A Social Network perspective of Conway's Law

Chintan Amrit, Jos Hillegersberg, Kuldeep Kumar
Dept of Decision Sciences
Erasmus University Rotterdam
{camrit, jhillegersberg, kkumar}@fbk.eur.nl

1. Introduction

To systematize software development, many process models have been proposed over the years. These models focus on the sequence of steps used by developers to create reliable software. Though these process models have helped companies to gain certification and attain global standards, they don't take into account interpersonal interactions and various social aspects of software development organizations. [9]

Conway's Law states "Organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations". [3]
Such a link means that "software" has to be reconceived as a socio-technical concept. Measurement, analysis, understanding, restructuring, and problem reduction for software is carried out by the same activities for the software organization too.

Parnas went on to clarify how the relationship between organization and product occurs within software development. His definition of a module as "a responsibility assignment rather than a subprogram" clearly shows that the dividing software system is simultaneously a division of labor [11]. It is this division of labor, among different individuals, that creates the need for them to coordinate, to align their efforts, in the production of software [6]. Thus, although the link between software and organization structure had been recognized decades ago in these classic papers, in today's software practices there seems to be little use and awareness of these ideas.

In this position paper we try to come up with a proposal to empirically test and visualize these ideas. In order to achieve this, we use the concept of *affiliation network* [5], which is a network formed by mapping the social network between the developers in a software organization to the small world network formed by the software [15]. By doing so we aim to improve the current design, execution and control of software process models.

2. Software Architecture Model and Conway's Law

In order to optimize software development several software architectures ranging from the waterfall model, spiral model to the present day models of Extreme Programming, Agile Development and the chaos model have been proposed. While the Waterfall model describes software development as a fixed sequence of discrete, irrevocable steps [14], the spiral model is described as a sequence of prototypes (each developed according to the waterfall model), each of which refines the previous prototype [2]. As suggested in the Chaos model the design, development and maintenance phases can be independent of each other, unlike the sequential structure suggested in the older Waterfall model. [13] Agile Software development among the most popular among modern software development methods is characterized by the following attributes: incremental, cooperative, straightforward, and adaptive [1]. All of these models have focused on the processes and have ignored the role of interpersonal communication and social interactions.

Previous work on social networks applied to software development has focused on a role-based approach to understanding the practices and organizational structures typical of productive software organizations [9, 10]. Where the typical examples of roles were designer, system tester, manager and end user, and the paper dealt with the interaction patterns between these Roles [9]. In this position paper we will be dealing with the social networks formed between the people involved in the three key areas of Design, Development and Testing.

There is also a huge difference between the design and implantation of software and as mentioned in one report [17], on an average only 52% of required features and functions make it to the released product. Diagram 1 is a representation of this disconnectedness between the process model and Software Architecture built during the design stage and the social network and Software Architecture in the implementation phase.

Just as the planned Software Architecture follows the Process Model we have the actual architecture follow the social structure of the company as explained in Conway's Law.

