Socio-technical Implications of Software Development

Giuseppe (Peppo) Valetto
Department of Computer Science
Drexel University, Philadelphia, PA - USA
Prologue
Introductions: who I am

- Giuseppe (Peppo) Valetto
- 15 years of experience as a researcher
  - Master in CS – Columbia University (1994)
- Worked for:
  - Xerox research
  - CEFRIEL – Politecnico di Milano
  - Telecom Italia Lab (TILAB)
  - IBM research
- Joined Drexel Computer Science faculty in September 2007 as an Assistant Professor in Software Engineering
My research interests

- Autonomic systems
- Engineering of self-adaptive traits in software systems
- Goal: self-management of complex distributed systems and applications

- SW process and collaboration
- SW development organizations
- Interplay between software design and software teams
My software engineering research

- I started studying environments and tools for cooperative development in the 90s
- We thought it was important to support teams developing SW
- Process-oriented SW Eng. environments (PSEEs)
  - They would support a prescribed process across one or more (possibly distributed) development teams
- The dominant idea at the time was that of the *Software Factory*
  - Focus on enforcing a pre-determined process, via tool support and automation
  - Tame complexity with healthy doses of determinism
What happened instead

- Research and practice on development tools diverged
- Focus on individual developer productivity

**IDEs:**

- VisualStudio for MS development
- NetBeans for Java
- Finally, Eclipse from Rational/IBM

- IDEs push the teamwork dimension to the background
  - data integration

- IDEs are process-agnostic
  - Nothing is prescribed or enforced
The importance of teamwork for SW development

- Teamwork concerns become essential when scale and complexity increase
- Success of a project is not achieved only through technical prowess of the individual developer

Recent emphasis on teamwork:

- **Agile methods**: aim at engineering better teams, not only better products
- **Globalized software development**: geographic barriers between people impact efficiency and quality of process
- **Open source development communities**: possibly the largest disrupting phenomenon in the SW industry
Some Research Questions

- How do development teams work?
  - Study structure and dynamics of teams
  - Situated in different contexts, practices, and process models
- Understand how the technical and the organizational dimensions influence each other
  - How can you best organize the structure of a team?
  - What is the mutual influence with the structure of the SW design?
  - What is the reason of efficiency? How do you find and correct inefficiencies? How much can that cost?
- For more:
  [https://www.cs.drexel.edu/~valetto/Peppo_Drexel/SE_research.html](https://www.cs.drexel.edu/~valetto/Peppo_Drexel/SE_research.html)
Introduction
(so … what is this class about?)
Software development trends

- Software continues to expand in size and complexity
  - What was once called “programming in the large” [DeRemer and Korn, 1975] has moved to a whole new level
- It is not uncommon for a software project to have hundreds of developers who work simultaneously
  - in different locations,
  - for different organizations,
  - collaborating without ever meeting
- Examples:
  - Consulting gigs
  - Global Software Development (GSD)
  - Open Source Software (OSS)
Scale and complexity in SW teams

Software Engineering = “the multi-person construction of multi-version programs” [Parnas, 1975]

- Contrast it with “solo programming”
Scale and complexity in SW teams

Software Engineering = “the multi-person construction of multi-version programs” [Parnas, 1975]

(from Jacques Tati’s “Playtime” – 1967)
Scale and complexity in SW teams

Software Engineering = “the multi-person construction of multi-version programs” [Parnas, 1975]

Geographically distributed interactions

© 2009, Valdis Krebs
Scale and complexity in SW teams

Software Engineering = “the multi-person construction of multi-version programs” [Parnas, 1975]

The SquirrelMail project [Crowston and Howison, 2005]
Scale and complexity in SW teams

Software Engineering = “the multi-person construction of multi-version programs” [Parnas, 1975]

- Emphasis on change and maintenance
  - How to write programs that are easily modified
  - How to write programs that are easily extended
  - How to write programs with useful subsets

- Emphasis on coordination among developers
  - How to break down work responsibilities
  - How to assign tasks
  - How to ensure circulation of knowledge and collaboration
Coordination / change interplay

- Large SW products have a huge number of dependencies
  - That directly impacts the level of complexity of the SW
- Driver of complexity is change
- Change is unavoidable and constant in SW development
  - When X is changed, it is possible that A must be changed as well
  - When A is changed, it is likely that a significant subset of Dep(A) must be changed as well
Bob, Chris and Diane do not know about the changes to A made by Alice.

