Software Process

Overview

• What is **software process**?
• **Generic process framework**

• Examples of process models
• **Unified Process (UP)**
• **Agile software development**
Software Process

- **Definition [Pressman]**
  - a framework for the tasks that are required to build high-quality software.
  - to provide stability, control and organization to an otherwise chaotic activity

What does SW process mean?

- **For a single programmer**
  - Planning (time, resources, assignments)
  - Design and development
  - Tracking and measuring progress

- **For a team of practitioners**
  - Organizational planning (time, resources, etc.)
  - Hiring, training, tool acquisition, etc.
  - Process assessment and improvement

- **For software engineering in general**
  - Helps organize SE around 'best practices'
Elements of SW process

<table>
<thead>
<tr>
<th>Term</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>People</td>
<td>Software developers, project managers, customers</td>
</tr>
<tr>
<td>Tasks</td>
<td>Analyze requirements</td>
</tr>
<tr>
<td>Work products</td>
<td>Requirements specification</td>
</tr>
<tr>
<td>Planning</td>
<td>Estimate needed resource, time, defects</td>
</tr>
<tr>
<td>Conducting</td>
<td>Track progress and work results</td>
</tr>
<tr>
<td>Assessing</td>
<td>Define and measure metrics like quality, progress, etc.</td>
</tr>
</tbody>
</table>

A process defines who is doing what, when and how to reach a certain goal.

Generic View of SW Process

Definition phase

Development phase

Support phase

Umbrella activities
Definition Phase

• Tasks related to problem definition
  – What? - requirements, constraints, environment, etc.
• Step 1: System engineering
  – Ascertain roles of hardware, software, people, databases, operational procedures, etc. in system
• Step 2: Analysis of the problem
  – Requirement analysis
    • Understanding what the users need and want
  – Domain analysis
    • Illustrate key concepts in a set of SW systems (reuse)
• Step 3: Project planning
  – Resources (e.g., people), cost, schedule

Development Phase

• Tasks related to problem solution
  – How? - architecture, programming, testing, etc.
• Step 1: software design (the blueprint)
  – Design models that describe structure, interactions, etc.
• Step 2: code generation/implementation
• Step 3: software testing
  – Goal: uncover as many errors as possible
Support (Maintenance) Phase

• Tasks related to software evolution
  – Changes? - Definition and development in the context of existing software
• Adaptation to change in the environment
  – New hardware, changes in OS, business rules, etc.
• Correction of defects (Y2K problem, $308B)
• Enhancements (new features, etc.)
• Refactoring (to ease future changes)

Some Umbrella Activities

• Project management
  – Tracking and control of people, process, cost, etc.
• Quality assurance (QA)
  – Formal technical reviews of work products
  – Software testing
  – Keeping docs consistent with code base
• Configuration management
  – Controls the changes in work products using systems like SVN, Git
Observations

• Process models are idealizations
  – The real world is a very complex place
• They can be very difficult to execute
  – Conformance can be faked
• But, they provide a roadmap for SE work to organize an otherwise chaotic activity

Code-and-Fix Process

• The first thing people tried in the 1950s
  1. Write program
  2. Improve it (debug, add functionality, improve efficiency, …)
  3. GOTO 1
• Works for small 1-person projects and for some CS course assignments
Problems with Code-and-Fix

- Poor match with user needs
- Bad overall structure - No blueprint
- Poor reliability - no systematic testing
- Maintainability? What's that?
- What happens when the programmer quits?

Code-and-Fix process
Code-and-Fix Process

From McConnell, After the Goldrush, 1999

N. Meng, B. Ryder

A More Advanced Process

From McConnell, After the Goldrush, 1999

N. Meng, B. Ryder
Examples of Process Models

- Waterfall model
- Prototyping model
- Spiral model
- Incremental model

Waterfall Model

- The “classic” process model since 1970s
  – Also called “software life cycle”

Analysis

Design

Implementation

Testing & Integration

Maintenance
Waterfall Phases

- **Analysis**: Define problems
  - requirements, constraints, goals and domain concepts
- **Design**: Establish solutions
  - System architecture, components, relationship
- **Implementation**: Implement solutions
- **Testing and integration**: Check solutions
  - Unit testing, system testing
- **Maintenance**: the longest phase

Key Points of the Model

- The project goes through the phases sequentially
- Possible feedback and iteration across phases
  - e.g., during coding, a design problem is identified and fixed
- Typically, few or no iterations are used
  - e.g., after a certain point of time, the design is "frozen"
Waterfall Model Assumptions

- All requirements are known at the start and stable
- Risks (unknown) can be turned into known through schedule-based invention and innovation
- The design can be done abstractly and speculatively
  - i.e., it is possible to correctly guess in advance how to make it work
- Everything will fit together when we start the integration

How was the model developed?

a) A group of researchers developed and proposed it as the best option of existing methods
b) A group of practitioners innovated a method that became the most widely used model
c) A person copied a picture of a method that he understood and could explain

Winston Royce wrote a recommendation about how to structure process for large software projects based on his experiences from NASA
Success story: space shuttle software
Charles Fishman, 1996

As the 120-ton space shuttle sits surrounded by almost 4 million pounds of rocket fuel, exhaling noxious fumes, visibly impatient to defy gravity, its on-board computers take command.

