Software Engineering

Process

Reading
- SE chapters 2, 12
- Eric S. Raymond: The Cathedral and the Bazaar
- read by 11/15: there will be a short quiz on it on Monday
  
  http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar.ps

Reasons for modeling a process
- To form a common understanding
- To find inconsistencies, redundancies, omissions
- To find and evaluate appropriate activities for reaching process goal
- To tailor a general process for the particular situation in which it will be used
Examples of process models

- Waterfall model
- Prototyping
- V-model
- Operational specification
- Transformational model
- Phased development: increments and iteration
- Spiral model
Operational Specification

Transformational Model

Phased Development
Agile Software Development

- 12 principles of agile software development

Agile Principles (1)

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
Agile Principles (2)

- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity—the art of maximizing the amount of work not done—is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Remarks on Agile Development (Martin Fowler)

- Beware the engineering metaphor. Building software is nothing like building bridges. When you’re spanning a river, design represents perhaps 15% of the cost; construction accounts for the rest. When you’re building a software application, in contrast, all the effort is in design; the actual coding (i.e., construction) is so cheap as to be essentially free.
- Creative processes, including application design, aren’t easily planned, and predictability may be an impossible target. Developers and customers thus need to be ready to change their methods and goals as a project continues.
- Agile programming is people-centered, not process-oriented. If you treat your developers as fungible, interchangeable programming units instead of the creative and talented individuals they are, the good people will leave.

Tools and techniques for process modeling

- Example: Lai notation
  - activity
  - sequence
  - process model
  - resource
  - control
  - policy
  - organization
Table 2.1. Artifact definition form for artifact "car" (Lai 1991).

<table>
<thead>
<tr>
<th>Name</th>
<th>Synopsis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car</td>
<td>This is the artifact that represents a class of cars.</td>
</tr>
<tr>
<td>Complexity type</td>
<td>Composite</td>
</tr>
<tr>
<td>Data type</td>
<td>(car_c, user-defined)</td>
</tr>
<tr>
<td>Artifact-state list</td>
<td>parked (state_of(car.engine) = off) (state_of(car.gear) = park)</td>
</tr>
<tr>
<td></td>
<td>Car is not moving, and engine is not running.</td>
</tr>
<tr>
<td></td>
<td>initiated (state_of(car.engine) = on) (state_of(car.key_hole) = has-key)</td>
</tr>
<tr>
<td></td>
<td>(state_of(car-driver(car.)) = in-car) (state_of(car.gear) = drive)</td>
</tr>
<tr>
<td></td>
<td>Car is not moving, but the engine is running.</td>
</tr>
<tr>
<td></td>
<td>moving (state_of(car.engine) = on) (state_of(car.key_hole) = has-key)</td>
</tr>
<tr>
<td></td>
<td>(state_of(car-driver(car.)) = driving) (state_of(car.gear) = drive)</td>
</tr>
<tr>
<td></td>
<td>(state_of(car.speed) = stand)</td>
</tr>
<tr>
<td></td>
<td>Car is moving forward or backward.</td>
</tr>
<tr>
<td>Sub-artifact list</td>
<td>doors</td>
</tr>
<tr>
<td></td>
<td>The four doors of a car.</td>
</tr>
<tr>
<td></td>
<td>engine</td>
</tr>
<tr>
<td></td>
<td>The engine of a car.</td>
</tr>
<tr>
<td></td>
<td>keyhole</td>
</tr>
<tr>
<td></td>
<td>The ignition keyhole of a car.</td>
</tr>
<tr>
<td></td>
<td>gear</td>
</tr>
<tr>
<td></td>
<td>The gear of a car.</td>
</tr>
<tr>
<td></td>
<td>speed</td>
</tr>
<tr>
<td></td>
<td>The speed of a car.</td>
</tr>
</tbody>
</table>

Dynamic process models

- Enables enaction of process to see what happens to resources and artifacts as activities occur
- Simulate alternatives and make changes to improve the process
- Example: systems dynamics model (Abdel-Hamid & Madnick)

Productivity Factors (Abdel-Hamid & Madnick)