They don’t know how to implement each other’s changes.

All of their work may need to be coordinated.

In SW development, coordination \(\equiv\) communication [Kraut and Streeter, 1995]
So ... what's the problem?

- Why should we pay attention?
- SW development is a knowledge-intensive activity
  - every arrow is a knowledge exchange
  - potentially quite complex

- Also, consider the case of GSD
  - say, Alice and Bob are dislocated wrt Chris and Diane
The cost of long-distance coordination, its level of “noise” and the implications for SW quality are significant.
Coordination and costs

- Coordination as a necessity
  - Triggered by continuous change and propagating because of SW dependencies

- Coordination as an overhead:
  - Developers spend much of their work time communicating in different ways [Perry et al., 1994] [Herbsleb et al., 1995]

- Coordination is one of the essential complexities of SW development activities [Brooks, 1986]
  - Can’t be eliminated, but can be addressed to an extent

- Strategies to streamline and optimize, and minimize related costs and effort
Socio-technical implications of software development

- **Socio-technical systems** show inextricable relationships between technology and people concerns [Emery, 1959]
  - Software is a socio-technical system
- The modular structure of the software product and the structure and dynamics of the development organization influence each other in complex ways.
- Governing the socio-technical implications of software development can lead to more effective management of contemporary software projects.
- It is important to investigate the interplay between social and organizational factors of development teams and characteristics of the software product.
The “Big 3” of socio-technical SW analysis

The “Big 3”: seminal observations of the importance of social factors to SW development success

1. Mythical man-month [Brooks, 1975]: Adding personnel to a late project delays it further
2. The double role of modularization [Parnas 1972]: Modularization is a means for subdivision of responsibility as much as for functional decomposition
3. Conway’s Law [Conway, 1968]: The structure of a software system is a direct reflection of the structure of the organization producing it

All from SW Engineering pioneers!
What we can cover today

- Trends in today’s SW development
- The problem of coordination of SW development teams
- The “Big 3” of socio-technical SE
  - Brooks’ Law – governing coordination overhead
    - Tools strategies
    - Process strategies
    - Design strategies
  - Parnas’ modularization principles
  - Conway’s Law
    - Socio-technical congruence
Brooks’ Law on coordination costs

“Adding personnel to a late project delays it further”
Brooks’ Law - value

- Brook’s Law is the epitome of why the socio-technical dimension of SW engineering is critical
  - It shows that without specific attention to this kind of concerns any project can fall into huge pitfalls

- It is also somewhat of an extreme view on the issue
  - Assumes undifferentiated coordination needs among all people in the team: $O(N^2)$ dyadic relationships
  - Assumes that pairwise coordination overhead is a significant fraction of the total amount of work

- Today we could consider Brooks’ Law only as the “ceiling” of coordination overhead [McConnell, 1999]
Attacking coordination overhead

- [Kraut and Streeter, 1995] observes that SW Engineering has devised three main lines of attack

1. via tooling ➔ boosting of developers’ productivity
2. via process ➔ orchestration of coordination via formalization of process
3. via design ➔ efficient partitioning of work
Advancements in development tools can:

1. Boost individual productivity
2. Decrease the need of explicit personal interactions to advance one’s work

You can do more technical work in less time …

- IDEs

… or spend less time on social interactions

- Awareness tools to overcome “bad isolation” [Sarma et al., 2003]
- Team-oriented development environments (TDEs)
Process-based attack strategies

- All the various process methodologies boil down to governing inter-dependent development activities
  - In socio-technical terms, they offer a framework for the orchestration of the interactions among developers attending to those activities

