Network analysis of the evolution of an open source development community

by

Zhaocheng Fan

A thesis submitted to the Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of

Master of Applied Science in Technology Innovation Management

Carleton University

Ottawa, Canada, K1S 5B6

September 2013

Abstract

This research investigated the evolution of an open source development community by undertaking network analysis across time. Several open source communities have been previously studied using network analysis techniques, including Apache, Debian, Drupal, Python, and SourceForge. However, only static snapshots of the network were considered by previous research. In this research, we created a tool that can help researchers and practitioners to find out how the Eclipse development community dynamically evolved over time. The input dataset was collected from the Eclipse Foundation, and then the evolution of the Eclipse development community was visualized and analyzed by using different network analysis techniques. Six network analysis techniques were applied: (i) visualization, (ii) weight filtering, (iii) degree centrality, (iv) eigenvector centrality, (v) betweenness centrality, and (vi) closeness centrality. Results include the benefits of performing multiple techniques in combination, and the analysis of the evolution of an open source development community.

Acknowledgements

Firstly, I thank my parents for their constant encouragement without which this research would not be possible.

I take this opportunity to express my profound gratitude and deep regards to my guides Prof. Michael Weiss and Prof. Steven Muegge for their exemplary guidance, monitoring and constant encouragement throughout the course of this thesis. I also would like to thank Prof. Tony Bailetti and Prof. Mika Westerlund, who also gave me many valuable suggestions on my research.

I want to thank Wayne Beaton in Eclipse Foundation. He provided the important input database that was used in this study.

Table of Contents

List of Figures

List of Tables

List of Appendices

Definitions of Key Terms

Open source software: The distribution terms of open-source software must comply with the following criteria: (i) free redistribution (ii) program must include source code, (iii) derived works among committers, (iv) integrity of the author's source code, (v) no discrimination against persons or groups, (vi) no discrimination against fields of endeavour, (vii) license must not be specific to a product, (viii) license must not restrict other software, (ix) license must be technology-neutral. (Open Source Initiative, 2013)

Project: A Project is the main operational unit at Eclipse. Specifically, all open source software development at Eclipse occurs within the context of a Project. Eclipse Projects are organized hierarchically. A special type of Project, Top-Level Projects, sits at the top of the hierarchy. Each Top-Level Project contains one or more Projects. A Top-Level Project is said to be the "parent" of those Projects. A Project that has a parent is oftentimes referred to as a Sub-Project (Eclipse, 2011).

Code Commit: A code commit refers to submitting the latest changes of the source code to a project. After a committer has made some changes he surely would like to save them to a remote location. This is when the commit operation is used (Eclipse, 2013a).

Committer: An Eclipse committer is an individual who has write access to the source code repository for the associated project (or component), or to other content on the Eclipse Foundation website. Each Project's set of Committers is distinct from that of any other Project, including Sub-Projects or parent Projects (Eclipse, 2011).

Eclipse developer community: The Eclipse developer community is a project-based meritocracy governed by the Eclipse Development Process and the applicable project charter document (Muegge, 2011). Companies have no status in the Eclipse developer community, but rather employ the individuals who have community status, and thus influence projects in that way. In this research, we focused to study the Eclipse developer community on the company level. On the company level, only the individual relations of a company can be considered.

Co-developer: Co-developers are companies who contribute sporadically by reviewing or modifying code on same open source development projects (Crowston & Howison, 2003).

Structural positions of companies:

This research adapts the developer categories proposed by Xu (2003). Based on an individual's contribution, Xu (2003) categorized individual software developers into three categories: peripheral developer, central developer, and core developer. Peripheral developers irregularly fix bugs, add features, provide support, write documents, and exchange other information. Central developers regularly fix bugs, add features, submit patches, provide support, write documents and exchange other information. Core developers extensively contribute to projects, manage releases, and coordinate peripheral developers and central developers.

For the purposes of this research, we modify Xu's categories in two specific ways. First, because this research examines the participation of companies rather than individuals, we categorize *companies* into peripheral developer, central developer, and core developer categories according to their structural positions and code contribution. Second, because this research examines only *code commits* rather than other forms of open source project participation, no other forms of participation – such as submitting bug reports, providing support, and contributing to documents – are included. The code commits of each individual developer are assigned to the company employing that developer. The modified definitions are as follows.

Peripheral Developer: A peripheral developer is a company that irregularly commits code to projects to either fix bugs or add features. Peripheral developers locate in the peripheral network.

Central Developer: A central developer is a company that regularly commits code to projects to either fix bugs or add features. Central developers locate in the center of the sub-network.

Core Developer: A core developer is a company that extensively commits code to projects. Core developers locate in the center of the whole network.

Definitions of network analysis techniques:

Degree: Degree is the number of nearest neighbors of a member in the network.

Eigenvector centrality: Eigenvector centrality represents if a node is located close to the center of whole network (Tsvetovat & Kouznetsov, 2011).

Closeness centrality: Closeness centrality is the length of shortest paths of a node to others (Newman & Girvan, 2004).

Betweenness centrality: Betweenness centrality is the number of times a node acts as a bridge along the shortest path between two other nodes (Freeman & Linton, 1977).

Chapter 1 Introduction

Muegge (2013) distinguishes between three interconnected levels of organization: platforms, communities, and business ecosystems. A *business ecosystem* includes companies, customers, suppliers, competitors, and other stakeholders, who "coevolve their capabilities and roles, and tend to align themselves with the directions set by one or more central companies" (Moore, 1996). A *community* is a voluntary group of people with common interests and a similar sense of identity. A *platform* is a set of technological building blocks and complementary assets that companies and individuals can use and consume to develop complementary products, technologies, and services (Cusumano & Gawer, 2002; Muegge, 2013, p. 7). According to Muegge (2013), field settings that combine these three subsystems are important contexts for innovation in the modern information economy. Interconnected networks of technologies, people, and organizations enable collaboration and value co-creation (Rosenkopf &Tushman, 1998; Van de Ven, 1993).

Open source software (OSS) projects often bring together these three subsystems (Milinkovich, 2008; Skerrett, 2008; Muegge, 2011b, Weiss 2011): (i) a platform of software assets distributed under open source licenses, (ii) communities of people who use, consume, maintain, and extend the platform, and (iii) a business ecosystem of companies that produce complementary products and services that consume and build on the software assets of the platform. One type of open source software community is the *development community* (West & O'Mahony, 2008), comprised of the individuals who maintain and extend the open source software. In many open source software projects,

the developers of the development community are employed by the companies of the business ecosystem (Skerrett, 2008; Muegge, 2011a). Within an open source development community, information, knowledge, and code resources flow more freely across organizational boundaries (Singh, 2010).

In this research, we focus on the *code commit* because every project in an open source community must include source code (Open Source Initiative, 2013). A code commit is the submission of a change to the source code to a project (Eclipse, 2013a). We focus on the companies that employ the people making code commits. Two companies that contribute code on the same project become *co-developers*: they co-innovate, co-create value, and co-develop the projects which they are concurrently working on. Information provided by the code commits of an open source development community includes when a company initiated a project, how many co-developers a company had, who the biggest committer was, and when a local development network got established. According to Grewalet et al. (2006), understanding relationships within an OSS community is critical for discovering the determinants of OSS project success, and code commits are one important aspect of those relationships. However, code commits cannot directly answer "how" or "why" questions such the reasons why companies choose to join or leave or to commit code to one project rather than a different project.

Social network analysis (SNA) techniques have been extensively used to study the development member relationships in OSS communities (Lopez-Fernandez et al. 2004; Jin et al. 2005; Masao et al. 2005; O Fleming et al. 2007; Grewal et al. 2006; Huang et al. 2007) and the association between developer relationships and outcomes (Fleming & Waguespack, 2007; Grewal et al. 2006; Hahn et al. 2006; Kuk, 2006). This thesis makes

improvements on how social networks were analyzed in previous research on open source projects. For example, previous research has often assumed that all members of a project collaborate equally. However, this is not true in a large project. By weight-filtering code commits of co-developers on the same project, we can ensure that only significant contributions are included in the analysis.

There are also several problems that previous network analysis studies of OSS communities did not answer, such as which social network analysis techniques can help provide insights into the evolution over time of an open source development community. To address such problems, we have collected and analyzed a code commit database of a well-known online OSS community, Eclipse (http://www.Eclipse.org), which includes code commits for 197 projects and 89 known companies over the past 12 years. There are at least three reasons why we chose the Eclipse development community to study. First, the Eclipse Foundation offices are located in Ottawa Canada, which gives us an opportunity to communicate easily with Eclipse Foundation staff. Second, the two co-supervisors of this thesis, Michael Weiss and Steven Muegge, have both been studying the Eclipse community for many years (Muegge, 2011a, 2011b; Weiss, 2011) and have experience with the field setting. Third, there is a body of prior research on the Eclipse field setting, including articles published in Carleton University's *Technology Innovation Management Review* (TIM Review; e.g., Smith & Milinkovich, 2007; Skerrett, 2008; Muegge, 2011b; Weiss, 2011), and graduate theses within Carleton University's Technology Innovation Management (TIM) program (e.g., Lombardi, 2008; Prattico, 2012). However, none of these prior studies have used network analysis to study the network evolution of the Eclipse development community.

In this study, we applied network analysis techniques to find what kinds of information that network analysis can provide about the evolution of an open source development community.

1.1 Objective

This research employs network analysis to examine the evolution of an open source development community. More specifically, it poses three research questions:

1. Which network analysis techniques are applicable to analyze the evolution of an open source development community?
2. What are the benefits of applying multiple techniques rather than a single technique?
3. How does an open source development community evolve?

1.2 Deliverables

There are four deliverables:

- Answers to the three research questions posed in Section 1.1.
- An evolution movie that shows the Eclipse development community growth each month over the past 12 years. It is a tool that helps researchers and practitioners to find out how the Eclipse project network evolved over time.
- Ranking lists of Eclipse companies which categorize companies based on the changes of their network structural position and project development involvement over the past 12 years.

- A software application with the capability to convert an open source development network to a company network.

1.3 Relevance

At least three groups will be interested in the outcomes of this research.

First, existing and potential Eclipse development companies can examine our results to learn how network positions change in the Eclipse development community over the time, identify popular projects to work on, and identify co-developers with high connectivity to work with. For example, degree centrality represents how many co-developer of a company has, and Eigenvector centrality represents if a particular co-developer of a company has more co-developers or not.

Second, the Eclipse Foundation staff can use the tools we created, and also the evolution movie and ranking lists, to arrange and reorganize the current resources to attract and help organizations that wish to collaborate on commercially friendly open source software. By visualizing the open source development network, we discovered that there are multiple local development networks that are weakly connected with each other.

Third students and researchers can employ the framework and tools developed here to study the evolution of other open source development communities.

1.4 Contributions

This study makes at least three contributions (more detail is provided in Section 5.3):

First, we combined analysis of both the project network and the company network in one study to gain a more complete view of the Eclipse development community than could be obtained from either network alone.

Second, we applied more network analysis techniques than previous studies (Sowe, Stamelos, & Angelis, 2006; Wiggins, Howison, & Crowston, 2009; Sadowski, Sadowski-Rasters, & Duysters, 2008; Hu & Zhao, 2009; Toral, Martínez-Torres, & Barrero, 2010; Crowston & Howison, 2012) to demonstrate that applying multiple techniques helped us to better understand each company's structural position changes and the changes to network characteristics over time.

Third, we provided a dynamic movie as a tool to show the evolution of the Eclipse development community. In comparison to previous studies using a static view (Xu & Madey, 2004; Sowe, Stamelos, & Angelis, 2006; Crowston & Howison, 2005), our dynamic view of the Eclipse network is a more effective way to visualize and understand the evolution of an open source development community over time.

1.5 Overview of method and results

This section is a brief overview of the dataset and method used to examine the Eclipse development community. We began by obtaining a copy of the Eclipse code commit database, which includes information on each code commit to each Eclipse project for the entire history of Eclipse. This database was provided by the staff of the Eclipse Foundation. We then assigned each code commit to the company that employed the committer. We constructed and visualized the project network (comprised of both companies and projects), converted the project network into a company network

(comprised of only companies), created an evolution movie as a tool to visualize how the network evolved over time, and computed network analysis metrics for each company for each year of the study interval.

Specifically, we reviewed, selected, and applied six network analysis techniques to study the evolution of the Eclipse open source development community: (i) network visualization, (ii) weight filtering, (iii) degree distribution, (iv) eigenvector centrality, (v) betweenness centrality, and (vi) closeness centrality. Tables 4.1, 4.2, 4.3, and 4.4 report these results to collectively show the network position changes over time of each company participating in the Eclipse open source development community.

Using company code commit data, we were able to study when and which companies joined or left the development community, which project was initiated by which company, which committer contributed the most of code on which project, and how many co-developers a company had at a certain time. We examined code commits only. We did not examine any of the other ways in which companies can contribute to an open source development community or interact with other companies. Likewise, we did not evaluate the relationship between companies' business strategies and their participation in development, or the motivations for company participation, or any other data not found in the code commit database.

1.6 Organization

This thesis is organized as six chapters. Chapter 1 presents the introduction. Chapter 2reviews the salient research literature on open source communities, network analysis, and the network analysis methods from prior studies. Chapter 3 explains the research

method, the network analysis framework, and the tools used to conduct the research. Chapter 4 presents the results of applying network analysis techniques. Chapter 5 discusses of the results, the contributions of this study, and the answers to the research questions. Chapter 6 concludes with the limitations of the present research and the opportunities for future research.

Chapter 2 Literature Review

The literature review is organized as three streams, each presented in its own section: open source communities and project participation (Section 2.1), structure of open source development communities (Section 2.2), and network analysis of open source communities (Section 2.3). Finally, section 2.4 presents the lessons learned from the literature review.

Table 2.1 is a summary of the three literature streams with highlights and key references.

Table 2.1 Summary of the literature review streams

Stream	Key highlights of the stream	Key references
Open source communities and project participation	An open source development community is a self-organizing network. Development companies can be categorized into different groups based on their project development participation.	Bailetti (2009) Carbone (2007) Xu (2003, 2009)
Structure of open source development communities	Many early studies of open source development communities were observational studies.	Crowston & McHugh (2008). Mockus, Fielding, Herbsleb (2002) Xu & Madey (2005)
Network analyses of open source communities	Various social network analysis methods can be applied to examine OSS communities; techniques include network visualization, network density, brokerage role, and centrality.	Gao, Y., & Madey, G. (2007) Huang (2010) Hossain, Wu, & Chung (2006) Mart (2012) Okoli & Oh (2007) Panchal (2009) Xu & Madey (2004) Xu, Christley, & Madey (2005)

2.1 Open Source Communities and Project Participation

There are two major member roles in open source communities: user and developer. Both roles participate in open source project development. Users can download and use open source products, report bugs, and suggest new features. Developers can do more than users; they are also able to write or modify the open source code, fix bugs, and deliver new features.

Xu (2003) distinguishes between active users and passive users. The passive users only download and use the product and never contribute anything. Active users have some code contributions to the open source projects. Xu (2009) divides open source developers into different groups based on their roles. The four roles are peripheral developers, central developers, core developers, and project leaders. The peripheral developers irregularly fix bugs, add features, provide support, write documents, and exchange other information. The central developers commit code, provide development support regularly. The core developers extensively contribute to projects, manage product releases and coordinate other developers. The project leaders provide directions of project development.

Carbone (2007) categorizes open source development companies into five different stages: (i) use stage, (ii) contribute stage, (iii) champion stage, (iv) collaborate stage and (v) redefine stage. The companies in “use” stage passively use open source product in this stage. The companies in “contribution” stage start to contribute code. The companies in “champion” stage start to take leaderships on one or more projects. The companies in "collaborate" stage become to be the open source strategists, they are able to highly

influence project development from customer driven resources to produce new versions of software. The companies in "redefine" stage change their structural positions in the value chain. Bailetti (2007) further discussed how the five level models of companies interacted with open source project development (Bailetti, 2007). The benefit for the companies in “use” stage is that using open source project can reduce the cost and development life cycle. The companies in “Contribute” stage fill gaps in feature set of company’s product for gaining better quality on interesting projects. The companies in “Champion” stage steer new project functionalities and its evolution. The companies in “Collaboration” stage have code commitment to health and advancement of open source project. The companies in “Redefine” stage drive resources to establish links with other open source projects and new project contributors. Previous studies also indicated and acknowledged that companies in different level of interaction with open source projects gains different level of profits. These two variables are directly proportional (Bailetti, 2007).

2.2 Structure of Open Source Development Communities

Previous literature on open-source processes primarily focused on open-source software development because of highly developed processes, large number of communities, and significant amounts of data on OSS development (Huang, 2010). Several researchers have reported that open source development communities have hierarchical or onion-like structures (Crowston & McHugh, 2008). Participation inequality allowed for the categorization of OSS community members into different groups (Mockus, Fielding, & Herbsleb, 2002; Xu, Madey, 2005). The book “The Success of Open Source” discussed different structure of organization structures in various OSS projects (Weber, 2004). For

example, the community structure of the Berkeley Software Distribution (BSD) project can be visualized as concentric circles, and Linux project has a pyramid community structure. Many previous studies analyzed open source communities, which were based on the observation of network structures. Later research started to apply social network analysis methods into OSS community network analysis, because OSS community network and social network had similar characteristics, such as network distribution, size, density, centrality, and so on.

2.3 Network Analysis of Open Source Communities

Social networks and OSS networks have similar structures. A network is a set of nodes that are connected by a set of ties (also called edges or links). A node can be used to represent a member or a set of members in a community. It can be any company/individual, team, or project, etc. In an open source ecosystem, nodes are used to represent companies and projects and ties connect pairs of nodes. Ties can be either directed or un-directed and either weighted or un-weighted. For example, in a network of several companies contributing to the same open source software project, the ties between projects and companies may be directed and weighted because company contributions may have different weights.

Table 1 summarizes four types of ties/edges connections in OSS networks. Within different databases, we can find weighted ties/edges connections in open-source project networks.

Table 2.2 Different types of connections in OSS ecosystem

	Directed	Un-directed
Weighted	Directed and Weighted	Weighted and Un-directed
Un-weighted	Directed and Un-weighted	Un-weighted and Un-directed

Some social network analysis methods are applicable to analyze an OSS community; others are not. Previous researchers analyzed OSS communities by using different combination of social network analysis. Xu & Madney (2004) studied role distribution and degree distribution in the Sourceforge community. Gao and Madey (2007) applied several SNA methods to study open source communities. They used degree distribution, clustering coefficient, and centrality. Because these researchers used different dataset, conclusions from their analysis were not the same. For example, Crowston & Howison (2005) analyzed the bug-tracking database in Sourceforge. They concluded that community's centralization was not a characteristic of OSS community (Crowston & Howison, 2005). However, after they analyzed Apache and Savannah open source communities. They found the large project was always less centralized comparing to small project (Howison, Inoue, & Crowston, 2006). In summary, they had at least one common conclusion; the network structure of OSS ecosystem directly influences the members' participation and decision-making process affecting the overall performance and growth of the community.

Table 2.3 is a summary of previous research. It is similar in structure to previous social network analysis literature reviews (Coulon, 2005; Provan, Fish & Sydow, 2007; Zschoch, 2007), identifying the dataset, methods employed, the contribution of each study.

Table 2.3 Summery of previous research contributions

Author(s)	Dataset	SNA method(s)	Contributions
Barcellini, Détienne, & Burkhardt (2009)	Python-dev mailing list	Distribution of participation (alternative to SNA methods)	This research provided a method; it confirms that participation must be investigated in the three interaction spaces. (Discussion, documentation and discussion)
Crowston & Howison (2012)	Sourceforge, the Apache Foundation and GNU's Savannah	Social structure, Centralization and Hierarchy	Large projects are less centralized. This research also compared centralized project teams and de-centralized ones.
Crowston &Howison (2005)	Several projects hosted by SourceForge (Based on Bug-tracker system)	Centralization, Out-Degree Distribution	The variance in communication structure was the way to differentiate between FLOSS projects. Centralization or decentralization in an open source community is not a characteristic of FLOSS projects when engaged in the task of bug-fixing.
Gao & Madey (2007)	Multiple monthly database dumps from SourceForge.net	Clustering Coefficient, Betweenness Centralization, Closeness Centralization, Degree Distribution	Applied several social network analyses on several communities, this research studied the network development and its influence to individual developments.
Huang (2010)	Drupal (drupal.org)	Degree Distribution, Clustering Coefficient, Diameter, Density	Drupal was a "Small-world network", this research extracted the mechanisms that drive the bottom-up evolution
Hossain & Zhu (2009)	Sourceforge	Network structure, density, centralization	There was a relationship between social network and project performance.

Author(s)	Dataset	SNA method(s)	Contributions
Hossain, Wu, & Chung (2006)	Enron corpus dataset	Betweenness Centralization, Closeness Centralization, Degree Distribution	This research proved the importance of designing a coordination and knowledge-sharing environment, and mapping social structure and position for coordinated groups.
Howison et al. (2006)	Several projects hosted by Sourceforge	Centralization, Out-Degree Distribution (over time)	This study proved that larger projects do not change central participants, but smaller projects change central participant more often.
Hu & Zhao (2009)	Ohloh dataset	Network Distribution, Topology Analysis	This study helped researchers and practitioners in OSS community to better understand the developers' participation choice behaviors and devise useful strategies for OSS project management.
Xu & Madey (2004)	Sourceforge	Degree Distribution	By visualizing Sourceforge community, the power law distribution proved that the Sourceforge developer network is a scale free network.
Xu, Christley, & Madey (2006)	Sourceforge 2003 data dump	Clustering Coefficient, Degree distribution	Different sized projects have different member distributions, also applying topological analysis on four subsets of the OSS development community.
Xu, Christley, & Madey (2005)	Projects from Sourceforge	Degree Distribution, Edge Density	This research studied the interaction and relationship between projects and the communication in OSS virtual organization.
Mart (2012)	Debian	Network density, Centralization, Clustering coefficient, Brokerage roles	This research provided genetic algorithms method to analysis OSS community. It classified community members into four different categories based on its size and degree.

Author(s)	Dataset	SNA method(s)	Contributions
Okoli & Oh (2007)	Wikipedia Database	Network Hierarchy (Structure), Density	Addressed one issue on how to keep the interest of participants and motivate them to continue to contribute, which can help to fill up the structure hole in network.
Panchal (2009)	Previous Researches	Network Hierarchy, Clustering Coefficient(Density), Centralization,	This research compared different network modules and interrelationships between them. This research can be used to understand the success of mass collaborative processes.
Toral, Martínez-Torres, & Barrero (2010)	Data from ARM Linux mailing list located at ‘‘lists.arm.linux.org.uk” (2001–2006)	Out-degree Distribution, Size of the network	This research developed and compared a macro-structural analysis and a micro-structural analysis. It provided new insights about the organization of virtual communities that can be useful for developing communities.
Sadowski, Sadowski-Rasters, & Duysters (2008)	Debian OSS community (meetings, conference, and interviews)	Distribution Structure	This research provided the understanding of specifics of the Debian community compared to other OSS communities.
Singh (2010)	Projects that were registered from 1999-2004 from the Sourceforge database hosted by the OSTG group.	Clustering Coefficient, Small-World network	Community’s network size influences the project success. Singh analyze the impact of macro-level properties of a community-level network on project success
Sowe, Stamelos, & Angelis (2006)	Debian lists server (lists.debian.org).	Community structure visualization	By using social network analysis to visualize the structure of the community, eventually found the major contributors and located knowledge broker in the network.

Author(s)	Dataset	SNA method(s)	Contributions
Valverde, & Solé (2007)	Sourceforge (http://sourceforge.net)	Measuring centrality: Strength and out-degree	It built up evidence for basic principles of self- organization, which is used to manage the software process.
Weiss, Moroiu, & Zhao (2006)	Large open source project (Apache) and sub-project (Httpd) mailing list	Community structure, Degree distribution	By using hypothesis-driven method to evaluate open source communities. This literature identifies different factors of community evolution, such as information flow, co-worker ties, and project dependencies.
Wiggins, Howison, & Crowston (2009)	FLOSS mole and Notre Dame Sourceforge repositories.	Network Centralization	In different venues, communication centralization represents different kinds of relationship.

2.4 Lessons Learned

The lessons learned from the literature are the following:

- An OSS community is a self-organized platform where individuals work together to provide modifications to the product.
- Companies located in the center co-create more value with others in development perspective. Companies located in the periphery make relatively less development contribution than others, which may also require assistance.
- The need to visualize and shape open source community creates new opportunities, especially for small technology companies.
- A company with low connectivity to the system has less commitment to the platform, increasing the risk that the company switches to another ecosystem.

- The structure of these relationships influences their potential for knowledge diffusion and creation.
- The way that companies choose to interact with Open Source projects determines the value they can appropriate.
- Many studies have acknowledged and emphasized the importance of developer relationships among the larger OSS community for their success

Chapter 3 Research Design and Method

There are three sections in this chapter. Section 3.1 describes the unit of analysis as the major entity that was analyzed in this study. Section 3.2 describes the study period we selected to collect data. Section 3.3 describes the network analysis framework that includes all applicable network analysis techniques can be used to study an open source development network.

3.1 Unit of Analysis

The unit of analysis is code commits by committers employed by a company.

3.2 Study Period

This study covered the 12-year period from 2001 to 2012, beginning from the launch of the Eclipse open source software project.

The input database was the Eclipse code commit database, which records code commitment activities since May 2001 (Eclipse, 2013a). The data were collected up to the end of 2012 so that complete annual data could be used to compare annual network changes.

3.3 Method

This research is atheoretical and exploratory.

Table 3.1 provides the steps of the research method and the rationale for each step.

Table 3.1 Steps of the research process

	Steps	Rationale Statement
1	Define research objectives	Set research objectives and study questions
2	Data collection	Collect data from Eclipse Foundation (code commitment database).
3	Network analysis techniques selection	Review network analysis techniques that have been used to study open source development communities.
4	Build network analysis framework	Based on the value of each technique, put them in to different category in different perspectives.
5	Create a software application to convert network	Build and use a software application that converts company-to-project network to company-to-company network
6	Network analysis approach	Use framework built in step 3 to analysis database.
6-1	Make a development network movie	A movie that shows how the Eclipse development community growth
6-2	Network characteristics Identification	Identify Eclipse community network structure and member's position
6-3	Rank of the Eclipse development members	Ranking list of the Eclipse development members based on their positions and project contribution
7	Comparison and Discussion	Discuss the results and values.

3.3.1 Define research objectives

The objective of this study is to use network analysis to study the evolution of an open source development community. Through this study, we can gain some insights into the Eclipse development community from analyzing its network structure changes over time

by using applicable network analysis techniques. In order to achieve this goal, we need to answer the following research questions:

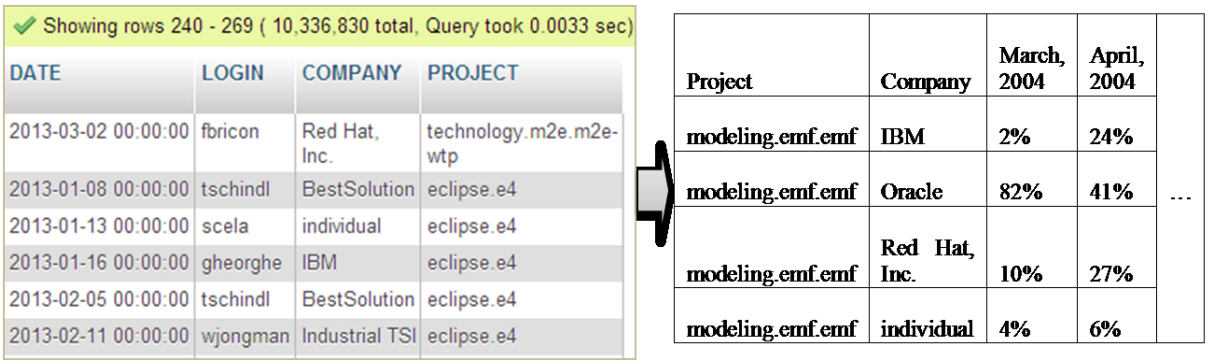
1. Which network analysis techniques are applicable to analyze the evolution of an open source development community?
2. What are the benefits of applying multiple techniques rather than a single technique?
3. How does an open source development community evolve?

3.3.2 Data collection

The Eclipse development network is a set of interrelated social entities. Network analysis is defined as an investigation of different network characteristics through studying the relationships between the entities in the network. Therefore, a data collection process must at least include the entities of the network and their relationships. In open source development community, an entity (node in network) could be a company or a project.

