Enabling Social Network Analysis in Distributed Collaborative Software Development

Tommi Kramer, Tobias Hildenbrand, Thomas Acker

{tkramer/thildenb/tacker}@rumms.uni-mannheim.de

Abstract: Social network analysis in software engineering attains an important role in project support as more and more projects have to be conducted in globally-distributed settings. Distributed project participants and software artifacts, such as requirements specifications, architectural models, and source code, can seriously impede efficient collaboration. However, collaborative software development platforms bear the potential information for facilitating distributed projects through adequate information supply. Hence, we developed a method and tool implementation for applying social network analysis techniques in globally-distributed settings and thus provide superior information on expertise location, co-worker activities, and personnel development.

1 Introduction

Social dependencies in globally distributed software development projects are critical, as most projects nowadays involve distributed stakeholders and respective information resources; moreover, the software under construction becomes more and more complex in terms of functional and technological requirements. Hence, collaborative development environments are utilized in order to provide one common information repository and global view on the project. Social dependencies evolve around shared artifacts and work processes and this information can be extracted and analyzed by different project stakeholders. Existing methods of social network analysis (SNA) allow for locating expertise, providing better co-worker awareness, and supporting personnel development in general, for instance.

In order to make the required information available even in distributed settings, we have developed an approach to extracting relevant project data from collaborative development environments and providing flexible analysis functionality for different types of stakeholder roles and software projects. Therefore, we aim at presenting a solution for enabling SNA in distributed collaborative software projects and making use of this information to support more efficient development processes through better information supply. Hence, we extend existing collaboration platforms towards a social software for software engineering (SE).

For achieving this objective, we present a brief introduction to SNA in software development and related work in Section 2. Section 3 analyzes the most relevant use cases for our SNA solution, whereas in Section 4 design and implementation of selected features are described and Section 5 concludes with an evaluative discussion [HMPR04] of the current use case implementations and future work.

2 Foundations and Related Work

This section describes the definition of Social Network Analysis (SNA) in practice as well as in software development projects in particular. It gives a basic understanding of what social networks are and how they are created and used in Collaborative Software Development Platforms (CSDP).

2.1 Social Network Analysis

Having *SNA* on the one hand as a logical construct and on the other hand as the network representation substitutional for visual aid, this kind of information is of great help in managing projects, especially software engineering projects, in a better and more successful way. Also the goals of SNA and the metrics for calculating that network are mentioned.

In order to communicate or to express themselves in daily private life, people create profiles in which they provide information about their current activities. Furthermore, they join groups of shared interests or equal abilities, e.g. people of the same university, family relations, or sports teams. This process of organizing and communicating via *social software* is often also called *social networking*. As an example, there are platforms for regional or scholar clustering called Facebook[1] as well as for graduates and business people the Linked-in[2] website.

As one individual generally shares more than one interest or ability with others, many different links are created. These altogether build a huge network among the participants, the so called *social network*. This network with its various characteristics e.g. number of activities and relations allows/facilitates an evaluation and analysis of the strength of participant ties [WF94].

The importance of social capital has been subject to various studies of social sciences [BRW04]. The forming and distribution of knowledge is seen as an integral part of it. It should however be distinguished between explicit knowledge, that is written down explicitly and tacit knowledge, which can hardly be transfered or saved. Most social network approaches cover the management of explicit knowledge, whereas the focus on the identification and better usage of implicit knowledge can only be found in more recent approaches.

With the importance of social capital recognized, social networks and social networking have found their way into companies. One major motivation was to overcome the missing competence transparency among employees within large enterprises. Relationships of

[1]www.facebook.com (as of Dec. 3, 2008)
[2]www.linkedin.com (as of Dec. 3, 2008)

employees within a company go far beyond the organizational structure and are mostly invisible to managers [CBP02]. These invisible networks are however very reflective of how work happens in the company and should therefore be investigated further in order to support and foster collaboration.

2.2 Social Networks in Software Engineering

As the basics of SNA (cf. 2.1) expose, team awareness is of major interest in many fields of research. [XCM03] empirically analyze and describe different roles of individiuals in large-scale (open source) SE projects. In these settings, SNA is considered very important in SE when using collaborative software platforms. The basic interrelations within software projects have been defined by [RJ01] as can be seen in figure 1. While making use of the object-to-object (O2O) relations, our focus is set on the stakeholders in particular.