- How processes can help diminishing coordination overhead:
  1. **Agile development**: encourages developers’ knowledge sharing and facilitates the integration of workforce in a project
  2. **Open Source development**: production mode that makes “transaction costs” for community interactions cheaper than in other types of organizations [Benkler, 2002]
Design-based strategies
From Brooks’ Law to Parnas’ principles

- [Parnas 1972]: Modularization is a means for subdivision of responsibility as much as for functional decomposition

- Parnas’ definition of module: an independent task assignment
  - modularization is a weapon to alleviate $O(N^2)$ effect in coordination overhead
  - criterion for modular decomposition is to create SW entities that can (mostly) be worked on independently
  - modular break down should correspond to work break down

- Modules become socio-technical concerns
When module = task …

- Then, work break down and product break down will closely match

- People that are assigned to tasks should mostly work “inside the box” of the various modules
  - At least as the main design assumptions remain stable
  - Changes that disrupt the design are going to disrupt also the structure and dynamics of the team
Module = task creates a situation in which the way developers are distributed is likely to match the way the design is constructed.

[Bird et al, 2008] have looked at OSS projects and found out that groups of developers who are closely linked tend to concentrate around the same set of worked-on files.

At Drexel we have looked at OSS projects to see whether independent modules yield less need for communication [Wong et al, 2009].
Example from [Wong et al., 2009]
## DR Hierarchy Clustering

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Physical_interface</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>3. Link_interface</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>5. Network_interface</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>7. Transport_interface</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>2. Physical</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>4. Link</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. Network</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>8. Transport</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9. Application</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Physical: Physical interface
- Link: Link interface
- Network: Network interface
- Transport: Transport interface
- Application: Application

**Table Explanation:**
- The table represents the hierarchical clustering of different types of interfaces.
- Each row corresponds to a different type of interface (Physical, Link, Network, Transport, Application).
- The columns represent the different levels of the hierarchy.
- The symbols (X) indicate the presence of an interface at that level.
## Task Coordination

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Physical_interface</td>
<td>.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Physical</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Link_interface</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Link</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Network_interface</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. Network</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. Transport_interface</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. Transport</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9. Application</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Task Coordination

Collaboration: Same Layer Same Module (SLSM)
## Task Coordination

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Physical Interface</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>2. Physical</td>
<td>X</td>
<td>X</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>3. Link Interface</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>4. Link</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>5. Network Interface</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>6. Network</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>7. Transport Interface</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>8. Transport</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>9. Application</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

Diagram of task coordination with icons representing different interfaces and applications.
## Task Coordination

### Coordination: Across Layers (AL)

<table>
<thead>
<tr>
<th>Layer</th>
<th>Physical Interface</th>
<th>Link Interface</th>
<th>Network Interface</th>
<th>Transport Interface</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Link</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transport</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Application</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

**Legend:**
- X: Inter-layer coordination
- .: No coordination

**Notes:**
- Coordination is crucial for effective communication across different layers of the network.
- The diagram illustrates the inter-layer dependencies and coordination points.
## Task Coordination

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Physical_interface</td>
<td>.</td>
<td>.</td>
<td></td>
<td></td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>3. Link_interface</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>5. Network_interface</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>7. Transport_interface</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>2. Physical</td>
<td>X</td>
<td>X</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>4. Link</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>6. Network</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>8. Transport</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>9. Application</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
### Task Coordination

**Independence:**

**Same Layer Different Modules (SLDM)**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Physical_interface</td>
<td>.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Link_interface</td>
<td></td>
<td>.</td>
<td></td>
<td></td>
<td>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Network_interface</td>
<td>.</td>
<td></td>
<td>.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Transport_interface</td>
<td></td>
<td></td>
<td></td>
<td>.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Link</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Network</td>
<td>X</td>
<td></td>
<td>X</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>8</td>
<td>Transport</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>9</td>
<td>Application</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The independence is achieved by ensuring that modules within the same layer are independent of each other, allowing for flexibility in designing and implementing different network systems.
Empirical evaluation

