The Role of Software Processes in DSD

CIS 423/523

Outline

• Lecture: software process
  – Review: usefulness of software processes
  – Fitting processes to development problems
  – Considerations in choosing an effective process for DSD
  – Implementing a process in a project plan

We need a plan ...

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 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

- Need a consistent way of representing processes
- Defn: we define a process as a 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

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

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”
  - “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
Summary Co-located vs. DSD

Co-located Development
- Free flow of information through informal means
- Shared process view
- Clear ideas 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 as people fill roles with well-defined responsibilities

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 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.

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

Project Plan

- Minimal plan contents
  - Clear role definitions
  - Risks and mitigation strategies
    - Evolves with progress and understanding
  - Tasks to be performed
  - Schedule
    - Person(s) assigned to roles and tasks
    - Deadline for each task and key milestones
    - Sequencing among tasks
      - Task dependencies
      - Gating (quality control: e.g. review and approval or testing)
  - Usually owned by team leads (one at each site)
  - Updated as project proceeds

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

Assignments

• Friday: project proposal due (goal is to fold this into the ConOps)
• Monday: initial draft of project plan
• Will grade interim deliverables as homework
  – Note that you should coordinate the tasks if possible, but go ahead and do your part when due

Questions?