The Role of Software Processes in DSD

DSD Team

Outline

- Review: usefulness of software processes
- Fitting processes to development problems
- Choosing a process for DSD
- Process in the project context
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 gaining and maintaining control over 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
  - Provide guidance to developers in what to produce and when to produce it
  - Provide a basis for planning and assessing development progress
Definition

- Process: we define a process as set of artifacts, activities, roles and the relationships between them
  - Roles produce artifacts by performing activities
  - A designer produces a design document as part of creating the design
- Will write the project plan in these terms
- Two basic questions
  - What kind of process is appropriate to DSD?
  - How formally (detail, precision) must it be defined?

Characteristic processes: The Waterfall Model

- Process viewed as a sequential set of activities
- Determine requirements, analyze & design, code, test, release
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

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

What is an Iteration?

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

Variability in Processes

- Emphasis vary on artifacts, types of artifacts, rules governing activities, gating, roles, for example:
  - 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
  - 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
  - Release criteria
  - Coding style
  - Roles: Project manager, systems engineer, architect, developer, tester
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

- Rules of thumb
  - The larger the project, the more formalized the process
  - The more critical the project, the more formalized the process
  - Avionics, medical software, automobile control

Compare & Contrast

**Waterfall Model**
- Goals
- Assumptions

**Agile model (e.g., scrum)**
- Goals
- Assumptions
Process Complexity and Project Complexity/Scale

- On large projects and complex projects, process helps assure that:
  - Requirements are feasible
  - Requirements express what the customer/market wants, or what the organization wants to build
  - Work assignments are properly apportioned
  - Work assignments result in code that works
    - Individual module correctly implements its interface
      - Interface specifier may not be implementer
  - Work assignments result in code that works together
    - Modules work together to produce the desired result, i.e., requirements are met
  - Teams at different sites understand the interactions among their work and the work at other sites
    - Our code uses yours, your code uses ours, you test what we do, etc.
  - Work assignments can be completed in time for the project to meet its release date(s)

Predicting Software Development (50 sampled projects)
Distributed development, innovation, new features, legacy adaptation all contribute to delays.
### When Process Complexity & Project Complexity/Scale Mismatch

- 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”

### Discussion: DSD Process Requirements

- What are some of the key risks of DSD?
- How might a well-defined process help address these risks?
- What kind of process is needed?
- How much formality is needed?
  - I.e., how much detail and specificity about the artifacts, activities, roles and relations?
Software Development Problems

- Typical problems in coordination and control of distributed software development
  - Difficulty establishing requirements (eliciting, understanding, negotiating)
  - Difficulty detecting and correcting conflicting assumptions
  - Difficulty detecting and correcting errors
  - Difficulty detecting and correcting slips in schedule
  - Difficulty managing change (especially requirements)

- Developmental risks
  - Deliver the wrong software (not what stakeholders wanted)
  - Deliver incorrect software (buggy)
  - Deliver software and other artifacts late

- Which are the most significant risk for your projects and what does this imply about the process?

Appropriate Process

- Primary risk is the immutable deadline, may not deliver working software before course ends.
- Secondary risk is required artifacts may be missing or incomplete
- Suggests some form of incremental development
  - Address risks in stages
  - Always have working version
  - Always have useful artifacts
- But, how formal should it be? i.e., more like scrum or more like spiral?
# 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

---

# Well-defined Process Benefits

- 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 goo job?
- However: does not mean everyone necessarily understands the process definition, must address this risk!

Project Relevance

- Process definition manifests itself in the project plan
- Maps roles to people and activities to dates
- 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 Planning

Project Plan

- Include each of these four sections (template):
  - Project organization: People and roles
  - Risk analysis, risk reduction strategies
    - Includes description of the process chosen and why
  - Work breakdown (>10 milestones) and project schedule (who will do what and when)
  - Monitor and report progress:
    - Keep a record of: Each task, assigned to whom, when assigned, due date, when completed, who did it, who signed off on it
    - Measurement Plan: specify what will be measured and how it will be reported (e.g., GQM)
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)
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

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
Projects

- To simplify startup, some decisions already made
  - Planned two major increments
  - Provide templates for project plan and some artifacts
- Allows you to focus on communication issues (the most difficult part)

Importance of Clearly Defined Roles
DSD Difficulties

- Coordination problems arising 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.

Need for Communication

- Best way to mitigate risk is
  *Frequent and consistent communication!*

- Helps make up for lack of informal communication pathways
- Difficult to do
  - May not know other party well
  - Mechanisms are relatively low bandwidth, high effort
  - Difficult for many technically oriented people
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

Questions?