Case Study: Apache Ant

- Open Source
- 600+ classes and interfaces
- Computed DR Hierarchy of version 1.6.5
- Analyzed mailing list and SVN activity during development of next version (19 months)
- For SVN commits, look at communication from prior two weeks
- Data population size:
  - 653 scenarios of developer pairs working concurrently
  - 266 scenarios of \textit{exclusively} SLDM
  - 356 scenarios of AL or SLSM (potential for Coord. Req – CR group)
Empirical evaluation

- Hypotheses:
  - Developers in SLDM scenario communicate less intensely
    Yes!
  - Developers in CR scenario communicate more frequently
    Yes!
- Pairwise Chi-squared test

<table>
<thead>
<tr>
<th></th>
<th>$\chi^2$</th>
<th>$p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLDM &lt; population</td>
<td>8.126</td>
<td>&lt; $10^{-2}$</td>
</tr>
<tr>
<td>SLDM &lt; CR</td>
<td>23.72</td>
<td>&lt; $10^{-6}$</td>
</tr>
<tr>
<td>CR &gt; population</td>
<td>8.252</td>
<td>&lt; $10^{-2}$</td>
</tr>
</tbody>
</table>
Empirical evaluation

- Hypotheses on probability of communications:
  - Is not correlated to amount of SLDM work
    Yes!
  - Is correlated amount of CR work
    Yes!

- Point-biserial correlation test

<table>
<thead>
<tr>
<th></th>
<th>$r$</th>
<th>$p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLDM</td>
<td>0.00206</td>
<td>&lt; 10^{-8}</td>
</tr>
<tr>
<td>CR</td>
<td>0.299</td>
<td>&lt; 10^{-8}</td>
</tr>
</tbody>
</table>
Empirical evaluation

Hypotheses on amount of communication:

- Is not correlated to amount of SLDM work
  Yes!
- Is correlated amount of CR work
  Yes.

Pearson correlation test

<table>
<thead>
<tr>
<th></th>
<th>$r$</th>
<th>$p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLDM</td>
<td>-0.0562</td>
<td>&lt; 10^{-2}</td>
</tr>
<tr>
<td>CR</td>
<td>0.189</td>
<td>&lt; 10^{-2}</td>
</tr>
</tbody>
</table>
Conclusions

- Communications between developers working on independent modules significantly less intense.
- Communications between developers working on inter-dependent modules in different layers significantly more intense.
- This work descends from Parnas’ modularization definition …
- … and goes in the direction of Conway’s Law.
Introducing Conway’s Law

- Parnas’ intuition about modules is shared by other research in non-SW product development literature (e.g. [Sosa et al., 2004])

- It is also consistent with the observation by Conway:

  *The structure of a software system is a direct reflection of the structure of the organization producing it*

- Conway’s Law has become part of SE lore with lots of anecdotic evidence
Conway’s Law implications

- Conway’s Law indicates correlation but not causation (in any direction)

- Coupled with Parnas’ modules definition it seems to indicate that there may be benefits if organization and product structures have a good fit
  - Those indications have been confirmed in non-SW product development literatures [Henderson and Clark, 1990], [Eppinger, 1991], [Sosa et al, 2004]

- A good fit, plus a good modularization, should help to minimize the interaction paths that are needed for a team to coordinate

- How do you measure that “fit”?
Validating Conway’s Law

- You need a way to quantitatively assess the fit between modularization and organizational structure.

- Moreover, you need a way to figure whether a better fit is conducive to better outcomes.

