The Role of Software Processes in DSD

DSD Team

Outline

Lecture: software process
Review: usefulness of software processes
Fitting processes to development problems
Considerations in choosing an effective process
Implementing a process in a project plan
Exercise: Which process best fits the goals, problems, and constraints of DSD?
Team Exercise: First Contact

We need a plan ...

How long? With what resources? In what order?
Who does what? How do they coordinate? What about changes?
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

Characteristic Processes: The Waterfall Model

Process viewed as a sequential set of activities Imposes separation of concerns on software development activities What is a fundamental assumption of this model?
Characteristic processes: The Spiral Model

Process viewed as repeating cycles of increasing scale
Identify risks & values & determine (next set of) requirements, build next version by extension,
increasing scale each time
Early iterations may be prototypes
Key concept: Risk control

Characteristic processes: Iterative development

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

Question: How does this differ from the spiral model?

An iteration is a distinct sequence of activities based on an established plan and evaluation criteria, resulting in an executable release (internal or external)

IBM

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

Need a consistent of representing and communicating about processes
Defn: we define a process as set of artifacts, activities, roles and the
relationships between them* where:
Artifact: 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

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
  - Officemate reads it over
  - None

Common Objectives

Predictability
- Make a plan with estimates of time and resources
  - Even if the plan is "we can quit whenever we want"

Visibility
- Check progress against the plan

(De)composition
- Break the work down into work assignments that can be assigned to individuals; compose the parts into a working system

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 Formality and Project Scale

As projects become more difficult to control, they typically require more formal and detailed processes.

- i.e., project complexity, size, duration, distributed, etc.
- 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 the project coordinated.

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.”

Prof. Einstein says...

- “Everything should be made as simple as possible, but not simpler.”
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 addresses 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
Team Exercise

Assume that you are the Product Manager for a small company doing distributed development. You have three teams of 8-10 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: Choose 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 group (3-4 people)
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

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 a
Globally Distributed Development

Which process?
- Too little communication
- Problems not detected until system generation
- Too much communication
- Assumes daily meetings replace written documents

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

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.

Examples: Process Specs

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

Break