In this study, the original Eclipse code commitment database was provided by the Eclipse Foundation. The input database includes all code commitments information. It shows which committer was committing code to which project at what time. It gave us much specific information. It provides us a raw data about the individual developer and the project name they contributed to. It contains more than 10 million commitment records, so we summarized the database information and extracted a simple data metrics that is suitable for applying network analysis techniques to study. Then it is easy to track which company was committing code to which projects on what time. In Figure 3.1, the table in the left side shows a segment of the code commitment database, and the table in the right

side shows a segment of the summarized data matrices. In comparison with Eclipse code commitment dashboard (Eclipse, 2013a), the data metrics used in this study is more specific, it shows network monthly changes. For example, in Figure 3.1, the project "modeling.emf.emf" was created in March 2004. At that time, Oracle was the most active committer. In the second month, IBM and Red Hat, Inc became more active on code commitment. In order to study the evolution of the Eclipse development network, the following information was collected: project names, company names, and the percentage of total commitment for each company monthly.



Showing rows 240 - 269 (10,336,830 total, Query took 0.0033 sec)

DATE	LOGIN	COMPANY	PROJECT
2013-03-02 00:00:00	fbricon	Red Hat, Inc.	technology.m2e.m2e-wtp
2013-01-08 00:00:00	tschindl	BestSolution	eclipse.e4
2013-01-13 00:00:00	scela	individual	eclipse.e4
2013-01-16 00:00:00	gheorghe	IBM	eclipse.e4
2013-02-05 00:00:00	tschindl	BestSolution	eclipse.e4
2013-02-11 00:00:00	wjongman	Industrial TSI	eclipse.e4

Project	Company	March, 2004	April, 2004	
modeling.emf.emf	IBM	2%	24%	
modeling.emf.emf	Oracle	82%	41%	...
modeling.emf.emf	Red Hat, Inc.	10%	27%	
modeling.emf.emf	individual	4%	6%	

Figure 3.1 Illustration of the Data extraction

3.3.3 Network analysis techniques selection

There are many network analysis techniques that have been used to study different type of network. Based on the type of input database, some network analysis techniques are applicable to analyze open source development community; the others are not. Based on the literatures we reviewed in Chapter 2, the most commonly used network analysis methods have been summarized in the Table 3.2. The value of applying each method has also been summarized and described in the table. The third column in the Table 3.2

shows how frequency each method has been used to study the open source development communities.

Table 3.2 Network analysis techniques

Method	Value of application	Number of occurrence in previous studies
Visualization	• Easy to track which/how many companies/projects joined or left this Eclipse ecosystem at specific time • Easy to compare which projects are more attractive than others.	27 (out of 27)
Degree Distribution and Eigenvector Centrality	• To find companies' position that if a company is located in the center or periphery of the development network. • To find if companies have healthy co-developers • To find core-peripheral structure in development network.	20 (out of 27)
Weight filtering and Most active committer	• To identify if two companies' development relationship is strong or weak. • To find if a company is the most active committer of certain projects.	13 (out of 27)
Betweenness Centrality and Closeness Centrality	• To find how fast it will take to spread information/knowledge to other companies. • To find if a company is working relatively independently	22 (out of 27)

Network analysis is a theoretical and methodological paradigm for examining complex network structures. The links in a network also carry weights to represent the strengths of the relationships between members (Hanneman, R. A. and Riddle, M., 2005). The following network analysis methods were used in this thesis: (i) network visualization, (ii) weight filtering, (iii) degree distribution, (iv) eigenvector centrality, (v) betweenness centrality, and (vi) closeness centrality.

i. Degree is the number of nearest neighbors of a member in the network. Degree centrality is defined as the number of links incident upon a node. The degree centrality value of node v can be calculated by counting its degree.

$$C_D(v) = \deg(v)$$

ii. Eigenvector centrality is an extensional use of degree centrality. The measurement of eigenvector centrality is like a recursive version of degree centrality. The value of eigenvector centrality simply represents if a node is located close to the center of whole network. The algorithm works roughly as follows: (Tsvetovat and Kouznetsov, 2011)

1) Assigning a centrality score of 1 to all nodes ($V_i = 1$ for i-th node in the network).

2) Recounting the scores of each node as a weighted sum of centralities of all neighbor nodes $X_{i,j}$.

$$v_i = \sum_{j \in N} x_{i,j} * v_j$$

3) Normalize v by dividing each value by the largest value.

4) Repeat steps 2 and 3 until the values of v stop changing.

iii. Closeness centrality represents the length of shortest paths of a node to others. Closeness centrality indicates how long it will take to spread information from the target node to all other nodes sequentially (M.E.J. Newman, 2005).The farness of a node *v* is defined as the sum of its distances to all other nodes in the same network. The formal equation of farness is:

$F(v) = \Sigma_{t \in v}\ d_G(v, t)$

Closeness centrality is the inverse of the farness (Sabidussi, G., 1966). So the closeness centrality equation is:

$C_c(v) = \frac{1}{\Sigma_{t \in v}\, dG(v,t)}$

iv. Betweenness centrality quantifies the number of times a node acts as a bridge along the shortest path between two other nodes. It is measured for quantifying the control of a node in a network between other nodes (Freeman and Linton, 1977).The algorithm works roughly as follows:

1) Computing the shortest paths between each two pair of nodes. For instance, node s and node t. Adding the total shortest paths between node s and t, we can get σ_{st}.
2) Determining the fraction of shortest paths that pass through the target node V.
3) Add these fractions over all pairs of vertices (s,t) to get $\sigma_{st}(v)$.

The formal equation is (Brandes and Ulrik, 2001):

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

3.3.4 Network analysis framework

A network analysis framework has been built in this study, which was used to analysis the Eclipse development community in this research. Figure 3.2 describes the study framework. On the left hand, we selected and summarized all applicable network analysis techniques that have been commonly used to study open source community. On the right hand, the Eclipse code commitment database has been summarized into a simple data metrics, which are suitable to apply network analysis techniques. By applying all

common applicable techniques, we can get a more complete view of an open source development community than could be obtained from applying each technique alone.

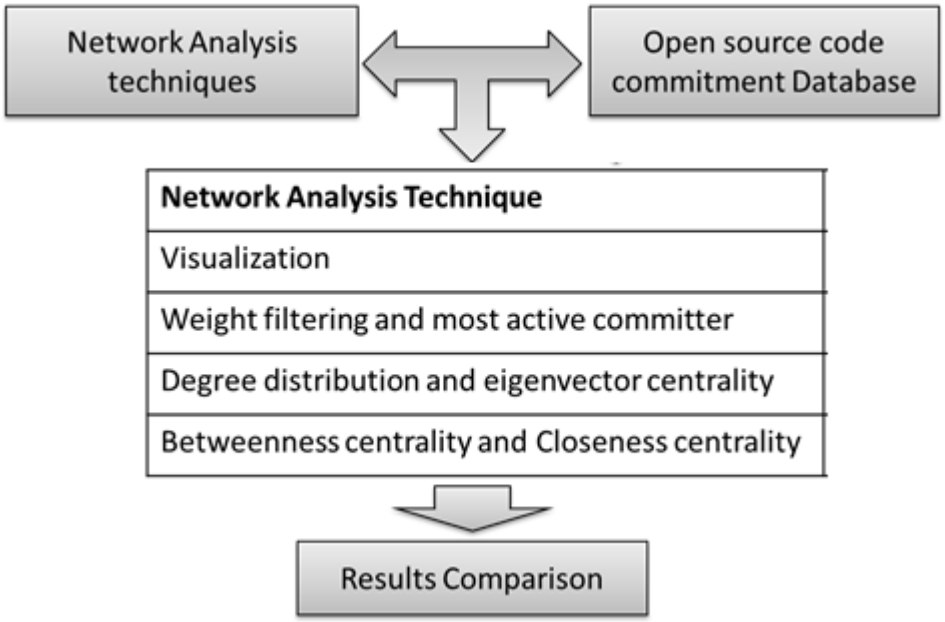


Figure 3.2 Study framework

3.3.5 Tools

3.3.5.1 Social network analysis application

There are many network analysis applications to study open source development community. They are able to generate same network structure if they have same inputted database. In this research, an application called "Gephi" had been used to visualize and analyze the Eclipse development community. Gephi is a well-known application that has even been used to teach and study social network analysis (Adamic, 2012; Bruns, 2011).

It has capabilities to visualize or measure each participant's structural position, degree, centrality, and so on.

3.3.5.2 Company network converter

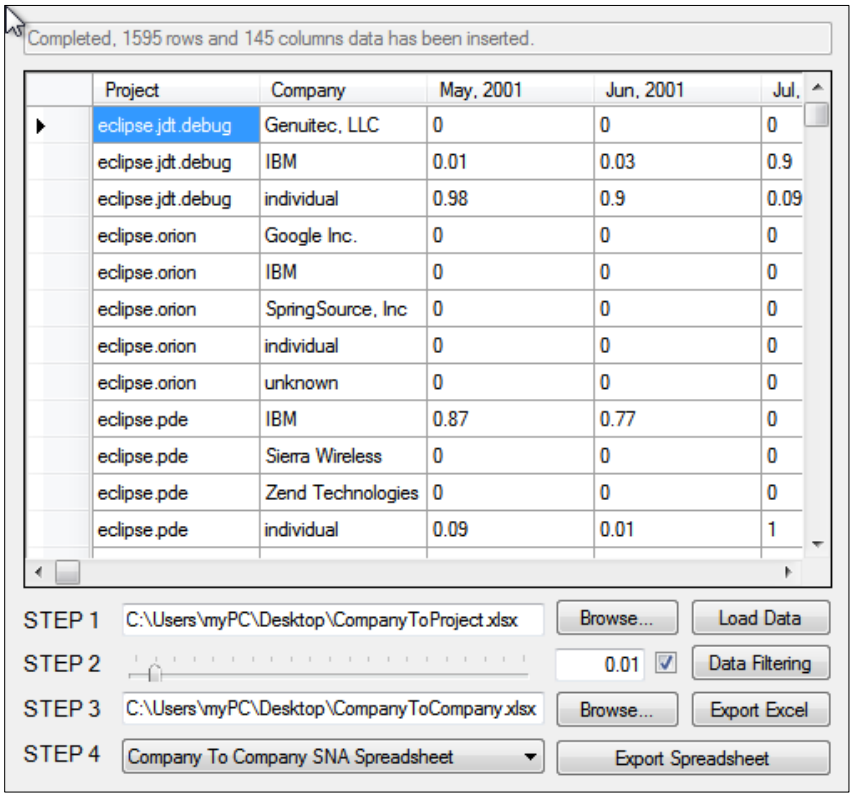


Figure 3.3 Company network converter

Figure 3.3 shows the software application that has been created in this research. This application was used to convert "company to project" network to "company to company" network. The reason why we need the company relationship network is that some network analysis techniques requires company network to analysis companies' structural position and network characteristics. In this research, the company network we generated only shows how companies shared development tasks on same projects, because we only studied the code commit of each company who participants in project development. Only

the co-developers that contribute code on the same projects will be linked in the company network in this study. Moreover, it also has capability to filter companies that has relatively small code contribution to certain projects. It can help us to find active co-developers and the most active committers.

3.3.6 Network Analysis Approach

The network analysis is based on the network analysis framework we built. Each analysis step and results are shown in Chapter 4 with details.

3.3.7 Results Comparison

In the end, we collected two different types of open source development network, which were the project network and the company network. Based on the type of network, different network analysis techniques were applied into study. The project network was used to study if a company contributed more or less than others. The company network was used to study how co-developers worked together on the same projects. Eventually, in Chapter 5, we compared the results from applying different network analysis techniques.

Chapter 4 Results

This section presents the results of our analysis of the Eclipse development network over the past 12 years. We systematically analyzed all participating companies' development relationship and their project participation. In Section 4.1, the first network analysis technique "visualization" was used to visualize the Eclipse project development network. We investigated how development companies change their network position over the past 12 years. The study in Section 4.2 was based on the results from Section 4.1. In Section 4.2, the weight filtering was used to measure and find the most active committers. In Section 4.3, degree centrality and eigenvector centrality techniques were used to measure and analysis companies' development relationship. The project development network was converted to company network, in order to show companies' development relationships. In Section 4.4, closeness centrality and betweenness centrality techniques were used to measure and analysis companies' network centrality characteristics.

4.1 Visualization

Network visualization technique is the most basic technique when studying a network, which is used to visualize the relationships among companies and projects in the Eclipse development community. In this research, it was used to identify the structure of a network, the project contributions of development companies in a network, and the attractiveness of projects in a network. The visualization makes the structural analysis of the Eclipse development communities possible, which helped us to identify core developers, central developer, and peripheral developers in the Eclipse development

network. As we learned from previous literatures in Chapter 2, a project initiator might be the project leader who takes the development leadership. Central company contributes the most on one or several projects. They regularly fix bugs, add features, submit patches, provide support, and exchange other information. Peripheral developers irregularly fix bugs, add features, and exchange information. Core developers extensively contribute to projects, manage CVS releases (Howison et al., 2006; Nakakoji, Yamamoto, Nishinaka, Kishida, Ye, 2001; Sadowski et al., 2008; J Xu & Madey, 2004). In this section, the study is not only looking for central developers in the Eclipse development community, but also identifies peripheral players in this development community. The advantage of applying visualization method is that a cluster of companies that has similar interests with high connectivity is placed together, which is segregated from dissimilar ones. Furthermore, we can identify how many local clustering network the Eclipse development community had in each year, and how these companies embedded in the local development network.

OSS communities should not have a strict hierarchical structure as prior studies discussed (Nakakoji et al., 2001), but the structure of the Eclipse development communities is not flat. Certain members have capabilities to influence other members and transfer information to others. Peripheral developers may have high development dependence on others. We have discussed several different developer roles in open source development community, such as core developer, central developer, peripheral developers, active users, and project leaders. The Eclipse open source community also maintains a balanced composition of all different roles of development companies. Otherwise, the community will be very unsustainable if all members are core developers or peripheral developers

(Nakakoji et al., 2001). Project Leader is often the person who initiates the projects, and the project leadership can be switched to other members. Project leaders might need helps from some core members (Weiss et al., 2006). So during evaluating the Eclipse development network, we tracked how development companies got involved into project development, in order to determine the project initiators, biggest project contributor, project-initiating helpers.

In Section 4.1.1, we explain the concept of the project development network, and how the network diagrams were generated. In Section 4.1.2, we describe the open source network structure with details. Because the open source network structure is a static view, it helps us to understand the dynamic grow that we will discuss in Section 4.1.3. In Section 4.1.3, we will focus on how these local development networks were established and who the central developers were. In Section 4.1.1 and Section 4.1.2, an overall the Eclipse development network diagram is used to explain and describe the Eclipse network structure. In Section 4.1.3, twelve annual Eclipse development network snapshots are used to show the network structure evaluation and changes of the company and project networks over 12 years. We built a dynamic movie to study the open source network evolution. The movie shows how the Eclipse development network changed monthly in past 12 years. This movie has more accurate information that reflects the network changes. Because there are too much information in the movie, we analysis and discuss the network changes in general in this chapter. The full dynamic movie can be found in YouTube.com (http://www.youtube.com/watch?v=SimFrZL9ylk), which is named as "Eclipse development network Apr, 2001 - Dec, 2012".

4.1.1 Project Development Network

The first step is to generate the global network diagram of the Eclipse development community. It showed all interactions among companies and projects living in the Eclipse development community. Visualizing a project development network is the prerequisite of identifying particular properties of this network, such as project initiator, core developer, central developer, peripheral developer, and active user. Project development network includes all participating companies, individuals, and projects. In this section, we can simply find which projects have more contributor than other projects, which companies are contributing to more projects than other companies. Furthermore, twelve annual project development network snapshots will be used to analysis network structure and sub-groups in next few sections.

As we described in Section 3.3.2 and Section 3.3.5, we extracted the input data from the Eclipse code commitment database. Then we used a network analysis application to generate and visualize the structure of the Eclipse development network. The original and clustered the Eclipse development networks are visualized as a collusion diagram corresponding to the code commitment matrices that has been generated previously. Figure 4.1 shows relationships among participating companies and projects in the Eclipse development community in past 12 years. Each node in the image represents a company or a project. It shows the overall Eclipse open source development community. Most previous studies used this kind of static network view to analysis open source community. In this study, we moved forwards, we separated this network into 12 network diagrams, which shows how the network got established in each year. Furthermore, we created a dynamic movie to show how the network grew and changed monthly in past 12 years.

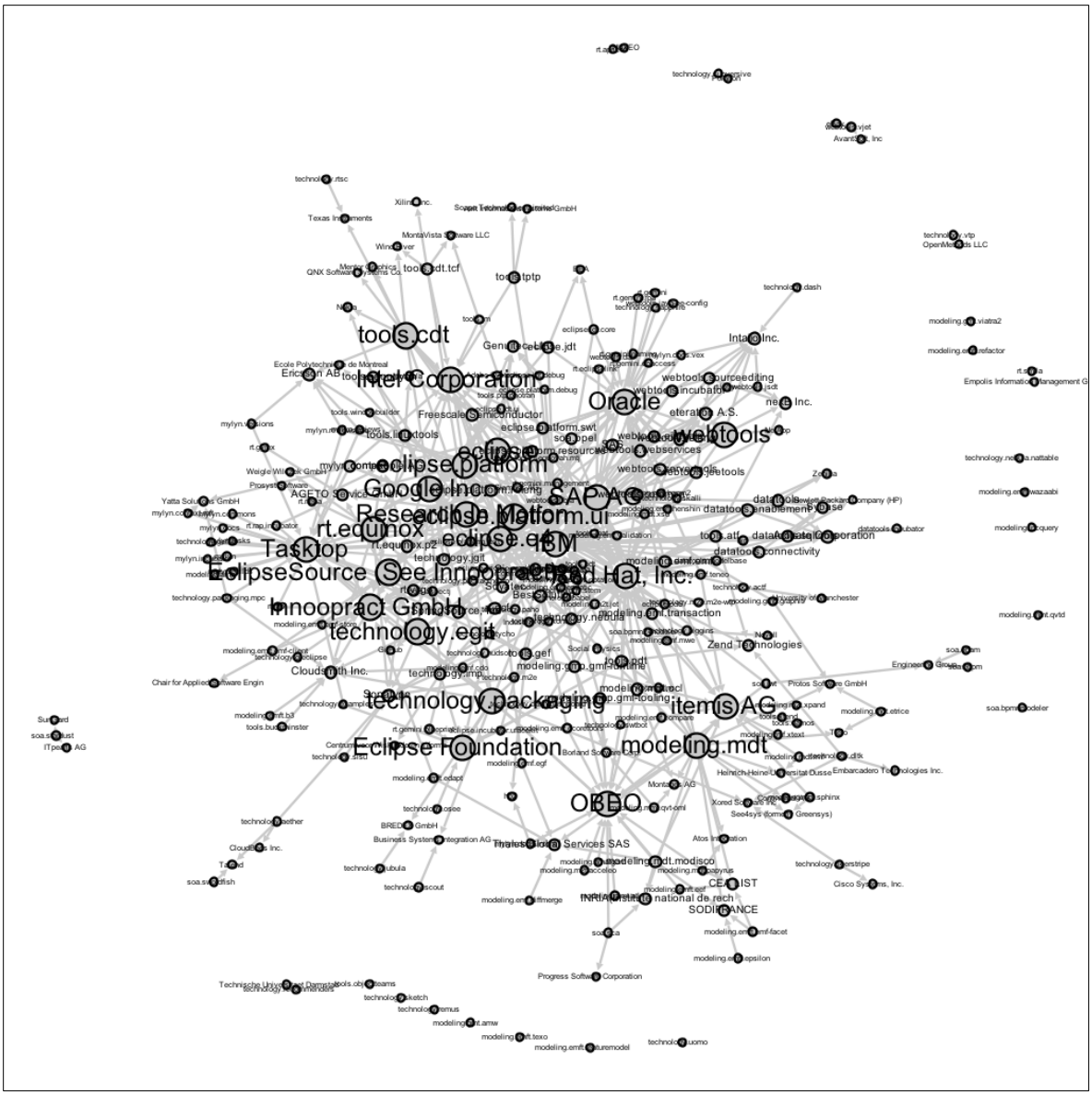


Figure 4.1 The overall Eclipse development network (2001-2012)

Figure 4.1 gave us an overall view of the Eclipse development community in past 12 years. Each link in the graph represents there is a contribution from a company to a project. There is no tie shown the relationship between projects or between companies. So all arrows are pointed at projects. This network structure cannot be represented as the out-tree model or network hierarchy because project development network is single direction networks. In Figure 4.1, we can easily observe that each two companies

working in the same project were co-developers. So projects acted as “hubs” that makes co-developers work together. For each project, some companies were contributing more than others. There are many clusters surrounding by some projects and companies in the Eclipse development network. Also it is easy to observe that a few of projects only have one committer, which is not ideal status in open source community. Another observation is some of the companies and projects were displayed at the center of network diagram, which normally have more connections than others on the peripheral companies and projects. The companies at the center often initiated and lead some projects. In another words, they were central developers or core developers in its particular local network. All studies in this chapter were based on observation.

4.1.2 Community Structure

Community structure is a common property of many networks. A large global network can be divided into multiple local networks. The connections between members in a local network are dense, but connections between local networks are sparser (Newman, Girvan, 2004).As shown in Figure 4.2, the Eclipse development global network has same network structure. A company located in local network (cluster-A) may also contribute a small weight of code to a project located in another local network (cluster-B). Such company can still be counted as a local network member in cluster-A, because this company has more relationship and contribute much more in Cluster-A. Finding and analyzing such local networks can provide us some help on understanding how development companies

involved into the Eclipse project development. In Section 4.1.3, we showed how this was achieved.

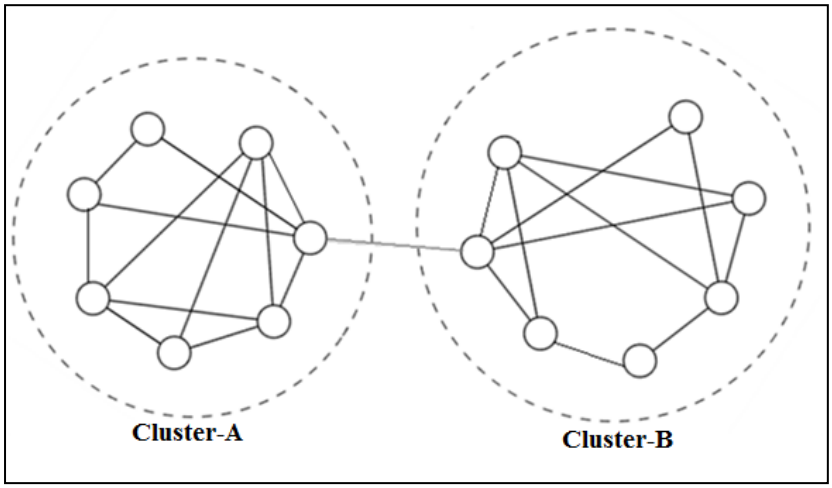


Figure 4.2 Connection between two local communities

The core developers in global communities often have strong similarity. They were connected in the early development stage. The core developers also have contribution to the projects located in multiple local networks. But central and peripheral development members have dissimilarity to others. They may have high contribution to a signal project or relatively less contribution to multiple projects as we learned from prior literatures. In each local Eclipse development network, we can see at least one central developer. However, the existing of peripheral developers is depending on whether the projects in the network have attractiveness to peripheral developers. In this study, the code commitment metrics have been put as modularity metrics into study to find sub-community or so-called local community (Iino & Iyetomi, 2012; Lambiotte, Delvenne, & Barahona, 2008; Milev, Muegge, & Weiss, 2009).

In this paper, therefore, we focus on how development companies were getting involved in the Eclipse project development. In each year, the global Eclipse development community was divided into local networks by using the method we discussed above. We measured and analyzed each year history of the Eclipse development community of 12 years in total by using the same methods. We depicted the Eclipse network structure annually. Next section discusses how development members changed their network position in global or local development community over past 12 years.

4.1.3 Community Evolution

New members joined into an open source community because the system can solve their problems. The evolution of an open source community is determined by two factors: the existing motivated members who want to play more important roles with larger influence, and the social mechanism of the community that encourages and enables such member role changes (Nakakoji et al., 2001). In this research, we studied the evolution of the Eclipse development community by analyzing different roles and role changes of participating members. Especially, we studied it deeply on how the role changes its members. As community companies changed the roles they play in the Eclipse community, they also indirectly changed and reshape the structure of the community. The role of an Eclipse company may get changed or not over time depending on how much this company wanted to get involved in the Eclipse community. A company can become a central member once this company joined the community, so the role is not assigned by anyone. It depended on how this member interacted with others. Company with low

connectivity to the system had less commitment to the platform and increased the risk that the company switches to another ecosystem. This can reduce the health of the Eclipse's ecosystem versus that of a competitor. (Hartigh, Tol, & Visscher, 2006)

By visualizing the Eclipse development community, we can find the core developer, central developers, peripheral developers, and active users of each local community. With time changes, the Eclipse has more and more local development network established based on local network members' personal business interests. At the early stage, from 2001 to 2003, the total amount number of local networks existed in the Eclipse was quite small. It rapidly grew during year 2004 to 2008, and year 2011 to 2012. Based on Crowston's study, small projects can be centralized or not, but the larger projects are decentralized (Howison et al., 2006). When participants started to have their personal interests and extra requirements, they needed to create new projects. Eventually new local development network got established. The growth of local development network was a result of new cluster of projects has been created.

Now we took twelve annual dynamic snapshots of the Eclipse development network to analyze how they changed their position and roles. Figure 4.3, Figure 4.4, and Figure 4.5 are the network snapshots demonstrate how the Eclipse development members got more and more involved in the development community monthly.

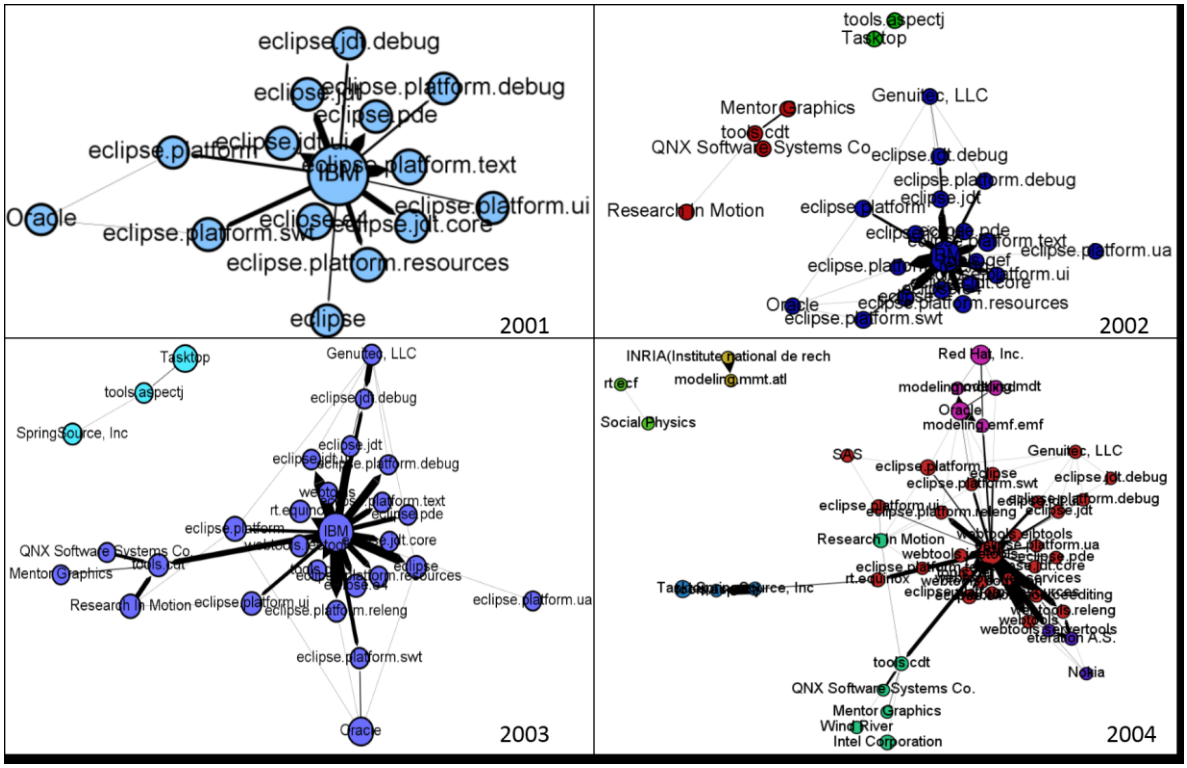


Figure 4.3 Eclipse project development network (2001-2004)

At the early stage, the Eclipse network structure can be speculated as a "star" network model. We found only a couple of companies were contributing to this network. The Eclipse development network did not seem like an open source network at early stage. Several companies lived inside of the Eclipse development network had full controllability at the early stage. The density of global development network was 1. In 2001, the global Eclipse development community only had one local network. It includes 13 projects and 2 known development companies, Oracle and IBM. Clearly, IBM was the initiator of multiple projects. The projects created in such early stage were representative. For instance, the Eclipse platform was the core projects for a long period. IBM can be looked upon as the project leader of these projects, because IBM was the initiator and contributed the most on these projects. Oracle only contributed code to two projects in

2001, which were “eclipse.platform” and “eclipse.platform.swt”, basically the Eclipse platform. But Oracle played a very significant role on co-developing both projects with IBM in 2001. IBM and Oracle was a couple of co-developers in 2001.

