Introduction to
Distributed Software Development
CIS 423/523

Stuart Faulk
Spring 2016

Outline

• Rationale for Distributed Software Development (DSD)
• Software engineering challenges of DSD
• Course structure and goals
• Team formation
Growth of DSD

• Software development by globally distributed teams 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
  – Also true of academic research!

Rationale for DSD

• Expanded pool of trained workforce
• Getting closer to customers and using local expertise to acculturate products
• National policy (regulatory locality requirements)
• Difference in development costs
• Promise of around-the-clock development that could lead to shorter intervals
Benefits Come with Risks

- 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
  - Does not 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 across 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
Probability of Technical Communication as a Function of Distance Between Work Stations

Source: Prof. Tom Allen, MIT Sloan School of Management Executive Programs
Informal Communication Pathways

- In co-located projects, people build up informal 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 misunderstanding
  - Relatively good understanding of relationships
    - People to tasks
    - Task dependencies
    - Professional and social
- Consider your CIS 422 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
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 intense
  – Work takes longer
  – Requires more effort

Useful to View as Risks

• Examples of increased project risks due to communication and control difficulties
• Risk of building the wrong software (behavior or qualities)
  – Misunderstand the requirements
  – Miss requirements or fail to address them
  – Functions needed by distributed team members not implemented or implemented incorrectly
• Management risks (budget, schedule, personnel)
  – Balancing workload across sites
  – Developing common understanding schedule, sequencing
• 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
Objectives

• Learn methods for, advantages of, and difficulties of working in a distributed team, similar to industry experience
• Work successfully in a team whose members are distributed among UO, Jilin University, Peking University, and Universidad Nacional de Colombia
• Produce a working application as part of a cooperative project with remote universities
• Learn to apply SE principles, methods and tools to support long-distance collaboration

Remote Collaboration
Industrial Scenario

• You will be part of a globally distributed team joining a large project already in progress
  – Will form teams with UNC, Jilin, and PKU students
  – Will develop one or more components of a larger piece of software
  – Will participate in integration
  – Will participate in global QA activities (review, test) to ensure quality
• Exact activities will vary depending on which part of the project each team selects
• Goals and expectations
  – Learn SE skills by encountering and overcoming DSD problems in simulation
  – Learn teamwork across cultural boundaries

Project: Remote Authentication

• Client sets up authentication protocols
  – E.g. train face recognition, collect data
• Authentication system controls access to clients application (e.g. testing services)
• User begins by providing picture with logon
System uses facial-recognition applications to authenticate user identity

http://mashable.com/2016/03/14/amazon-selfie-payment/#BOc2yFsmsPqd
• Conceptual view of system
• What do I need to do to have this developed globally?
Progress and Schedule

- Requirements
  - Use cases, models
  - Review/validation
- Design
  - Architectural views
  - Interface specification
  - Reviews
- Detailed design
- Implementation
- Unit testing
- Integration testing

Choosing Teams

- Will have collaborative teams (½ each from two schools) and some local
  - Too many students to have every team collaborative
- Teams needed
  - 2 teams of 3 or 4 students to work with UNC
  - 1 team to work with JLU (~3)
  - 1 team to work with PKU (~3)
  - 3 to cover the remaining modules + integration
- Collaborative teams will work on client apps
Interest Survey

• Remote team needs and project interests
• UNC (Columbia) ~3 each
  – Testing authentication (Java API): 2 developer, architect/integration, documentation, CM
  – Android client: developer, architect/integration, CM
• JLU (Changchun): tester, architect
• PKU (Beijing) ~3: TBD

Progress Reporting

• Each team will give a progress report midway through the semester, and each team will demonstrate its product at the end of the semester.
• Each team will meet often with the instructor during the semester to discuss progress.
• Each team will hold a retrospective at the end of the semester and produce a retrospective report.
Weekly Schedule

• Two 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 instructor
  – Small group discussion of SE and project ideas
  – Design reviews
  – Progress reviews
  – Course assessment

Student Evaluation

• Primarily interested in what you learn about software engineering and teamwork
• Rough decomposition
  – Quality of development artifacts: 35%
  – Quality & functionally of code 20%
  – Project management & communication 15%
  – Teamwork and participation 20%
  – Written evaluations 10%
Questions?

Evolution of DSD

Global Sourcing is Here to Stay and is Evolving

Evolving From...  |  Evolving To
--- | ---
Offshore labor primarily in India | Labor in multiple geographies around the globe
Offshoring provides cheaper labor | Offshoring provides efficient access to a larger talent pool and leading-edge technologies
Numerous tactical vendors | 3-5 strategic outsourcing vendors
Predominantly maintenance | New development lifecycle, design and test
Predominantly technical programming | Consulting, business process outsourcing, SQA, and infrastructure
Global delivery is a specialty | Global delivery (chain) is the standard

Global delivery will continue to grow and evolve—away from a simple cost play for lower value services

*IBM Global Initiative

© S. Faulk 2010
Globally Distributed Software Development

Projects

• Phase 1
  – Initial definition of requirements, then design, accompanied by prototyping (as needed)
  – Projects will be based on a core set of requirements and modular design
  – Produce documentation and code sufficient to convey project understanding to remote site
  – Prepare for collaboration-at-a-distance
    • Learn to use tools
    • Develop teams and teamwork skills
• Two iterations of the project, each 5 weeks
  – Distributed teams cooperate to define extensions to core requirements and design, and to implement and test
• Iteration 1: instructor guided
  – Instructors interact closely with teams
  – Focus on developing effective collaboration-at-a-distance
• Iteration 2: team guided
  – Students negotiate and define requirements for increment
  – Work collaborative to deliver software on time
  – Complete product including documentation and tests