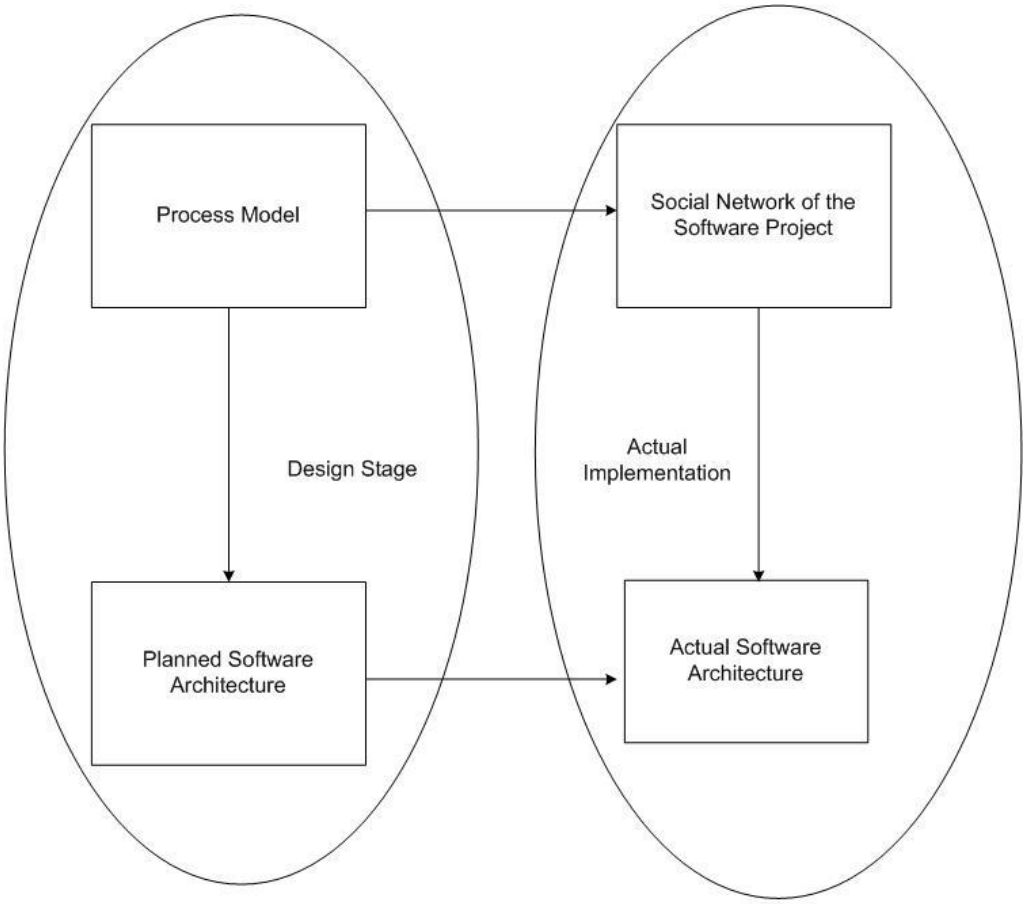


Fig 1. The Planned architecture of the Software Project vs. the actual implementation of the Architecture.

3. Conceptual Foundations

A social network exhibits the *small-world phenomenon* if, roughly speaking, any two individuals in the network are likely to be connected through a short sequence of intermediate acquaintances. [7]

Since the pioneering series of experiments conducted by Stanley Milgram and his co-workers [8], *small world phenomena* has come a long way and recent work has shown the phenomena is pervasive in networks arising in nature and technology. [8, 16]

Social networks often represent groups of people and the connections among them. The strengths of social network analysis have resulted in increasing use for understanding a range of small through large group interaction. In general, social networks originated from a descriptive and analytic discipline, but there is a trend toward embedding social networks into systems with the goal of facilitating new or renewed collaboration [15].

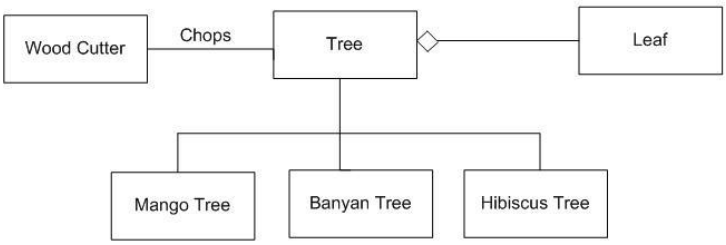
A way to understand natural and artificial systems is by looking at the structure of the relationships between their constituting components. On observation of large software packages, a pattern of social network structure is observed [15]. The following is a way of visualizing the formation of such a social network structure.

The components (classes) from the class diagram can be mapped into a node of a graph. If two classes relate through inheritance or composition, then we have a directed link in the associated graph.

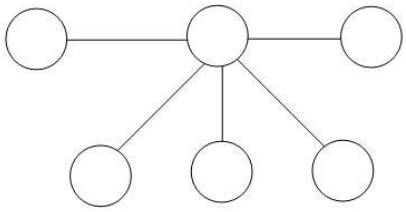
For example:

An edge in a class collaboration graph is directed from **B** to **A** if **B** makes a reference to **A** in its definition (either through inheritance or aggregation), and an edge in a call graph points from node **g** to node **f** if subroutine **g** calls subroutine **f** from its scope (**Fig 2**.).

A.



B.



A) A simple UML diagram.

B) A social network created from the above UML diagram. Every class maps into a single graph node. The relationships between the classes are represented as links in the graph. The directions and type of the linkages are ignored.

Fig 2.

We also have a social network of the people involved in the development of the Software (**Fig 3**.). These people are represented in the figure as rectangles colored in three colors. The people involved in all the different activities of Design, Coding and Maintenance are shown as large rectangles subdivided with the three aspects of development represented in 3 colors respectively. A different social network can exist for each aspect of Software development. Here we are concerned only with social linkages that translate to concrete work in the project. So we ignore linkages, say between specialist Designers and specialist Developers. Let us term the network formed by these "useful" linkages as a **production network** for the purposes of this position paper.

