Lecture 01.2:
Introduction to Software Engineering

Dietmar Pfahl
email: dietmar.pfahl@ut.ee

Fall 2019
Schedule of Lectures

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

**Week 05: Development Infrastructure**
- Week 06: Continuous Development and Integration

**Week 07: Architecture and Design I**
- Week 08: Architecture and Design II

**Week 09: Verification and Validation I**
- Week 10: Verification and Validation II
- Week 11: Refactoring (and TDD)
- Week 12: Agile/Lean Methods
- Week 13: Software Craftsmanship
- Week 14: Course wrap-up, review and exam preparation

**Week 15: Reserve time slot (no lecture scheduled as of today)**
Software Engineering

What?
Why?
Seventh Inning Stretch

# Task:
Talk to your neighbour and exchange your personal experience (or opinion) with (about) SE
Software Engineering

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

- Software Development

Project or Iteration
Software Development – Three Ps

• Software Development

Products

People

Iteration

Processes
Software Development – Three Ps

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

class ISort
{
    public static void insertionSort(int[] a, String direction)
    {
        int in, out;
        for(out=1; out<a.length; out++)
        {
            int temp = a[out];
            in = out;
            while((in>0 && a[in-1] >= temp && direction.equalsIgnoreCase("A"))
                || (in>0 && a[in-1] <= temp && direction.equalsIgnoreCase("D")))
            {
                a[in] = a[in-1];
                --in;
            }
        }
        a[in] = temp;
    }
}
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
  - ...

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

- Types of Software:
  - Embedded/real-time
  - Information System
  - Web application
  - Mobile application
  - Systems of systems
  - ...

- Models
Software in a Car

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

ECU = Electronic Control Unit

Autonomous driving
Secure cloud communication

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

Online information: http://www.uml.org
Software Development – Three Ps

- Software Development
- Products
- Processes
- People

Project or Iteration
People in Software Development

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

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

Skills:
- Must match roles

Training:
- Must fill skill-gaps

Education:
- Curricula (ACM/IEEE)

User models
Software Development – Three Ps

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

Coding

Deploying

Processes

?
Software Development Process

Find Requirements
Analysis / Designing
Coding
Testing
Deploying
Software Development Process

(System Requirements)

(Preliminary Program Design)

(Analysis)

(Program Design)

(Coding)

(Testing)

(Operations)

(Usage)

(Preliminary Design)

(Analysis)

(Program Design)

(Coding)

(Testing)

(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
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
Comparison of Basic Process Types

- Waterfall
- Design
- Implementation
- Testing
- Requirements
- 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
Process Frameworks Used in 26 Countries

<table>
<thead>
<tr>
<th>Frameworks/Methods</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>14%</td>
</tr>
<tr>
<td>Kanban</td>
<td>14%</td>
<td>18%</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>19%</td>
<td>19%</td>
</tr>
<tr>
<td>Lean Software Development</td>
<td>20%</td>
<td>16%</td>
</tr>
<tr>
<td>Domain-Driven Design</td>
<td>21%</td>
<td>18%</td>
</tr>
<tr>
<td>Scrum-Ban</td>
<td>11%</td>
<td>12%</td>
</tr>
<tr>
<td>Feature Driven Development (FDD)</td>
<td>10%</td>
<td>15%</td>
</tr>
<tr>
<td>V-shaped Process (V-Model)</td>
<td>11%</td>
<td>12%</td>
</tr>
<tr>
<td>Phase / Stage-gate model</td>
<td>19%</td>
<td>12%</td>
</tr>
<tr>
<td>Model-Driven Architecture (MDA)</td>
<td>33%</td>
<td>15%</td>
</tr>
<tr>
<td>Scaled Agile Framework (SAFe)</td>
<td>20%</td>
<td>16%</td>
</tr>
<tr>
<td>Team Software Process</td>
<td>11%</td>
<td>12%</td>
</tr>
<tr>
<td>Personal Software Process</td>
<td>9%</td>
<td>12%</td>
</tr>
<tr>
<td>Nexus</td>
<td>9%</td>
<td>12%</td>
</tr>
<tr>
<td>Large-Scale Scrum (LESS)</td>
<td>12%</td>
<td>10%</td>
</tr>
<tr>
<td>SSADM</td>
<td>33%</td>
<td>14%</td>
</tr>
<tr>
<td>Spiral Model</td>
<td>8%</td>
<td>23%</td>
</tr>
<tr>
<td>Dynamic Systems Development Method</td>
<td>7%</td>
<td>36%</td>
</tr>
<tr>
<td>Crystal Family</td>
<td>20%</td>
<td>19%</td>
</tr>
<tr>
<td>PRINCE2</td>
<td>11%</td>
<td>12%</td>
</tr>
<tr>
<td>Rational Unified Process</td>
<td>10%</td>
<td>16%</td>
</tr>
</tbody>
</table>
Process Frameworks Used in Estonia