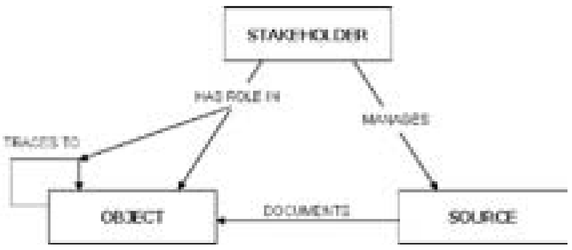

Figure 1: Traceability Metamodel after Ramesh and Jarke, 2001, p. 66 [RJ01]

[NT95] have recognized that globally acting companies have to cross great distances and thus propose early collaboration through dialog among the team members. They call the process of tacit knowledge exchange ”socialization”. A prerequisite for this process is the ability to identify colleagues working on similar issues.

Another important motivation to be aware of social structures within a company is risk management. The assessment of who is important to the project and who can be withdrawn can be crucial in scenarios [CH04]. In projects which are subject to high fluctuation, it is for example important to identify the key experts of a project. If a development team is reliant on a small group of developers, the sustainability of a project cannot be guaranteed.

Human beings are the most important critical success factor in an SE process. According to [GKSD05] it is important to enhance one’s personal development during a project lifetime. Humans have the intrinsic ability to develop their own status as well as that of the remaining project staff. The demand for creating ties to colleagues arises as everyone strives for as many connections to others as project members of equal status have obtained themselves [LFRGBH06]. This intrinsic behavior enables a project manager to quickly recognize key players.

[XCM03] and [Gra83] also investigated the capabilities of SNA in a SE environment. Knowledge and the spread of knowledge were the main focus in their studies. The difference between strong and weak ties as well as the spread of new ideas and knowledge can be explained by an SNA. That way the visual support of SNA provides a strong sense of common understanding and better awareness within the team. [Gab90] and [McG84] found out that a team is not understood as a composition of individuals and single workers any more. SNA helps create a better working atmosphere, better collaboration, and a higher satisfaction of the staff.

To analyze a social network in detail, metrics are of fundamental importance. Therefore, [LFRGBH06] made some initial propositions for adequate metrics in SE, like *Betweenness* and *Diameter* or *Distance*. Besides adapting these and other metrics, another approach for the identification of expertise of a colleague in an SE environment is to extract information from the documents the individual users have developed. These documents can provide a heap of information about authorship, topic, involvement and so on. This approach is further investigated in this paper.

3 Use Cases and Functional Requirements

As the importance of social networks was outlined both in daily life and in companies, the following section focuses on the investigation of how this kind of awareness can improve collaboration and coordination within software projects, in particular. Moreover, actual requirements based on the user value for team members and management are deducted.

3.1 Use Case 1: Expertise Location [UC1]

The importance of expert location is confirmed in a field study by [KDV07]. In this study, awareness about artifacts and co-workers was identified to be the most frequent information need around developers. It was found that developers often had to defer tasks because the only source of knowledge were unavailable coworkers.
In various other field studies, [dSHR07] found specific scenarios that identified major problems during the development of software. These especially illustrated the impact of a lack of awareness. The situation could be solved by the identification of developers who worked on the same or at least similar artifacts within a project. In order to find an expert, it is important to specify what is considered to be central in an organization. Communication habits as well as working habits of individual teams need to be regarded in order to obtain the desired result.

3.2 Use Case 2: Co-worker Awareness [UC2]

As partly outlined in the previous section, co-worker awareness is generally of significant importance. Not only the purposeful identification of an ascertained number of colleagues but also the awareness of ”what my professional environment is doing and how this relates

to me” is a prerequisite for efficient development planning.
Especially in large distributed teams like open source development projects, the global awareness of who is working on the same or related artifacts can save a lot of redundant work. Moreover, as mentioned in section 2.2, this kind of awareness allows the project members to identify colleagues working on related issues more efficiently or in the first place. Knowing these colleagues can thus provide a means to exchange tacit knowledge and even to enhance the affiliation within the team.

3.3 Use Case 3: Personnel Development [UC3]

By means of SNA, it is possible to analyze a project worker’s development from the management level. A new developer joining the team might only have a few ties to other project members. SNA providing special functions as part of this work enables the manager to track the development of the newcomer via SNA screenshots every once in a while. The progress (or even regress) in his development can be traced by that functionality. In a positive scenario the numbers of ties for example increase in every step.

