The Role of Software Processes in DSD

DSD Team

Outline

- Lecture: software process
  - Review: usefulness of software processes
  - Fitting processes to development problems
  - Choosing a process
- Lab Exercise: Which process is best for DSD?
- Lecture: from process to project plan
  - Importance of well defined process
  - Implementing a process in a project plan
- Lab Exercise: familiarize with assembla tools

Problems in software development

- Knowing what the customers want; knowing the requirements
- Predicting time to develop
- Predicting resources needed to develop
- Managing change
- Knowing when you’re done
- Designing to enable distribution of work
- Coordinating work
- Tracking time, resources, quality, productivity, effectiveness, etc.
Addressed by Software Processes

- Developed as a tool for controlling complex software developments
- Answers the "who", "what", "when", etc. questions
  - What product should we work on next?
  - What kind of person should do the work?
  - When is the work finished?
- Intended use
  - Provide a basis for planning and assessing development
  - Provide guidance to developers in what to produce and when

Intended use

- Different kinds of projects required different processes

Characteristic Processes: The Waterfall Model

- Process viewed as a sequential set of activities
- Imposes separation of concerns on software development activities

Characteristic Processes: The Spiral Model

- Process viewed as repeating cycles of increasing scale
- Identify risks and determine (next set of) requirements, build next version by extension, increasing scale each time
- Early iterations may be prototypes
Characteristic Processes: The Iterative Model

- Process viewed as a sequence of iterations, each building on the last
  - Build minimal useful subset, test, release, build next version by extension
  - Early iterations may be prototypes

Characteristic Processes: Agile (scrum)

- Process viewed as nested sequence of builds (sprints)
  - Each build adds small feature set
  - Customer in loop, code centered (little or no documentation)
  - Problem detection and correction through daily team meetings (scrum)

Formal Definition

- Process: we define a process as a set of artifacts, activities, roles, and the relationships between them where:
  - Artifacts: any work product of the software development process (requirements specifications, design documents, code, etc.)
  - Activities: the tasks that produce the work products (requirements analysis, design, coding)
  - Roles: responsibility for performing specific activities (requirements analyst, software architect, coder)
  - Relationships: the relations between artifacts, activities, and roles that structure the process (precedes, responsible-for)

- Intuitively: roles produce artifacts by performing activities
  - A coder is responsible for implementing module code as part of coding
  - A tester is responsible for writing test cases as part of verification
Process Definition Graphic

- **Roles:** who does the work?
- **Artifacts:** what is produced?
- **Activities:** what tasks are done?
- **Relationships:** how do processes vary?

**How do processes vary?**

- **Content:** processes vary in the specific activities performed, artifacts produced, roles required, and the relationships between these, for example:
  - Which specific activities are performed
  - Which role performs which activities
- **Formality:** processes vary in how detailed, complex, and prescriptive they are
  - How much detail is defined on the activities, etc.
  - How closely developers are required to follow the written process

**How do processes vary?**

- Emphasis varies on artifacts, types of artifacts, rules governing activities, gating, roles, for example:
  - Differ in form of requirements, design, test plan:
    - Written document, conforming to standard template, reviewed by peers and users using standard review process, benchmarked and configuration controlled
    - Notes on a web site
    - Knowledge in the heads of the development team
  - Differ in review procedures for documents and code:
    - Formalized inspections with criteria for passing, e.g., Fagan inspections or active reviews
    - Informal peer review meeting
    - Office mate reads it over
    - None
- Compare spiral to agile
Why do processes vary?

- Different processes reflect different assumptions about the developmental context and goals
  - Context: project size, complexity, availability of stakeholders
  - Goals: time-to-market, reliability, usability, maintainability, control of risk

- Process is something we can design to address project needs
- Must consider
  - What kind of process do we need: which kinds of activities, artifacts, etc. fit our goals and risks?
  - How much formality/complexity do we need?

Process Formality and Project Scale

- As projects grow larger and more complex, they typically require more formal and detailed processes
  - Difficult for individual developers, or management to track the overall state of the development
  - Difficult to keep track of who is supposed to be doing what ("Who do I talk to?")
  - Difficult to know when your job should be finished or what quality criteria it should satisfy

- A clear, well-defined process helps keep a large project coordinated

Distributed Development Contribution to Delay
When Process Complexity & Project Complexity/Scale Mismatch

- But, there can be too much process as well
  - Process is overhead
  - Unnecessary process overhead leads to problems
- Developers feel frustrated
  - "I want to write code, not documents"
  - "I can't understand what I'm supposed to do"
  - "I'm afraid to touch this code"
- Progress is slowed
  - "I have to wait for that other team to finish"
  - "I have to wait for my code to be inspected"
  - "We have an integration problem"

Prof. Einstein says...

