Exploring Coordination Structures in Open Source Software Development

Chintan Amrit
J.H. (Eric) Hegeman
Prof Jos van Hillegersberg

University of Twente
The Netherlands
Agenda

- Problem Definition
- Existing Scenerio
- Research Questions
- Some Definitions
- Socio-Technical Structure Clashes
- Conceptual Model
- Tool Design
- Tool Demo
- Open Source Data
- Results Discussion
- Questions
Problem Description
Existing Scenario
Problem Definition
Research Contribution
Socio-Technical Structure Clashes (STSC)
Software Pattern

It is a solution to a specific and generic problem in a software development context


# Pattern Representation

<table>
<thead>
<tr>
<th>Field</th>
<th>Explanation/Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Ideally a meaningful name that will be part of the shared in design vocabulary. Many existing patterns do not satisfy this requirement for historical reasons.</td>
</tr>
<tr>
<td>Also known as</td>
<td>Other names of pattern.</td>
</tr>
<tr>
<td>Intent</td>
<td>A short specification or rationale of the pattern, used as a principal index for goal oriented pattern search</td>
</tr>
<tr>
<td>Applicability</td>
<td>An outline of the circumstances in which the pattern may be applicable and, perhaps more importantly, when it should not be applied.</td>
</tr>
<tr>
<td>Structure</td>
<td>A diagrammatic representation of the pattern</td>
</tr>
<tr>
<td>Consequences</td>
<td>Discusses the context resulting from applying the pattern. In particular, trade-offs should be mentioned</td>
</tr>
<tr>
<td>Implementation</td>
<td>Advices on how to implement the patterns, and other language specific issues. The implementation will depend on the abstractions (objects, parameterized types, . . . supported by the target language.</td>
</tr>
<tr>
<td>Known uses</td>
<td>Patterns are by essence derived from existing systems. It is therefore important that they be justified by their use in several real systems.</td>
</tr>
<tr>
<td>Related patterns</td>
<td>Patterns are often coupled or composed with other patterns, leading to the concept of pattern language; e.g. a visitor may be used to apply an operation to the closed structure provided by a composite.</td>
</tr>
</tbody>
</table>


Socio-Technical Pattern

Socio-Technical Patterns refers to those patterns involving problems related to both the social and technical aspects of the software process. So, in this sense Socio-Technical patterns would be a subset of Process Patterns.

Example: a pattern which describe the preferred relationships between team communication structure (the social network) and technical software architecture
Socio-Technical Structure Clash

An STSC occurs if and when a Socio-Technical Pattern exists that indicates that the social network of the software development team does not match the technical dependencies within the software architecture under development.
A healthy open source community
## Socio-Technical Structure Clashes

<table>
<thead>
<tr>
<th>Structure Clash</th>
<th>Explanation/Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gatekeeper Needed</td>
<td>Isolated and non communicating developers are not good for the company.</td>
</tr>
<tr>
<td>Betweenness Centrality Match</td>
<td>People who are not central to the software development or management take a central role in coordination</td>
</tr>
<tr>
<td>Code Ownership</td>
<td>When a code module is not changed by one ”owner” it is difficult to keep track of the different changes to it.</td>
</tr>
<tr>
<td>Conway’s Law</td>
<td>When the organizational structure is not compatible with the product architecture and vice-versa.</td>
</tr>
</tbody>
</table>
STSCs based on these Socio-Technical Patterns are:

1. If the developers in the periphery do not move towards the centre core
2. If the developers in the core move to the periphery of the project
3. If the ownership of important modules are shared by a large group of developers with no developer taking sole responsibility
Fig 6. The functioning of the software tool.
Clustering Motivation

Figure. Screenshot from TESNA. Dots represent software modules (blue/dark, lots) and developers (red/light, few).
clusters findClusters (param inputDSM) {
    Set clusters = one cluster for each module;
    int cost = calculate initial total cost;
    int failedattempts = 0;
    Do {
        m = a random module from inputDSM;
        For every element c from clusters {
            calculate bid of c for m;
        }
        cmax = highest bidding cluster;
        best_bid = bid from cmax;
        If (best_bid > 0) {
            Reassign m to cmax;
            For every element c from clusters {
                If (c is empty) remove c;
            }
            cost = cost – best_bid;
            failedattempts = 0;
        } else {
            failedattempts++;
        }
    } while (failedattempts <= FAILED_LIMIT)
    Return clusters;}

\[
CC(i) = \sum_{j=1}^{n} (SDM(i,j) + SDM(j,i)) * size(i,j)^2
\]
Propogation Cost/Clustered Cost

Raising the dependency Matrix to successive powers of $n$ and summing the rows and the columns we get the Fan out and Fan in cost or in general the: Propagation Cost

(MacCormack, Rusnak, Baldwin, Mang Sci 2006)

Assumption behind propagation cost is that dependency between elements incurs the same cost, wherever these elements. Clustered cost attributes different cost to the dependency according to where the elements are located.

(MacCormack, Rusnak, Baldwin, Mang Sci 2006)
Figure: The Tool at Work (displaying clustering results for a random input matrix)
Figure: Subversion activity of JAIM (taken from Sourceforge)
Case Studies

• Pilot Case Study with Masters Students, 4 teams of students each partly in Netherlands and India.
• Case Study in a Small start up software company in Rotterdam.
• Open source software development
• Future case: large globally distributed company
## STSCs in the Case Studies

<table>
<thead>
<tr>
<th>Company Size</th>
<th>Case Study Conducted</th>
<th>Conway's Law Pattern</th>
<th>Gatekeeper Required</th>
<th>Betweenness Centrality Match</th>
<th>Code Ownership</th>
<th>Core/Periphery</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small/teams Companies</td>
<td>4 GD Student Teams/Mendix</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Medium Size</td>
<td>(EMaxx)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Large Size</td>
<td>Open Source</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Contributions/Expected Contributions

• A formal approach to the identification of STSCs
• Development of a tool to identify STSCs: TESNA
• Empirical validation of this approach and its utility through a series of case studies:
  – Student case (ICMGDW)
  – MENDIX case (ECIS)
  – Opensource Case
Questions ?