3.4 Functional Requirements

Based on the use cases outlined above, functional requirements will be derived for our SNA-enabled Trace Visualization (TraVis) solution [HGKA08] in order to provide an effective means to meet the mostly unresolved issues identified in the literature (cp. Section 2.2).
First of all, the novel version of TraVis must provide functionality to extract existing tracability and rationale data from current data structures. With an adequate algorithm and by implementing some of the metrics discussed in section 2.1, this data needs to be transformed and prepared for visual output with users also grouped by their *Edge Betweenness*, a measure to calculate the importance of a user in a network.
The tool *shall* be enhanced by role-based filtering in order to be able to distinguish between developers, project managers, etc. The possibility of a view that highlights only the outgoing relationships of the user might also be a valuable addition. If he already has an idea of whom to contact for expertise knowledge within the project, this view will support him by proposing the shortest path regarding the strength of the ties to others.
To obtain a more specific view on the expertise that is searched for, a tracker item view *shall* be implemented that displays only those relationships between users that were identified within a specific tracker item. As the standard SNA-view visualizes relationships of projects based on their participation in all trackers, tracker items and artifacts, the social network becomes very abstract. This problem is enhanced in large-scale projects where many users participate. Where a holistic view on the social network is important to find out the overall relationships within a project, a more detailed view can be used for the location of expertise. The user *shall* therefore be able to choose a specific tracker item and the social network of users related to that item will be displayed on the graph.

Finally, a further measure to locate the project's experts *shall* be implemented: centrality. Centrality measures the involvement of the actor in a network by the ties he is involved in. More concrete, the tool *shall* use the betweenness centrality measure. Here, an actor is central if he lies between many other actors in their geodesics, their shortest paths between other users [WF94]. This measure *shall* locate individual experts in the project, where the user can decide, e.g. by the ego-view, how and whom within the experts he will contact [HGKA08]. Table 3.4 presents an overview of the functional requirements with respect to the use cases outlined before.

	Use Case 1	**Use Case 2**	**Use Case 3**
Different Node and Edge Sizes	X		X
Clustering		X	
Role Based Filtering	X	X	
Shortest Paths	X		
Tracker Centralization	X	X	
Betweenness Centrality	X		

Table 1: Functional Requirements with Respect to Use Cases

4 Solution Design and Implementation

The following chapter will provide the implementation details for the SNA-enabled version of TraVis. In doing so, the approach for extracting social network information will be explained against the background of existing approaches. After the description of other major implementation details, the process of calibrating the SNA relationship computation method is further elaborated.

4.1 Underlying Technology

We use codeBeamer, a collaborative software development platform developed by the company Intland[3], as the basis for creating the traceability and rationale network, we discussed in section 3.4. The platform is a solution especially suited for distributed collaborative software development, as it is an Internet-based application and therefore accessible all around the world [Rob05].
JUNG[4] (Java Universal Network/Graph Framework) is an open source Java library that provides an extendible language for the modeling, analysis, and visualization of data that can be represented as a graph or network. The JUNG framework provides functionality to visualize entities and their relations, represented as vertices (software artifacts) and edges (relations). JUNG also provides numerous algorithms of the graph theory, including

[3]http://www.intland.com (in this paper, codeBeamer version 4.3.2 is used)
[4]http://www.jung.javaforge.net (the current version of JUNG is 1.7.6 which is also used for TraVis)

centrality, betweeness, HITS, etc. This framework is the basis for the implementation of TraVis.

4.2 Trace Visualization and Social Network Analysis Method

In [Hil08] a tool, called TraVis, was developed for visualizing tracability and rationale information. Technical details are also described in [HGKA07]. Based on that we are complementing some classes and build new data structures to achieve social networking functionality.

4.2.1 Algorithm

As mentioned before, there are two basic approaches for finding dependency information in software development projects. Social dependencies can be obtained by collecting code dependency information. Authorship information of the source code is retrieved in order to associate users with code dependencies [dSHR07]. On the other hand, TraVis is based on more general artifact-related dependencies. Software development platforms supporting software projects with document management functionality, repositories, etc. (cp. section 4.1) contain valuable information on how individual teams collaborate that goes far beyond source code. This information can be utilized in order to obtain authorship information and analyze social dependencies throughout the project. By means of the JUNG framework, dependencies are displayed in a graph: vertices represent users and edges their relation to each other.