Choosing a Process for DSD
Process Development

- Can view process development like software development:
  - Choose/create a process to address specific project needs and constraints
  - Think in terms of requirements, design, etc.
- Must ask the questions:
  - What are the key problems or risks of DSD?
  - What features of a process would help address the risks of DSD?
  - How much formality is needed?
    - I.e., how much detail and specificity about the artifacts, activities, roles and relations?

DSD Issues and Risks

- Key Problem: coordination at a distance
  - i.e., the key difficulty is getting all the people involved to do the right task the right way at the right time
- Key risk factors:
  - Restricted communication, flow of information
  - Different organization, language, culture
  - Lack of visibility into what remote teams are doing
- Potential difficulties:
  - Different views of the problem (requirements)
  - Different views of what the process is supposed to be
  - Misunderstanding of what remote teams are doing
  - Difficult to detect and correct problems
  - Difficult to manage synchronize the work
  - Difficult to detect and correct slips in schedule
Exercise: Chose a process

- Assume that you are part of a small company doing distributed development
  - Three teams of developers in USA, China, and Germany
  - Teams are 8-10 people in mixed roles (some coders, testers, etc. at each site)
  - Many team members have not worked together remotely before

- Discuss as a team
  - Each person must take turns contributing
  - Decide on team answer

- Determine which kind of process would be the best fit for a DSD project and how formal

- Hint: think about what happens over time

Team Exercise

Assume that you are the Product Manager for a small company doing distributed development. You have three teams of 6-8 developers in three countries (China, Germany, USA). Your boss has asked you to choose between a conventional waterfall process, an iterative process, and an agile process (Scrum).

As a team, choose which process you believe will best address the risks and constraints of a distributed development. Justify your answer to management by answering the following questions:

1. For each of the two processes you do not choose, briefly describe why you think the process is not a good choice (or not the best choice) for a distributed development. Specifically, describe which characteristics of the process are likely to cause problems in terms of the risks of DSD.

2. For the process you choose, briefly describe which characteristics of the process are likely to help address specific risks of DSD.

3. Characterize how formal the process should be: for example, should it be written down and very detailed or informally described?

Summary Co-located vs. DSD

<table>
<thead>
<tr>
<th>Co-located Development</th>
<th>DSD Risks*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free flow of information through informal means</td>
<td>Restricted flow of information, mostly formal</td>
</tr>
<tr>
<td>Shared process view</td>
<td>Possibly different process views</td>
</tr>
<tr>
<td>Clear idea of expertise, responsibility</td>
<td>Unclear idea of expertise, responsibility on remote teams</td>
</tr>
<tr>
<td>Common culture eases understanding</td>
<td>Possible misunderstandings due to cultural/language differences</td>
</tr>
<tr>
<td>Understand relationships</td>
<td>Vague or incorrect understanding of relationships</td>
</tr>
<tr>
<td>- People to tasks</td>
<td></td>
</tr>
<tr>
<td>- Task interdependencies</td>
<td></td>
</tr>
</tbody>
</table>

*Standardizing the process helps mitigate these risks a people fill roles with well-defined responsibilities.
Which process?

- Too little communication
  - Problems not detected until system generation

- Too much communication
  - Assumes daily meetings replace written documents

Lecture:
From Process to Project Plan

Incremental Development Over Time

- Acts as a feedback loop with a calibration point at each delivery
  - Allows cross checking of assumptions, understanding
  - Early check if remote sites are doing what is expected
  - Early check for communication effectiveness
  - Allows plan adjustments at each increment
Well-defined Process Benefits

- Process should also be relatively formal
  - Written down in detail
  - Required for all of the distributed sites
- Well-defined process clearly specifies
  - The artifacts to be produced
  - The set of activities that must be performed (e.g., specify requirements, review design, write code)
  - The set of roles (e.g., coder, tester, designer)
  - The relationships
    - Which roles perform which activities to produce which artifacts
    - The order of activities
    - Which artifacts are needed as input to produce which other artifacts

Well-defined Process Benefits

- Helps address risks
  - Everyone has common definition of the process
  - Assigning roles clearly defines responsibilities
  - Helps make clear what people should be working on
  - Helps make clear when a task is finished
- Should answer for individuals the questions
  - Is this my job?
  - What do I do next?
  - Am I done yet?
  - Did I do a good job?
- However: not enough just to define the process, must check that people understand and follow it.

Importance of Clearly Defined Roles

- DSD coordination problems arise from communication problems
- Lack of contextual information makes unclear
  - Exactly who knows what (who has expertise)
  - Exactly who is doing what (work allocation)
  - What questions or problems people have
  - What assumptions people are making
  - Etc.
Roles Help!