In 2002, since some new members joined in this community and created extra projects, the Eclipse development network has been divided into three relatively independent local networks. Tasktop worked on a tool project independently. QNX, Research in Motion, and Mentor Grapics started to co-develop a new project called “tools.cdt”. None of them contributed any code to projects initiated by IBM, so the Eclipse development network had three independent local development networks as shown in Figure 4.3. A new company called Genuitec joined in 2002. This company built a relatively stronger network with existing members, because each two of them were co-developing projects with others. In 2003, two local networks have been merged. QNX and its co-developers joined IBM’s local development network. Because IBM contributed the most on the project “tools.cdt”, then IBM had development relationship with other three companies that IBM never worked with before in the Eclipse development community. We also noticed that Oracle, Genuitec, and IBM concurrently worked together on more projects by times, which built up a stronger development relationship among them. They were able to share information and improve products more frequently. Tasktop found its first co-developer called SpringSource. SpringSource can be counted as peripheral developers in that time, because this company contributed much less than others, and it only had one co-developer. It is also easy to find in the network, SpringSource was located far away from others. In 2004, Tasktop and SpringSource started to contribute code to projects located in the main local network. But they were not merged into the main network.

There are two reasons that can cause a develop member gets merged into another local network, either such development member contributes the most to multiple projects located in another local network, or another member contribute the most on multiple projects such company is currently working on. Either way works. However, Tasktop and SpringSource were not merged in another local network as we can see in the network diagram, because both companies only worked on the same projects, and did not get involved in developing others. This kind of network evolution appeared all the time along with the Eclipse development network growth. Oracle started to create its own local network with a new joiner RedHat. Intel, a central developer in the future, joined in this year as a peripheral developer, which was co-developing “tools.cdt” with others. Another two independent local network has been established in the same year. In prior year, it had only two local communities. But in this year, it had 7 local communities, which means most companies were not highly connected.

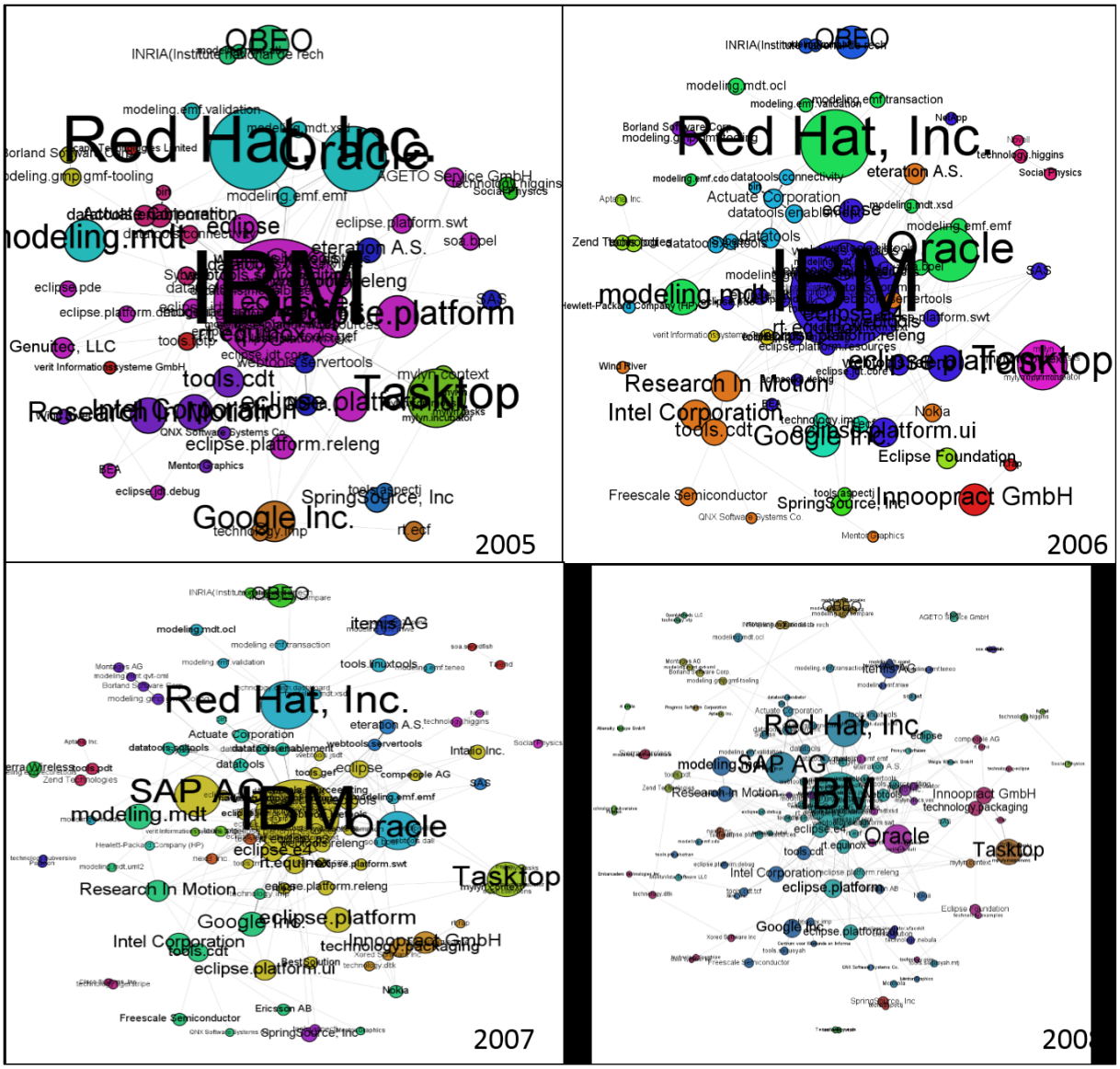


Figure 4.4 Eclipse project development network (2005-2008)

In 2005, it had 12 local development networks. The Eclipse development network had many company positions changed during that year. As we can see in annual network diagram, it had a relatively loose network, but only two local networks were completely independent from the main network. INRIA, the central developer of an isolated local network, started to find its co-developer, several new joiners, OBEO, Google, HP joined this network and worked as peripheral developers at that time. But it changed its position

rapidly in next couple of years. Tasktop established a local community with 5 independent projects but no co-developers.

In 2006, the Eclipse development community had 13 local development networks, which was similar to prior year. But some companies changed their network positions in this year. Red Hat and Actuate became central developers in their own local community. Other local communities remained the same as previous year. In 2007, Tasktop started to contribute code to "eclipse.flatform" and other projects located in other local communities. This behavior helped Tasktop to find more co-developers, in order to share information, build stronger connection to the network, and build up stronger development relationship with other members. Red Hat's local community has been divided into several local networks, because the project that Red Hat worked on were very attractive to others. This change helped Red Hat built up a relatively stronger development relationship. Red Hat also became a co-developer of OBEO. Even through OBEO was still a peripheral developer at that time, OBEO has been connected to main development network eventually. Because of the large contribution to project called "eclipse.jdt", SAP got closer to the center of global development network. It differentiated its network position from peripheral developers, like Google, QNX, Research in Motion and so on. In 2008, there were seven completely isolated local communities located in the Eclipse development network. Each local community only contained one committer and one project. The independent project initiator and leaders were Polarion, Sierra Wireless, Attensity, OpenMethods, Teaxs Instruments, and Talend, AGETO Service. Each of them was independently working on their own projects. These members did not get involved into the main development network, which means they did not have any co-developer at

that time. Besides, Red Hat got more embedded in the global development network. Red Hat was one of the strongest co-developer of IBM who was always the core developer of the global network. Intel also got more involved on multiple project co-developments with others by times.

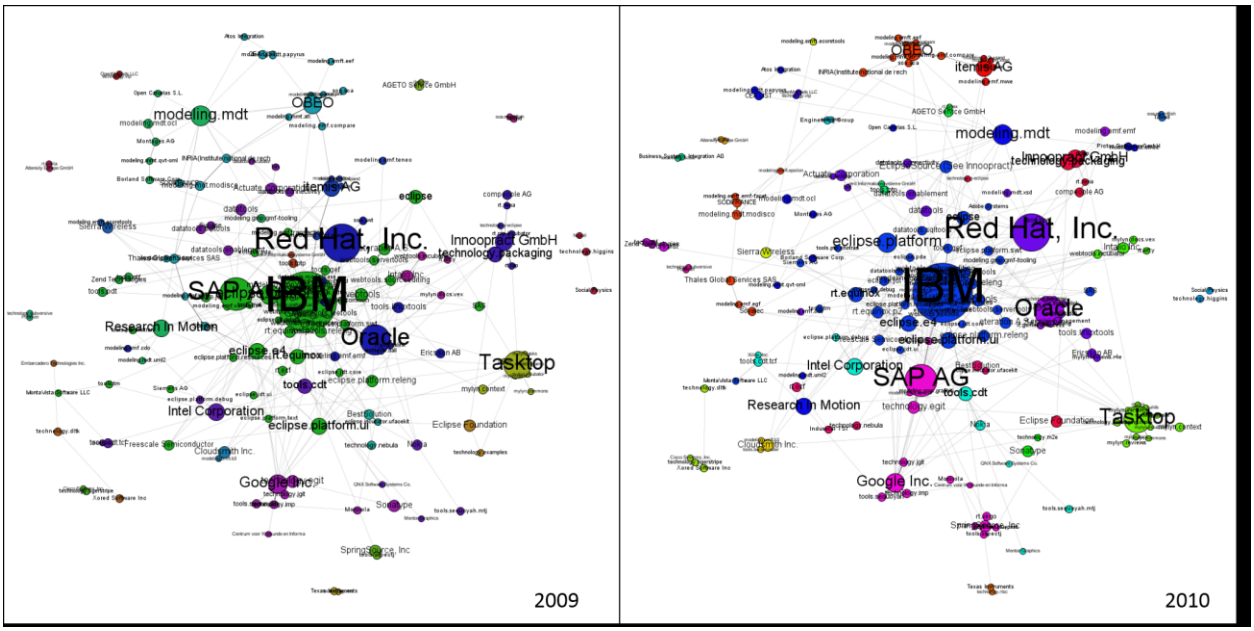


Figure 4.5 The Eclipse project development network (2009-2010)

In 2009, the local development network structure tends to stable. Tasktop, Innoopract, itemis AG, OBEO, IBM still remained the central developer in their own local network. They contributed more or less than before, but generally the network structural positions of these central developers did not change. There were some small changes in the periphery. Cisco lost its only co-developer, Xored. Cisco has been isolated from the main development network, because it does not have any co-developers located in any other local development network in the Eclipse. Google built up its own local network and worked as the central developer. In 2010, Oracle's local community became clear, which means that Oracle contributed the most on the project it was working on. Taxes

Instruments also found its co-developer in this year. Only two local communities were isolated from the main network, Talend and OpenMethods, which still worked on their own independently. In 2010, 33 local communities were built in the Eclipse development network. Central developers built up their local network rapidly since 2006. Till 2011, the central developers were still IBM, SAP, Tasktop, itemis AG, OBEO, and Oracle. In the 2012, based on the size and position of each local community, we can see these local development networks still exist but larger than 2010.The central developers' names are labeled larger than usual. They were Oracle, Red Hat, SAP, Tasktop, Google, Intel, OBEO, itemis AG, Innoopract, Research in Motion.

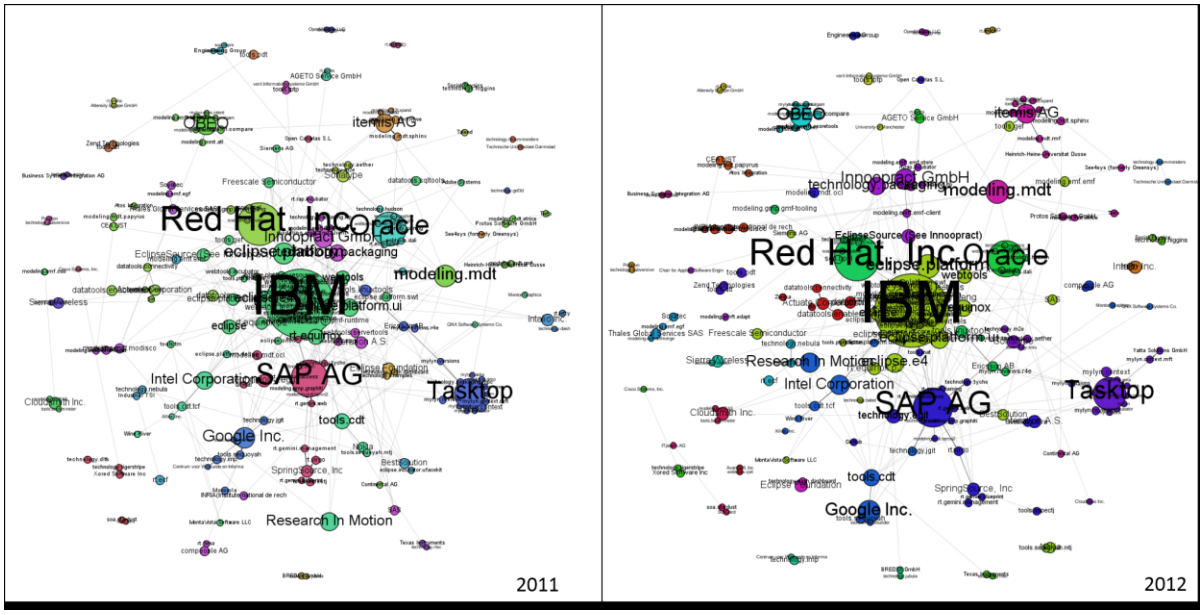


Figure 4.6 The Eclipse project development network (2011-2012)

There are several conclusions we drawn by studying the Eclipse development network over past 12 years. IBM joined first and continuing initiated the most projects. Based on its network structural position, IBM was always the core developer in the Eclipse development network. Central developers always joined at the early stage. They

established their own local development network and made the most contribution on these projects. Their projects were attractive and helped them to find more co-developers. The peripheral developers can move their position close to the core developer or central developers if they contribute more to projects located in other local development network. Companies that established their local networks were the central developer in the local development network; their positions were replaceable by other participants. But in the Eclipse, central developers are not more likely to undergo change at their centers. Surprisingly, most peripheral developer did not leave this community. Most joined peripheral developers still participated to developing projects in 2012.

4.2 Weight Filtering and Most Active Committer

By applying weight filter to generate the company network, we will be able to find the primary project committers. A company can be a committer of only one project with little code contribution. It can also be a primary committer of multiple projects. It will give us an idea if the project a company is working on is attractive to others, and if other companies are willing to contribute a big share of total commitment. For a project with several committers, some committers may commit most of its code, and some other companies commit relatively lower. Weight filter can be used to cut off low contribution committer, in order to find the development relationship among companies who make relatively large code contribution. Furthermore, the weight filtering has been applied to find how many projects a company acts as the primary committer in the second section of this chapter.

4.2.1 Weight Filtering

To reduce the complexity of a large company interaction network, it is necessary to filter the company network for selectively displaying commentary companies network with less dense and complicity. The most important reason of applying the weight filtering is that it allows us to identify if a pair of co-developers are having strong development relationship, in other words, it will tells us if co-developers are both actively working together. Based on this idea, the filter has been introduced because of such reasons. Firstly, the original company network is still dense, that requires filter to reduce complexity. Secondly, it can help us to identify development relationships between companies. In another words, a company with lower contribution to a project will be removed in the network after applying the filters as explained at the beginning of this chapter. However, it is hard to determine what is the percentage of contribution from a company to a project can be counted as "low" contribution. It depends on the original network and database we have here. We found that the percentage of contribution from a company to certain project has been changed over time; the range of percentage of contribution is wide. The contribution from a company to a project can be as low as 0.1% out of total contribution. It can also be as high as 100%. An iterative filtering process has been applied in this study, in order to reduce the network complexity and identify the complementary company network. For a company with 99.9% contribution and a company with 0.1% contribution to a same project, they do not have a strong or healthy complementary relationship. This iterative filtering process is trying to arrive at a clear view of complementary company network with all stronger complementary connections. It is a process of repeating rounds of analysis; it was start with a small number (0.1%) to

cut off all lower percentage of contribution. For each iteration, the filter number increased by a small amount until a clear network got displayed. If the network becomes too thin, we had to reverse back to get desired result.

Because the data metrics is quite large, a software application has been built to automatically run the iterative filtering process. This application is called company network converter that has been described in previous chapter. By comparing with Figure 4.7, it is easy to understand how it was used to filter the network. Input data is a "company to project" contribution percentage metrics, which is a metrics with percentage of contribution from each company to a project calculated in each month. Then by comparing companies' contributions in each month, the application automatically cut off companies with lower contribution in that time. Eventually, the applications set up a new metric with all company to company relationship linked by projects whey are working on. For instance, if Company-A contributed 50% of total code changes on Project-A in April 2001; Company-B contributed 49% of total code changes; Company-C contributed 1% of total code changes. Then the link between Company-A and Company-B will be generated. The links between Company-A and Company-C, Company-B and Company-C will be ignored, if the filter has been set above 1%. Furthermore, If Company-A and Company-B also contributed some other project, and both companies contributed above the filter; the application will count how many projects they are working on simultaneously.

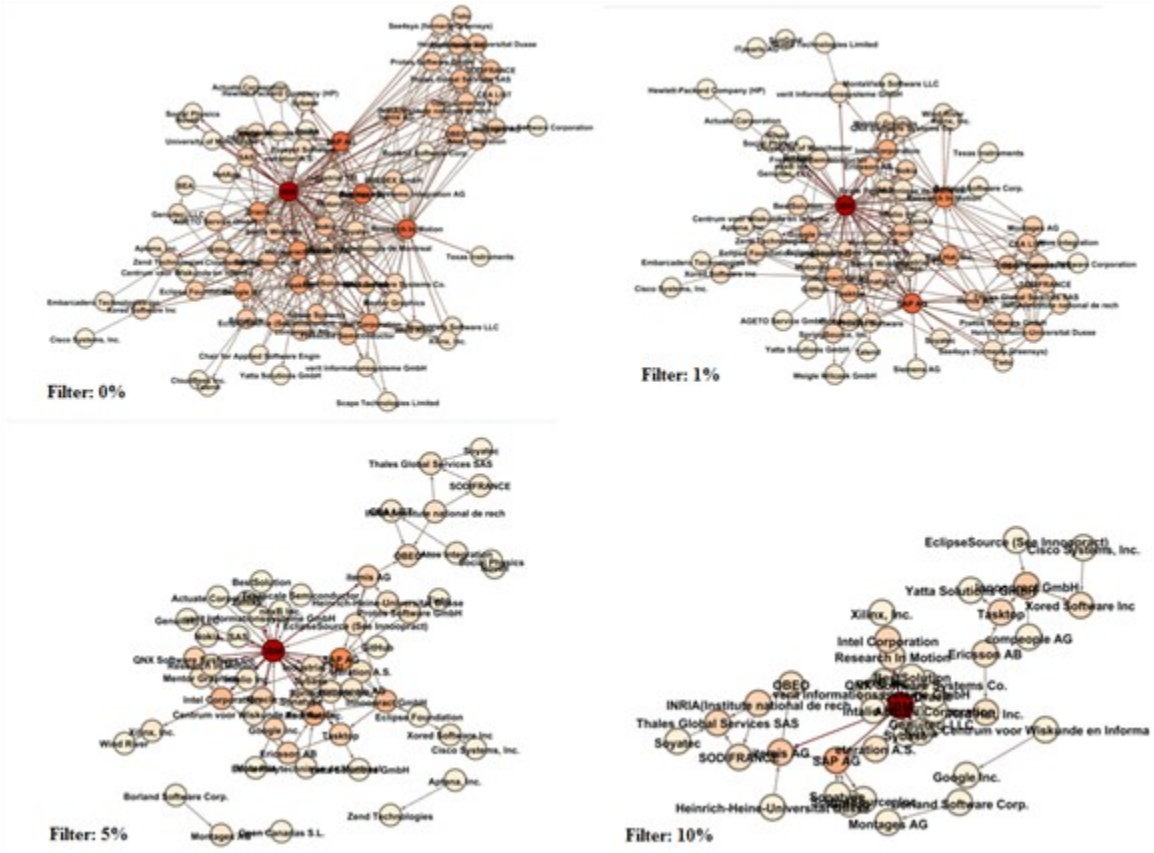


Figure 4.7 The Eclipse company network after applying weight filters

In this study, the filter has been set as 0.5%, 1%, 2%, 3%, 4%, 5%, and so on, till 20% to reduce the complexity of company network. Figure 4.7 above shows the network after applying several typical filters. Without applying any filter, filter coefficient is 0%, we have 375 complementary links between companies. Most of companies are well connected, 89.89% known companies are highly connected in the Eclipse. After applying a small filter coefficient, for instance 5%, the company network only carries 91 complementary links. By raising the filter coefficient, the company network will have less and less complementary links as shown in Figure 4.8. From Figure 4.8, we can see most companies contributed a small amount of total code commitment. It can help us to identify which companies are located in periphery in either global or local network. IBM

has many strong co-developers to work with, which is a typical company located in company network.

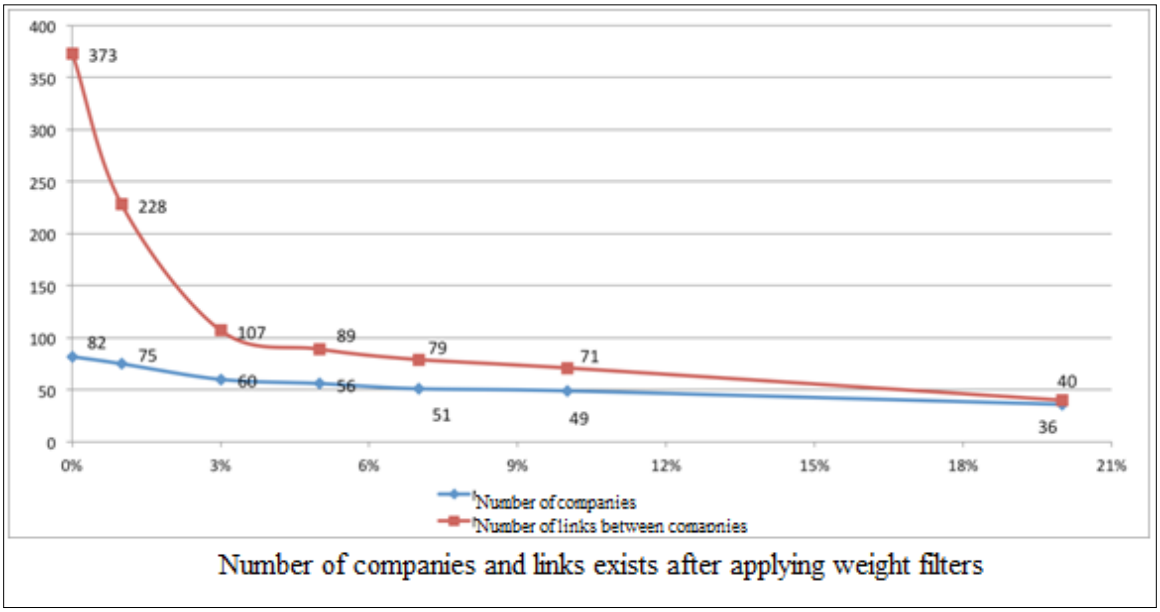


Figure 4.8 Number of companies and links exist after applying weight filters

There are two results that can be easily found by studying filtered company network. For one thing we are easy to see that 82 companies share 373 links between them, averagely each of them has more than 4 co-developers to work with, and many peripheral developers contribute a small amount of total commitment on certain projects helps to improve the project quality. With a relatively high filter value, 10%, all the central developers we studied in project network are still exist in the filtered company network, INRIA, IBM, SAP, Tasktop, Research in Motion, itemis AG, OBEO, Google and so on. That means they get help on the projects they are working on. But by comparing with project network, several peripheral developers are disappeared in filtered company network; those peripheral developers, like OpenMethods, NUXEO, Polarion, could not get enough help from others, but there are only six independent committers who does not

have co-developers to work with. In summary, The Eclipse development network is highly connected, because most development companies are complimenting other's work by contributing on same projects.

4.2.2 Primary Committer

The global Eclipse development company network contains many local development networks. Each of them has one or two central developers located at the center of the local network. We have discussed the core developer, central developers, and peripheral developers at the beginning of current chapter. But it is necessary to find how many projects these companies contributed the most on. By raising the weight filter coefficient, each project's one or two most active committer will be appeared. In the project development network diagrams, the project node is colored by following its most active committer node's color. It is easy to find if a company is contributing the most in its particular local development network.

Here, this section is focusing on studying central developers and core developers in the Eclipse development community, in order to find that how many projects a particular company contributed as a central developer.

A project could have as low as one committer, or has as many as twenty committers in the Eclipse development community. If a company contributes most of code changes for a project by comparing with other committers, then this company is the central developer of this particular project. Each year or even month, a company may have different

position in the development community. We have noticed that it is unnecessary to go through all stages for a company in the community. In the early stage, some big companies directly jumped into network center once they joined the Eclipse development community, because they initiated some projects and were taking some projects leadership while joining the Eclipse's development. Central developers and core developers always have higher connectivity to the development system than general peripheral developers. Additionally, if a company is concurrently working on multiple projects and making most of commitment to those projects would have more connectivity; this company will have stronger relationship than the company that only is working on one project.

Time has wrought a lot of changes in the Eclipse development community. The total number of projects has been continually increasing. Some companies has been transformed from peripheral development company to central ones, some of them becomes to be project leaders on some projects, and some of them remain their network position since they joined. We found that some project initiator stop contribute code to the project they established. SAP AG is a good example we discussed before. This company has quite different network positions in its different stage. This is a clear example to show different roles development member play and how a development member grew in an open source community. SAP AG started to commit code in the Eclipse since 2007. In 2007, SAP a peripheral developer, which SAP was simultaneously contributing to five projects. But SAP had quite small code contribution on four out of five target projects; SAP only committed less than 1% of total code commitment on those projects in 2007. In 2008, SAP started to take some development responsibility in local

network of the Eclipse. SAP was the project initiator, project leader, and biggest contributor on one project called "Eclipse Memory Analyzer"(MAT). But SAP could not find any co-developer on "Eclipse Memory Analyzer" together at that time. Some companies joined to develop "Eclipse Memory Analyzer" in the second year after it got initiated, 2009. So SAP did not have its own local development network in 2008. In 2009, SAP started to contribute more and more, SAP was the biggest contributor on two projects; and initiated a new project is called "EMF Validation". At that time, SAP started to build its local development network in the development community. Because SAP found co-developers on both projects SAP was leading on. After that, SAP takes more and more project development work in the Eclipse development community. SAP was the central developer in its local development network. SAP was also the biggest contributor on 4 projects in 2010, 8 projects in 2011, and 9 projects in 2012. The example SAP AG has been chose to clearly show how a peripheral company got more and more involved in project development and eventually became a central developer. Some of companies directly became one of the biggest code contributors once they joined the development network. For instance, Tasktop was leading five projects in 2005, the same year Tasktop joined the Eclipse development community.