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

1. Scrum
2. Iterative Development
3. Kanban
4. Classic Waterfall
5. XP
# Dev. Practives Used in 26 Countries

<table>
<thead>
<tr>
<th>Practices</th>
<th>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 Management</td>
<td>60%</td>
</tr>
<tr>
<td>Continuous Int.</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>50%</td>
</tr>
<tr>
<td>Definition of done / ready</td>
<td>49%</td>
</tr>
<tr>
<td>Pair Programming</td>
<td>46%</td>
</tr>
<tr>
<td>End-to-End (System) Testing</td>
<td>46%</td>
</tr>
<tr>
<td>Collective code ownership</td>
<td>45%</td>
</tr>
<tr>
<td>Test-driven Development (TDD)</td>
<td>44%</td>
</tr>
<tr>
<td>Use Case Modeling</td>
<td>44%</td>
</tr>
<tr>
<td>Automated Code Generation</td>
<td>43%</td>
</tr>
<tr>
<td>On-Site Customer</td>
<td>43%</td>
</tr>
<tr>
<td>Velocity-based planning</td>
<td>41%</td>
</tr>
<tr>
<td>Iteration Planning</td>
<td>40%</td>
</tr>
<tr>
<td>Destructive Testing</td>
<td>39%</td>
</tr>
<tr>
<td>Scrum-of-Scrums</td>
<td>35%</td>
</tr>
<tr>
<td>Model Checking</td>
<td>33%</td>
</tr>
<tr>
<td>Formal Specification</td>
<td>31%</td>
</tr>
<tr>
<td>Automated Theorem Proving</td>
<td>30%</td>
</tr>
<tr>
<td>Do not know it</td>
<td></td>
</tr>
<tr>
<td>Do not know if we use it</td>
<td></td>
</tr>
<tr>
<td>We never use it</td>
<td></td>
</tr>
<tr>
<td>We rarely use it</td>
<td></td>
</tr>
<tr>
<td>We sometimes use it</td>
<td></td>
</tr>
<tr>
<td>We often use it</td>
<td></td>
</tr>
<tr>
<td>We always use it</td>
<td></td>
</tr>
</tbody>
</table>

*LTAT.05.003 / Lecture 01.2 / © Dietmar Pfahl 2019*
### Dev. Practices Used in Estonia

<table>
<thead>
<tr>
<th>Practice</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release Planning</td>
<td>92%</td>
</tr>
<tr>
<td>Refactoring</td>
<td>84%</td>
</tr>
<tr>
<td>Daily Stand-Up</td>
<td>84%</td>
</tr>
<tr>
<td>Coding Standards</td>
<td>83%</td>
</tr>
<tr>
<td>Continuous Code Ownership</td>
<td>67%</td>
</tr>
<tr>
<td>Code Review</td>
<td>75%</td>
</tr>
<tr>
<td>Iteration/Sprint Review</td>
<td>75%</td>
</tr>
<tr>
<td>Backlog Management</td>
<td>67%</td>
</tr>
<tr>
<td>Automated Unit Test</td>
<td>66%</td>
</tr>
<tr>
<td>End-to-End (System) Testing</td>
<td>58%</td>
</tr>
<tr>
<td>Test-driven Development (TDD)</td>
<td>58%</td>
</tr>
<tr>
<td>Use Case Modeling</td>
<td>58%</td>
</tr>
<tr>
<td>Automated Code Generation</td>
<td>58%</td>
</tr>
<tr>
<td>On-Site Customer</td>
<td>58%</td>
</tr>
<tr>
<td>Velocity-based planning</td>
<td>58%</td>
</tr>
<tr>
<td>Iteration Planning</td>
<td>58%</td>
</tr>
<tr>
<td>Destructive Testing</td>
<td>58%</td>
</tr>
<tr>
<td>Scrum-of-Scrums</td>
<td>58%</td>
</tr>
<tr>
<td>Model Checking</td>
<td>58%</td>
</tr>
<tr>
<td>Formal Specification</td>
<td>58%</td>
</tr>
<tr>
<td>Automated Theorem Proving</td>
<td>58%</td>
</tr>
</tbody>
</table>
# Dev. Practices Used in Sweden

Which of the following practices do you use?