- In SW engineering, better outcomes may be:
  - Better SW quality (some quality property)
  - Better project efficiency

- How do you tackle the two issues above?
Socio-technical congruence
Socio-technical congruence

Concept borrowed from organizational theory

- Shows whether the organization is well-structured to carry out its tasks
- It’s a measure of fit between organizational structure and design structure

In SW, superimposes the organization to the modular structure

One goal is to verify Conway’s postulate in a quantitative way
Congruence and value

- [Cataldo et al., 2006] proposed a method for measuring congruence in SW projects

- Finding: resolution time of assigned tasks improves when coordination requirements highly congruent with patterns of coordination
  - verification of Conway’s postulate: a project that has higher congruence also tends to have better performance

- If the design structure and the organizational structure match well, it may increase project value
How do you compute congruence?
Construction of congruence

Task Assignments:
- go back to Parnas’ definition of modules

Task Dependencies:
- Tasks \((t_1, t_2)\) are dependent iff the corresponding change sets \((cs_1, cs_2)\) contain software dependencies
- ties back tasks to structural dependencies of SW artifact

Observed coordination: usually equated to explicit communication through some channel of technical communication that can be mined
- E.g. project mailing lists
Leveraging socio-technical congruence

- Requires understanding where there is room for improvement
- Analyze potential benefits

Method:
1. Individuate instances of lack of congruence
2. Evaluate their importance for the project
3. Quantify cost and value of re-structuring the socio-technical system
Assessing congruence “gaps”

- [Valetto et al., 2007]
- Look at the socio-technical ensemble of
  - Team
  - SW artifacts
  - Work by persons on SW artifacts
- Represent it as a graph
  - Demonstrably equivalent to matrix form
  - Can be extracted by looking at traces of software development in project repositories
Example: A set of SW artifacts (S)
Example: Relationships between artifacts

\[ G_s(S, A_s) \]

- e.g. static analysis deps. or change sets relationships
Example: a set of persons (P)
Example: The social network plane

$G_p(P, A_p)$

e.g. communication traces
Example: People to artifacts

Joins (J)

e.g. Task assignment
Example: Complete Network Model
Properties of the network model

- Topology of the socio-technical system
- Visualization of organizational “patterns”

- It is possible to isolate places in the networks that have a significant impact on congruence and study them
The importance of gaps

- A **gap** is a missing communication arc in an occurrence of an organizational pattern.
- Each gap decreases overall congruence.
  - It is an option for increasing value.
- Gaps can be ranked according to their congruence contribution.
Organizational Patterns: Collaboration (Node Tie)

- Two people are each involved with a common software artifact
- Collaboration is observed via a tie
  - A link in the social network plane
Organizational Patterns: Coordination (Arc Mirroring)

- Two people are each involved with a software artifact
  - Observable by two joins

- There is a dependency between the two artifacts
  - Observable by a software artifact link

- Coordination observed via a link in the social network plane
A major gap

P5 and P9 lack coordination
The importance of \((P5,P9)\)

Coordination between P5 and P9 would double overall congruence
Many ties have no or low impact on the congruence metric while a few have high impact.
Closing gaps

- Closing a gap is satisfying a coord. requirement
  - Positive impact on congruence metric
  - Possible impact on performance
- Close high-priority gaps!
  - Given the exponential distribution of high priority gaps
- What are your options?
How do you close gaps?
How do you close gaps?

Increase Communication

➢ Costs
  ▪ Communication overhead
  ▪ Brooks’ Law may kick in

➢ Risks
  ▪ Logistical burden (e.g. in distributed projects)
How do you close gaps?

Refactor Design

Costs
- Development re-work
- Communication overhead must still be paid to negotiate interface semantics (transitory)

Risks
- Variance in change cost estimation
How do you close gaps?
Modify task scheduling

- Costs
  - Resource slack

- Risks
  - Ripple effect that can impact project timeline
  - Must re-assess project schedule

Modify task scheduling
How do you close gaps?
Modify task assignment

➢ Costs
  ▪ Resource slack

➢ Risks
  ▪ Work centralization
  ▪ Resource overload
  ▪ Ripple effect that can impact project time line
  ▪ Must re-assess project schedule

Modify task assignment
How do you close gaps? Leverage indirect channels

- **Costs**
  - Burden on broker

- **Risks**
  - Noisy information transfer
  - Broker could become a bottleneck for decisions
Evaluating risk vs. value