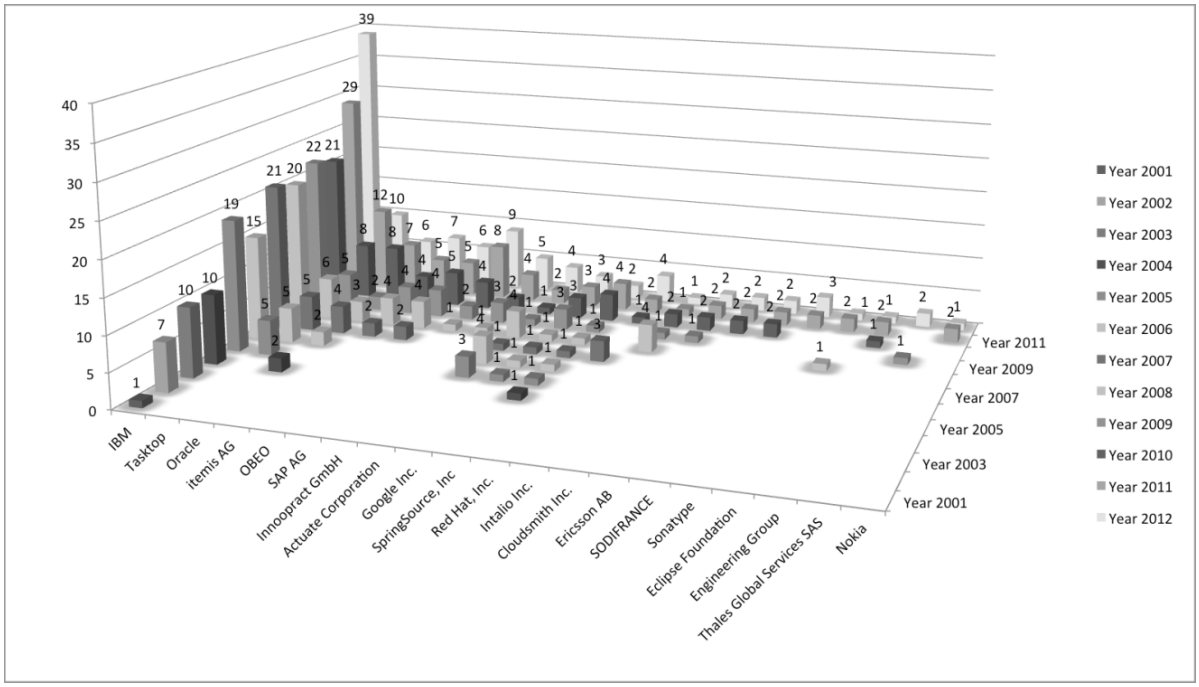


Figure 4.9 The biggest Committers and number of project they most contributed

Until the end of 2012, there are 56 known companies have been contributing the most on their projects, and each of them has led at least one project since 2001. Figure 4.9 shows how many projects that a particular company contributed the most during 2001 to 2012. For example, IBM was 39 projects' biggest contributor in 2012. Figure 4.9 only shows companies that have led two or more projects. There are twenty companies have been simultaneously contributing most on multiple projects. These companies have stronger relationship than other companies. In past 12 years, most of companies have been leading or contributing most on more and more projects. All of the 20 companies were still central developers in 2012. By comparing with development companies involved in the Eclipse development network, we found that companies that were leading or contributing most on multiple projects never left the Eclipse development community. This result confirmed following theory; it is depended on how this member interacts with others.

Members with low connectivity to the system have less commitment to the platform, increasing the risk that the member switches to another ecosystem (Hartigh et al., 2006).

But several companies that were contributing most to one project eventually left this community. For these 20 companies shown in Figure 4.9, they had most commitment to the Eclipse platform, which decrease the risk for them to switch to another ecosystem. They have significant strong relationships in this ecosystem. In the 20 companies, there are seven companies has even relatively stronger relationships in the Eclipse community. They are IBM, Tasktop, Oracle, itemis AG, OBEO, SAP AG, and Innoopract GmbH. They have been described as central developers at the very beginning of current chapter also. In 2012, each of them contributed most to more than 5 projects. These seven companies were share the most work on developing those 82 projects, which is almost half number of total projects amount.

4.3 Degree Distribution and Eigenvector Centrality

In Section 4.2.1we explained how the project network has been converted into company network. In the section 4.2.2, we used the company network to discuss which companies have more co-developers to work with and their positions in company network. Degree centrality is used to indicate how many co-developers of a company have. In the section 4.2.3, we used eigenvector centrality techniques as an extension of degree centrality to measure if a company had stronger co-developer to work with, and if they were located in a healthy development environment (Grewal, 2006).

4.3.1 Company network

Figure 4.10 includes several typical networks. Circles represent companies and squares represent projects. In the top left network shown in Figure 4.10, if a project only has one committer, it means this committer does not have any co-developer to work with. The top right diagram shows how the converter works. Considered three co-developers concurrently working on a same project, Company-1 has relatively small contribution to the project, and then only Company-2 and Company-3 are linked if we applied a weight filter 1%. If we do not apply any filter, all three companies will be linked each other. Compared two diagrams at the bottom, the right diagram shows the network has relatively stronger network than the one in the left diagram. Each two companies are concurrently working on two projects in the right network. If a particular company stops to contribute to a certain project, this company network still exists. However, as the network shown in the left bottom diagram, if a particular company stops to contribute to a

certain project, the current company network will no longer exist, because each pair of co-developers work on only one project.

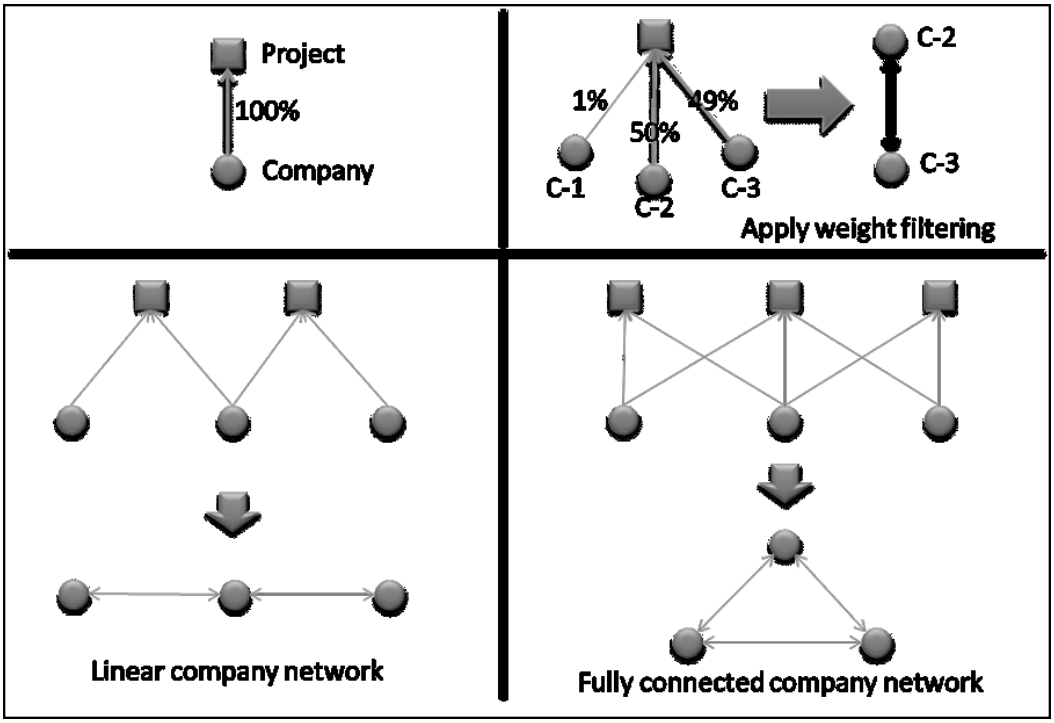


Figure 4.10 Company network conversion

The project network can be converted into company network by using the application we built for this study, which has been described in Section 3.3.5. The further study in Section 4.4 requires company networks to analysis centrality values of companies. Figure 4.11, Figure 4.12 and Figure 4.13 represent the Eclipse development company network during 2001 to 2012 annually. There are another five companies are not included in the company network, because they were independent players with no co-developers as we described in project networks.

Figure 4.11The Eclipse development company network (2001-2004)

In 2001, the company network looked like the network diagram shown in the top right of Figure 4.9. IBM and Oracle was a couple of co-developers, concurrently working on two projects. In 2002, after six new development companies joined in, the network had a dramatic change. There were actually there local community, but only two are shown in the Appendix-B. The local company network established by QNX, Research in Motion, Mentor Graphics was a fully connected network as shown in the bottom right network diagram in Figure 4.9, although this local company network was not that strong, because they were only working on one project. Genuites joined into IBM and Oracle's local company network, but Genuites did not work on the projects that Oracle was working on, so these two companies was not a couple of co-developers, that is the reason there was no

link between them. This local company network is like a linear company network shown in the bottom right in Figure 4.9.

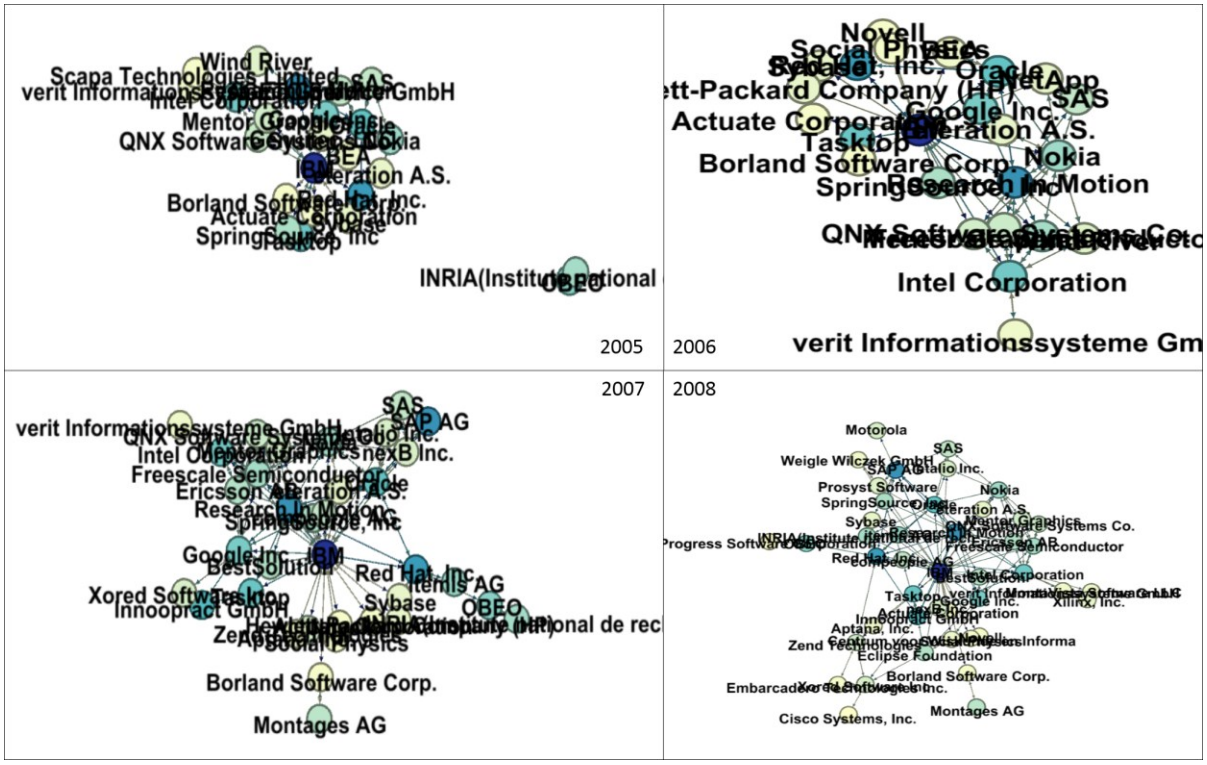


Figure 4.12 The Eclipse development company network (2005-2008)

By comparing with the project networks explained in Section 4.1, we can see the main network's center is IBM geophysical. Many isolated local company network gets involved into the main network. SpringSource, Tasktop have been merged into main company network in 2004; INRIA was isolated from the main network in 2005, but get involved into the main network eventually. The reason has been discussed before, core developers and central developers give supports and co-developed these new projects, which help independent developers get involved into the main network. The company networks used heightening color to display who has more degree (more co-developers to work with). It confirms the central developers we analyzed with the project development

networks. In most recent years, IBM, SAP, Oracle, Red Hat, OBEO, itemis AG, and Research in Motion are shown with relative dark color and surrounding by other companies. For instance, IBM has as many as 53 links connected with, which means IBM has been working with 53 co-developers to work with. We can also see these central developers also have co-development relationship with other central developers.

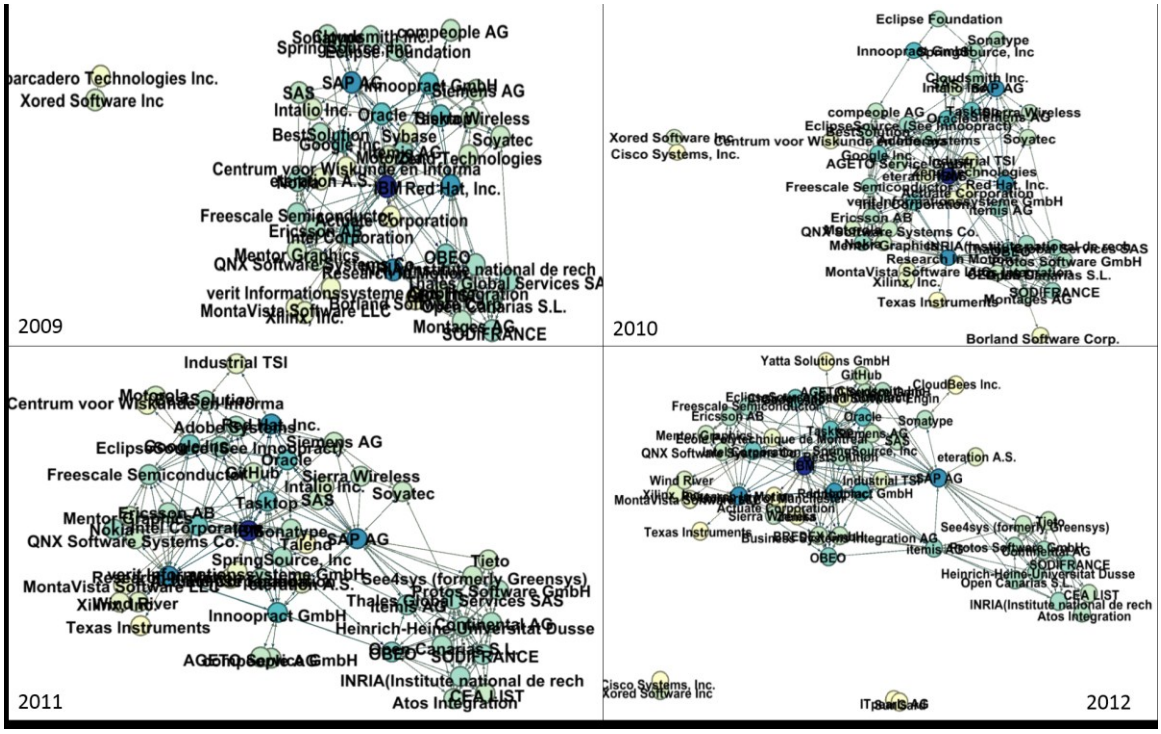


Figure 4.13 Eclipse development company network (2009-2012)

4.3.2 Degree Distribution

We discussed the importance of the co-developers role in the development community. It is necessary to find how many co-developers that a company has, and how they connect together.

Degree centrality can simply tell us the number of co-developers that a particular development member had in the Eclipse development community in the past. For each development company, its degree centrality value has been measured annually. Most of them had more co-developers to work with during past 12 years. Their degree centrality values have been calculated and represented in Table 4.1. A company can find co-developer and build strong development relationship if it chooses to work on a popular project. For instance, a company called Continental AG contributed code to a project called “modeling.mdt”, which helped this company found 12 co-developers in 2010. Or a company can gain higher degree centrality value if it works on many projects, like IBM does. But at least one of the projects attracts co-developers.

We also need to understand how those companies got connected when they were concurrently working on the same projects. Degree distribution at this point helps us to clearly see how those companies were connected and how the company network got established. By looking at the overall degree centrality table of this company network in Appendix-C, we can find most of companies, 73 out of 82, had at least two co-developers to work with. That means most of them can find multiple co-developers on one or more projects. Figure 4.14 shows the Eclipse development company network overall. Excluding the local network SunGard and ITpearls AG built, and five other independent developers, most development members are more or less connected. Degree filtering has been applied here to analyze which companies have more co-developers and how they connected. By comparing with Appendix-C, we can see the central developers we discussed in project network are shown in the top of the table. If we set degree filter as 15, there are 11 companies left in the network, but they are not fully connected, that is not a

bag sign, because they have different projects to work on. SAP, Research in Motion, IBM, and Red Hat, the top four ranked companies are fully connected in the company network. That simply tells us, even they have their own local development communities and interesting projects, they still co-create, co-develop some concurrently interesting projects. These high degree members connect to each other and to some of the low degree members at the same time.

Companies have been changing their positions over time. In the company network diagram, a company is located closer to the center if this company has more development relationships with others, we can also call it co-development. The core development companies should be at the center of company network collusion. As we described in the literature review, the central companies locate off-centered surrounding by peripheral developers who also have same development interests. So it is necessary to find development companies' positions and categorize them in to different group. In other words, we need to find which companies have more influence in local network or even whole network. Based on prior studies we discussed earlier, a company located close to the central network has more influence in the development network.

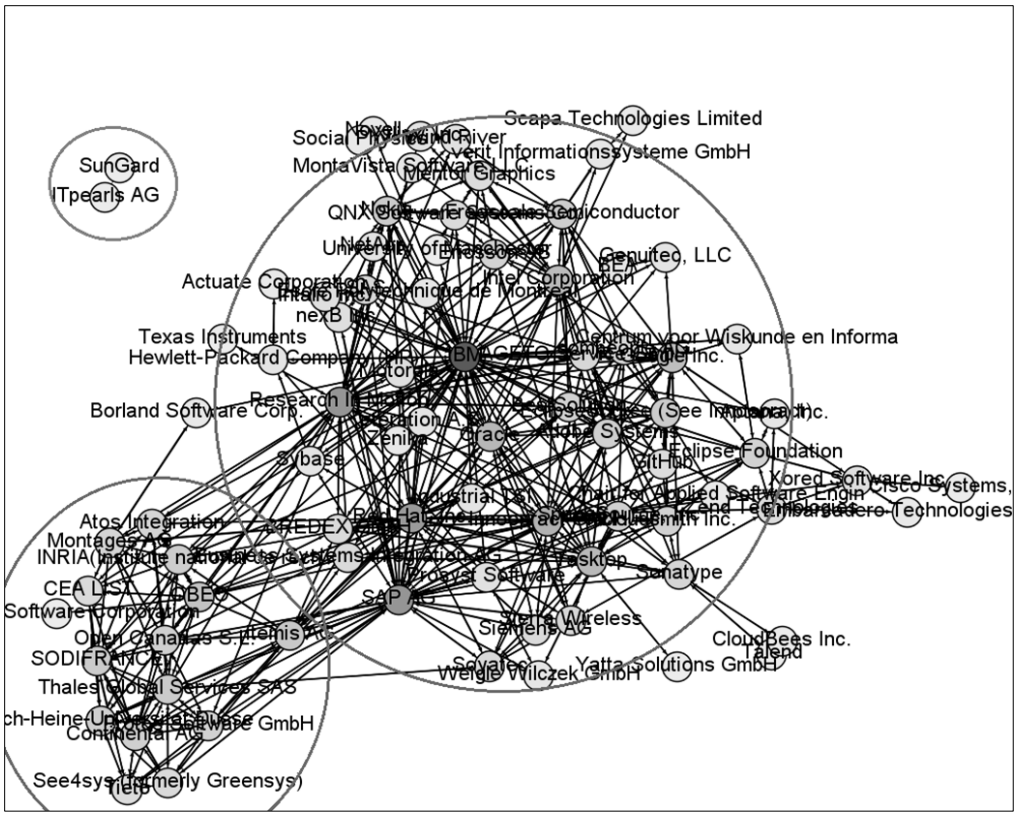


Figure 4.14 The Eclipse development company network (2001-2012)

Table 4.1 Degree centrality ranking table

Company	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	Overall
IBM	1	2	5	9	17	20	25	31	30	32	27	29	53
Research In Motion		2	4	7	9	11	14	18	16	21	15	17	34
SAP AG							4	11	11	13	26	22	34
Red Hat, Inc.				1	3	4	7	11	18	18	10	19	32
Innoopract GmbH							3	9	12	3	10	13	24
Oracle	1	1	2	3	8	8	9	11	13	11	12	10	22
Tasktop			1	1	2	1	3	3	9	12	15	16	21
Intel Corporation				5	7	7	7	12	13	13	17	15	20
Google Inc.					5	3	8	9	11	13	12	9	19
OBEO					1	1	2	3	9	11	12	7	17
itemis AG							1	3	4	3	7	14	15
Nokia				2	4	10	11	10	10	7	6		14
EclipseSource (See Innoopract)										11	10	11	14
SODIFRANCE									6	8	11	10	14
Thales Global Services SAS									7	9	12	1	14
Freescale Semiconductor						7	8	8	9	12	11	9	13
INRIA(Institute national de rech					1	1	1	5	9	9	8	6	13
SpringSource, Inc			1	2	2	2	3	6	4	6	6	6	12
Open Canarias S.L.									7	8	8	7	12
Adobe Systems										12	10		12
Continental AG											12	10	12
Heinrich-Heine-UniversitatDusse											12	10	12
BestSolution							3	3	4	6	7	7	11
Ericsson AB							8	8	8	8	10	9	11
Cloudsmith Inc.									6	7		7	11
Protos Software GmbH										5	8	8	11
Eclipse Foundation						2		5	4	2	3		11

Company	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	Overall
Atos Integration									10	10	6	4	10
Montages AG							1	1	10	10			10
Sonatype									4	3	6	4	10
SAS				1	4	4	5	4	4	4	4	3	9
Sierra Wireless									8	4	4	1	9
compeople AG							3	4	1	7	1		9
AGETO Service GmbH					6					2	1	4	9
QNX Software Systems Co.		2	3	4	4	6	7	7	7	7	7	7	8
Siemens AG									6	7	7	3	8
Soyatec									6	7	4	1	8
CEA LIST									5	5	6	5	8
See4sys (formerly Greensys)											8	7	8
BREDEX GmbH												8	8
Business Systems Integration AG												8	8
Mentor Graphics		2	3	4	4	6	7	7	7	7	7	6	7
Zend Technologies						2	2	5	3	1			7
GitHub											5	6	7
Tieto											6	6	7
Intalio Inc.							6	5	5	4	4		6
nexB Inc.							6	1					6
Xored Software Inc							2	5	1	1	1	1	6
Motorola								1	3	3	1		6
Wind River				2	2	4					3	3	5
Prosyst Software								5					5
eteration A.S.				2	2	2	2	2	2	1	3	1	4
Sybase					2	3	3	3	3				4
Genuitec, LLC		1	1	1	4								4
Xilinx, Inc.								2	2	3	4	4	4
Hewlett-Packard Company (HP)						4	4						4

Company	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	Overall
Industrial TSI										1	2	4	4
NetApp						4							4
MontaVista Software LLC								2	2	3	3	3	3
BEA					3	2							3
veritInformationssysteme GmbH					2	1	1	1	1	1	1		3
Centrum voorWiskunde en Informa								3	2	1	1		3
Aptana, Inc.						2	2	2					3
EcolePolytechnique de Montreal												3	3
WeigleWilczek GmbH								3					3
Actuate Corporation					1	2	2	1	1	1	1	1	2
Borland Software Corp.					1	1	2	2	2	1			2
Novell						2	2	2	1				2
Social Physics						2	2	2	1				2
Embarcadero Technologies Inc.								2	1				2
Chair for Applied Software Engin												2	2
Scapa Technologies Limited					2								2
Zenika												2	2
Cisco Systems, Inc.								1		1	1	1	1
Texas Instruments										1	1	1	1
Yatta Solutions GmbH												1	1
CloudBees Inc.												1	1
ITpearls AG												1	1
Progress Software Corporation								1					1
SunGard												1	1
Talend											1		1
University of Manchester												1	1

4.3.3 Eigenvector Centrality

Eigenvector centrality is an extension of degree centrality (Adamic, 2007), which allows us to rank developers by their activity and their position in the network (Martinez-romo, Robles, Gonzalez-barahona &Ortun, 2008). Degree centrality was used to help companies to find how many co-developer they have. In this section, Eigenvector centrality will be used to find if their co-developers have many co-developers. Specifically in the Eclipse development network, Eigenvector centrality measure is used to find if companies are located in a healthy development network. It assigns scores to all members in the Eclipse development network; a company can get a raised score if this company is connected to high-scoring companies. The Eigenvector degree of a company in the network indicates if this company is connected to relatively stronger co-developers. The basic idea is very simple; a node in the network can indirectly influence other companies. A highly influential company may only impact one or two companies, but impacted companies may also influence others. Not all neighbors are equally important in the network, connecting to more central nodes increases importance of members. Based on this idea, previous study indicated that the Eigenvector centrality measurement gives us good view about project leadership and information control flows for each member in the development network.

A member's eigenvector centrality value can be large if it has many neighbours or it has important neighbours. In year, the eigenvector centrality value of each company in the Eclipse development network will be measured separately, which will be a relative number. The score "1" will be assigned to the company that has the highest eigenvector centrality value.

In Table 4.2, we can see changes of the eigenvector centrality value of each company changes during 2001 to 2012. The network at the early stage is always a good sample to explain its relationship inside. In 2001, there were only two companies in the network, which were co-developing several projects, so both Oracle and IBM had eigenvector centrality value as “1” at that time. In 2002, there were three local development networks. In the local development network established by Research in Motion (RIM), QNX, and Mentor Graphics, the companies were fully connected; each of them had two co-developers. So they had eigenvector centrality score as "1". In IBM's local development network, IBM had higher eigenvector centrality value than Oracle and Genuitec, because IBM had more co-developer than them. Tasktop did not have any co-developer at that time, so Tasktop does not have Eigenvector centrality score shown in the table. Once companies like RIM and QNX have development relationship with IBM or Oracle located in the main local development network, they will have more accurate relative value. Since 2003, IBM always has the highest eigenvector centrality value in the Eclipse development network, because IBM has most co-developer by comparing all members in the network. In 2004, the new joiner Intel had relative high eigenvector centrality value, because Intel had the only but important co-developer IBM. That is the reason why a peripheral company could get such high eigenvector centrality. So this result tells us a company with relative high eigenvector centrality may not necessarily contribute a lot, but it must be located in a healthy position, which is working with important companies or having many co-developers. Intel is not a special example, any company can get such benefits by joining the network with positioned in a healthy place. There were some companies had very low eigenvector centrality value, such as INRIA, OBEO. In 2005,

INRIA's eigenvector centrality value was 0.003. Because these two companies had a local development network that was isolated from the main network, both of them did not have any neighbor's neighbors. So they could not get any benefits from the main network, such as information transformation, or development support. This question will be able to answer in section "Betweenness Centrality & Closeness Centrality". It is also easy to find that late joiners may also have high eigenvector centrality value. Adobe joined into the Eclipse development network in 2010, but in overall, this company has been ranked as the 14th company in eigenvector centrality table. Because Adobe joined and worked with many relatively stronger and important co-developers, such as Google, Intel, Tasktop, Innoopract, RIM and so on. Working with these important co-developers gives Adobe a relative healthy position. By comparing the results from analyzing the network evolution, in 2012, the central developers have relatively higher eigenvector centrality value.

In summary, by measuring a member's eigenvector centrality value, we are able to find the importance of each member located in the Eclipse development network. It indicates if a member either has many co-developers to work with, or has relatively stronger co-developer, which tells us if such company is located in a healthy position. It also verified the project initiators and central developers may have relatively higher eigenvector centrality value, but late joiners can also have high eigenvector centrality value if such members are working with strong and important members. So the analysis in this section confirms some results we found from analyzing the network structure and evolution. But it cannot tell us how much contribution of each member made on the projects such member was working on by comparing with all co-developers, which we will be able to answer after the study in next section.

