Outline

- Rationale for Distributed Software Development (DSD)
- Software engineering challenges of DSD
- Course structure and goals
  - Course plan
  - Student resources
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 round-the-clock development that could lead to shorter intervals
### Evolution of DSD

Global Sourcing is Here to Stay and is Evolving

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

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

*IBM Global Initiative*

© S. Faulk 2010

### Benefits Come with Risks

- What could go wrong? 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

© S. Faulk 2010
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 (not offshoring)
  - 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
- Experience suggests that coordination at a distance is different
- Managing these differences is a central issue in DSD

Distance Measurement

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 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

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
  – 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

Approach and Rationale

• Goal: gain real experience with problems by simulating an industrial DSD project
  – Similar to way in-class project simulates co-located development
• Collaborate with students at another university on a software project
  – 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

Collaboration Goals

• 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 remote site especially important
• Will try to have 5-6 teams on different projects

© S. Faulk 2010
Current Status and Issues

- Two collaborating schools
- Peking University
  - Traditional collaboration partner
  - Only 5 students this year
- Kean University (US, China)
  - US campus, international student body
  - New to DSD
  - Semester ends in May 15
- Will need to patch together collaborations

Likely Projects

- Data analysis and visualization (insurance company) with PKU
  - Real application from business client
  - Currently collecting requirements
- Machine learning and translation: analyze accuracy of Google translation of different languages (Kean University)
  - Prior experience with algorithms
  - Would add new versions with analysis, visualization
Risk Mitigation Strategy

• Goal is to have everyone have some experience with distributed development
• Different project structures
  – PKU collaboration
    • 8-9 week project
    • Likely 2 teams (Controller, View)
  – Kean collaboration
    • 5 week project
    • 4 teams
    • May continue development or switch for 2nd half
  – Risk mitigation option: simulate distribution within class

Assignments

• Goal is to settle collaboration plans this week
  – Team meetings by week end
• Fill out team formation survey
• Identify Liaison(s)
• Reading (by Thursday): Cultural Surprises in Remote Software Development
Resources

• Class web page:
  – Class web pages are in progress and will be updated ASAP
  – Will track current assignments

• Assembla workspaces
  – Have created work spaces and will assign as teams are created

Student Evaluation

• Primarily interested in what you learn about software engineering and teamwork
  – Will adjust for risks, likely problems

• Rough decomposition
  – Quality of development artifacts: 30%
  – Quality & functionally of code 20%
  – Project management & communication 15%
  – Teamwork and participation 15%
  – Written evaluations/tests 20%
Questions?