“This software is bug-free”

• Impressive statistics
  – The last 3 versions of the program--420,000 lines of code had just 1 error each
  – The last 11 versions of the software had a total of 17 errors
  – Commercial programs of equivalent complexity would have 5,000 errors
How did they write the right stuff?

• 1/3 of the process before coding
• NASA and Lockheed Martin groups agree in the most minute detail about everything
• Specs are almost pseudo-code
• Nothing in the specs is changed without agreement and understanding
• Task: upgrade software to add GPS navigation
  – 1.5% changes in program/6366 LOC
  – 2500 page specs for the change

How expensive is the software?

• 260 people
• >40,000 pages of specifications
• 20 years
• $35 million Annual budget
• $700 million overall budget
• 700 million/420k = $1600/line of code
Pros and Cons

• Pros: widely used, systematic, good for projects with well-defined requirements
  – Makes managers happy

• Cons:
  – The actual process is not so sequential
    • A lot of iterations may happen
  – The assumptions usually don’t hold
  – Working programs are not available early
    • High risk issues are not tackled early enough
  – Expensive and time-consuming

When would you like to use waterfall?

• Work for big clients enforcing formal approach on vendors
• Work on fixed-scope, fixed-price contracts without many rapid changes
• Work in an experienced team
Observation

Standish group 1995

- Top three reasons for at least partial failure projects
  - lack of user input
  - incomplete requirements, and
  - changing requirement

Prototyping Model

- Build a prototype when customers have ambiguous requirements
Key Points of the Model

• Iterations: customer evaluation followed by prototype refinement
• The prototype can be paper-based or computer-based
• It models the entire system with real data or just a few screens with sample data
• Note: the prototype is thrown away!

Success stories of prototyping

• Organizations of all types do it
  – Boeing builds digital prototypes of its aircraft allowing the detection of design conflicts
  – Disney uses storyboards to work through the process of producing feature-length films
• Online systems and web interfaces
Pros and Cons

• Pros
  – Facilitate communication about requirements
  – Easy to change or discard
  – Educate future customers

• Cons
  – Iterative nature makes it difficult to plan and schedule
  – Excessive investment in the prototype
  – Bad decisions based on prototype
    • E.g., bad choice of OS or PL

When would you like to use prototyping?

• When the desired system has a lot of interactions with users
Spiral Model

- A risk-driven evolutionary model that combines development models (waterfall, prototype, etc.)

Spiral Phases

- **Objective setting**
  - Define specific objectives, constraints, products, plans
  - Identify risks and alternative strategies
- **Risk assessment and reduction**
  - Analyze risks and take steps to reduce risks
- **Development and validation**
  - Pick development methods based on risks
- **Planning**
  - Review the project and decide whether to continue with a further loop
What Is Risk?

- **Something that can go wrong**
  - People, tasks, work products
- **Risk management**
  - risk identification
  - risk analysis
    - the probability of the risk, the effect of the risk
  - risk planning
    - various strategies
  - risk monitoring

### Risk Planning [Sommerville]

<table>
<thead>
<tr>
<th>Risk</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recruitment problems</td>
<td>□ Alert customer of potential difficulties and the possibility of delays, investigate buying-in-components</td>
</tr>
<tr>
<td>Defective components</td>
<td>□ Replace potentially defective components with bought-in components of known reliability</td>
</tr>
<tr>
<td>Requirements changes</td>
<td>□ Derive traceability information to assess requirements change impact, maximize information hiding in the design</td>
</tr>
<tr>
<td>Organizational financial problems/ restructuring</td>
<td>□ Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business</td>
</tr>
<tr>
<td>Underestimated development time</td>
<td>□ Investigate buying-in components, investigate the use of a program generator</td>
</tr>
</tbody>
</table>
Key Points of the Model

- Introduce risk management into process
- Develop evolutionary releases to
  - Implement more complete versions of software
  - Make adjustment for emergent risks

Pros and Cons

- Pros
  - High amount of risk analysis to avoid/reduce risks
  - Early release of software, with extra functionalities added later
  - Maintain step-wise approach with “go-backs” to earlier stages
- Cons
  - Require risk-assessment expertise for success
  - Expensive
When to use the model?

- Large and mission-critical projects
- Medium to high-risk projects
- Significant changes are expected

Incremental Model

- A sequential of waterfall models

Feedback, adaptation

Analysis → Design → Implementation → Testing & Integration

Iteration n: 3 weeks (for example)

Analysis → Design → Implementation → Testing & Integration

Iteration n+1: 3 weeks (for example)
Key Points of the Model

• Iterative: many releases/increments
  – First increment: core functionality
  – Successive increments: add/fix functionality
  – Final increment: the complete product
• Require a complete definition of the whole system to break it down and build incrementally

Pros and Cons

• Pros
  – Early discovery of software defects
  – Early delivery of working software
  – Less cost to change/identify requirements
• Cons
  – Constant changes ("feature creep") may erode system architecture
When to use the model?