Table 4.2 Eigenvector centrality ranking table

Company	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	Overall
IBM	1.000	0.243	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Research In Motion		1.000	0.932	0.916	0.733	0.797	0.773	0.797	0.691	0.729	0.768	0.640	0.804
SAP AG							0.174	0.437	0.396	0.453	0.845	0.765	0.778
Red Hat, Inc.				0.095	0.297	0.278	0.330	0.490	0.730	0.732	0.757	0.770	0.768
Innoopract GmbH							0.089	0.424	0.481	0.077	0.383	0.601	0.650
Oracle	1.000	0.172	0.568	0.435	0.660	0.553	0.513	0.533	0.550	0.495	0.481	0.468	0.610
Tasktop			0.012	0.053	0.191	0.138	0.220	0.229	0.417	0.496	0.634	0.691	0.589
Intel Corporation				0.803	0.472	0.544	0.470	0.538	0.539	0.574	0.611	0.620	0.527
Google Inc.					0.529	0.321	0.454	0.445	0.446	0.505	0.474	0.425	0.487
OBEO					0.003	0.002	0.043	0.084	0.429	0.459	0.559	0.389	0.445
itemis AG							0.040	0.209	0.229	0.197	0.370	0.554	0.441
EclipseSource (See Innoopract)										0.521	0.471	0.504	0.421
INRIA(Institute national de rech					0.003	0.002	0.007	0.301	0.427	0.415	0.498	0.147	0.384
Adobe Systems										0.542	0.469	0.469	0.374
SpringSource, Inc			0.012	0.231	0.191	0.245	0.236	0.299	0.171	0.218	0.330	0.352	0.370
Cloudsmith Inc.									0.225	0.318	0.333	0.335	0.358
Freescale Semiconductor						0.622	0.581	0.469	0.424	0.529	0.443	0.403	0.353
Ericsson AB							0.581	0.469	0.407	0.391	0.439	0.406	0.348
BestSolution							0.262	0.231	0.193	0.306	0.315	0.343	0.342
Open Canarias S.L.									0.306	0.323	0.462	0.229	0.336
Thales Global Services SAS									0.264	0.320	0.502	0.001	0.324
SODIFRANCE									0.242	0.297	0.488	0.244	0.322
Sierra Wireless									0.375	0.183	0.188	0.088	0.321
Nokia				0.279	0.355	0.736	0.660	0.485	0.417	0.314	0.262	0.262	0.319
BREDEX GmbH												0.445	0.316
Business Systems Integration AG												0.445	0.316
Atos Integration									0.454	0.443	0.444	0.072	0.309
compeople AG							0.236	0.260	0.047	0.313	0.277	0.277	0.303
Protos Software GmbH										0.218	0.401	0.259	0.297
AGETO Service GmbH					0.578	0.578	0.578	0.578	0.578	0.136	0.145	0.196	0.288
Sonatype									0.122	0.091	0.253	0.179	0.286

Company	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	Overall
SAS				0.196	0.366	0.332	0.242	0.179	0.156	0.194	0.196	0.197	0.276
QNX Software Systems Co.		1.000	0.803	0.739	0.417	0.579	0.533	0.428	0.368	0.349	0.298	0.363	0.276
Siemens AG									0.269	0.286	0.283	0.168	0.273
Eclipse Foundation						0.008	0.008	0.224	0.128	0.029	0.183	0.183	0.272
Continental AG											0.417	0.283	0.266
Heinrich-Heine-UniversitatDusse											0.417	0.283	0.266
Motorola								0.046	0.209	0.140	0.118	0.118	0.263
Soyatec									0.245	0.272	0.285	0.001	0.257
Montages AG							0.016	0.012	0.378	0.363	0.374	0.374	0.254
GitHub											0.266	0.300	0.248
Mentor Graphics		1.000	0.803	0.739	0.417	0.579	0.533	0.428	0.368	0.349	0.298	0.301	0.230
CEA LIST									0.258	0.243	0.325	0.101	0.208
Intalio Inc.							0.349	0.273	0.244	0.194	0.196	0.196	0.202
nexB Inc.							0.349	0.104	0.104	0.104	0.104	0.104	0.202
Prosyst Software								0.274	0.274	0.274	0.274	0.274	0.193
See4sys (formerly Greensys)											0.285	0.239	0.184
Industrial TSI										0.090	0.161	0.253	0.183
eteration A.S.				0.279	0.216	0.237	0.195	0.153	0.136	0.090	0.173	0.068	0.174
Sybase					0.207	0.212	0.184	0.201	0.205	0.205	0.205	0.205	0.170
Zend Technologies						0.008	0.136	0.185	0.202	0.090	0.078	0.078	0.165
Tieto											0.223	0.194	0.157
NetApp						0.358	0.358	0.358	0.358	0.358	0.358	0.358	0.140
EcolePolytechnique de Montreal												0.190	0.134
Wind River				0.366	0.191	0.363	0.363	0.363	0.363	0.363	0.118	0.123	0.134
Hewlett-Packard Company (HP)						0.233	0.200	0.200	0.200	0.200	0.200	0.200	0.128
Genuitec, LLC		0.172	0.295	0.216	0.383	0.383	0.383	0.383	0.383	0.383	0.383	0.383	0.120
Centrum voorWiskunde en Informa								0.174	0.139	0.046	0.037	0.037	0.112
Zenika												0.155	0.112
BEA					0.323	0.214	0.214	0.214	0.214	0.214	0.214	0.214	0.110
Xored Software Inc							0.066	0.096	0.001	0.001	0.001	0.001	0.103
veritInformationssysteme GmbH					0.094	0.073	0.054	0.055	0.052	0.090	0.078	0.078	0.100

Company	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	Overall
Xilinx, Inc.								0.063	0.058	0.129	0.127	0.134	0.099
Aptana, Inc.						0.008	0.136	0.125	0.125	0.125	0.125	0.125	0.092
MontaVista Software LLC								0.063	0.058	0.129	0.118	0.123	0.091
WeigleWilczek GmbH								0.106	0.106	0.106	0.106	0.106	0.085
Borland Software Corp.					0.159	0.138	0.122	0.106	0.133	0.033	0.029	0.029	0.080
Actuate Corporation					0.159	0.172	0.143	0.104	0.096	0.090	0.078	0.088	0.072
Novell						0.162	0.136	0.117	0.001	0.001	0.001	0.001	0.069
Social Physics						0.162	0.136	0.117	0.001	0.001	0.001	0.001	0.069
Chair for Applied Software Engin												0.096	0.067
University of Manchester												0.088	0.064
Texas Instruments										0.066	0.060	0.056	0.051
Scapa Technologies Limited					0.094	0.094	0.094	0.094	0.094	0.094	0.094	0.094	0.040
Yatta Solutions GmbH												0.060	0.037
Progress Software Corporation								0.010	0.010	0.010	0.010	0.010	0.028
Embarcadero Technologies Inc.								0.033	0.001	0.001	0.001	0.001	0.018
CloudBees Inc.												0.016	0.018
Talend											0.020	0.020	0.018
Cisco Systems, Inc.								0.012	0.012	0.001	0.001	0.001	0.007
ITpearls AG												0.001	0.001
SunGard												0.001	0.001

4.4 Betweenness Centrality and Closeness Centrality

From the perspective of the global Eclipse development networks, this section focuses on the analysis of the betweenness centrality and closeness centrality of the Eclipse development community, to measure each member's centrality and analyze the impact of the success of the Eclipse project development community. Methodologically, depending on which specific structural features the researcher wants to illuminate, the suitable type of centrality should be measured. Betweenness centrality and closeness centrality are two important network characteristics, which has been commonly used to measure open source development community. The centrality describes the level of connectivity of a company located in a network. The general discussion on how development members involved in a network is to look at their centrality scores, members located in the center has relatively higher score and others are located in peripheral network.

4.4.1 Betweenness Centrality

Basically, betweenness centrality measures the extent to which the development company acts as the connecting link among other projects and other companies in the development network.

A development member with a higher betweenness centrality is able to spread its various tasks (bug fix or feature delivery) over a larger number of co-developers, leading to increased technical success and improved efficiency and quality in serving users' needs (Toral, Martínez-Torres, Barrero, & Cortés, 2009). Companies with relatively higher betweenness centrality are working closely within the development network or clustering of mass projects, these strong relationships benefit the project through the exchange of

information and knowledge (Rowley et al., 2000). On the other hand, there is no shortest path to peripheral members in the network (Newman & Girvan, 2004). Betweenness centrality is based on a company's network position to measure the number of shortest paths to other companies or projects passing through this member, which indicates the ability of this company to interact in the control of others through an intermediary path to share information or tasks (Huang, Li, & He, 2013).

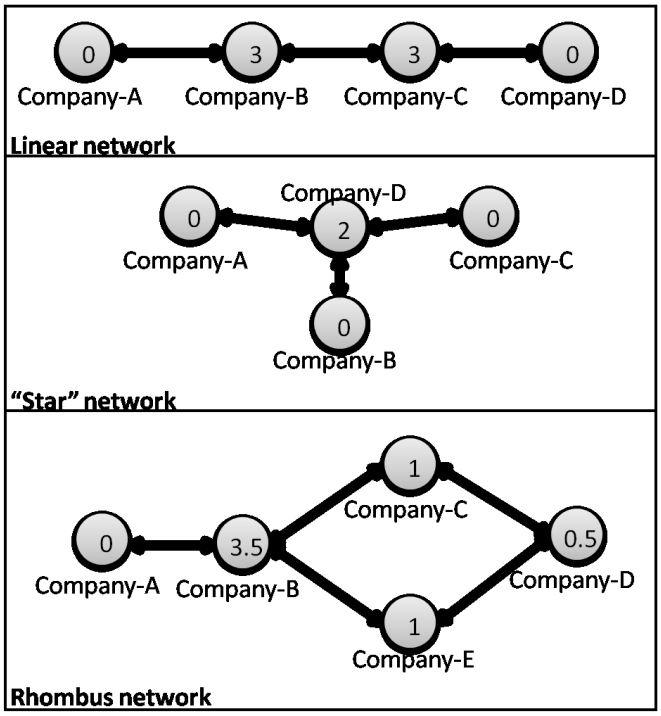


Figure 4.15 Betweenness centrality measurement

If a local network is like a linear network shown in Figure 4.15, Company-A does not lie between two other companies, company-B lies between company-A and two others. We can see company located in the periphery does not have betweenness centrality value. If a local network is a "star" network, the central node plays very important role. There are no alternate paths for these pairs to take, in other words, Company-D in the "star" network take the responsibility to pass information or knowledge to others. If a local network is a

"Rhombus" network, Company-B has two alternative ways to pass information to Company-D. Specifically in the Eclipse development network, we saw a mixed network, which contains all different types of local networks. Then calculating the betweenness centrality is quite difficult in such network. Excluding the six companies who do not have any co-developers, the company network has 22 companies that are located in the periphery has zero betweenness centrality value. This means these companies have hard time to pass and gain information in the network.

In this research, we consider the number of times the Eclipse development member appears on the shortest path between any two projects in the network. IBM has been ranked as the first company who has betweenness centrality value is 2209.929, which is almost triple of 796, the second company SAP AG has. That means IBM connects other companies. If the companies that IBM connected with are all connected, IBM should not have such high betweenness centrality value. It indicated that IBM is working on many projects that have advantages to share information or task with a large number of its co-developers. By analyzing the complete betweenness centrality ranking list, company that has higher value plays relatively more important role between other pairs. By comparing each two adjacent companies in the table, we can find that betweenness centrality that between IBM and SAP AG has the biggest differences. Below the SAP AG, the next company always has more than half of betweenness centrality value of previous company has. It confirms that IBM made relatively more contributions to other companies and it plays a crucial role on passing information through the global network.

Table 4.3 Beteenness centrality ranking table

Company	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	Overall
IBM		2	10	102	275.1	333.7	693.24	1018.72	743.22	1013.43	926.5	647.91	2209.93
SAP AG								153.68	63.66	110.12	499.48	687.54	796.15
Research In Motion			2	42	51.23	44.47	114.12	152.58	156.67	408.18	512.69	224.79	762.8
Red Hat, Inc.					1.4	2.8	195.8	172.9	216.14	234.62	205.12	240.1	565.2
Innoopract GmbH							2.67	109	152.54	14.2	49.61	108.1	370.53
Intel Corporation				8	87.33	49.67	66	257.95	269.91	99.42	187.83	216.77	353.58
Sonatype									1.17		109.62	96	313.12
Tasktop							21.77		41.13	97.04	102	171.91	251.54
Google Inc.					1.67		106.36	21.3	75.35	170.48	184.37	32.15	233.97
OBEO							66	84	18.46	45.82	52.43	4.83	232.29
Xored Software Inc								97.33					198.41
Oracle				24	20.47	22.1	83.49	65.88	92.48	32.62	48.98	51.95	197.65
Zend Technologies								109.12					143.34
itemis AG									2.06		19.95	243.24	90.38
SpringSource, Inc				24				23.25	5.69	92.84	15.95	1.88	70.55
EclipseSource (See Innoopract)										19.71	21.14	48.16	64.51
Eclipse Foundation								34.55	5.22	1.5			64.01
Nokia					5.9	32.63	45.36	32.1	41.54	10.77	9.61	9.61	57.4
INRIA								46.11	22.65	27.56	35.25	23.28	48.47
Thales Global Services SAS									10.45	18.7	53.58		40.08
Freescale Semiconductor						6.33	7.49	3.88	16.6	32.18	26.5	9.69	33.49
SODIFRANCE									0.29	2.77	28.17	62.28	31.94
Montages AG									22.04	101.41	112.61	112.61	28.06
Ericsson AB							7.49	3.88	5.3	3.7	15.97	13.79	26.51
Adobe Systems										42.65	34.14	34.14	26.44

Company	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	Overall
Protos Software GmbH										2.32	24.46	3.08	25.74
verit Informationssysteme GmbH													23.38
Cloudsmith Inc.									27.62	30.88	11.57	11.34	22.72
BestSolution									12.05		7.77	12.53	21.45
CEA LIST									1.97	7.62	17.01	1	20.7
Atos Integration									26.76	33.96	17.16	0.9	20.05
Sierra Wireless									14.63	6.75	4.48		17.7
Open Canarias S.L.									5.36	4.21	16.49	43.03	16.37
Prosyst Software								18.48	18.48	18.48	18.48	18.48	16.01
AGETO Service GmbH					7.23	7.23	7.23	7.23	7.23		0.8	1.08	15.2
SAS					1.83	0.5	1.73	1.02	0.68				14
Continental AG											11.58	39.88	11.38
Heinrich-Heine-Universitat Dusse											11.58	39.88	11.38
Soyatec									15.74	24.22	9.69		10.75
compeople AG								1.98		36.2	3.8	3.8	4.57
Hewlett-Packard Company (HP)						2	5	5	5	5	5	5	4.14
GitHub											1.55	8.26	3.49
Borland Software Corp.							66	84	1.97				3.29
Wind River						0.67	0.67	0.67	0.67	0.67			3.12
BREDEX GmbH												6.61	2.57
Business Systems Integration AG												6.61	2.57
Sybase													2.5
Motorola									0.33	0.8	0.8	0.8	2.28
QNX Software Systems Co.						0.67	1.07	1.14	1.14	0.33	0.62	2.59	2.2

Company	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	Overall
Siemens AG									1.42	4.67	3.96		2.19
eteration A.S.													2.13
Genuitec, LLC					1.33	1.33	1.33	1.33	1.33	1.33	1.33	1.33	1.9
See4sys (formerly Greensys)											1.9	2.35	1.27
Intalio Inc.							18.68	8	4.63				0.95
nexB Inc.							18.68						0.95
Xilinx, Inc.											0.67	0.67	0.67
BEA					0.5								0.64
Industrial TSI												1.2	0.64
Mentor Graphics						0.67	1.07	1.14	1.14	0.33	0.62		0.47
Ecole Polytechnique de Montreal												0.62	0.29
NetApp						3.8	3.8	3.8	3.8	3.8	3.8	3.8	0
Actuate Corporation, MontaVista Software LLC, Tieto, Centrum voor Wiskunde en Informa, Novell, Social Physics, Aptana, Inc., Cisco Systems, Inc., Embarcadero Technologies Inc., Texas Instruments, Weigle Wilczek GmbH, Chair for Applied Software Engin, Scapa Technologies Limited, Yatta Solutions GmbH, Zenika , CloudBees Inc., ITpearls AG, Progress Software Corporation, SunGard, Talend, University of Manchester													0

4.4.2 Closeness Centrality

Closeness centrality refers to the extent to the ability of a particular company in the development network to quickly get the information. If a development company has relatively high closeness centrality degree, it indicates that this company is close to central network, consequently, to access information faster, and lower the cost of communication (Huang et al., 2013). A company's closeness centrality degree represents the potential for analysis its work independence and efficiency, and the capability to avoid impact from other members (Hossain, Wu, & Chung, 2006). Prior study also argued that closeness indicates nodes that can spread a message to others in the network in a minimal amount of time (Hossain et al., 2006).

There are several possibilities for the Eclipse development company to get high closeness centrality degree. Companies committing code to one or several most popular projects will have high closeness centrality degree. Companies that have stronger/healthy co-developers may also have high closeness centrality degree, because the stronger co-developers are located close to the central network. These two circumstances are not duplicated; the popular projects do not inevitably attract strong co-developers to join to develop. For example, Company-D in the center of the "star" network in Figure 4.9 has a relatively high closeness, because the total distance to all other members from C is relatively small. The members located on the periphery have relatively lower closeness, because the total distance to all other members in the network is relatively large. If some local network is disconnected, then closeness centrality is undefined, since some pairs of nodes are unreachable and the distance between them is infinite.

Because the development company network in 2001 only contains two companies, IBM and Oracle, which were simultaneously working on same projects. Then they were highly connected closely, each of them had a very small closeness centrality degree: one. With the passing time more and more companies joined this network; existing companies' closeness centrality degree has been changed. In 2002, Oracle had a larger closeness centrality value, which means Oracle's position in network has been changed because of the network growth. Specifically, Oracle had development relationship with all other companies except a company called Genuitec, LLC, and a new joined company. In 2003, IBM and two new joined companies called Tasktop and SpringSource were located in the center of the small network. But in 2004, we start to have a relatively loose network, Tasktop and three new joined companies: Red Hat, SAS, and Wind River were located in the periphery. The farthest path between two peripheral members is 4 steps, which means there are three members between them. The network became closer since 2005 by comparing with the network in 2004. By analyzing the network closeness centrality from 2004 to 2012, we can find that the top rated companies built up the core network, and top 15 companies joined this network before 2007. But joined early does not make a company can always keep its network position. Since the network was keep growing, some of the members were isolated from the main network. This kind of circumstance firstly happed in 2009, four companies: Novell, Social Physics, Xored Software Inc, and Embarcadero Technologies Inc. were isolated from the main network, that means they lost some benefits on developing their interested projects as we discussed before. However, their positions in 2008 were located in the peripheral network already.

We did an overall consideration of the companies' closeness centrality at last. Generally speaking, most of the Eclipse development companies have a close development relationship with others, because the longest distance between two companies is five, only 6 companies were located in such peripheral network in 2012. IBM has the lowest closeness centrality value all the time. By evaluating the most centered company's closeness centrality and eccentricity, we can easily compare companies' positions. The full ranking table is shown below.

Table 4.4 Closeness centrality ranking table

Company	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	Overall
IBM	1,1	1,1	1,1	1.308,2	1.19,2	1.13,2	1.294,3	1.326,3	1.302,2	1.354,3	1.4,3	1.469,3	1.367,3
Red Hat, Inc.				2.769,4	2,3	1.913,3	1.912,3	1.791,3	1.581,2	1.667,3	1.673,3	1.653,3	1.608,3
SAP AG							2.706,4	1.953,4	1.884,3	1.833,3	1.545,3	1.571,3	1.608,3
Research In Motion		1,1	1.2,2	1.538,3	1.571,2	1.522,2	1.647,3	1.674,3	1.674,3	1.625,3	1.582,3	1.939,4	1.62,3
Innoopract GmbH							2.824,5	1.907,3	1.744,3	2.646,4	1.873,3	1.796,3	1.722,3
Oracle	1,1	1.5,2	1.6,2	1.846,3	1.714,3	1.696,3	1.824,3	1.884,4	1.744,3	1.875,3	1.818,3	1.878,3	1.759,3
Tasktop			1,1	3,4	2.095,3	2.087,3	2.147,4	2.209,4	1.837,3	1.854,3	1.727,3	1.755,3	1.772,3
Intel Corporation				1.769,3	1.667,2	1.696,2	2.059,4	1.86,3	1.721,3	1.813,3	1.727,3	1.776,3	1.785,3
Google Inc.					1.857,3	1.913,3	1.941,4	2.047,4	1.814,3	1.771,3	1.782,3	1.918,3	1.797,3
OBEO					1,1	1,1	2.824,4	2.698,4	1.953,3	1.854,3	1.818,4	2,3	1.873,3
SpringSource, Inc			1,1	2.077,3	2.095,3	1.957,3	2.206,4	2.116,4	2.14,3	2.125,4	1.964,4	1.98,3	1.886,3
Ericsson AB							1.941,4	2.047,4	1.907,3	1.958,3	1.855,3	2.163,4	1.937,3
itemis AG							2.882,4	2.186,4	2.093,3	2.125,3	2,3	1.837,3	1.937,3
INRIA(Institute national de rech					1,1	1,1	3.794,5	2.116,4	1.953,3	1.896,3	1.855,4	2.633,4	1.962,4
Cloudsmith Inc.									2.023,3	2.021,3	2,3	2.02,3	1.962,3
Freescale Semiconductor						1.696,2	1.941,4	2.047,4	1.86,3	1.813,3	1.891,3	2.184,4	1.975,3
BestSolution							2.147,4	2.256,4	2.163,3	2.125,4	2.109,4	2.082,3	1.975,3
EclipseSource (See Innoopract)										1.875,3	1.945,3	2.082,4	1.987,3
BREDEX GmbH												1.959,3	1.987,3
Business Systems Integration AG												1.959,3	1.987,3
Atos Integration									1.93,3	1.875,3	1.891,4	3.122,5	2,4
Sierra Wireless									1.93,3	2.188,4	2.109,4	2.449,4	2,3
Adobe Systems										1.854,3	1.927,3	1.927,3	2,3
SAS				2.462,4	2.143,3	2.174,3	2.529,4	2.558,5	2.372,3	2.208,4	2.109,4	2.143,3	2.013,4
Nokia				2.154,3	1.952,3	1.609,3	1.912,4	2.07,4	1.93,3	2,3	2.073,4	2.073,4	2.025,4
Sonatype									2.535,4	2.729,4	2.2,4	2.245,3	2.025,3
compeople AG							2.206,4	2.163,4	2.721,4	2.083,4	2.2,4	2.2,4	2.025,3
Motorola								2.93,5	2.116,3	2.354,4	2.364,4	2.364,4	2.025,3

Company	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	Overall
AGETO Service GmbH					1.81,3	1.81,3	1.81,3	1.81,3	1.81,3	2.271,4	2.309,4	2.367,4	2.038,3
GitHub											2.055,4	2.061,3	2.051,3
Prosyst Software								2.14,4	2.14,4	2.14,4	2.14,4	2.14,4	2.063,4
QNX Software Systems Co.		1,1	1.4,2	1.846,3	1.81,2	1.739,2	2.029,4	2.07,4	1.953,3	2,3	2.036,3	2.184,4	2.076,4
Eclipse Foundation						1,1	1,1	2.116,4	2.535,4	3.042,5	2.255,4	2.255,4	2.076,3
eteration A.S.				2.154,3	2.048,3	1.957,3	2.206,4	2.256,4	2.233,3	2.333,4	2.127,4	2.551,4	2.089,3
Intalio Inc.							2.059,4	2.14,4	2.116,3	2.208,4	2.109,4	2.109,4	2.101,4
Open Canarias S.L.									2.163,3	2.188,4	1.945,3	2.286,4	2.101,4
nexB Inc.							2.059,4	2.302,4	2.302,4	2.302,4	2.302,4	2.302,4	2.101,4
Mentor Graphics		1,1	1.4,2	1.846,3	1.81,2	1.739,2	2.029,4	2.07,4	1.953,3	2,3	2.036,3	2.265,4	2.114,4
Sybase					2.095,3	2,3	2.118,3	2.14,4	2.093,3	2.093,3	2.093,3	2.093,3	2.127,4
Zend Technologies						1,1	2.235,4	2.07,3	2.14,3	2.333,4	2.382,4	2.382,4	2.127,3
Industrial TSI										2.333,4	2.236,4	2.122,3	2.127,4
CEA LIST									2.093,3	2.125,3	2.055,4	3.102,5	2.139,4
Protos Software GmbH										2.292,4	1.982,3	2.265,4	2.139,4
Thales Global Services SAS									2.302,3	2.208,4	1.945,3	1,1	2.152,4
SODIFRANCE									2.465,4	2.375,4	1.982,3	2.306,4	2.165,4
Siemens AG									2.233,3	2.25,3	2.164,3	2.449,4	2.165,3
Montages AG							3.176,5	3.233,5	2.093,3	2.146,4	2.109,4	2.109,4	2.19,4
Soyatec									2.302,3	2.229,3	2.164,3	1,1	2.203,3
Hewlett-Packard Company (HP)						1.957,3	2.088,3	2.088,3	2.088,3	2.088,3	2.088,3	2.088,3	2.215,4
EcolePolytechnique de Montreal												2.347,4	2.228,4
veritInformationssysteme GmbH					2.571,3	2.652,3	3.029,5	2.837,4	2.698,4	2.333,4	2.382,4	2.382,4	2.241,4
Zenika												2.367,4	2.241,4
Genuitec, LLC		1.5,2	1.8,2	2.231,3	1.952,3	1.952,3	1.952,3	1.952,3	1.952,3	1.952,3	1.952,3	1.952,3	2.253,3
NetApp						1.87,3	1.87,3	1.87,3	1.87,3	1.87,3	1.87,3	1.87,3	2.253,4
Centrum voorWiskunde en Informa								2.209,4	2.233,3	2.75,4	2.764,4	2.764,4	2.266,3
BEA					2,3	2,3	2,3	2,3	2,3	2,3	2,3	2,3	2.278,4

Company	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	Overall
Borland Software Corp.					2.143,3	2.087,3	2.206,4	2.256,4	2.163,3	3.125,5	3.091,5	3.091,5	2.291,4
Aptana, Inc.						1,1	2.235,4	2.209,4	2.209,4	2.209,4	2.209,4	2.209,4	2.291,4
Continental AG											2.182,3	2.224,4	2.329,4
Heinrich-Heine-UniversitatDusse											2.182,3	2.224,4	2.329,4
Actuate Corporation					2.143,3	2.043,3	2.235,4	2.302,4	2.279,3	2.333,4	2.382,4	2.449,4	2.342,4
Novell						2.043,3	2.235,4	2.279,4	1,1	1,1	1,1	1,1	2.342,4
Social Physics						2.043,3	2.235,4	2.279,4	1,1	1,1	1,1	1,1	2.342,4
University of Manchester												2.449,4	2.354,4
See4sys (formerly Greensys)											2.255,3	2.286,4	2.392,4
Wind River				2.385,4	2.381,3	2.217,3	2.217,3	2.217,3	2.217,3	2.217,3	2.4,3	2.592,4	2.418,4
Tieto											2.291,3	2.388,4	2.443,4
Xored Software Inc							2.853,5	2.674,4	1,1	1,1	1,1	1,1	2.456,3
Xilinx, Inc.								2.814,4	2.674,4	2.458,4	2.382,3	2.571,4	2.481,4
MontaVista Software LLC								2.814,4	2.674,4	2.458,4	2.4,3	2.592,4	2.494,4
WeigleWilczek GmbH								2.86,5	2.86,5	2.86,5	2.86,5	2.86,5	2.544,4
Texas Instruments										2.604,4	2.564,4	2.918,5	2.608,4
Chair for Applied Software Engin												2.633,4	2.646,4
Scapa Technologies Limited					2.571,3	2.571,3	2.571,3	2.571,3	2.571,3	2.571,3	2.571,3	2.571,3	2.759,4
Yatta Solutions GmbH												2.735,4	2.759,4
Progress Software Corporation								3.674,5	3.674,5	3.674,5	3.674,5	3.674,5	2.861,4
CloudBees Inc.												3.224,4	3.013,4
Talend											3.182,5	3.182,5	3.013,4
Embarcadero Technologies Inc.								2.953,4	1,1	1,1	1,1	1,1	3.051,4
Cisco Systems, Inc.								3.651,5	3.651,5	1,1	1,1	1,1	3.443,4
ITpearls AG												1,1	1,1
SunGard												1,1	1,1

Chapter 5 Discussions

According to prior research, the structure of a network affects information transparency, work efficiency, and growth of the network (Xu & Madey, 2004; Grewal, 2006), and the growth of companies within the network (Xu & Madey, 2004), so it is important for managers of participating companies to understand the structure and dynamic evolution of an open source community. This chapter discusses the results reported in chapter 4 and positions those results with respect to the literature reviewed in chapter 2. It is comprised of four sections. The first section provides a concise summary of the thesis results. The second section provides answers to the research questions. The third section identifies and explains the contribution. The fourth section discusses reliability and validity.

5.1 Summary of results

Chapter 4 presented the thesis results. First, we collected the code commit database of the Eclipse development community, which includes information on when and which committer contributed code to which project in the Eclipse development community. Second, we replaced these committers by their employer (companies that employs committers) in the dataset, which shows how companies employed committers that commit code to their interested projects. Third, we visualized the project network (comprised of both companies and projects), and converted it into the company network (comprised of only companies). Finally, we created the evolution movie as a tool to analysis how the project evolved over time, and we measured the change of network position of each company participated in the Eclipse development community.

We observed the evolution of the Eclipse development community, analyzed companies' network position changes over time, and quantitatively measured companies' network characteristics annually. The evolution movie can help researchers to find out how the Eclipse development network evolved over time. Four network characteristics of each company were measured and reported: (i) degree centrality in Table 4.1, (ii) eigenvector centrality in Table 4.2, (iii) betweenness centrality in Table 4.3, and (iv) closeness centrality in Table 4.4. These four tables show the network position changes of each company that participated in the Eclipse development community.

