Introduction to Distributed Software Development

DSD Team

Outline

- Admin
  - Course purpose
  - Projects and teams
- Lecture: Introduction to Distributed Software Development (DSD)
Course Format

- Ten week project course
  - Lectures: Foundations and background
  - Projects: Learn how to apply SE concepts to globally distributed software development
  - Collaborative development: work remotely with computer science students at Peking University
  - Project Reviews and Presentations: Critique and guidance
  - Course assessment: help improve course

- One project, two (large) iterations
  - First iteration, focus is to develop teams, learn tools, establish communication
  - Second apply skills to globally distributed development

Software Engineering Emphasis

- Emphasis on life-cycle management and teamwork in a globally distributed context
- Understand coordination and communication challenges of distributed development
- Participate in planning and coordination of distributed teams
- Learn effective software engineering techniques to address DSD risks
- Learn effective communication and collaboration skills supporting distributed teamwork

© S. Faulk 2009
Projects

- Two iterations of the Schematic Map project, each 5 weeks

Iteration 1: instructor guided
- Instructors will provide most of the supporting infrastructure such as detailed requirements, document templates
- Focus on developing effective collaboration-at-a-distance
  - Learn to use tools
  - Develop teams and teamwork skills

Iteration 2: team guided
- Students negotiate and define requirements for increment
- Work collaborative to deliver software on time
- Complete product including documentation and tests

Teams

- Team size depends on class size
- Teams will include both UO and PKU students
Fix before class
Stuart Faulk, 2/15/2010
Course Evaluation & Development

- Experimental course designed to be a model for distribution
  - New course for the instructors as well as students
  - Goal is to develop a model course
    - Create lectures, projects, other course materials
    - Provide as "open source" materials for other universities to give similar collaborative courses

- Hence, assessment is part of the course
  - Required to do formal assessments of your learning
    - Will have evaluations
    - Will track interactions (e.g., team communication)
  - Need to get your feedback on what needs improvement

© S. Faulk 2009

Questionnaire

- Purpose
  - Formation of balanced Iteration 1 teams
  - Beginnings of evaluation database
  - Introduction to teammates
Weekly Schedule

- Monday and Wednesday lectures
  - Mix of lectures, discussions, group exercises
  - Some lecture times or parts thereof will be used for team meetings and project discussions

- Meetings with the professor
  - Design reviews
  - Progress reviews
  - Course assessment

Introduction to Distributed Software Development

Purposes of DSD
Issues and Risks Introduced
The Role of Software Engineering
Growth of DSD

- Increasingly used by companies of all but the smallest size
- Outsourcing growth by an order of magnitude over past few years
- Over half of the Fortune 500 companies
  - Software industry: Google, Microsoft, IBM, etc.
  - Other industry: telecom, cell phone, etc.
- If you work in industry, you will likely work in a distributed team

Sample of Development of 52 Major Projects at Avaya Corp. 2006

Approximately 75% multi-site development and growing
Rationale for DSD

- Expanded pool of trained workforce
- Necessity of getting closer to customers and using locality specific expertise to customize/localize products
- National policy in some countries where the government may be a customer requiring suppliers to locate R&D facility in that country as a condition of sale or a favorable tax treatment
- Difference in development costs
- Promise of round-the-clock development that could lead to shorter intervals
Benefits Often Come at a Cost

- With DSD benefits come increased risks compared to similar co-located developments
- Schedule delays – same work takes longer
- Higher risk of failure
- Reduced product capabilities
  - Decreased functionality, qualities
  - Doesn’t meet some customer requirements
- Increased cost
  - May cost more in spite of lower labor costs
  - Schedule delays and rework increase costs

Observed Difficulties (1)

- Nature of a software project
  - Software development produces a set of interlocking, interdependent work products
    - E.g. Requirements -> Design -> Code
  - Implies dependencies between tasks
  - Implies dependencies between people
- Successful development requires effective coordination between people and tasks!
  - Must coordinate work (need product A to produce product B)
  - Must coordinate schedule (must finish A before starting B)
  - Must coordinate people (person P has expertise need to produce A but is busy)
Observed Difficulties (2)

- Key property distinguishing DSD from co-located development
  
  "The key phenomenon of DSD is coordination over distance." – J. Herbsleb (2007)

- All software projects require coordination
- Suggests that coordination at a distance is different
- Managing these differences is a central issue in DSD

How is DSD Different?

