./Sprint Review (75%)
9-Backlog Mnmt. (67%)
10-Autom. Unit Test (66%)

Practices
Which of the following practices do you use?

- Coding standards
- Code review
- Prototyping
- Refactoring
- Release planning
- Automated Unit Testing
- Expert/Team based estimation
- Design Reviews
- Backlog Management
- Continuous integration
- User Stories
- Architecture Specifications
- Iteration/Sprint Reviews
- Limit Work-in-Progress
- Retrospectives
- Daily Standup
- Continuous deployment
- Detailed Designs
- Definition of done / ready
- Formal estimation
- Security Testing
- Burn-Down Charts
- Pair Programming
- End-to-End (System) Testing
- Collective code ownership
- Test-driven Development (TDD)
- Use Case Modeling
- Automated Code Generation
- On-Site Customer
- Velocity-based planning
- Iteration Planning
- Destructive Testing
- Scrum-of-Scrum
- Model Checking
- Formal Specification
- Automated Theorem Proving

- Do not know it
- Do not know if we use it
- We never use it
- We rarely use it
- We sometimes use it
- We often use it
- We always use it
Dev. Practices Used in Sweden

<table>
<thead>
<tr>
<th>Practices</th>
<th>Usage Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Daily Stand-Up (85%)</td>
<td></td>
</tr>
<tr>
<td>Release Planning (84%)</td>
<td></td>
</tr>
<tr>
<td>Backlog Mnmt. (77%)</td>
<td></td>
</tr>
<tr>
<td>Retrospectives (70%)</td>
<td></td>
</tr>
<tr>
<td>Automated Unit Test (70%)</td>
<td></td>
</tr>
<tr>
<td>Architecture Spec. (69%)</td>
<td></td>
</tr>
<tr>
<td>Design Review (69%)</td>
<td></td>
</tr>
<tr>
<td>Exp./Team Estimat. (62%)</td>
<td></td>
</tr>
<tr>
<td>Coding Standards (61%)</td>
<td></td>
</tr>
<tr>
<td>Code Review (61%)</td>
<td></td>
</tr>
<tr>
<td>Security Testing</td>
<td></td>
</tr>
<tr>
<td>Pair Programming</td>
<td></td>
</tr>
<tr>
<td>End-to-End (System) Testing</td>
<td></td>
</tr>
<tr>
<td>Collective code ownership</td>
<td></td>
</tr>
<tr>
<td>Test-driven Development (TDD)</td>
<td></td>
</tr>
<tr>
<td>Use Case Modeling</td>
<td></td>
</tr>
<tr>
<td>Automated Code Generation</td>
<td></td>
</tr>
<tr>
<td>On-Site Customer</td>
<td></td>
</tr>
<tr>
<td>Velocity-based planning</td>
<td></td>
</tr>
<tr>
<td>Iteration Planning</td>
<td></td>
</tr>
<tr>
<td>Destructive Testing</td>
<td></td>
</tr>
<tr>
<td>Scrum-of-Scrums</td>
<td></td>
</tr>
<tr>
<td>Model Checking</td>
<td></td>
</tr>
<tr>
<td>Formal Specification</td>
<td></td>
</tr>
<tr>
<td>Automated Theorem Proving</td>
<td></td>
</tr>
</tbody>
</table>
Company Sizes of Respondents: Estonia vs. Sweden vs. 26 Countries

Company Size
What is your company’s size in equivalent full-time employees (FTEs)?

All the countries

Estonia

Sweden

- Not Answered
- Small (11 - 50 employees)
- Medium (51 - 250 employees)
- Large (251 - 2499 employees)
- Very Large (>2500 employees)
Software Engineering

Consistent application of engineering principles and methods to the development of software (intensive) systems

Engineering:
Application of systematic (i.e., predictable, repeatable, scalable) procedures
- with well-defined goals (e.g., quality, functionality/scope, cost, time)
- with well-defined/structured products, processes, and organization
Adherence to existing body of knowledge
Observation of constraints (standards, time/cost/quality requirements, etc.)
Development and use of models
Magic Triangle of SE

- **6 Quality Characteristics** (ISO 9126 / ISO 25000):
  - Functionality
  - Reliability
  - Usability
  - Efficiency
  - Maintainability
  - Portability

- **Effort/Cost and Time**:  
  - Development
  - Maintenance
Software Engineering

A bridge from customer/user needs to software product

Customer, User Needs

Developer (SW Engineer)

Software Product/System
Next Lecture

• Date/Time:
  • Friday, 14-Sep, 10:15-12:00

• Topic:
  • Requirements Engineering I  ➔  1st Homework!

• For you to do:
  • Have a look at the course wiki
  • Make sure you know to which lab group you have been enrolled + start forming project teams
  • MOST IMPORTANTLY: Go to the labs next week!