- Just shown 5 different options for improving congruence
  - All options bring in the same value, but have different kinds and intensity of risk
- Risk vs. benefit analysis: must be able to estimate cost and other risks for each option
- Some things are “easy” to estimate:
  - we know how to estimate change costs in design and development
  - we can project effect of changes in schedule and resource allocation
- Some other things are tougher (e.g. human-centered measures):
  - how do you estimate the cost of communication?
  - how do you estimate overload of a resource?
  - How do you recognize the benefit of brokers?
STC - recap

- STC is a quantitative technique to analyze socio-technical concerns in SW development
- Leverages the ability to mine the information about the interplay between team, tasks and design
- Has a potential to impact the efficiency of the process
- We have shown how to locate places for improvement and ways to implement them
- Cost/benefit analysis is the next logical step in this research
Congruence implications on design

- Congruence integrates socio-technical concerns into design decisions
  - In a topological and quantitative manner

- Do we need to refactor our design?
- How do we refactor it?
- Be aware of coordination requirements
  - Consider how coordination requirements are affected
What covered today

- Trends in today’s SW development
- The problem of coordination of SW development teams
- The “Big 3” of socio-technical SE
  - Brooks’ Law – governing coordination overhead
    - Tools strategies
    - Process strategies
    - Design strategies
- Parnas’ modularization principles
- Conway’s Law
  - Socio-technical congruence
A final few words …

- The modular breakdown of the software product and the structure and dynamics of the development organization influence each other in complex ways.
  - We have seen how socio-technical concerns are always present in SW engineering (process, tooling, design …)

- To govern large-scale development effort, we have to understand the implication of the socio-technical nature of software development, and how they can be leveraged

- We have started to build actionable techniques to do systematic analysis and take decisions

- We must work on further analytic insights, as well as figure how they can be used in the SW development practice
Bibliography (1)

[De Souza et al., 2004] “How a Good Software Practice Thwarts Collaboration – The multiple roles of APIs in Software Development”
[Krauta and Streeter, 1995] “Coordination in Software Development”
[Hinds and McGrath, 2006] “Structures that work: social structure, work structure and coordination ease in geographically distributed teams”
[Herblseb et al, 2001] “An empirical study of global software development: Distance and speed”
[Cataldo and Nambiar, 2009] “Quality in global software development projects: A closer look at the role of distribution”
Bibliography (2)

[Perry et al., 1994] “People, organizations and process improvement”

[Herbsleb et al., 1995] “Object-oriented analysis and design in software project teams”


[Parnas, 1972] “On the criteria to be Used in Decomposing Systems into Modules”

[Sarma et al., 2003] “Palantir: Raising Awareness among Configuration Management Workspaces”

[de Souza and Redmiles, 2007] “The awareness network: To whom should I display my actions? And, whose actions should I monitor?”
Bibliography (3)

[Damian et al, 2007] “Awareness in the wild: Why communication breakdowns occur”

[Cheng et al, 2003] “Jazzing up Eclipse with Collaborative Tools”

[Kersten and Murphy, 2005] ” Mylar: A degree-of-interest model for IDEs”

[Robles et al., 2005] “Self-Organized Development in Libre Software Projects: A Model Based on the Stigmergy Concept ”

[Bolici et al., 2009] “Socio-technical congruence and Stigmergy in Free and Open Source Software projects”


[Raymond, 1999] “The Cathedral and the Bazaar”
[Crowston et al, 2006] “Core and Periphery in Free/Libre and Open Source Software Team Communications”


[Cataldo et al., 2006] “Identification of Coordination Requirements: Implications for the Design of Collaboration and Awareness Tools”

[Valetto et al., 2007] “Using Software Repositories to Investigate Socio-technical Congruence in Development Projects”

[Ehrlich et al., 2008] “An Analysis of Congruence Gaps and their Effect on Distributed Software Development”

[Damian et al., 2007] “Collaboration Patterns and the Impact of Distance on Awareness in Requirements-centered Social Networks”