In order to identify these relationships, TraVis implements the following algorithm: For all users of the project, relationships to artifacts are identified where the user has some kind of active involvement (e.g. creator, modifier, etc.). For each of these artifacts in turn, associations are extracted from the lists and their respective users are put into relation with the current user to be analyzed in the loop. The overall procedure is illustrated in figure 2. For every user (shown in the middle of the figure) an iteration is run. It should be noted that within two iterations, relations to the depth of 3 can be obtained by the algorithm. These relationships however have a decreased weighting as the social relationship is of less direct nature. By that algorithm a foundation for extracting social data is set which delivers the required information for the named use cases [UC1], [UC2] and [UC3].

These associations can include many types. As authorship information about users to define the type of linkages between them and therefore decides about the strength of the tie, weightings need to be included. These weightings consist of the participation of the user regarding the individual artifact and the weighting of the artifact itself. In the algorithm, both values are multiplied and added to the SNA-value of the respective pair of users. For that, TraVis regards the roles: Creator, Modifier, Submitter, Approver, Owner, Last Modifier, Assigned Person, Locking Person. Moreover, various weightings can be distinguished:
TRACKER, TRACKER ITEM ATTACHMENT, TRACKER ITEM COMMENT, FORUM,

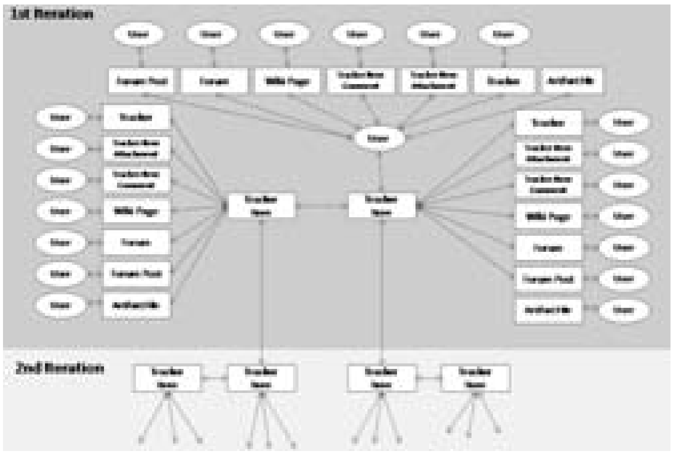

Figure 2: Illustration of the SNA Computation Algorithm

FORUM POST, WIKI PAGE, WIKI PAGE COMMENT, ARTIFACT FILE.

4.2.2 Customizable Weighting

The fact that the strength of the tie between users depends on these values implies that the right adjustment of these weightings is of great relevance for the SNA data basis. Where in its previous version, constant values were used in order to calculate the weightings, a means to customize the SNA by modifying the weightings is introduced in this paper. This will allow the user to select his preferred set of weightings before the algorithm is run (cp. section 3.4). The implementation of this feature is shown in figure 3. The values can be modified by means of individual sliders.

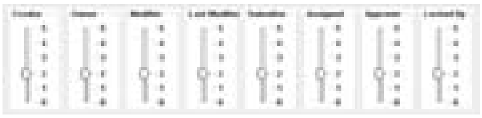

Figure 3: Adjustment of Role and Artifact Weightings

The right choice of weightings both for authorship information as for the respective artifacts can be crucial for the adequate representation of the social network of a project. In order to achieve a valid representation of social dependencies, the habits of use of the software development platform must be assessed prior to the excution of the SNA. Thus, finding experts [UC1] and personal co-workers [UC2] will be outlined more precisely.

4.2.3 Pre-Defined Views

To achieve the functional requirements (cp. section 3.4), we created different types of views to support and implement the specified use cases (cp. section 3). The following views combine multiple requirements at once.

Role-based View. As various roles in a software development team can be distinguished, it should be possible to identify these roles in social networks and cluster groups of roles. Therefore, the possiblity of identifying the color of the vertices as individual roles of the users was implemented. As individual roles can be created by an administrator in the code-Beamer platform, the roles are extracted dynamically from the project. Also the ability to remove certain roles from the graph is enabled in order to provide a more specific view on the SNA.

Ultimately, the possibility of clustering the users by their role is implemented within this paper. This shall allow a visual separation of the different parts of the teams and also provide a means to analyze how different roles and hierarchies collaborate within a specific project. Figure 4 shows how users can be clustered (by different colors) within a project according to their specific roles.

