Research in Distributed Software Development

Review & discussion of readings
Readings

• Assigned
  – Culture Surprises in Remote Software
  – Lessons Learned by DSD Participants
  – Stakeholders in Global Requirements Engineering
  – Challenges of Global Software Development
  – Distance Matters

• Other
  – Conway’s Law Revisited (Herbsleb & Grinter)
Why (globally) distributed development?

• Sometimes to reduce cost
• Often for other reasons
  • Special expertise
  • Local knowledge
  • Policies and laws
  • (24-hour cycle? Maybe not, according to Komi-Sirvio and Tihinen in “Lessons learned ...”)

[188x515]Why (globally) distributed
development?

• Sometimes to reduce cost
• Often for other reasons
  • Special expertise
  • Local knowledge
  • Policies and laws
  • (24-hour cycle? Maybe not, according to Komi-Sirvio and Tihinen in “Lessons learned ...”)

[280x468]• Some@mes to reduce cost
• O2en for other reasons
• Special exper@se
• Local knowledge
• Policies and laws
• (24‐hour cycle?  Maybe not, according to Komi‐Sirvio
and Tihinen in “Lessons learned ...”)

[83x413]• Sometimes to reduce cost
• Often for other reasons
• Special expertise
• Local knowledge
• Policies and laws
• (24-hour cycle? Maybe not, according to Komi-Sirvio and Tihinen in “Lessons learned ...”)
We brought this on ourselves

• “Maturation of the technical infrastructure has enabled ...”

– Possible corollary: Distributed knowledge work appeared early in software development, but will spread to other fields. ??
Basic Challenge

• **More** communication needed ...
  • To deal with differences in corporate and national culture, misunderstanding, visibility problems in distributed teams, ...

• **Less** communication possible ...
  • Communication slowed by language differences, time zone differences, limited media, less familiarity
  • Informal communication (the lunch time chat) particularly missing
We can apply the research to ...

• Ameliorate the limitations ...
  • The better we understand what the problems are, the better our chance to address it
  • Example: “awareness” technology in version control
  • Example: team-building meetings at outset of a project

• Adjust to the limitations ...
  • By choosing appropriate processes and team organization
  • By choosing appropriate software organization
Aside: Sir David King at ICSE 2010

• Keynote on global warming and resource use ... ironically to a room full of people who flew thousands of miles to Cape Town
  • We’ve got to quit meeting like this ...
• Question to King: Replace meetings with teleconferencing?
  • anecdote on high-end telemeeting:
    Forgot screen, got up to shake hands
  • but even this addresses only one of the barriers
    (as noted by Olson and Olson)
Software development processes

• Software processes are designed.
  • They have goals and constraints. What are some of the dimensions of design decisions?

• What aspects of process design does global development impact?
  • Example: Consider an agile method like XP or Scrum. What are its goals and constraints? What design tradeoffs have been made? How does global distribution impact those tradeoffs?
Software architectural design

• What does globally distributed development have to do with the architectural design of a software system?
  • Anything different, in kind or in degree, from a software system developed by a co-located team?
  • Anything different in how the architectural design is recorded and communicated?
  • How does this relate to the challenges described in the papers we read?
Conway’s Thesis

• The structure of a system reflects the structure of the organization that builds it

(What does that mean?)
Organization Structure

- Organizations are hierarchically structured
- There are interfaces (interactions) between their parts
- This includes organizations that design and build software systems
(Software) System Structure

- Systems are hierarchically structured
- There are interfaces (interactions) between their parts

Architectural design ("software architecture") comprises several relations; "part of" and "uses" are two of them
The “responsible for” relation

- Each module or subsystem is assigned to some part of the organization
  - One part of the organization may be responsible for zero or more modules
Conway’s observation: System structure reflects organization

Interface between team A and team B

Interface between the modules they design
Why?

• Interfaces between modules (and subsystems) are interfaces between people
  – We must communicate (negotiate) to design the interface between our respective modules
  – We each depend on the other obeying the interface contract
Suppose we violated the law ...

Little or no interaction between us ...

... but our modules share an interface. So who do I call when it doesn’t work?
Information Hiding and Conway’s Law

• Information hiding says: each module holds a design secret
  – “Secret” may change without affecting other modules
  – Abstract interface does not expose the secret

• Translation: We have to talk about the interface, but not module internals

![](Image)
Process and Conway’s Law

• The *build order* of a project should allow everyone to make and measure progress
  – Avoid making some wait for others

• Key milestones are in terms of available functionality at interfaces
Distributed Development and Conway’s Law

• Distributed teams magnify all the issues
  – Even more important for an interface to be stable, abstract, well-documented
  – But there will be questions, bugs, maybe even changes; module interfaces are still human interfaces
Technical Fixes

• What problems of GSD can be addressed by technology?
  • How?

• What problems are likely to persist regardless of technology?
  • Are there other ways to address them?
Summary

• Understanding the challenges of distributed teams
  • Distance still matters. So do differences in corporate and local culture.
  • Some problems (e.g., slow networks) will get better on their own, and some (e.g., awareness) can be addressed with technology, but some (e.g., time zones, cultural differences) are essentially non-technical.

• From understanding, better design
  • Of the software, and of our processes for building it