- Well defined roles provide a badly need structure
  - Define who is responsible for what
  - Gives guidance for expected expertise
- Relations between roles tell you
  - Who needs to talk to each other (e.g., shared responsibility, handoff, etc.)
  - What you need to be talking about
  - Provides bases for forming professional relationships
- Upshot: in DSD it is critical that
  1. Roles and their responsibilities are clearly defined
  2. Well defined lines of communication are established between roles at different sites
  3. People consistently perform the role’s responsibilities

Examples: Process Specs

- Release_06/Process Template Design Document.doc
- Release_06/GEN-001RevisionControl.html

Project Planning in DSD
From Process to Plan

- Process definition manifests itself in the project plan
  - Process definition is an abstraction
  - Many possible ways of implementing the same process
- Project plan makes process concrete, it assigns
  - People to roles
  - Artifacts to deliverables
  - Activities to tasks over time
- Project plan should be one of the first products but expect it to evolve
- For DSD, essential that distributed teams agree on the project plan

Project Plan

- Minimal plan contents
  - Tasks to be performed
  - Person(s) assigned to roles and tasks
  - Deadline for each task
  - Sequencing among tasks
  - Risks and mitigation strategies
- Usually owned by project manager
- Updated as project proceeds

Project Roles & Responsibilities

<table>
<thead>
<tr>
<th>Role</th>
<th>Number of team members taking on the role</th>
<th>Artifacts for which the role is responsible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Systems Engineer</td>
<td>1</td>
<td>Use cases, requirements, preliminary screenshots</td>
</tr>
<tr>
<td>Architect</td>
<td>1 or 2</td>
<td>Module tests, system structure, interface specifications</td>
</tr>
<tr>
<td>Developer</td>
<td>&gt;1</td>
<td>Module implementation, test results report</td>
</tr>
<tr>
<td>Tester &amp; Integrator</td>
<td>&gt;1</td>
<td>System generation and verification plan, test results report</td>
</tr>
<tr>
<td>Project Manager</td>
<td>1</td>
<td>Project plan, project measures, retrospective report</td>
</tr>
</tbody>
</table>
Work Breakdown Structure

This is a technique to analyze the content of work and cost by decomposing it into its component parts. It is produced by:
- Identifying the key elements
- Decomposing each element into component parts
- Continuing to decompose until manageable work packages have been identified. These can then be allocated to the appropriate person

The WBS is used to allocate responsibilities
For the software, the WBS depends on the software architecture (discuss next)

Example: work packages by project phase

Milestone Planning

Milestone planning is used to show the major steps that are needed to reach the goal on time
Milestones typically mark completion of key deliverables or establishment of baselines
- Baseline: when a work product is put under configuration management and all changes are controlled
- Often associated with management review points
  - E.g., Requirements baseline, project plan complete, code ready to test
- Can use Gantt or PERT charts, put milestones in Assembla
Gantt Charts

- Method for visualizing a project schedule showing
  - The set of tasks
  - Start and completion times
  - Task dependencies
  - Responsibilities

Example Gantt Chart

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
</tr>
</tbody>
</table>

Example Gantt Chart

http://www.spottydog.u-net.com/guides/faq/faq.html

DSD Project Plan

- Common project plan is key to coordination
  - Clear definition of roles and responsibilities
  - Clear dependencies between tasks hence, what needs to be done next
  - Provides basis for tracking progress

- Just one part of necessary communication!
  - Teams must agree on project plan but...
    - Still easy to have misunderstanding about meaning of plan
    - Still may go off track
      - Must detect and correct as soon as possible
      - This is not easy
    - Plan must be continuously updated

http://www.spottydog.u-net.com/guides/faq/faq.html
Plans vs. Reality
(Rational vs. Irrational)

Plan

Reality

Making The Reality Seem Rational

- Document as if it had been rational
  - Readers can follow a sequential story
- Explain significant changeable decisions
  - What alternatives were (are) there?
  - Why did we choose A'' rather than A or A’
- Later readers (maintainers) understand the trade-offs and can be guided by them

Planned
Actual w/ backtracking
Measurement

- Measuring progress is an important part of any real project plan
- Addressed in Prof. Zhou Minghui’s lectures

Summary

- Processes provide methods for managing software developments over time
- Must choose the right process to address a project’s specific problems and constraints
- Incremental process provide the feedback between distributed teams required for DSD
- Processes must be defined and realized in a project plan
- The project plan must evolve as the project progresses
Questions?

Break

Lab Exercise

- Use some of the assemble project management tools
- Assume we will do a small software development project with your distributed team
- Communicate with your team members to choose roles for each person
  - Requirements engineer, architect, project manager, coder, reviewer, tester
  - One person may have multiple roles and vice versa
  - Each person should try out two or three roles
- Add a milestone to define roles
- Add roles to your wiki page