These social networks can be documented using means of e-mail, phone, chat or casual meetings over lunch, dinner etc. Though the nodes of the graph may seem to denote physical proximity, in fact some of the people in the social network may be located at far away places. As these networks change over a period of time, we can learn and draw different conclusions on how the social networks affect the Software development process.

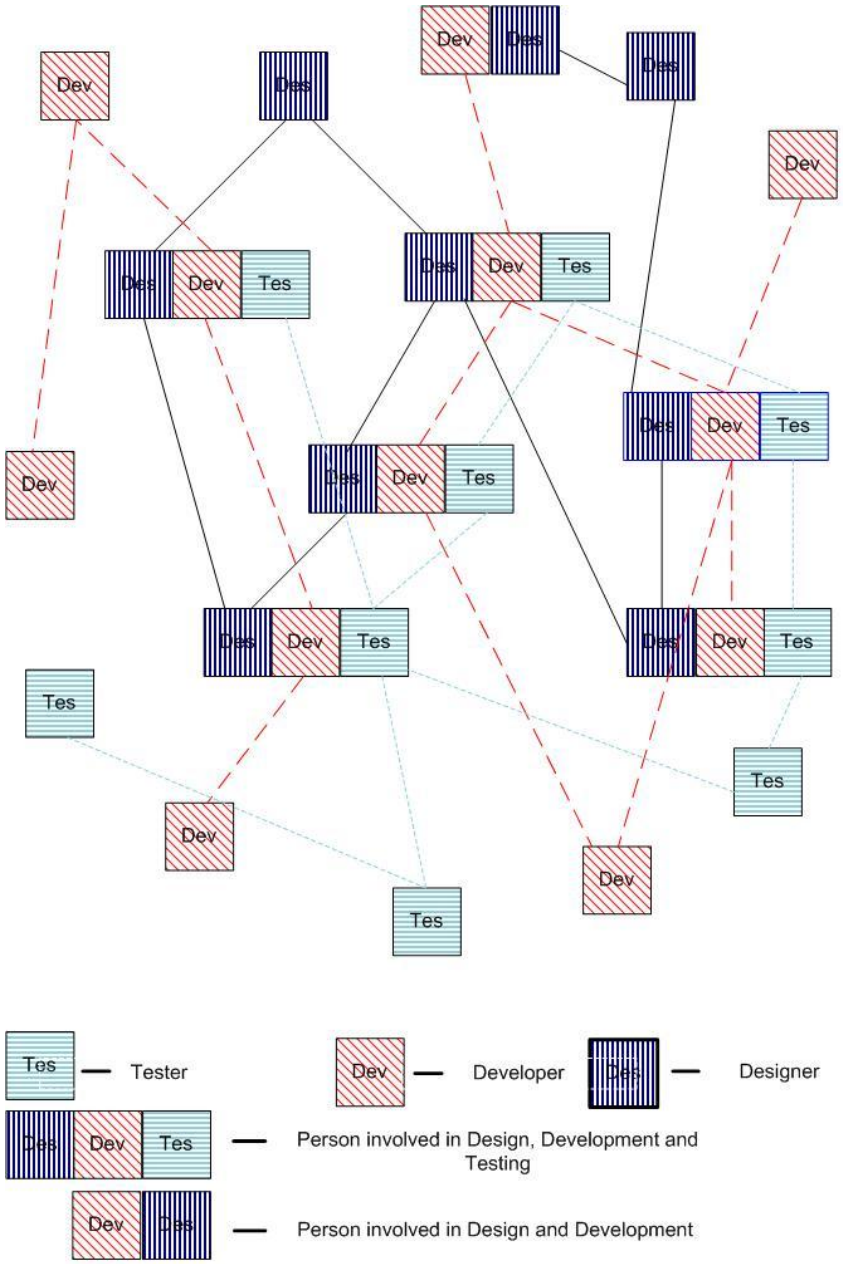


Fig 3. Each rectangle represents a person involved in the Software Development network. Some people who are involved in more than one activity are represented as larger rectangles.