- In co-located projects, people build up ways of coordinating work
  
  - Shared process view (implicit or explicit)
  - Common vocabulary, viewpoint
  - Clear idea of expertise, responsibility
  - Free flow of information through informal channels
  - Common language, culture, backgrounds help avoid miscommunication
  - Relatively good understanding of relationships
    
    - People to tasks
    - Task dependencies
    - Professional and social
DSD is Different…

- In DSD many of the mechanisms for coordinating work are absent or disrupted
  - Much less communication
    - Temporal distance
    - Socio-cultural distance, e.g., language
    - Spontaneous communication declines rapidly with distance
  - Less effective communication
    - Fewer overlapping work hours
    - Low bandwidth links (e.g., email and other asynchronous)
  - Lack of awareness
    - Lack context hence knowledge of history, relationships
    - What people are doing day to day, concerns, availability
  - Incompatibilities
    - Differ in tools, processes, work products
    - Leads to confusion, misunderstandings, inconsistencies
- Some issues are observed with even small distances (e.g., 30 meters)

© S. Faulk 2009

Software Development Problems

- Manifests as problems in coordination and control of software development
  - Difficulty establishing requirements (eliciting, understanding, negotiating)
  - Difficulty effectively distributing work
  - Difficulty detecting and correcting conflicting assumptions
  - Difficulty detecting and correcting slips in schedule
  - Difficulty managing change (especially requirements)
  - Difficult managing development resources (schedule, personnel, budget)
- Similar to traditional SE problems, but more so
  - Work takes longer
  - Requires more effort

© S. Faulk 2009
Useful to View as Risks

- Examples of increased project risks due to communication and control difficulties
- Building the correct software (behavior or qualities)
  - Fully understand (or misunderstand) the requirements
    - Developers build the wrong thing
    - Inadequate verification, validation
  - Decomposing the work into work assignments that together address behavioral and developmental goals
  - Ensuring each team builds what is needed by others
- Managing development (budget, schedule, personnel)
  - Balancing workload, skills across sites to meet developmental and quality goals
  - Developing common understanding of who should be developing what in which timeframe
- We will examine risks in detail as we develop projects
- Fundamental issue is how to mitigate DSD risks

The Role of Software Engineering

Brief overview of relationship of SE to managing developmental risks.
Purpose of SW Engineering

The *purpose of software engineering* is to gain and maintain intellectual and managerial control over the products and processes of software development.

- “Intellectual control” means that we are able make rational technical choices based on an understanding of the downstream effects of those choices
- Managerial control means we are able to make rational choices about development resources (budget, schedule, personnel) to deliver software on time and within budget

Application to DSD

- Risks: in DSD communication difficulties and context difference lead to coordination and control problems
- Approach: apply software engineering processes, methods, and tools to mitigate risks
- Examples:
  - Software processes
  - Software architecture
Process Definition

- Process: A set of artifacts, activities, and roles
  - Activities produce artifacts and are performed by roles
  - A designer produces a design document as part of creating the design

- The process determines
  - What activities are done
  - The order in which they are done
  - Who does them

- Different processes address different goals and risks

Characterizing 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
Characterizing 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)

Discussion: Role of Process

- How can process help address DSD risks associated with communication and control?
  - How might it help to have a well-defined process?
    - Sequence of activities?
    - Clearly delineated roles?
    - Well-defined products?
  - How might the choice of process help?
Software Architecture

Working Definition

- “The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them.” *
- Architecture divides the system into parts and defines
  - The relationships among the parts
  - The interface for each part
- Familiar in terms of decomposing a system into classes/objects
  - Defining relations, e.g., inheritance, calls
  - Defining interfaces, e.g., service

*From Software Architecture in Practice, Bass, Clements, Kazman
Fit in the Development Cycle

...The earliest artifact that enables the priorities among competing concerns to be analyzed, and it is the artifact that manifests the concerns as system qualities.

Influence of Architecture on DSD

- How can architecture help address risks associated with DSD?
- How might the decomposition into modules with well-defined interfaces affect the need for communication?
  - How does software decomposition influence communication?
  - How does software decomposition influence work breakdown structure?
How should components be distributed among teams?

Goal: distribution of work on components requiring least inter-team communication

![Diagram showing distribution of components among teams A, B, and C.]

Some SE Topics Covered

- Topics presented (roughly) in order needed for projects
- Software processes and project planning
  - Which life cycle model to use
  - What kinds of activities and roles support distributed development
  - How to allocate work to distributed teams
- Software architecture
  - How to use architecture to manage dependencies
  - Which design principles and methods support concurrent development
  - Role of interfaces as contracts
- Software requirements
  - How to ensure mutual understanding of expected behavior and system qualities
- Software verification and validation
  - The role of feedback in maintaining control
  - Effective reviews
  - Distributed testing
The Development Environment

Effects of the development environment
- Incompatible development infrastructures can result in inconsistencies between collaborating teams
- Common environment can support communication and coordination

Our projects will use a common environment (Assembla)
- Provides common development workspace
- Tools for synchronous and asynchronous communication
  - Email, IM
  - Wicki, document management
- Source control
- Some scheduling
Required for Collaboration

- Use of Assembla tools is required for collaboration
- Ensures common tools
- Gives instructors visibility into emerging difficulties
- Provides a record for measurement and assessment

Questions?