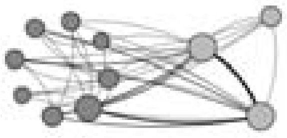

Figure 4: SNA Information Overview

Ego View. To enable an overview of the outgoing relations of a user, the so-called "‘ego view"' is implemented. This feature uses the Dijkstra-Algortihm for the calculation of the shortest paths [CLRS01]. The result is displayed in a SparseTree, which is provided as a means for the display of vertices and edges by JUNG. Based on the node of the current TraVis user, the shortest paths to all of the other nodes of the project are calculated. The Dijkstra algorithm had to be sligtly modified as a shortest path from one node to another is usually defined as having the lowest edge weight. In the SNA however, the higher the edge weight the stronger is the tie between two nodes. An example representation of the ego view can be seen in figure 5.

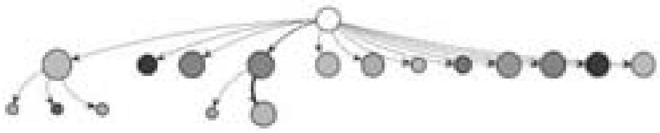

Figure 5: Ego View

Betweeness Centrality. To integrate the results of the calculation of this measure, the value is written on the graph next to the particular vertices. Therefore, the interface `VertexStringer` from the JUNG framework was implemented and now shows the values according to each vertex. Figure 6 depicts the result of applying betweeness centrality on a social network of a real-life project. The highest values are distributed between developers.

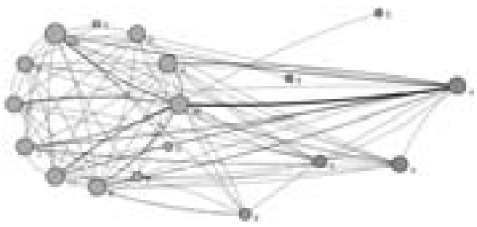

Figure 6: Betweeness Centrality View

Tracker Item View. The purpose of this feature is the refinement of the initial view on the social network that is provided by the SNA. The SNA algorithm (cp. figure 2) inlcudes all trackers, tracker items and software artifacts of a project. In order to filter the relationships within a tracker item, a list is added to the `UserUserRelations` list. Apart from the `UserDtos` and the SNA value, this list contains the specific tracker item, where the relation was identified. In order to run the TrackerItem view, the checkbox in the option pane is enabled and a popup-menu appears. Here, the user can choose between the specific trackers of the project within a dropdown-menu. If the tracker was chosen, another dropdown-menu appears for the user to choose between the tracker items of the tracker chosen in the previous step.

When completed, the social network for the specific tracker item appears. It should be noted that the vertices and therefore the users, similar to the standard view remain on the graph, even if they have no relation within the tracker item chosen. This shall provide a means to compare participations within different project tasks. If this is not desired, the *Hide Inactive* checkbox can be selected. Figure 7 shows the dropdown menu that allows choosing between trackers and tracker items, respectively. The menu does not dissapear after the tracker item is chosen, so the view can be changed dynamically while the checkbox is activated. An example for the application of this feature will also be given in the next chapter.

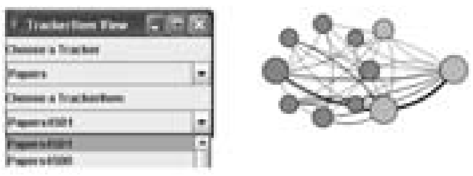

Figure 7: Tracker Item View

Concluding, every developed view helps to concentrate collaboration data for the different use cases [UC1], [UC2] and [UC3] as discussed in section 3.

5 Evaluation and Discussion

As has been demonstrated in the preceding paragraphs, TraVis' SNA-based views provide multiple functionalities for expertise location, co-worker awareness, and personnel development (cp. section 3). The underlying SNA method is innovative in that a large set of *various* relevant artifacts and user roles is included in the computation model and that the model can be easily adjusted to different project types in terms of artifact and role weighting.

Artifact and role weighting have been adapted by means of data from 17 replicated software experiments, where 9 teams used a prior version of TraVis and the other 8 just a state-of-the-art collaboration platform [Hil08]. In this sample, all teams developed an application based on the same set of requirements and identical technology (Java Platform, Standard Edition, including Java 3D). The experiment included 108 graduate students and had an overall duration of 6 months.