4. Mapping of Social Networks

Typically, small-world studies of social networks have examined networks involving only one type of social entity (e.g. individuals or WebPages). These social networks can be represented as a 1-*mode* graph, where the term 1-*mode* refers to the fact that there is only one type of social entity or *actor*. In the graph, nodes represent actors and edges represent the presence of a *social tie* between actors. Networks of interlocking directors

are another, more complex type of network, *affiliation networks*. An affiliation network can be represented as a 2-*mode* or *bipartite graph*, with two types of nodes, and edges only possible between nodes of different types. [5]

The tasks of each developer in the network consist of designing, or coding, or testing of one or more objects or files of the Software project. We can map these tasks between the network of developers and the network of the software to come up with an ***affiliation network*** [5] as shown in **Fig 4**.

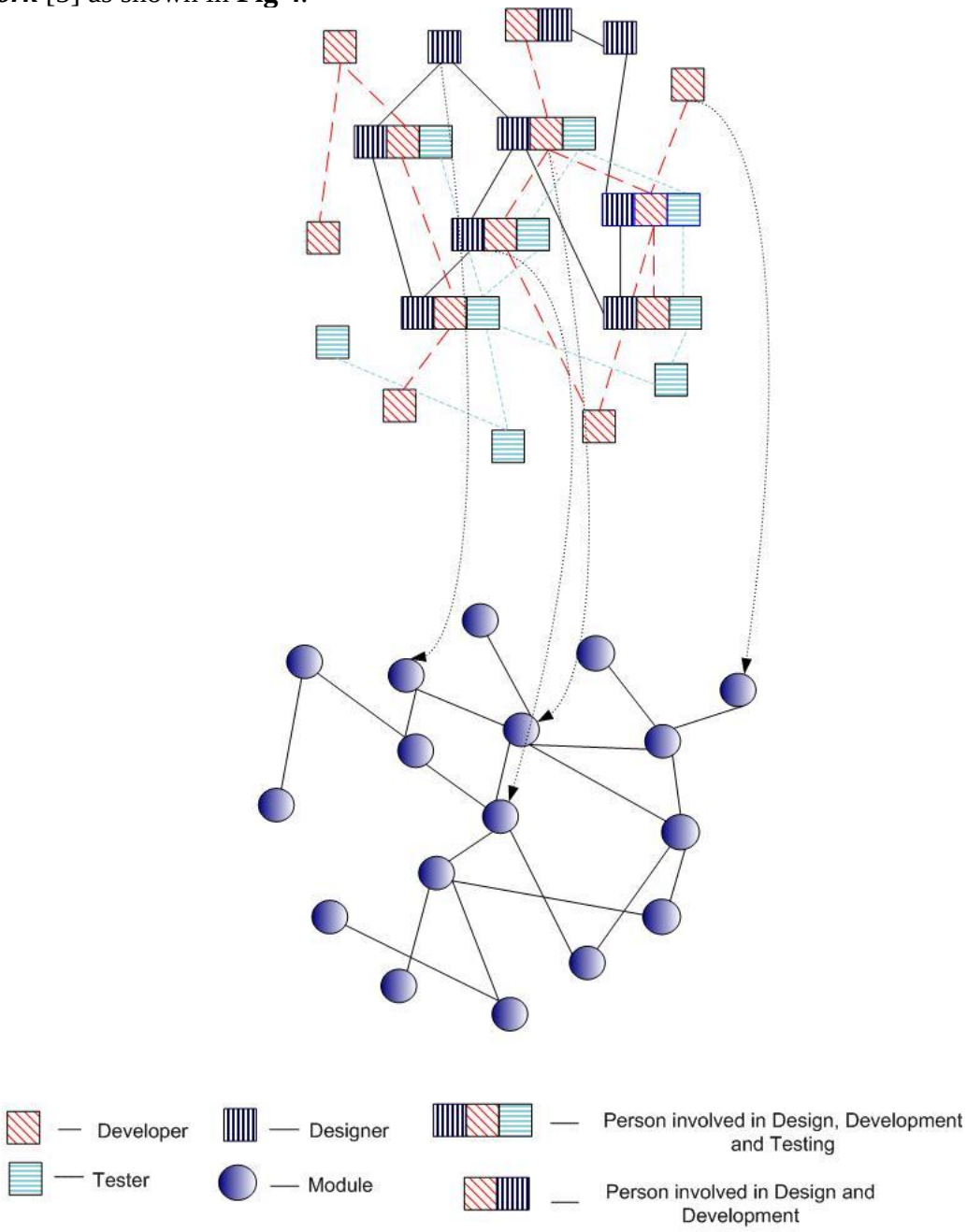


Fig 4. The mapping between the Social Network of people and the Small World Network of the Software. Only a part of the entire assignment of tasks is shown with an indication of graph isomorphism and one to one mapping.

