Software processes

Kristian Sandahl
krs@ida.liu.se

Contents
- Definitions
- Software life-cycle processes
  - activities
  - software life-cycle models
- Process models

Software process
- Sequence of steps
- Result: software items
- Input
- Output
- Resource consumption
- Feed-back

The effector process
- A process that verifies itself
- A process that exits under certain criteria
- ETVXM-architecture:
  - Entry
  - Task
  - Verify
  - Exit
  - Measure

Example:
- Plan
- Decide
- Write SRS
- SRS
- Inspect
- # defects
- X

feed-back
resources
side-effects
analysis
design
implementation
in
out
Process levels

- Universal:
  - Processes suitable for many projects
- Worldly
  - Processes adapted to a certain project or product
- Atomic
  - Detailed processes for teams and individuals

The software life-cycle

= time for concept -> time for “unavailability”

The SLC is made up by:
- Software life-cycle model
- Activities

Software life-cycle models

- Waterfall model
- Incremental model
- Spiral model
- Win-win spiral model
- Iterative model: “RUP”

Waterfall model

de Facto reference model
- forward engineering
- manageable
- fixed documents
Negative:
- one-step delivery
- the negative circle:
  - sensitivity to changes -> more time for planning -> shortened design time -> sensitivity to changes....
The win-win negotiation

- Find stake-holders’ win condition
- Infer design attributes
- Negotiate a suitable architectural solution

See: [http://sunset.usc.edu/research/WINWIN/index.html](http://sunset.usc.edu/research/WINWIN/index.html)

The win-win spiral model

1. Identify stakeholders’ win conditions
2. Define stakeholders’ win conditions
3. Establish next level objectives, constraints, and alternatives
4. Evolve product and process alternatives, resolve risks
5. Define next iteration of product and process, including iterations
**RUP – Rational Unified Process**

- **Business Modeling**
- **Requirements**
- **Analysis & Design**
- **Implementation**
- **Deployment**

**Concept Exploration**
- Requirements
- Design

**Implementation**
- Design
- Test

**Test**

**Synchronise and Stabilise**
- Idea: to always be prepared to deliver
- Incremental method
- Frequent increment installation and test
  - Daily Build
  - Smoke tests
  - Regression testing
  - The MS way

**Cleanroom process model**
- Incremental method
- Committed to formal specification
- Dedicated use of ETVXM
- Usage-based verification

**Prototypig**
- Sometimes called RAD (Rapid Application Development)
- Focus on feedback
- Negative:
  - too early commitment
  - hard to obtain quality?
eXtreme Programming

Some XP-rules

- User stories are written
- Make frequent small releases
- Move people around
- Simplicity
- Choose a system metaphor
- Create spike solutions to reduce risk
- Refactor whenever and wherever possible

- The customer is always available
- Code the unit test first
- All production code is pair programmed
- Leave optimisation till last
- No overtime
- All code must pass all unit tests before it can be released
- Acceptance tests are run often and the score is published.

Open source

- The code is published
- An interested community voluntarily evolves the code
- All results are free to use
- Success #1: Linux
- "Software culture"