<table>
<thead>
<tr>
<th>Practice</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding standards</td>
<td>8%</td>
</tr>
<tr>
<td>Code review</td>
<td>31%</td>
</tr>
<tr>
<td>Prototyping</td>
<td>46%</td>
</tr>
<tr>
<td>Refactoring</td>
<td>15%</td>
</tr>
<tr>
<td>Release planning</td>
<td>15%</td>
</tr>
<tr>
<td>Automated Unit Testing</td>
<td>8%</td>
</tr>
<tr>
<td>Expert/Team based estimation</td>
<td>15%</td>
</tr>
<tr>
<td>Design Reviews</td>
<td>46%</td>
</tr>
<tr>
<td>Backlog Management</td>
<td>46%</td>
</tr>
<tr>
<td>Continuous integration</td>
<td>46%</td>
</tr>
<tr>
<td>User Stories</td>
<td>46%</td>
</tr>
<tr>
<td>Architecture Specifications</td>
<td>46%</td>
</tr>
<tr>
<td>Iteration/Sprint Reviews</td>
<td>46%</td>
</tr>
<tr>
<td>Limit Work-in-Progress</td>
<td>46%</td>
</tr>
<tr>
<td>Retrospectives</td>
<td>46%</td>
</tr>
<tr>
<td>Daily Standup</td>
<td>46%</td>
</tr>
<tr>
<td>Continuous deployment</td>
<td>46%</td>
</tr>
<tr>
<td>Detailed Designs</td>
<td>46%</td>
</tr>
<tr>
<td>Definition of done/readiness</td>
<td>46%</td>
</tr>
<tr>
<td>Formal estimation</td>
<td>46%</td>
</tr>
<tr>
<td>Security Testing</td>
<td>46%</td>
</tr>
<tr>
<td>Burn-Down Charts</td>
<td>46%</td>
</tr>
<tr>
<td>Pair Programming</td>
<td>46%</td>
</tr>
<tr>
<td>End-to-End (System) Testing</td>
<td>46%</td>
</tr>
<tr>
<td>Collective code ownership</td>
<td>46%</td>
</tr>
<tr>
<td>Test-driven Development (TDD)</td>
<td>46%</td>
</tr>
<tr>
<td>Use Case Modeling</td>
<td>46%</td>
</tr>
<tr>
<td>Automated Code Generation</td>
<td>46%</td>
</tr>
<tr>
<td>On-Site Customer</td>
<td>46%</td>
</tr>
<tr>
<td>Velocity-based planning</td>
<td>46%</td>
</tr>
<tr>
<td>Iteration Planning</td>
<td>46%</td>
</tr>
<tr>
<td>Destructive Testing</td>
<td>46%</td>
</tr>
<tr>
<td>Scrum-of-Scrums</td>
<td>46%</td>
</tr>
<tr>
<td>Model Checking</td>
<td>46%</td>
</tr>
<tr>
<td>Formal Specification</td>
<td>46%</td>
</tr>
<tr>
<td>Automated Theorem Proving</td>
<td>46%</td>
</tr>
</tbody>
</table>

1. Daily Stand-Up (85%)
2. Release Planning (84%)
3. Backlog Mnmt. (77%)
4. Retrospectives (70%)
4. Automated Unit Test (70%)
6. Architecture Spec. (69%)
6. Design Review (69%)
8. Exp./Team Estimat. (62%)
9. Coding Standards (61%)
9. Code Review (61%)
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: 21.76% Small, 14.97% Medium, 41.67% Large, 23.95% Very Large, 11.58% Not Answered
Estonia: 27.54% Small, 8.33% Medium, 41.67% Large, 11.58% Very Large, 11.58% Not Answered
Sweden: 46.15% Small, 7.69% Medium, 38.46% Large, 7.69% Very Large, 7.69% Not Answered

Legend:
- Blue: Not Answered
- Orange: Micro (<10 employees)
- Red: Small (11 - 50 employees)
- Green: Large (251 - 2499 employees)
- Teal: Medium (51 - 250 employees)
- Yellow: 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, 13-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!