In the above figure we see that not all the nodes of the social network are connected to the modules of the Small World network of the software, this was done to facilitate better understanding. This diagram depicts a specific case where there is isomorphic similarity of the 2 graphs, which goes to indicate that the software architecture follows the social network. The mapping shows the possibility that the task of designing/testing of the object in the software small world could be assigned to the corresponding person in the social network of developers.

Proposition:

We can use this idea of the affiliation network to improve the current design, execution and productivity of software process models.

5. Software Development and future research directions

In the book by Harrison and Coplien [10] there are lots of patterns that deal with the production of Software in the context of Conway's Law. These patterns are broad ideas for structuring an organization and do not answer many specific questions in the context of affiliation networks, like: How should the social network of the Software development team be structured, so that they are most productive? We can observe the changes in the *production networks* associated with the various aspects of software development, with time. Especially interesting would be the changes that occur when the deadline for an important project is nearing. We can ask the following question regarding the Software small world, how can the small world network of the Software be structured so that it's most productive and easy to maintain? Further considering our *affiliation network* we can ask the following question: How is the social network of the company connected to the Small World network of the software? Also interesting are the changes that can occur to the Small world network of the software if a highly connected person in the social network leaves the company.

6. Conclusion

Though many software architecture models have been suggested, there still exists a discontinuity between the Design and Implementation of processes in a software company [17]. What we have is complex development cycles that are governed by the social networks between the developers in the company. In this position paper we have tried to propose a model which can be used to empirically test and visualize principles of modularity as well as the ideas behind Conway's Law. By doing so, we aim to improve the design and execution of software process models. Further, we ask 3 principle questions regarding the *affiliation network* that could serve as pointers for future research in this field, namely:

1. How should the social network of the Software development team be structured, so that they are more productive?
2. How can the small world network of the Software be structured so that it's more reliable and easier to maintain?
3. How is the social network of the company connected to the Small World network of the software?

References

[1] Abraharnsson P., Salo O., Ronkainen J., and Warsta J., *Agile software development methods:Review and Analysis.* Espoo, Finland: Technical Research Centre of Finland, VTT Publications.

[2] Barry Boehm (1986) *A Spiral Model of Software Development and Enhancement,* in *Software Engineering Notes*, Volume 11, Number 4,

[3] Conway M.E., "How do Committees Invent?" *Datamation, 14(4):28-31, 1968*

[4] David W. McDonald, "Recommending Collaboration with Social Networks: A Comparative Evaluation", *Proceedings of the conference on Human factors in computing systems (2003)*

[5] Garry Robbins, Malcolm Alexander, "Small Worlds among Interlocking Directors: Network Structure and Distance in Bipartite Graphs", *Computational and Organization Theory (2004)*

[6] James D. Herbsleb, Rebecca E. Grinter, "Splitting the Organization and Integrating the Code: Conway's Law Revisited", *Proceedings, International Conference on Software Engineering (1999), pp 85-95*

[7] Kleinberg Jon, "The Small World Phenomenon: An Algorithmic Perpective", *Proceedings of the thirty-second annual ACM symposium on Theory of computing (1999)*

[8] Milgram S., "The small world problem," *Psychology Today* 1, 61 (1967)
Pages 14 to 24, August 1986, ACM Press.

[9] Neil B. Harrison and James O. Coplien, "Patterns of Productive Software Organizations", *Bell Labs Technical Journal (1996)*

[10] Neil B. Harrison and James O. Coplien, "Organizational Patterns of Agile Software Development", *Prentice Hall (2004*

[11] Parnas D.L., "On the Criteria to be Used in Decomposing Systems into Modules", *Communications of the ACM*, Vol. 15, No 12, 1972, pp. 1053-1058

[13] Raccoon L.B.S (1995) *The Chaos Strategy*, ACM SIGSOFT

[14] Raccoon L.B.S (1997*) Fifty Years of Progress in Software Engineering*, ACM SIGSOFT

[15] Sergi Valverde and Ricard V. Solé (2004) Hierarchical Small Worlds in Software Architecture, *IEEE Transactions (2004)*

[16] Travers J. and Milgram S., "An experimental study of the small world problem," S*ociometry* 32, 425 (1969)

[17] The Standish Group, http://www.standishgroup.com/press/article.php?id=2 (2003)