In Section 4.1 ("Visualization"), we visualized the Eclipse project network and created an evolution movie as a tool to study how the Eclipse project evolved overtime. We observed when and which Eclipse development companies got involved in project development network over the past 12 years. We studied when and which companies joined or left the development community, and which projects were initiated by whom.

In Section 4.2 ("Weight filtering and most active committer"), we compared the code contribution of co-developers that were concurrently working on the same projects. We studied whether some companies had unequal code contribution or relatively equal contribution on the same projects. Weight filtering can tell us if a company was the most active committer on a particular project. in our Eclipse dataset, the global development network initiator, IBM, maintained its core developer position. We identified the central developers, which was based on how many projects they initiated, how many projects they worked as the most active committer, and how many co-developers they had. Most central developers joined at the early stage, others joined in later than 2007. For example, Research in Motion, SAP, Red Hat, Innoopract, Oracle, Tasktop, Intel, Google, OBEO,

itemis AG were central developers in the Eclipse development network. Their network characteristics were measured by using different network analysis techniques, which confirmed our observation. Joining early and initiating projects are two advantages that can help companies to build up their own local development network. The central developers were always the initial submitter or managers of the project. According to Huang (2003), central developers accelerate the progress of development projects and enhance the quality and efficiency in development (Huang, 2003).

In Section 4.3 (“Degree distribution and eigenvector centrality”), the Eclipse company network was extracted from the development network. The company network only comprises development companies. Two companies are linked only if they commit code to the same projects. It simply tells us that how many co-developers of each company had over the past 12 years. The ranking lists we generated in Section 4.3 can tell us if a company was sharing development tasks with other companies. A company that has high eigenvector centrality value is located in a healthy network position, but it does not mean such company contributed more than others. Adobe is a good example to explain this conclusion. Adobe was working on a healthy project, which had 12 co-developers, but Adobe only contributed less than 1% of total commitment. That is the reason why we introduced the weight filtering in following analysis. The most active committer study helped us to identify if a company was the biggest committer, but it did not have the capability to tell us if those projects were attracting other development members.

In Section 4.4 (“Betweenness centrality and closeness centrality”), we studied and compared each company's information control capability and its importance in the network. Closeness centrality is a structural position symbol. A company with low

closeness centrality has a peripheral network structural position. A company may have higher closeness centrality, but has low degree centrality. That means this company has development relationship with central developers, which enhanced its ability to access information. There are some companies with high eigenvector centrality, but lower closeness centrality and betweenness centrality. The explanation is that such companies were contributing to one or several popular projects. There are some companies with higher degree centrality, but lower closeness centrality and betweenness centrality. The reason is that the local development network they are involved are away from the other nodes of the network. A company with low betweenness centrality means that such company was located in periphery or independent development environment. A company with high betweenness centrality and low degree centrality gives the whole network better information transparency.

5.2 Answers to the Research Questions

In Section 1.1, we posed three research questions that guided the work of this thesis. This section answers those questions.

Question 1: Which network analysis techniques are applicable to analyze the evolution of an open source development community?

Six network analysis techniques are applicable to analyze the evolution of an open source development community: (i) visualization, (ii) degree centrality, (iii) eigenvector centrality, (iv) weight filtering, (v) betweenness centrality, and (vi) closeness centrality. All six techniques were employed in this study to examine the Eclipse developer community.

There are many network analysis techniques that can be used to study different kinds of network communities. Some techniques are applicable to open source communities and others are not. In section 2.3 and subsection 3.3.3, we reviewed previous research on network analysis of open source development communities, identified the applicable network analysis techniques that were effective in those studies, and selected the six techniques for this research. In chapter 4, we applied these six techniques to examine the Eclipse developer community. This was more techniques than previous studies (Sowe, Stamelos, & Angelis, 2006; Wiggins, Howison, & Crowston, 2009; Sadowski, Sadowski-Rasters, & Duysters, 2008; Hu & Zhao, 2009; Toral, Martínez-Torres & Barrero, 2010; Crowston & Howison, 2012).

Table 5.1 shows the six applicable techniques that were selected and used in this study. The third column in Table 5.1 shows the value of applying each technique.

Table 5.1 Network analysis techniques and their applicabilities

Method	Type of network	Value of application
Visualization (Section 4.1.1, Section 4.3.1)	Project network and company network	• Easy to track which/how many companies/projects joined or left this Eclipse ecosystem at specific time • Easy to compare which projects are more attractive than others. • To find when and which local development network got established • To find companies' position that if a company is located in the center or periphery of the development network. • To find core-peripheral structure in development network.
Weight filtering (Section 4.2)	Project network	• To identify if a company contributed more or less than others on the same projects • To identify if a company is the most active committer of a particular project.
Degree Centrality (Section 4.3.2)	Company network	• To find how many co-developers of a company has to contribute to same projects. • To find a company is located in a healthy development environment, which gains help from others.
Eigenvector Centrality (Section 4.3.3)	Company network	• To find if a company's co-developers have more or less development relationships in the network
Betweenness Centrality (Section 4.4.1)	Company network	• To find how fast it will take to spread and gain information/knowledge to other companies.
Closeness Centrality (Section 4.4.2)	Company network	• To find if a company is working relatively dependently or independently. • To find if a company is located in center or periphery in global network.

In Section 4.1 and 4.2, we studied when and which companies joined or left, which project was initiated by whom, which company was the biggest code committer on which project. In Section 4.3 and Section 4.4, we measured each company's annual network

position changes; companies' network characteristics changed over time, confirming what we observed in the visualization, but with quantitative metrics that can be used to directly compare the project involvement of companies.

Question 2: What are the benefits of applying multiple techniques rather than a single technique?

Using multiple network analysis techniques rather than a single technique can provide a more complete view of an open source community. The value of applying each network analysis technique is unique, as shown in Table 5.1.

Specifically, to visualize an open source development community, we created a evolution movie that shows how the Eclipse project evolved over time. The evolution movie is a tool to help researchers and practitioners to identify when and which companies joined or left the community, and which project was initiated by whom. But the evolution movie is only a partial view of the community, because it cannot help researchers to find out which company employed committers who committed the most of the code of a project, how many co-developers a company had, the connectivity of a company, or the importance of a company at a certain time. That is the reason why we need to also apply other techniques to measure companies' centrality characteristics.

By measuring a company's degree centrality, we found how many co-developers each company had at each time. By measuring a company's eigenvector centrality, we found if a company had high connectivity with its co-developers. Closeness centrality provided additional information about structural position. A company with low closeness centrality has a peripheral network structural position. A company may have higher closeness

centrality, but has low degree centrality. That means this company has development relationship with central developers, which enhanced its ability to access information. There are some companies with high eigenvector centrality, but lower closeness centrality and betweenness centrality. The explanation is that such companies were contributing to one or several popular projects. There are some companies with higher degree centrality, but lower closeness centrality and betweenness centrality. This implies that the local development networks in which the companies are involved are away from the other nodes of the network. A company with low betweenness centrality was located in the periphery or an independent development environment. A company with high betweenness centrality and low degree centrality gives the whole network better information transparency.

Question 3: How does an open source development community evolve?

Our results on the Eclipse development community provide some insights on the evolution of an open source development community over time.

Consistent with the results of prior studies (e.g.,Nakakoji et al. 2001), the Eclipse open source development community did not have a strict hierarchical structure. However, our results showed that the structure is also not flat. There are many local development networks that have been established in the Eclipse development community over the past 12 years. Each local development network contained at least one central developer. The central developers led project development in certain areas, surrounded by their most contributed projects and peripheral developers. This does not mean that the central developer of a local development network cannot contribute code to projects located in

other local development network, but those contributions were relatively less. We observed that the Eclipse project initiators were always the most active committers on that project in the future. Core developers and central developers joined at the early stage, and never changed their central positions. We found that if a peripheral developer contributed more code, the peripheral developer could become a central developer who eventually establishes its own local development community. An Eclipse project could have more committers if it was located in a larger local development network. The Eclipse projects that were initiated at the early stage had only few companies to work on. But since 2005, a popular project can attract many co-developers to concurrently work on at the first year when the project was created. That means an open source project could have more committers if it was located in a larger open source development community. Prior studies have acknowledged and emphasized the importance of developer relationships among the larger OSS community for their success (Singh, 2010). We further suggest that with a network of larger size there is greater chance to find co-developers to work with.

5.3 Contributions

Previous studies had proposed several promising future research directions on how to study the evolution of an open source development community. This study deliberately implemented two of these suggestions.

One recommendation from prior studies (Hinds & Lee, 2011; Huang, 2013; Wasserman, 1994) was to employ multiple network analysis techniques in order to find the indicators that best reflects the core position of the open source community developers. In this study,

we employed six applicable techniques in combination to study the evolution of an open source community. By employing multiple techniques, we were able to study different aspects of one open source community. The results show the evolution of an open source development community and the changes in network position of the companies located in the development community.

A second suggested research direction was building a dynamic network movie (Grewal, 2006), which can provide new insights on when and which members get involved in project development. Previous researchers studied open source development communities through a static view. Grewal (2006) indicated that the dynamics of the network and the environment might have even more powerful effects on the nature of the relationship among projects and developers. In this study, we built a dynamic movie as a tool that helps researchers and practitioners visualize how the Eclipse project evolved dynamically over time.

5.3.1 Methodological contributions

This study made at least three contributions to research methodology.

First, we confirmed that we could get a more complete view of an open source network by applying multiple network analysis techniques in one study. Previous studies indicated the possibility of applying more network analysis methods on studying one open source development community (Hinds & Lee, 2011; Huang, 2013; Wasserman, 1994). This study responds to that call by building a more complete network analysis framework that including more network analysis techniques than previous studies (Sowe, Stamelos, & Angelis, 2006; Wiggins, Howison, & Crowston, 2009; Sadowski, Sadowski-Rasters, &

Duysters, 2008; Hu & Zhao, 2009; Toral, Martínez-Torres, & Barrero, 2010; Crowston & Howison, 2012). Combining the results from applying these network analysis techniques, we got a more complete view of an open source development community than could be obtained from applying each technique alone. For example, measuring eigenvector centrality can tells us if a company has strong co-developers to work with, but it cannot tell us if a company contributed more or less. Our more complete network analysis framework can also be used to study other open source networks in future. New network analysis techniques can be added in the future.

Second, we confirmed the feasibility of using a dynamic network movie to visualize the evolution of an open source community. In comparison to previous studies using a static view (Grewal, 2006; Xu & Madey, 2004; Sowe, Stamelos, & Angelis, 2006; Crowston & Howison, 2005), our dynamic view of the Eclipse network gives more visual and contextual information about the network evolution. Each frame of the evolution movie is like a snapshot of the network, but with a shorter time period than the annual network snapshot. We found that the evolution movie makes it easy to track a company's activities. For example, we can easily track which company initiated what projects at what specific time. The movie provides a visualization of when a company joined or left, when a company initiated a project, and whether a company frequently or occasionally commits code to some projects. In an evolution movie, the network position of each company and project is easily observed and tracked. We recommend the evolution movie as a better tool than static snapshots to study how the network changes over time.

Third, we confirmed the feasibility of analysing two types of networks within one study. In the vocabulary of network analysis, a general network includes only one type of node.

For instance, one node may represent a person in the Facebook network (Mallapragada, Grewal, & Lilien, 2008). But an open source project network is comprised of two types of nodes: project nodes and company nodes. We demonstrated here the *project network* (with two node types) can be transformed into a *company network* comprised of only one node type: company nodes. Previous studies had analyzed either project networks or company networks and not both together. This study combined and analyzed both network types, which gave us a more compete view of the open source development community with both company project participation and company development relationships. In this research, because we only studied the code commit of each company, the company network we generated only shows how companies shared development tasks on same projects. It only links two companies if they simultaneously worked on same projects. Eventually, it became possible to analysis and compare the project network and the company network in one study, because both networks were extracted from the same database. In previous researches, two networks were always analyzed separately (Toral, Martínez-Torres& Barrero, 2010, Barcellini, Détienne & Burkhardt, 2008; Crowston & Howison, 2003).

Additionally, this study corrected an error found in the prior literature. Previous studies indicated that eigenvector centrality value indicates the importance of a company in development community (Martinez-Romo, 2008). In section 5.1, we proposed a sharper and more limited interpretation of the eigenvector centrality metric. Our findings show that eigenvector centrality indicates if a company has strong co-developers, but this is not necessarily an indication of importance in an open source community.

5.3.2 Practical contributions

This study made at least three contributions to practice for managers interested in the Eclipse platform, developer community, and business ecosystem.

First, we provided four different ranking lists of the Eclipse development companies (Tables 4.1, 4.2, 4.3, and 4.4). The four different ranking lists were created based on different perspectives, which categorized companies based on their structural position changes and project contribution. We measured four different centrality characteristics of each company in the Eclipse development community annually over the past 12 years. This is relevant to managers at the Eclipse, because it can help them to identify the network position changes of their companies. It can also help them to find relatively stronger development companies to work with. For example, degree centrality represents how many co-developers of a company have. Eigenvector centrality represents if a company has high or low connectivity with its co-developers. A company may have higher closeness centrality, but has low degree centrality. That means this company has development relationship with central developers, which enhanced its ability to access information. There are some companies with high eigenvector centrality, but lower closeness centrality and betweenness centrality. The explanation is that such companies were contributing to one or several popular projects. There are some companies with higher degree centrality, but lower closeness centrality and betweenness centrality. The reason is that the local development network they are involved are away from the other nodes of the network. A company with low betweenness centrality means that such company was located in periphery or independent development environment. A company with high betweenness centrality and low degree centrality gives the whole network

better information transparency. This is relevant to researchers and practitioners; Table 4.1, 4.2, 4.3, 4.4 show different network characteristics of each company participants in an open source development community, which help researchers to find out the network position changes of each company participated in the Eclipse development community.

Second, we provided a software application with the capability to convert a project network to a company network with weight filtering. The converted network (the company network) links two companies if both companies concurrently commit code to same projects within a time period of interest. It is relevant to researchers and managers who can easily find if any two companies are concurrently committing code to the same projects, and can evaluate and find similarities and differences among other open source development communities. The converted network (the company network) and the original network (the project network) are extracted from the same input database. Now managers and researchers are able to compare the results from analyzing both networks separately to study how the Eclipse project evolved over time.

Third, we built a dynamic network movie as a tool to help managers and researchers to visualize how the Eclipse development community evolved over the past 12 years and how companies got more or less involved in the development network over time. By visualizing the Eclipse development network, we found there are many local development networks. We also provided annual project network and company work snapshots in section 4.1 and section 4.3. The tool we created helps researchers and practitioners to find out how the Eclipse project evolved over the past 12 years. We emphasized that making a sustainable growing local development network impacts the global development network in the open source development community. Furthermore,

managers of companies in the Eclipse ecosystem can use this movie and the annual network snapshots to track their companies' structural positions and to identify popular projects to work on and active companies to work with.

5.4 Reliability and Validity

Reliability is the extent to which an experiment, test, or any measurement procedure yields the same result on repeated trials (Howell et al. 2012). Each of the six techniques used in this study has been employed in previous research (Crowston & Howison, 2005; Gao & Madey, 2007; Hossain & Zhu, 2009; Xu & Madey, 2004; Valverde & Solé, 2007). Each technique is well specified by a formal equation; with the same dataset, research procedures, tools, and methods, repeated applications of the technique will produce identical results. However, the Eclipse code commit database has a special issue that may impact trials conducted at different times. Each individual committer has a single field for "employer", and that field is updated to the current employer when a committer changes jobs; thus, people who have changed companies are reported as employees of their current company for all of their history. So future researchers who use a different "snapshot" of the Eclipse code commit database to study the evolution of the Eclipse development community may gain different results from employing the same methods. We recommend that future researchers employ the methods developed here to analyze the evolution of the Eclipse development community, but we urge caution in comparing new results with these results. Without detailed information on the number of committers who switched companies, the impact of this issue is difficult to quantify. It is likely that the impact is greater as more time passes, and more committers change employers.

Validity refers to the degree to which a study accurately reflects or assesses the specific concept that the researcher is attempting to measure (Howell et al. 2012). In this study, we focused on the evolution of the Eclipse development community. There are at least three possible threats to validity that may affect our result.

First, the community construct can take many different forms (Muegge, 2013; West & Lakhani, 2008). The Eclipse user community (Eclipse, 2011) is particularly relevant to the Eclipse development community. A user community is a community of people with common interests on using and consuming product and services (Muegge, 2013). We studied only the developer community, but users' activities also influence the evolution of a development community. For example, users can report project defects, then developers can fix the defects found by users. At this point, the users who report project defects actually have contributions to the development community. Because user do not commit code to the development community source code repository (Xu & Madey, 2004), company relationships through the user community were not analyzed in this study.

Second, we used only one type of database – the Eclipse development community code commit database. The code commit database provides development information on when and which company joined to commit code, which company initiated which project on what time, and which company was the biggest code contributor on which project. But there is other development information in the Eclipse development community which we did not analyze. For example, the Eclipse Bugzilla database is a "Defect Tracking System" that allows developers to report and track defects (Bugzilla, 2012), and the "Eclipse Mailing lists" are a record of project communications within the community by developers engaged in day-to-day development (Eclipse, 2013b). It allows developers to

discuss and vote decisions on the project development. A prior study claimed that these two data sources – the defect tracking system and developer mailing lists – can be used to study the work performance of the development community (Kidane & Gloor, 2007). In this study, we limited the analysis and discussion based on the information provided by the only database we used. There are some other ways that companies can contribute to the development community, with the only data source we analyzed, we could only analyzed their code contribution in the Eclipse development community. With more sources of data, we could potentially gain more insights into the Eclipse development community.

Third, there are some limitations of the input database; committers who have changed companies are being reported as having worked for their current company for all of history. This issue was previously discussed as a threat to reliability, but it is also a threat to validity. For example, if a committer previously worked on Eclipse as an employee of IBM, then became employed by Intel, the database shows the committer as an employee of Intel. This influences our results, because we do not have correct information of committers who switched companies. This issue only occurred in the Eclipse code commit database, and there is another common issue for all open source code commit databases, only final committers get credits. For example, if a person reports a bug fix and provides code changes, but another person commits the code changes; the database only records the second person who actually commits the code. It also influences our study, because some code commits were from committers who are missing in the database. In this study, we could not avoid this issue, so it affects some parts of the results in this study. The impact of these issues is difficult to quantify.

Chapter 6 Conclusions

In this study, we examined the evolution of a specific open source development community and its growth over the past 12 years. We observed the evolution of the Eclipse development community, analyzed companies' network position changes, and quantitatively measured companies' annual network characteristics. We applied six network analysis techniques to analyze both the project network (comprised of both companies and projects) and the company network (comprised of companies only). The six network techniques were (i) visualization, (ii) degree centrality, (iii) eigenvector centrality, (iv) weight filtering, (v) betweenness centrality, and (vi) closeness centrality.

We created a tool that helps researchers and practitioners to find out how the Eclipse projects and companies evolved over time. Based on our observation and measurement, the Eclipse development network was not flat. In fact, it contains many local development networks. The central developers were leading the project development in certain areas, surrounded by their most contributed projects and peripheral developers. An Eclipse project could have more committers if it was located in a larger local development network. The Eclipse projects that were initiated at the early stage began with only one or two participating companies. More and more companies got involved in the Eclipse project development over the past 12 years. Companies showed their interests to develop projects together in Eclipse. As reported in Chapter 4, most projects initiated since 2005 attracted multiple co-developers in the first year. Eventually, most of the Eclipse projects (192 out of 197) got code commits from two or more companies. Previous studies indicated that the project leader can be replaced by other companies, but

we could not find any instances of this in the Eclipse development network. We observed that the Eclipse project initiators were always the most active committers on that project in the future. Central developers joined at the early stage, and they never changed their central positions. Initiating multiple popular projects gave a company an opportunity to build its own local development network, which attracts companies to work together. It is not necessary to initiate many projects; one popular project can help its initiator to build its own local development network and find many companies to work together. So a popular project is a key to establish a local development network. Each local development network contains at least one popular project that attracts co-developers.

Four different ranking tables were provided to examine how companies' network structural characteristic changed over time (Table 4.1, 4.2, 4.3, and 4.4). We studied not only how many co-developers that each company had at a certain time, but we also measured if these co-developers tightly worked together. We found that 73 out of 82 known companies had at least two companies to work with on the same projects over the past 12 years. That means most Eclipse projects had multiple companies to share the development tasks. We also found that working on a popular project can give a company an opportunity to work with very active co-developers and committers.

By analyzing both the Eclipse project network and company network, we gained a more complete view of the Eclipse development network than could be obtained from either network alone. Much previous research studied SourceForge (Crowston & Howison, 2005; Gao & Madey, 2007; Hossain & Zhu, 2009; Xu & Madey, 2004; Valverde & Solé, 2007). In SourceForge, 268,554 projects have only one developer, and only 21 projects have more than 100 committers (SourceForge, 2013). In comparison to SourceForge,

Eclipse had a smaller development network, but participating companies worked more tightly together.

Prior research has reported that companies can be linked by many types of connections and flows in a development community, including information, materials, services, and social support (Provan et al. 2007). In this study, we focused exclusively on analysis of code commits from committers employed by companies. With could commit data, westudied when and which companies joined or left the community, which project was initiated by whom, which company was the biggest code committer on which project, and how many co-developers a company had on certain time. However, code commit data provided no information on the other developer community connections and flows identified by Porvan et al. (2007), and provided no insights into "how" and "why" questions such as how and why companies joined, left or committed code to open source projects.

6.1 Limitation

This study has at least five limitations.

First, an open source software development community is only one type of technical community (West & Mahony, 2008; West & Lakhani, 2008). Prior research reports that firms gain benefit from participating in technical communities by gathering information on potential alliances, identifying opportunities for future inter-firm collaboration (Rosenkopf et al. 2001). We only studied one type of "inter-firm collaboration": committing code to the same projects. Companies can also participate in projects and interact with other companies without committing code, but those activities are not

captured in our analysis. For example, the Eclipse BIRT project ("Business Intelligence and Reporting Tools") has a "Project Management Committee" (PMC) that governs the project, and that PMC includes committee members employed by IBM and Innovent. However, employees of IBM and Innovent rarely commit code to the source code repository of the BIRT project, so the contributions and interactions of those companies on the PMC of the BIRT project are not captured in our analysis

Second, we studied how company employees contributed code to open source projects, but our analysis did not include committers who are not employed by member companies. The Eclipse developer community also includes individuals who commit code based on their personal interests (Panchal, 2009). Those contributions are excluded from our analysis.

Third, we studied when and which companies participated in an open source development community, but we did not study the motivation for company participation. According to West & O'Mahony (2008), there are at least three reasons why companies participate in technical communities: (1) to interpret, support, extend and diffuse path-breaking innovations, (2) to further develop and refine innovations, and (3) to coordinate the work of firms and individuals. In order to examine motivation, we would need data not found in the code commit database.

Fourth, we used data from only one type of database – code commits to the Eclipse source code repository. There are many other sources of data which we did not analyze, including the Eclipse Bugzilla database (Bugzilla, 2012) and the Eclipse developer mailing lists (Eclipse, 2013b). Both of these sources were previously discussed in section

5.4. With more sources of data, we may be able to gain more insights into the Eclipse development community.

Fifth, there are known limitations of the Eclipse code commit database. For example, some committers' names were changed in the database when Object Technology International (OTI) joined IBM in the early days of the Eclipse project (Eclipse Foundation, 2013). These old log-ins are categorized as "unknown". People categorized under "individual" in the code commit database may actually be associated with a company but that information is missing. These two groups, unknown and individual, were deliberately excluded from our analysis of company code commits. In addition, committers who have changed companies are reported employees of their employer all of that committer's history (this was previously discussed in section 5.4)

6.2 Opportunities for future research

We offer five recommendations for future research.

The first opportunity for future research is analysis of how the *user community* is involved with the open source development community. The Eclipse user community is particularly relevant to the Eclipse development community. This opportunity addresses the first limitation of the present research.

The second opportunity for future research is analysis of *individual developer motivation*. This study analyzed the evolution of an open source development community at the company level and thus excluded the motivations of individual developers. Prior studies have reported individual motivations of many forms – both intrinsic and extrinsic,

pecuniary and non-pecuniary, and self-interested and socially motivated (Ghosh, 2005; Lakhani & Wolf, 2005; Roberts et al. 2006). This opportunity addresses the second limitation of the present research.

The third opportunity for future research is the relationship between *companies' business strategies* and their project involvement. This is a young but active area of research (Dahlander, 2007; Feller et al. 2008). This opportunity addresses the third limitation of the present research.

The fourth opportunity for future research is use of *other sources of data*, including the Eclipse Bugzilla database (Bugzilla, 2012) and the Eclipse developer mailing lists (Eclipse, 2013b). With more sources of data, we could study other aspects of the open source development community. This opportunity addresses the fourth limitation of the present research.

The fifth opportunity for future research is a more complete and verified Eclipse code commit database that accounts for committers with different employers at different points in time. This opportunity addresses the fifth limitation of the present research.

References

Bailetti, T. 2009. ***Open source maturity curve and ecosystems.*** Eclipse Summit Europe. 2009.

Barcellini, F., Détienne, F., & Burkhardt, J. M. 2008. User and developer mediation in an Open Source Software community: Boundary spanning through cross participation in online discussions. ***International Journal of Human-Computer Studies***, 66(7): 558–570.

Barcellini, F., Détienne, F., & Burkhardt, J.-M. 2009. Participation in online interaction spaces: Design-use mediation in an Open Source Software community. ***International Journal of Industrial Ergonomics***, 39(3): 533–540.

Bossink, B. 2002. The development of co–innovation strategies: Stages and interaction patterns in interfirm innovation. ***R&D Management***, 32(4): 311–320.

Breiger, R. 1976. Social Structure from Multiple Networks. ***American Journal of Sociology***, 81(6): 1384–1446.

Bugzilla. 2012. ***What is Bugzilla?***. http://www.bugzilla.org/about

Chaikalis, T., Melas, G., & Chatzigeorgiou, A. 2012. SEANets: Software evolution analysis with networks. ***2012 28th IEEE International Conference on Software Maintenance (ICSM)***, 634–637.

Coulon, F. 2005. ***The use of social network analysis in innovation research: A literature review***, Danish Research Unit for Industrial Dynamics, Skorping, Denmark.

Crowston, K., & Howison, J. 2003. The social structure of open source software development teams. ***First Monday,*** 10(2): 1-16.

Crowston, K., & Howison, J. 2012. Hierarchy and centralization in Free and Open Source Software team communications, ***Knowledge Technology Policy***, 18(4): 65-85.

Dahlander, L. 2007. Penguin in a new suit: a tale of how de novo entrants emerged to harness free and open source software communities. ***Industrial and Corporate Change***, 16(5): 913-943.

De Noni, I., Ganzaroli, A., & Orsi, L. 2012. The evolution of OSS governance: a dimensional comparative analysis. ***Scandinavian Journal of Management***, 29(3): 247-263.

Durón, R. M. 2008. ***Management issues in open source software networks***. Unpublished Thesis, Universidad Carlos III de Madrid. Spain.

Dyer, J. H., & Hatch, N. W. 2006. Relation-specific capabilities and barriers to knowledge transfers: creating advantage through network relationships. ***Strategic Management Journal***, 27(8): 701–719.

Eclipse, 2011. ***Eclipse Development Process,*** http://www.eclipse.org/projects/dev_process/development_process_2011.php

Eclipse, 2013a. ***Eclipse Subversive – Documentation.*** http://www.eclipse.org/subversive/documentation/teamSupport/SVNaction/commit.php

Eclipse. 2013b. ***Eclipse mailing lists home***. http://www.eclipse.org/mail/

Ethiraj, S. 2007. Allocation of inventive effort in complex product systems. ***Strategic Management Journal***, 584(1): 563–584.

Faust, K. 1997. Centrality in affiliation networks. ***Social Networks*** 19(2): 157–191.

Feller, J., Finnegan, P. & Hayes, J. 2008. Delivering the whole product: business model impacts and agility challenges in a network of open source firms. ***Journal of Database Management***, 19(2): 95-108.

Foer, A. 2004. ***Do the "New Dynamics of Business Ecosystems" Spell the End of Antitrust?*** Working paper, Loyola University, Chicago, United States.

Freeman, C. & Linton, A.. 1977. A set of measures of centrality based upon betweenness. ***Sociometry***, 40: 35–41.

Gao, Y., & Madey, G. 2007. Network analysis of the SourceForge.net community. ***Open Source Development, Adoption and Innovation***, 234(1): 187-200.

Goth, G. 2005. Beware the march of this IDE: Eclipse is overshadowing other tool technologies. ***IEEE Software***, 22(4): 108-111.

Grewal, R. 2006. Location, location, location: How network embeddedness affects project success in open source systems. ***Management Science***, 52(7): 1043-1056.