Compared to existing approaches, novel views based on a broader range of project data can be created and analyzed by different developer and management roles. Hence, superior decision support, primarily regarding expertise knowledge and workplace awareness, is provided. Further experiments and industrial case studies will be conducted in order to gain evidence for TraVis' effect on development efficiency and effectiveness (cp. [Hil08]). Moreover, weighting, view design, and platform integration will be further improved as more and more evaluation studies are conducted.

References

[BRW04] Andreas Becks, Tim Reichling, and Volker Wulf. Expertise Finding: Approaches to Foster Social Capital. *Social Capital and Information Technology*, pages 333–354, 2004.

[CBP02] Rob Cross, Stephen Borgatti, and Andrew Parker. Making Invisible Work Visible: Using Social Network Analysis to Support Strategic Collaboration. *Network Roundtable at the University of Virginia*, 2002.

[CH04] Kevin Crowston and James Howison. The Social Structure of Free and Open Source Development. *Syracuse Floss research working paper*, 2004.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald Rivest, and Clifford Stein. *Algorithmen - Eine Einfhrung*. Oldenbourg Wissensverlag GmbH, 2001.

[dSHR07] Cleidson R. B. de Souza, Tobias Hildenbrand, and David Redmiles. Towards Visualization and Analysis of Traceability Relationships in Distributed and Offshore Software Development Projects. In *Proceedings of the 1st International Confer-*

ence on Software Engineering Approaches for Offshore and Outsourced Development (SEAFOOD'07). Springer, 2007.

[Gab90] J. Gabarro. The development of working relationships. *Intellectual Teamwork: Social and Technological Foundations of Cooperative Work*, 1:79–110, 1990.

[GKSD05] T. Girba, A. Kuhn, M. Seeberger, and S. Ducasse. How developers drive software evolution. *Proceedings of the International Conference on Software Maintenance*, 1:113–122, 2005.

[Gra83] M. Granovetter. The strength of weak ties. *American Journal of Sociology*, -:1360–1380, 1983.

[HGKA07] Tobias Hildenbrand, Michael Geisser, Lars Klimpke, and Thomas Acker. Designing and Implementing a Tool for Distributed Collaborative Traceability and Rationale Management. *Working Paper des Lehrstuhls fr ABWL und Wirtschaftsinformatik der Universitt Mannheim*, 2007.

[HGKA08] Tobias Hildenbrand, Michael Geisser, Lars Klimpke, and Thomas Acker. Designing and Implementing a Tool for Distributed Collaborative Traceability and Rationale Management. In *Proceedings of the Multikonferenz Wirtschaftsinformatik (MKWI'08)*, Munich, Germany, 2008. accepted for publication.

[Hil08] Tobias Hildenbrand. *Improving Traceability in Distributed Collaborative Soaftware Development - A Design Science Approach*. Dissertation, University of Mannheim, Germany, Mannheim, Germany, 2008.

[HMPR04] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design Science in Information Systems Research. *MIS Quarterly*, 28(1):75–105, 2004.

[KDV07] Andrew Ko, Robert DeLine, and Gina Venolia. Information Needs in Collocated Software Development Teams. In *29th International Conference on Software Engineering (ICSE '07)*, 2007.

[LFRGBH06] Luis Lopez-Fernandez, Gregorio Robles, Jesus M. Gonzalez-Barahona, and Israel Herraiz. Applying Social Network Analysis Techniques to Community-Driven Libre Software Projects. *Int. J. of Information Technology and Web Engineering, Universidad Rey Juan Carlos, Spain*, 2006-09:22, 2006.

[McG84] J. McGrath. Groups, interaction and performance. *Prince-Hall*, 1:–, 1984.

[NT95] I. Nonaka and H. Takeuchi. *Knowledge Creating Company*, volume 77. Harvard Business Review, 1995.

[RJ01] Balasubramaniam Ramesh and Matthias Jarke. Towards Reference Models for Requirements Traceability. *IEEE Transactions on Software Engineering*, 27(1):58–93, 2001.

[Rob05] Jason Robbins. Adopting Open Source Software Engineering (OSSE) Practices by Adopting OSSE Tools. In Joseph Feller, Brian Fitzgerald, Scott A. Hissam, and Karim R. Lakhani, editors, *Free/Open Source Processes and Tools*, pages 245–264. MIT Press, Cambridge, USA, 2005.

[WF94] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, Cambridge, UK, 1994.

[XCM03] Jin Xu, Scott Christley, and Gregory Madey. Application of Social Network Analysis to the Study of Open Source Software. 2003.