The Role of Software Processes in DSD

1. Standup progress
2. Usefulness of processes in controlling DSD risks
3. Project planning

Questions We Encounter

• What do my clients want? What do they need?
• How long would it take? With what resources?
• Oops, something changed ... now what?
• Are we there yet? How much longer?
• How can we share this work? How can we coordinate?
• How did we do? How could we do better?

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?
  – What information is needed to do the work?
  – When is the work finished?
• Intended use
  – Guidance to developers in what to produce and when to produce it
  – Basis for planning and assessing development progress
• But, … different types of developments need different processes
  – i.e., different processes answer the questions differently
Formal Definition

• Need a consistent framework for representing and communicating about processes
• Defn: we define a process as set of artifacts, activities, roles and the relationships between them where:
  – Artifacts: any work product of the software development process
  – Activities: the tasks that produce the work products
  – Roles: skill set responsible for performing specific activities
  – Relationships: the relations between artifacts, activities, and roles that structure the process
• 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

Exercise: Which Model?

How do processes vary?

• **Content**: processes vary in the specific activities performed, artifacts produced, roles required, and the relationships between these, e.g.:
  – 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
**Why do processes vary?**

- Must understand this to deploy processes effectively
- 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
  - Primary risks: feasibility, cost, schedule, communication, etc.
- Process is something we can *design to address project needs*
- Must consider
  1. What kind of process do we need: which kinds of activities, artifacts, etc. fit our goals and risks?
  2. How much formality/complexity do we need?

---

**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?

---

**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?

---

**Exercise: Which Model?**

- [Diagram of different software development models]

- [Diagram of agile development process]

- [Diagram of Waterfall model]

- [Diagram of Iterative model]
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

Summary Co-located vs. DSD

Co-located Development

- Free flow of information through informal means
- Shared process view
- Clear idea of expertise, responsibility
- Common culture eases understanding
- Understand relationships
  - People to tasks
  - Task interdependencies

DSD Risks*

- Restricted flow of information, mostly formal
- Possibly different process views
- Unclear idea of expertise, responsibility on remote teams
- Possible misunderstandings due to cultural/language differences
- Vague or incorrect understanding of relationships

*Standardizing the process helps mitigate these risks.

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 activates 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.
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.

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
• For DSD, it is essential that distributed teams agree on the project plan

Project Plan Template

• Use the template provided in your Assembla team workspace (under the Wiki tab)
• This should be a *living document*
  – Changed as the project progresses
  – For the reader, reflects both the planned activities and provides a snapshot of the current project state

On the 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 their role’s responsibilities

Examples: Process Specs

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

Questions?