Hamill, L., & Gilbert, N. 2009. Social circles: A simple structure for agent-based social network models. ***Journal of Artificial Societies and Social Simulation***, 12(2): 3-12.

Hartigh, E. Den, & Asseldonk, T. Van. 2004. Business ecosystems: A research framework for investigating the relation between network structure, firm strategy,

and the pattern of innovation diffusion. ***ECCON 2004 Annual Meeting: Co-Jumping on a Trampoline, The Netherlands.***

Hartigh, E. Den, Tol, M., & Visscher, W. 2006. The health measurement of a business ecosystem. ***ECCON 2006 Annual Meeting: Co-Jumping on a Trampoline, The Netherlands.***

Hauptman, O. 2003. Platform leadership: How Intel, Microsoft, and Cisco drive industry innovation. ***Journal of Strategic Management Education***, 2(1): 1–4.

Hinds, D., & Lee, R. 2011. Communication Network Characteristics of Open Source Communities. ***Disciplinary Advancement in Open Source Development***, 1(4): 1–28.

Hinds, D., & Lee, R. M. 2008. Social Network Structure as a Critical Success Condition for Virtual Communities. ***Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)***, 1530-1605.

Hossain, L., Wu, A., & Chung, K. K. S. 2006. Actor centrality correlates to project based coordination. ***Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work - CSCW '06***, 363-372.

Hossain, L., & Zhu, D. 2009. Social networks and coordination performance of distributed software development teams. ***The Journal of High Technology Management Research***, 20(1): 52–61.

Howell, J., Miller, P., Park, H., Sattler D., Schack T., Spery E., Widhalm S., Palmquist M. 2012. ***Reliability and Validity***. Working Paper, Colorado State University, US.

Howison, J., Inoue, K., & Crowston, K. 2006. Social dynamics of free and open source team communications. ***Social Dynamics,*** 203(1): 319-330.

Hu, D., & Zhao, J. 2009. Discovering Determinants of Project Participation in an Open Source Social Network. ***International Conference on Information Systems (ICIS)***. 16-31.

Huang, H. 2010. Analysis of the structure and evolution of an open-source community, ***Journal of Computing and Information Science in Engineering***, 11(3): 1-14.

Huang, Y., Li, B., & He, P. 2013. Project Development Promoting Strategy in Open Source Community Based on Social Network Centrality Analysis. ***Journal of Computational Information Systems***, 9(2): 721–728.

Iansiti, M., & Levien, R. 2002. ***The New Operational Dynamics of Business Ecosystems: Implications for Policy, Operations and Technology Strategy***. Harvard Business School.

Iansiti, M., & Levien, R. 2004. ***Keystones and dominators: framing operating and technology strategy in a business ecosystem,*** Working Paper, Harvard Business School.

Iansiti, M., & Richards, G. 2006. ***Information Technology Ecosystem: Structure, Health, and Performance,*** Working Paper, 6-34, Harvard Business School,

Iino, T., & Iyetomi, H. 2012. Subcommunities and Their Mutual Relationships in a Transaction Network. ***Progress of Theoretical Physics Supplement***, 194: 144–157.

Jap, S. D., & Anderson, E. 2003. Safeguarding Interorganizational Performance and Continuity Under Ex Post Opportunism. ***Management Science***, 49(12): 1684–1701.

Jermakovics, A., Sillitti, A., & Succi, G. 2011. Mining and visualizing developer networks from version control systems. ***Proceeding of the 4th international workshop on Cooperative and human aspects of software engineering 2011***, 24-31.

Karhiniemi, M. 2009. ***Creating and sustaining successful business ecosystems***. Unpublished thesis, Helsinki School of Economics, Finland.

Kidane, Y. H., & Gloor, P. A. 2007. Correlating Temporal Communication Patterns of the Eclipse Open Source Community with Performance and Creativity Contact, ***Computational & Mathematical Organization Theory***, 13(1): 17-27.

Lambiotte, R., Delvenne, J., & Barahona, M. 2008. ***Laplacian dynamics and multiscale modular structure in networks.*** Unpublished thesis. Institute for Mathematical Sciences, London.

Lakhani, K.R. & Wolf, R.G. 2005. Why hackers do what they do: understanding motivation and effort in free/open source software projects. In J. Feller, B. Fitzgerald, S.A. Hissam & K.R. Lakhani (eds.), ***Perspectives on free and open source software***, Cambridge, MA: The MIT Press: 4-21.

Le, Q., & Panchal, J. H. 2012. Network-Based Analysis of the Structure and Evolution of an Open Source Software Product. ***45th Hawaii International Conference on System Sciences***, 3436–3445.

Liao, S. 2005. Technology management methodologies and applications. ***Technovation***, 25(4): 381–393.

Liu, X., & Iyer, B. 2007. Design architecture, developer networks, and performance of Open Source Software projects. ***Proceedings of the Twenty-Eighth International Systems****, Montreal.*

Lombardi, S. 2008. ***Interactions between eclipse foundation members and eclipse projects***. Unpublished thesis, Carleton University, Canada.

Long, Y., & Siau, K. 2007. Social Network Structures in Open Source Software Development Teams. ***Journal of Database Management***, 18(2): 25–40.

Mallapragada, G., Grewal, R., & Lilien, G. 2008. ***Born to win? How foundational network structure affects the success of open source development projects***, Unpublished thesis, Pennsylvania State University.

Mart, R. 2012. Expert Systems with Applications A genetic search of patterns of behaviour in OSS communities. ***Expert Systems with Applications journal***, 39: 13182–13192.

Martinez-Romo, J. 2008. Using social network analysis techniques to study collaboration between a floss community and a company. ***IFIP International Federation for Information Processing***, 275: 171–186.

Mockus, A. 2000. A case study of open source software development: the Apache server. Unpublished conference paper, International Conference on Software Engineering, 2000.

Moore, G.A. 1991. ***Crossing the chasm: Marketing and selling high-tech products to mainstream customers***. New York, NY: Harper-Business.

Moore, J. 1996. ***The Death of Competition: Leadership and Strategy in the Age of Business Ecosystems***. HarperCollins, Canada.

Milinkovich, M. 2008. TIM Lecture Series: A Practitioners Guide to Ecosystem Development. ***Technology Innovation Management Review***, October: 40-42.

Muegge, S., 2011a. ***Institutions of Participation: A Nested Case Study of Company Participation in the Eclipse Foundation, Community, and Business Ecosystem,*** Doctoral Thesis, Carleton University, Canada.

Muegge, S., 2011b. Business ecosystems as institutions of participation: A systems perspective on community-developed platforms, ***Technology Innovation Management Review***, November: 4- 13.

Muegge, S. M. 2013. Platforms, communities, and business ecosystems: Lessons learned about technology entrepreneurship in an interconnected world***. Technology Innovation Management Review***, February: 5-15.

Nachira, F., Dini, P., & Nicolai, A. 2007. ***A network of digital business ecosystems for Europe: roots, processes and perspectives.*** Unpublished thesis, London School of Economics and Political Science.

Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K., & Ye, Y. 2001. ***Evolution Patterns of Open-Source Software Systems and Communities***, No. 76-85,

Proceedings of the International Workshop on Principles of Software Evolution, 2001.

Newman, M., & Girvan, M. 2004. Finding and evaluating community structure in networks. ***Physical review***, 69(2): 1–16.

Novak, S., & Stern, S. 2008. How does outsourcing affect performance dynamics? Evidence from the automobile industry. ***Management Science***. 54(12): 1963-1986.

Okoli, C., & Oh, W. 2007. Investigating recognition-based performance in an open content community: A social capital perspective. ***Information & Management***, 44(3): 240–252.

Open Source Initiative 2003. ***The Open Source Definition,*** http://opensource.org/osd

Panchal, J. 2009. Agent-based modeling of mass-collaborative product development processes. ***Journal of Computing and Information Science in Engineering***, 8949: 1–31.

Panchal, J. 2009. ***Co-evolution of products and communities in mass-collaborative product development-a computational exploration.*** 17th International Conference on Engineering Design. 49-60.

Peltoniemi, M. 2005. ***Business ecosystem: A conceptual model of an organization population from the perspectives of complexity and evolution.*** Unpublished thesis, Tampere University of Technology, Finland.

Peltoniemi, M., & Vuori, E. 2004. Business ecosystem as the new approach to complex adaptive business environments. ***Frontiers of E-business Research***, 20(22): 1–15.

Prattico, L. 2012. ***Examining governance of open source software foundations***. Unpublished thesis, Carleton University, Canada.

Provan, K. G., Fish, a., & Sydow, J. 2007. Interorganizational networks at the network level: A review of the empirical literature on whole networks. ***Journal of Management***, 33(3): 479–516.

Roberts, J., Hann, I., & Slaughter, S. 2006. Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the Apache projects. ***Management Science***, 52(7): 984-999.

Robertsa, J., Hann, I., & Slaughter, S. 2006. Communication networks in an open source software project. ***IFIP Intemational Federation for Information Processing***, 203: 297–306.

Rosenkopf, L., & Tushman, M. 1998. The co-evolution of community networks and technology: Lessons from the flight simulation industry. ***Industrial and Corporate Change***, 7(2): 311-346.

Sadowski, B. M., Sadowski-Rasters, G., & Duysters, G. 2008. Transition of governance in a mature open software source community: Evidence from the Debian case. ***Information Economics and Policy***, 20(4): 323–332.

Sagers, G., Wasko, M., & Dickey, M. 2004. ***Coordinating efforts in virtual communities: examining network governance in open source.*** 10th Americas Conference on Information Systems, New York.

Singh, P. V. 2010. The small-world effect. ***ACM Transactions on Software Engineering and Methodology***, 20(2): 1–27.

Smith, D. & M. Milinkovich. 2007. Eclipse: A premier open source community. ***Technology Innovation Management Review***, July: 7-10.

Skerrett, I. TIM lecture series: Building technical communities. ***Technology Innovation Management Review***, June: 29-32.

Sowe, S., Stamelos, I., & Angelis, L. 2006. Identifying knowledge brokers that yield software engineering knowledge in OSS projects. ***Information and Software Technology***, 48(11): 1025–1033.

SourceForge, 2013, ***Developer counts – the stats***, http://sourceforge.net/blog/developer-counts-the-stats/, Mar 9, 2012.

Stevens, G., & Draxler, S. 2010. Appropriation of the Eclipse Ecosystem: Local Integration of Global Network Production. ***Computer Supported Cooperative Work (CSCW)*** 12(4): 465-490.

Toral, S. L., Martínez-Torres, M. R., & Barrero, F. 2010. Analysis of virtual communities supporting OSS projects using social network analysis. ***Information and Software Technology***, 52(3): 296–303.

Toral, S. L., Martínez-Torres, M. R., Barrero, F., & Cortés, F. 2009. An empirical study of the driving forces behind online communities. ***Internet Research***, 19(4): 378–392.

Tsvetovat, M., & Kouznetsov, A. 2011, ***Social network analysis for startups***. O'Reilly Media.

Valente, T., & Coronges, K. 2008. How correlated are network centrality measures? ***Connections (Toronto, Canada)***, 28(1): 16-26.

Valverde, S., & Solé, R. V. 2007. Self-organization versus hierarchy in open-source social networks. ***Physical review. E, Statistical, nonlinear, and soft matter physics***, 76(4): 75-89.

Valverde, S., & Theraulaz, G. 2006. Self-organization patterns in wasp and open source communities. ***IEEE Computer Society***, 21(2): 36–40.

Van de Ven, Andrew. H. 1993. A community perspective on the emergence of innovations. ***Journal of Engineering and Technology Management***, 10 (1-2): 23-51.

Wasserman, S. 1994. ***Social network analysis: Methods and applications***. Cambridge University Press.

Watts, D. J., & Strogatz, S. H. 1998. Collective dynamics of "small-world" networks. ***Nature***, 393: 440–442.

Weiss, M., & Gangadharan, G. R. 2009. Modeling the mashup ecosystem: structure and growth. ***R&D Management***, 40(1): 40–49.

Weiss, M. 2011. ***Company-led open source***, International Open and User Innovation Workshop (OUI), Vienna, Austria, July, 2011.

Weiss, M., Moroiu, G., & Zhao, P. 2006. Evolution of open source communities. ***Open source systems***, 203: 21–32.

Weiss, M. 2011. Economics of Software Product development collectives. ***Technology Innovation Management Review***, October: 13-18.

West, J. & Lakhani, K. 2008. Getting clear about communities in open innovation. ***Industry and Innovation***, 15(2): 223-231.

West, J. & O'Mahony, S. 2008. The role of participation architecture in growing sponsored open source communities. ***Industry and Innovation***, 15(2): 145-168.

Wiggins, A., Howison, J., & Crowston, K. 2008. Social dynamics of FLOSS team communication across channels. ***Open Source Development***, 275: 131–142.

Williams, H. W. 1996. The social capital of entrepreneurial managers. ***Financial Times***: 96-98.

Xu N. 2003. ***An exploratory study of open source software based on public project archives***, Thesis, the John Molson School of Business, Concordia University, Canada, 2003.

Xu, J., Christley, S., & Madey, G. 2005. ***The open source software community structure.*** Conference Paper, North American Association for Computational Social and Organizational Science.

Xu, J., Christley, S., & Madey, G. 2006. Application of social network analysis to the study of open source software. ***The economics of open source software development***, 205–224.

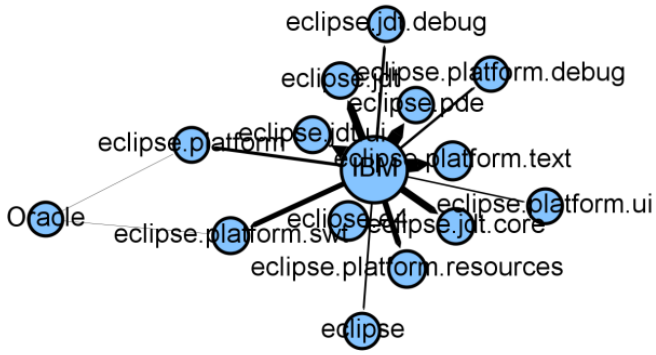
Xu, J., & Madey, G. 2004. ***Exploration of the open source software community.*** Conference Paper, Computational Social Science Society of the Americas conference.

Zschoch, M. 2007. The success of open source. ***Canadian Journal of Political Science***, 40(1): 250-252.

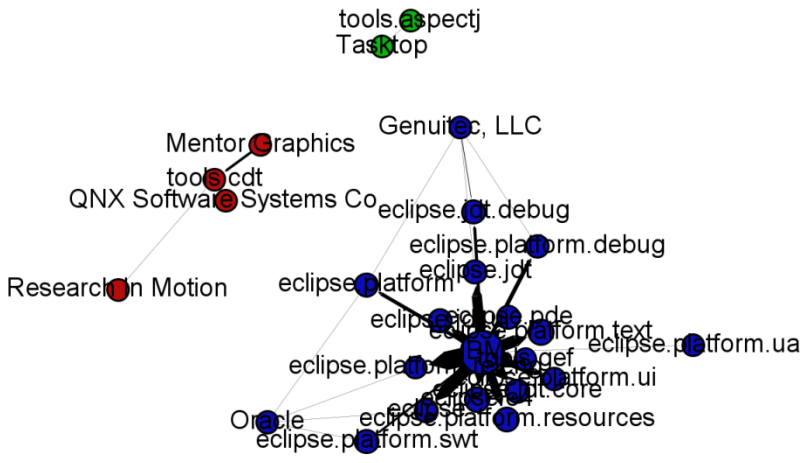
Appendix

Appendix 1 Eclipse project network (2001-2012)

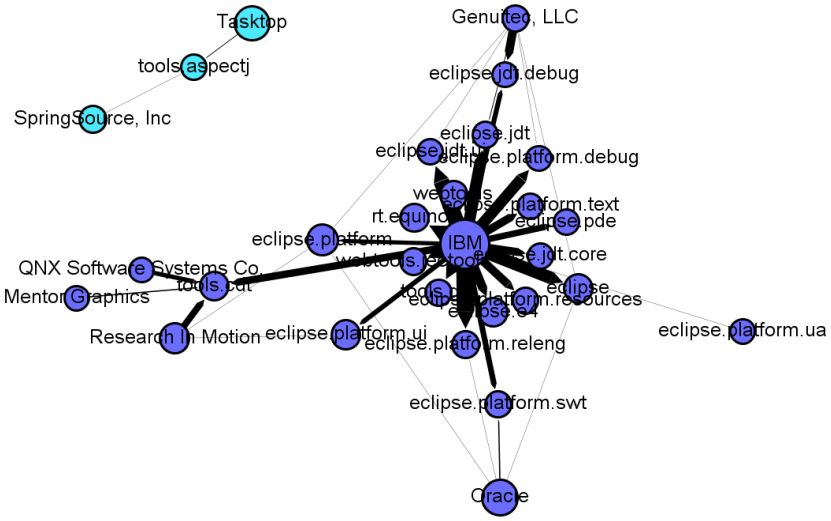
The Eclipse project network represents the relationships between participate companies and projects. Each node represents a project or company. The links shows if there is a contribution from a company to a project. Appendix 1 shows the company and project relationships in Eclipse development community in past 12 years.



