Introduction to Distributed Software Development

Stuart Faulk & Michal Young
CIS
University of Oregon

Lian Yu
School of Software and Microelectronics
Peking University

Outline

- Challenges of Distributed Software Development (DSD)
- Course structure and goals
  - Course plan
  - Project

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 acculturate products
- National policy in some countries (government may require suppliers to locate R&D facility in that country as a condition of sale or favorable tax treatment)
- Difference in development costs
- Promise of round-the-clock development that could lead to shorter intervals

Evolution of DSD

Global Sourcing is Here to Stay and is Evolving

- Offshore labor primarily in India
- Labor in mature geographies around the globe
- Diffusing private (cheaper labor) to broader talent pool and using R&D technology
- Numerous technical Analyzers
- R&D strategic outsourcing vendors
- Prolonging maintenance
- New development lifecycle, design and test
- Prolonged technical Programming
- Consulting, focused process outsourcing, valued and integrated
- Global delivery is a specialty
- Global delivery (base) is the standard

IBM Global Initiative
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

Working Definition

- Distributed Software Development (DSD): teams in geographically distant locations collaborate to produce the work products of a software development
- Synchronize in phases of the life cycle
- Collaborate on artifacts from requirements to code
- Coordinate activities among members of distributed teams

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
- Consider your 422/522 experience

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)

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)
  - Difficulty managing development resources (schedule, personnel, budget)
- Similar to traditional SE problems, but more so
  - Work takes longer
  - Requires more effort

© S. Faulk 2010
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
    - 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 we will address is how to mitigate DSD risks

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

Course Structure
Approach and Rationale

- Simulate DSD
- Similar to way in-class project simulates co-located development
- Collaborate with students at Peking University on a software project
  - Each team will be roughly half UO, half PKU
  - Work together on a software application
  - Collaborate over the web to create, review, and present the results of development
- Learn to apply SE principles, methods and tools to support long-distance collaboration

Remote Collaboration

Beijing Summer School
Weekly Schedule

- Three class meetings a week
  - Mix of lectures, discussions, group exercises
  - Some lecture times or parts thereof will be used for team meetings and project discussions

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

Resources

- Class web page:
    - Syllabus: updated with assignments, lecture slides, and class videos
    - Project: documents and activities related to the project
    - Tools: descriptions of the development and communication tools we will use
    - Readings: download reading assignments
    - Announcements
    - FAQ: ask questions or find answers about the project, lectures, or anything else.

- Assembla workspaces
  - On-line collaboration environment for project teams

Teams

- Will try to have three to five teams of UO/PKU students
- Goals and expectations
  - Learn SE skills by encountering and overcoming DSD problems in simulation
  - Learn teamwork across cultural boundaries
  - Active participation with your whole team is critical to this
    - It is expected that you will fulfill the duties of your role(s) in your team by actively collaborating with your teammates
    - Interacting with your teammates at PKU especially important
  - Also need student liaison to work with student at PKU (student-to-student)
    - Communicate about course issues
    - Help us provide resources and help needed
Student Evaluation

- Primarily interested in what you learn about software engineering and teamwork
- Rough decomposition
  - Quality of development artifacts: 30%
  - Quality & functionality of code: 20%
  - Project management & communication: 15%
  - Teamwork and participation: 20%
  - Written evaluations (exams): 15%

Assignments

- Familiarize yourself with the class web pages
- Fill out team formation survey and bring to next class
- Reading: *Cultural Surprises in Remote Software Development*

Questions?