- The requirements of the complete system are clear
- Major requirements must be defined while some details can evolve over time
- Need to get a product to the market early

Spiral model vs. increment model

- Iterative models
  - Most projects build software iteratively
- Risk-driven vs. client-driven
Unified Process (UP)

- An example of iterative process for building object-oriented systems
  - Very popular in the last few years
  - By the same folks who develop UML
- It provides a context for our discussion of analysis and design

A Little History

- “The three amigos”: Grady Booch, Ivar Jacobson, James Rumbaugh
  - Early 90s: Separated methodologies for object-oriented analysis and design (OOAD)
  - 1996: Created the Unified Modeling Language (UML)
  - 1999: Defined the Unified Process (UP) in Rational Software Inc.
    - Refinement: Rational Unified Process (RUP)
      - Adaptable process framework + tools
**Phases in UP**

<table>
<thead>
<tr>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
</table>

- **Inception**: preliminary investigation
- **Elaboration**: analysis, design, and some coding
- **Construction**: more coding and testing
- **Transition**: beta tests and development
- Each phase may be enacted in an iterative way, and the whole set of phases may be enacted incrementally

**Inception Phase**

- Investigate approximate, business case, scope, and vague estimates
  - Should we even bother?
- Some basic analysis to decide whether to continue or stop
- Inception is NOT “requirement” in waterfall
Elaboration Phase

- Most requirement analysis
- Most design
- Some coding and testing
  - Implementation and testing for core architecture and high-risk requirements
- Deeper investigation of scope, risks, and estimates
- Work products
  - Requirement models (UML use cases)
  - An architectural description
  - A development plan

Construction Phase

- More coding and testing
  - Implementation and testing for the remaining lower risk and easier elements
  - Integration
- Work products ready for delivery
  - A working software system
  - Associated documentation
Transition Phase

• Beta tests and deployment
  – Moving the system from the development community to the user community
  – This is important but ignored in most software process model

• Work products
  – A documented software system that is working correctly in its operational environment

Iteration Length

• Iteration should be short (2-6 weeks)
  – Small steps, rapid feedback and adaptation
  – Massive teams with lots of communication - but no more than 6 months

• Iterations should be timeboxed (fixed length)
  – Integrate, test and deliver the system by a scheduled data
  – If not possible: move tasks to the next iteration
Reasons for Timeboxing

• Improve programmer productivity with deadlines
• Encourage prioritization and decisiveness
• Team satisfaction and confidence
  – Quick and repeating sense of completion, competency, and closure
  – Increase confidence for customers and managers

UP Disciplines

• Discipline: an activity and related artifact(s)
• Artifact: any kind of work product
• We will focus on artifacts related to two disciplines
  – Requirement modeling
    • requirement analysis + use-case models, domain models, and specs.
  – Design
    • design + design models
Agile Software Development

• A timeboxed iterative and evolutionary development process
• It promotes
  – adaptive planning
  – evolutionary development,
  – incremental delivery
  – rapid and flexible response to change

Any iterative method, including the UP, can be applied in an agile spirit.

The Agile Manifesto

Kent Beck et al. 2001

• We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
  – Individuals and interactions over Processes and tools
  – Working software over Comprehensive documentation
  – Customer collaboration over Contract negotiation
  – Responding to change over Following a plan
Key Points of Agile Modeling

• The purpose of modeling is primarily to understand, not to document
• Modeling should focus on the smaller percentage of unusual, difficult, tricky parts of the design space
• Model in pairs (or triads)
• Developers should do the OO design modeling for themselves
• Create models in parallel
  – E.g., interaction diagram & static-view class diagram

Models are inaccurate

• only tested code demonstrates the true design
• treat diagrams as throw-away explorations
• Use the simplest tool possible to facilitate creative thinking
  – E.g., sketching UML on whiteboards
• Use “good enough” simple notation
Agile Methods

- Agile Unified Process (Agile UP)
- Dynamic systems development method (DSDM)
- Extreme programming (XP)
- Feature-driven development (FDD)
- Scrum

Agile UP

- Keep it simple
  - Prefer a small set of UP activities and artifacts
  - Avoid creating artifacts unless necessary
- Planning
  - For the entire project, there is only a high-level plan (Phase Plan), to estimate the project end date and other major milestones
  - For each iteration, there is a detailed plan (Iteration plan) created one iteration in advance
Pros and Cons

- **Pros**
  - Customer satisfaction by rapid, continuous delivery of useful software
  - Close, daily cooperation between business people and developers
  - Better software quality and lower cost

- **Cons**
  - People may lose sight of the big picture
  - Heavy client participation is required
  - Poor documentation support for training of new clients/programmers

When to use agile methods?

- Changing requirements
- Faster time to market and increased productivity
- Frequently used in start-up companies
A Borrowed Joke

How many software engineers does it take to change a light bulb?

Five. Two to write the specification, one to screw it in, and two to explain why the project was late.