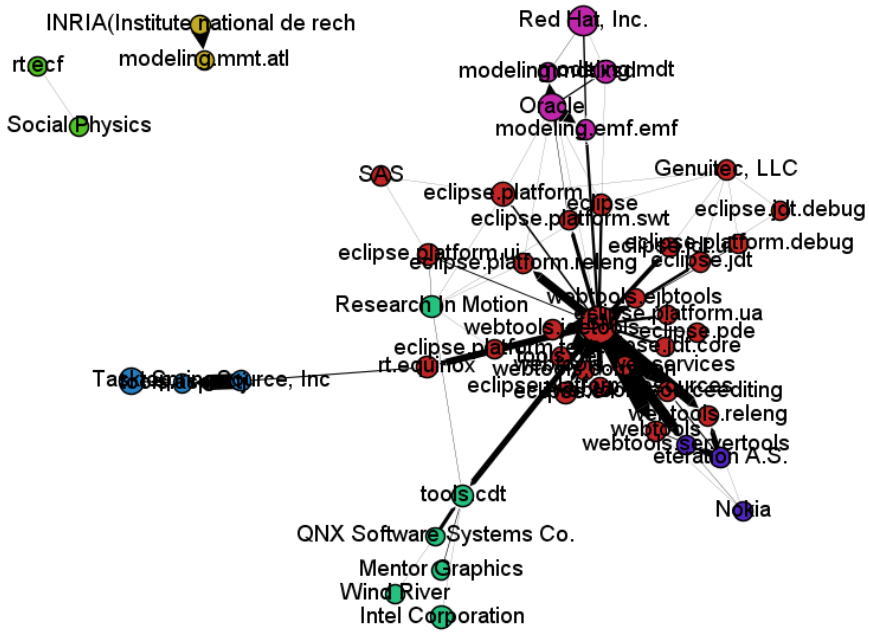
Eclipse project network in 2001



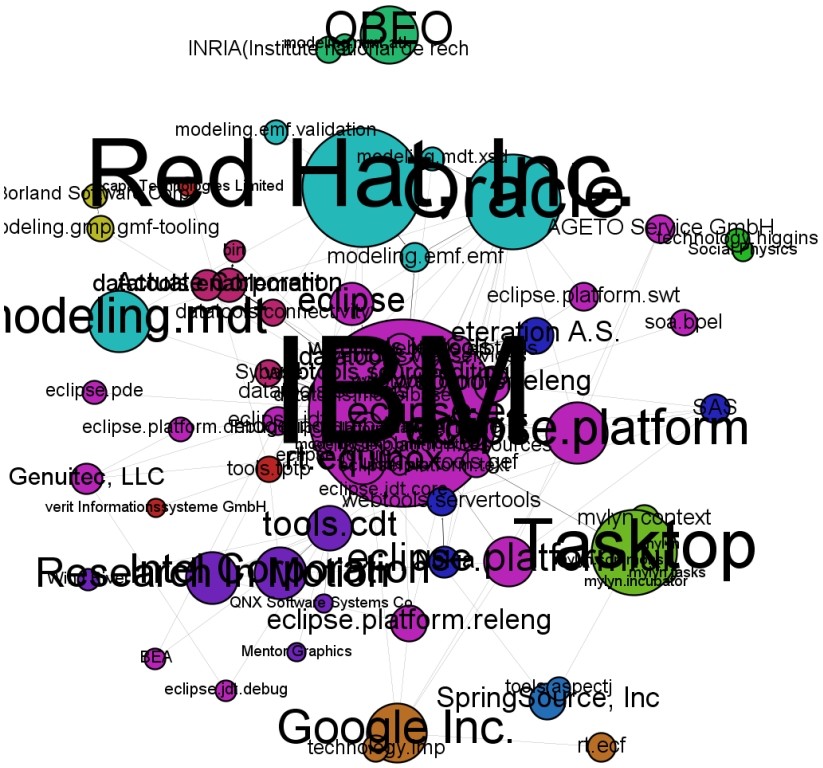
Eclipse project network in 2002



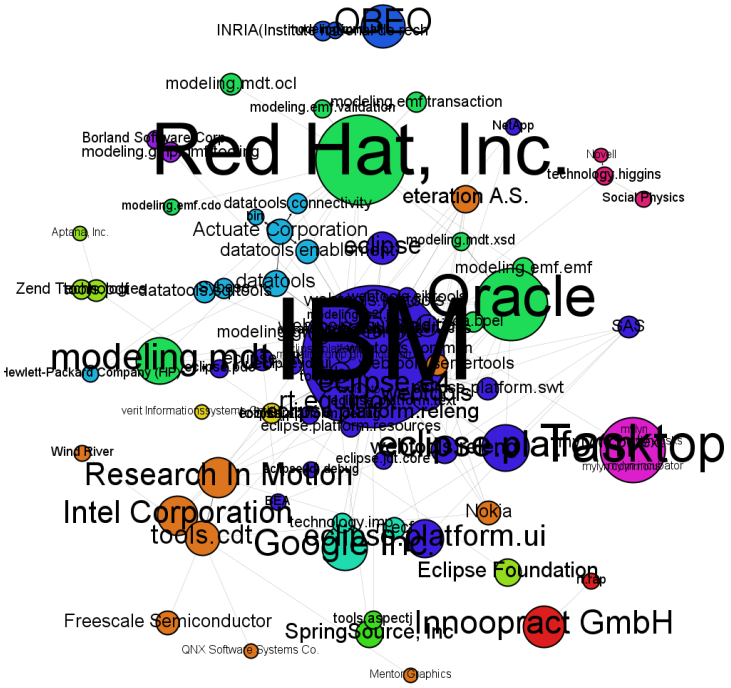
Eclipse project network in 2003



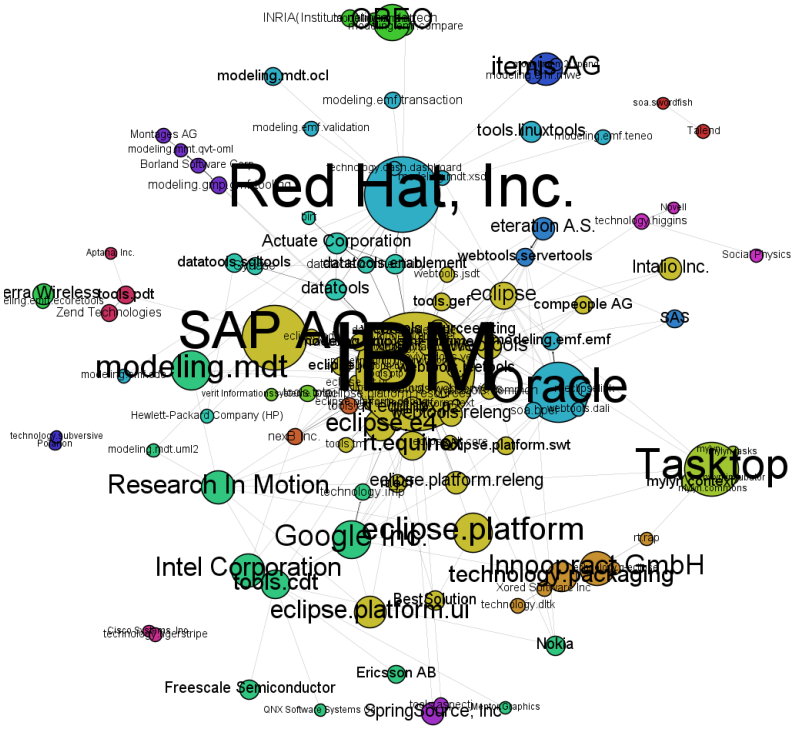
Eclipse project network in 2004



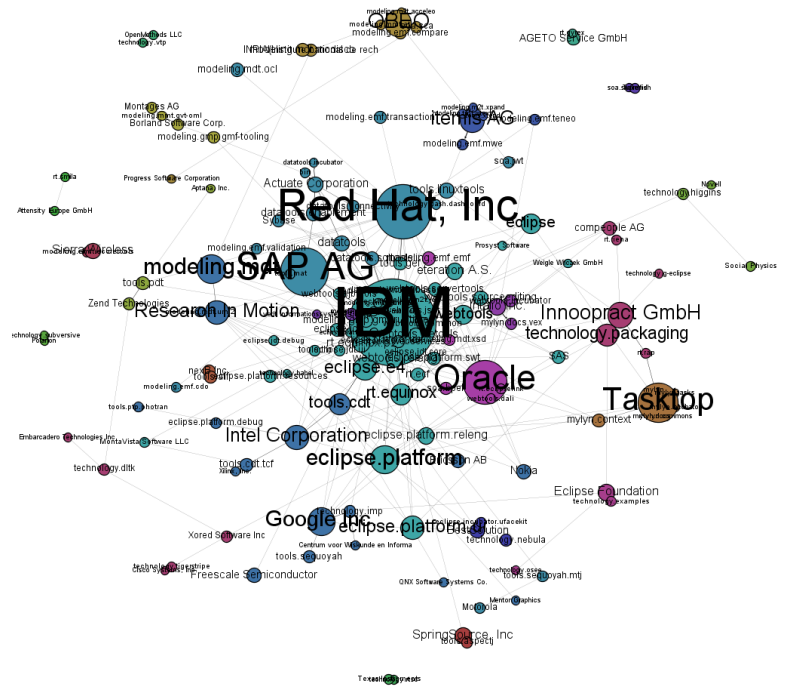
Eclipse project network in 2005



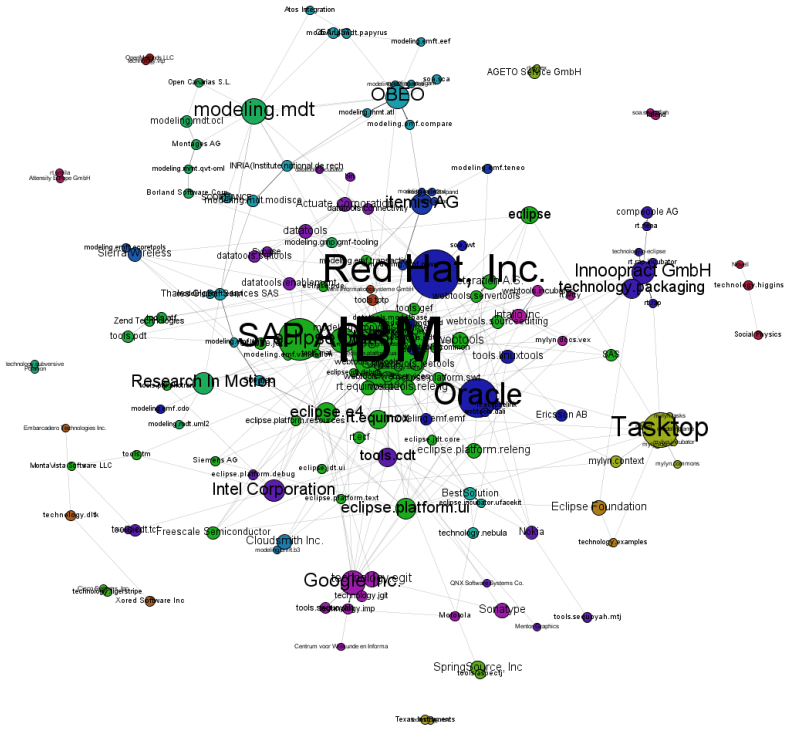
Eclipse project network in 2006



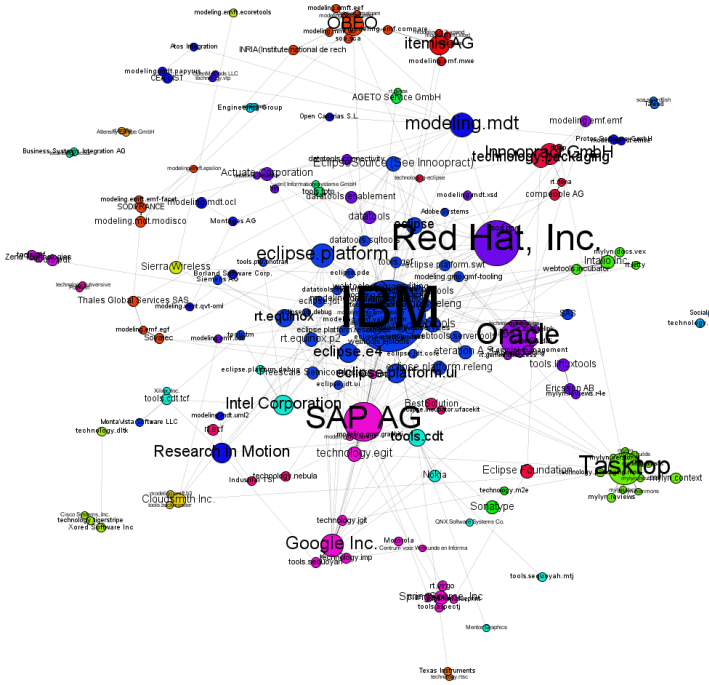
Eclipse project network in 2007



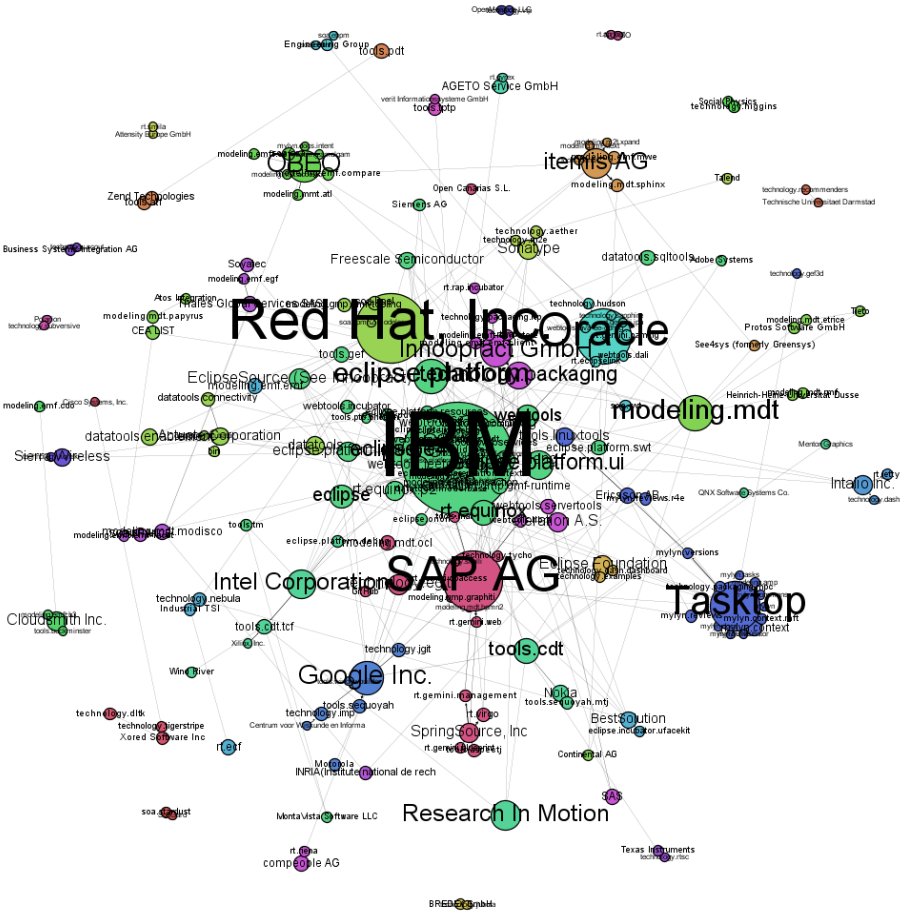
Eclipse project network in 2008



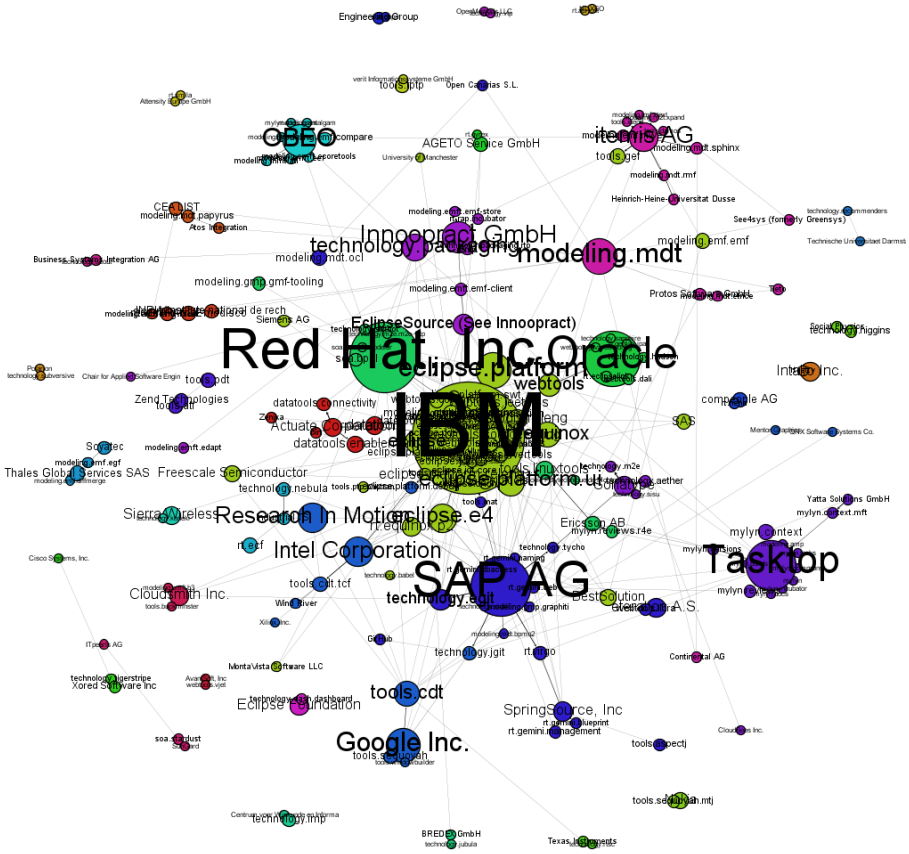
Eclipse project network in 2009



Eclipse project network in 2010



Eclipse project network in 2011

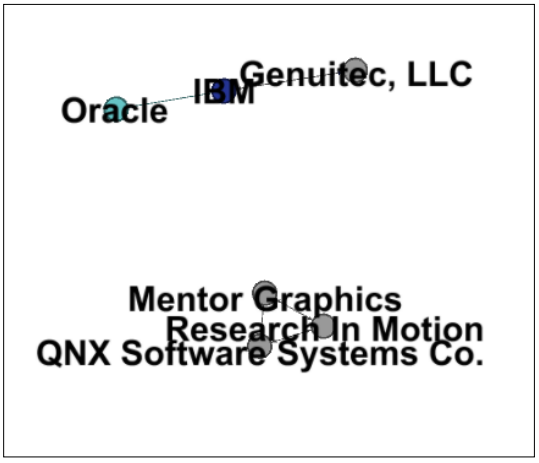


Eclipse project network in 2012

Appendix 2 Eclipse company network (2001-2012)

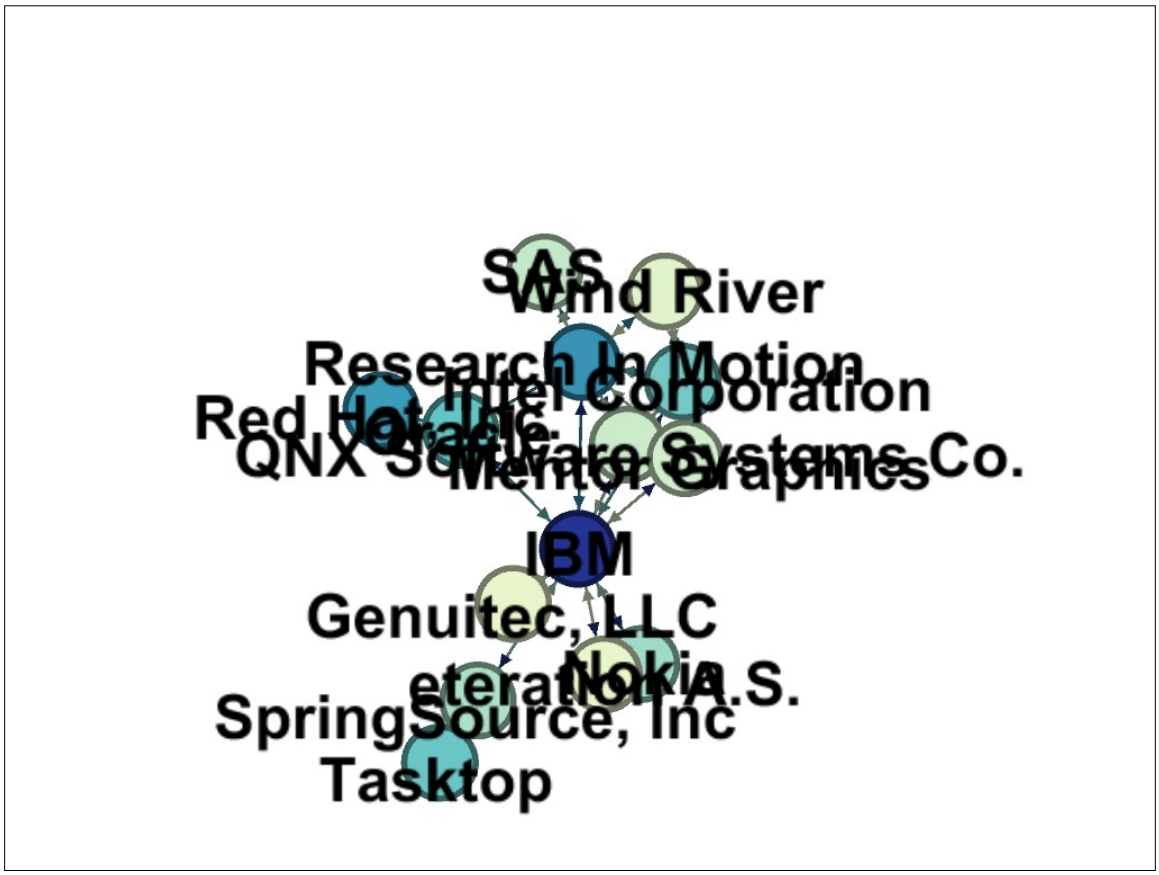
The Eclipse company network represents the relationships between participate companies. Each node represents a company. The links between companies shows if they were a couple of co-developers, which represents if they concurrently contributed to same projects. Appendix 2 shows the company development relationships in Eclipse development community in past 12 years.

Eclipse company network in 2001

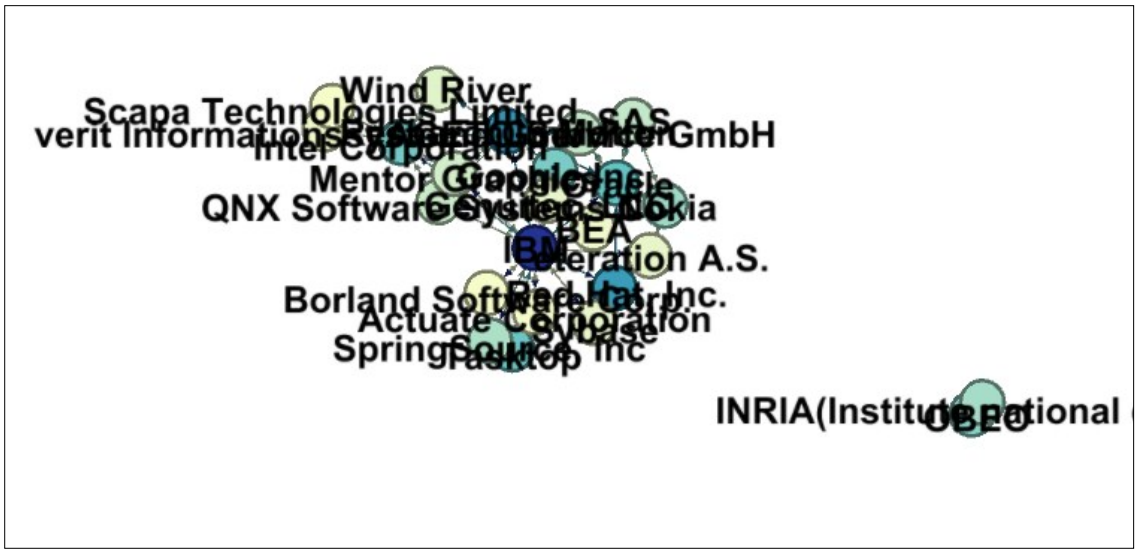


Eclipse company network in 2002

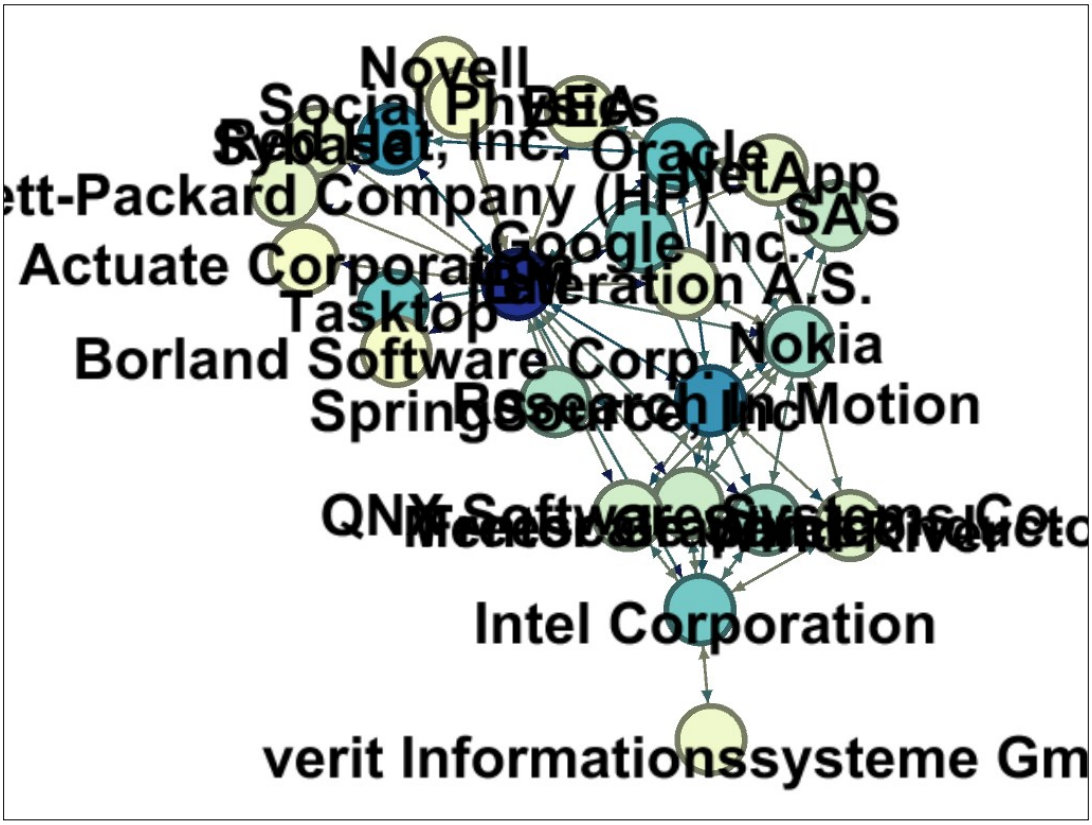
Eclipse company network in 2003



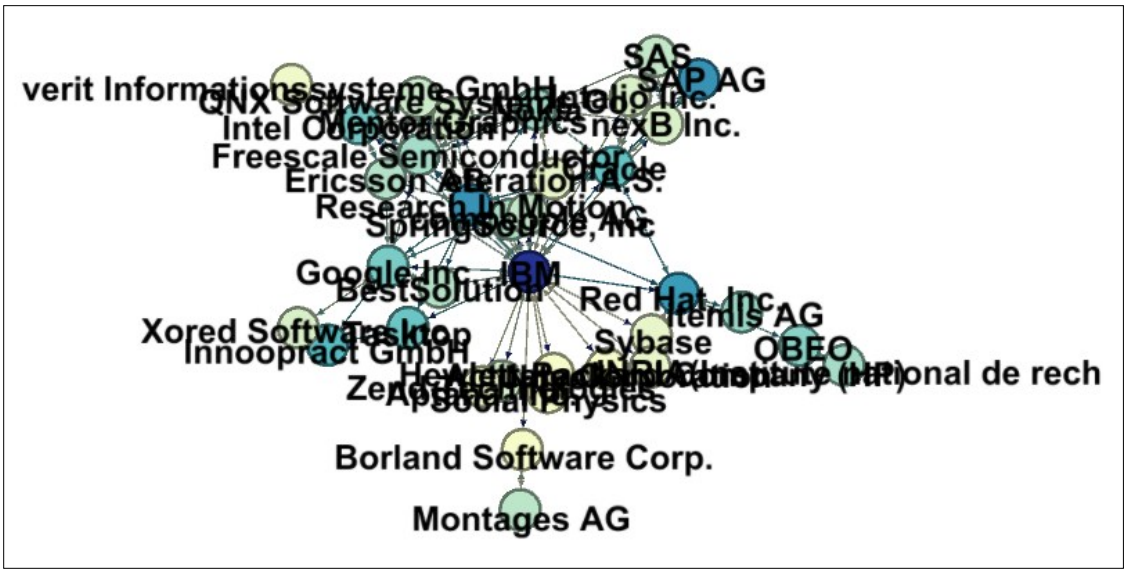
Eclipse company network in 2004



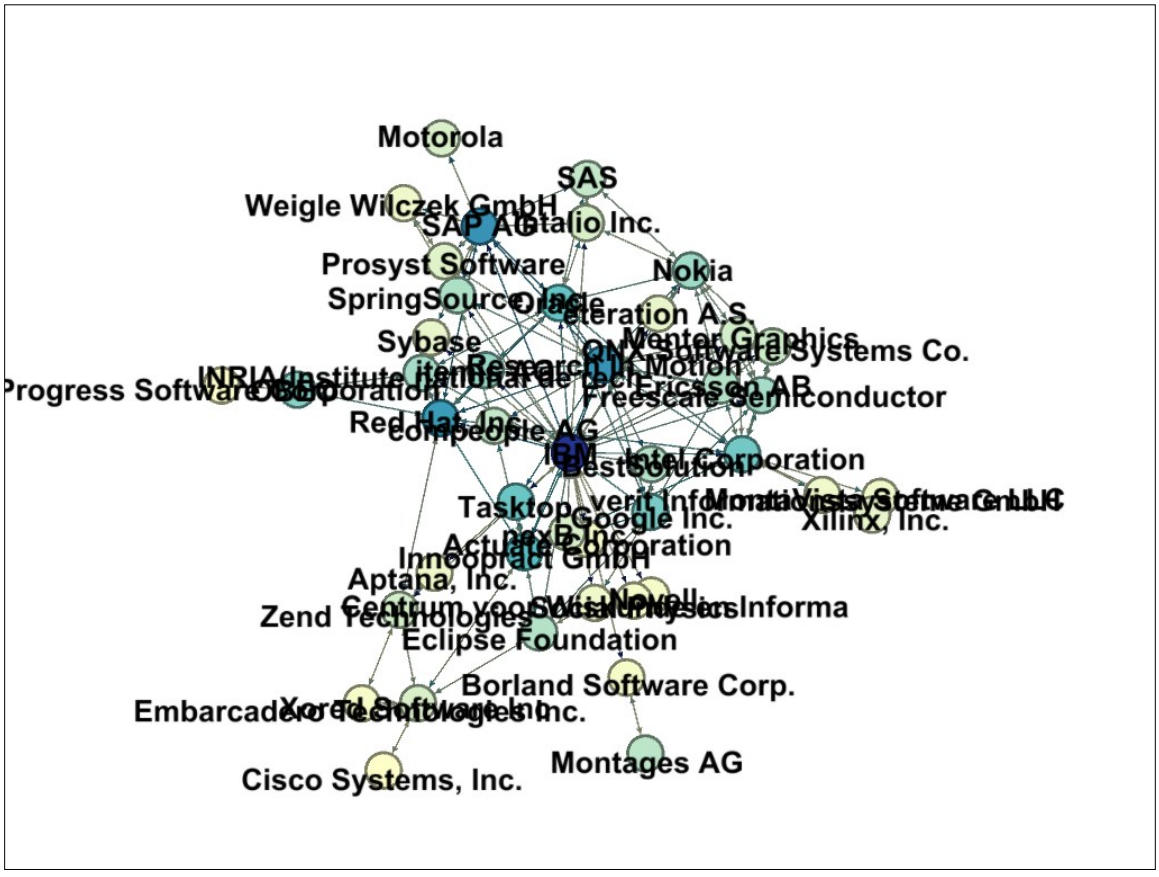
Eclipse company network in 2005



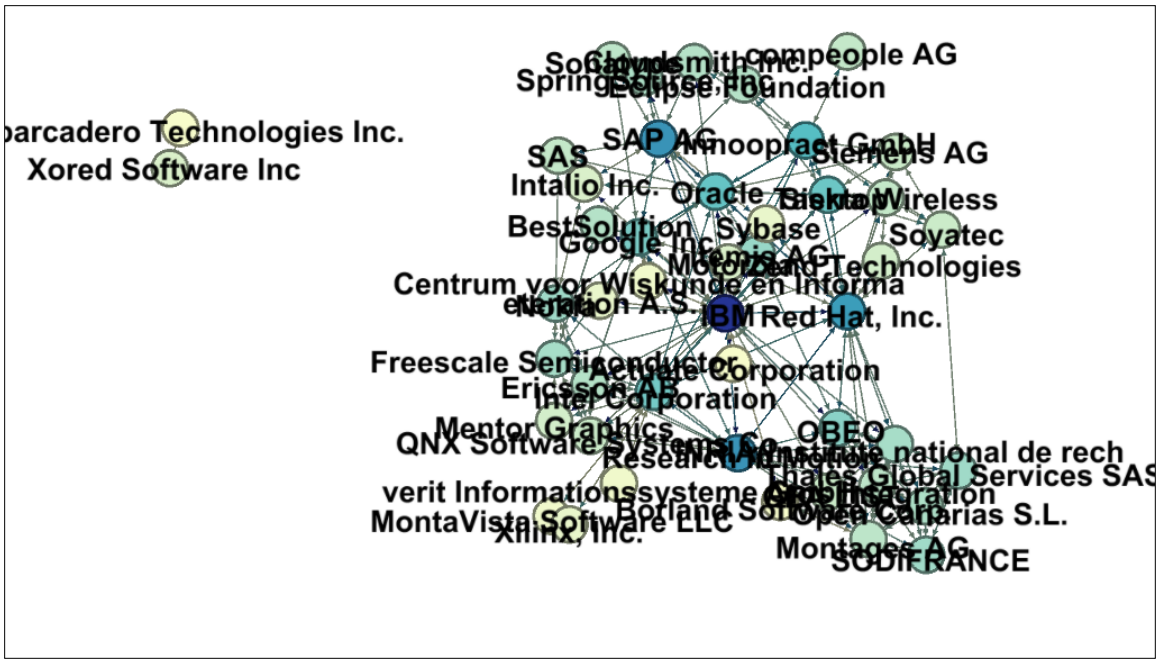
Eclipse company network in 2006



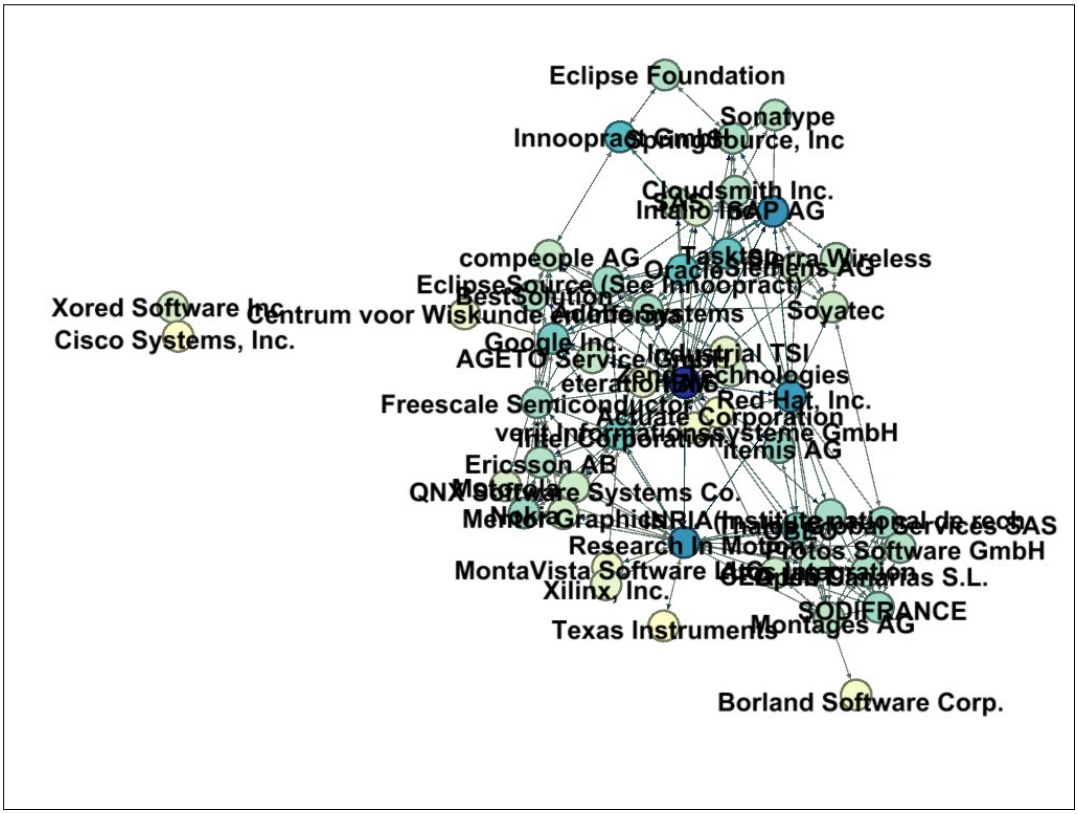
Eclipse company network in 2007



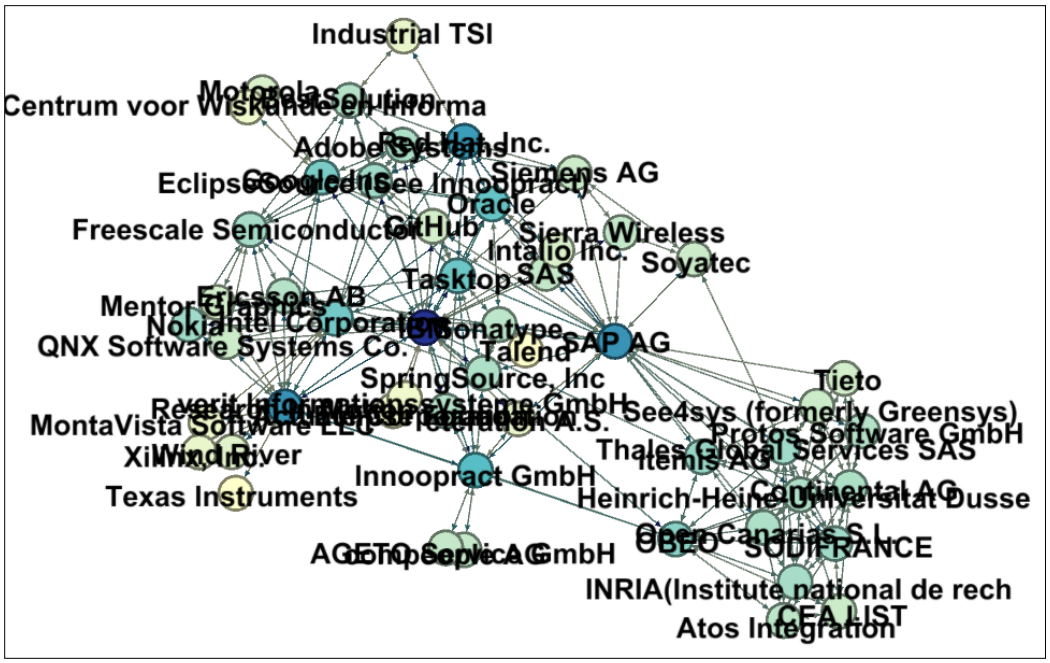
Eclipse company network in 2008



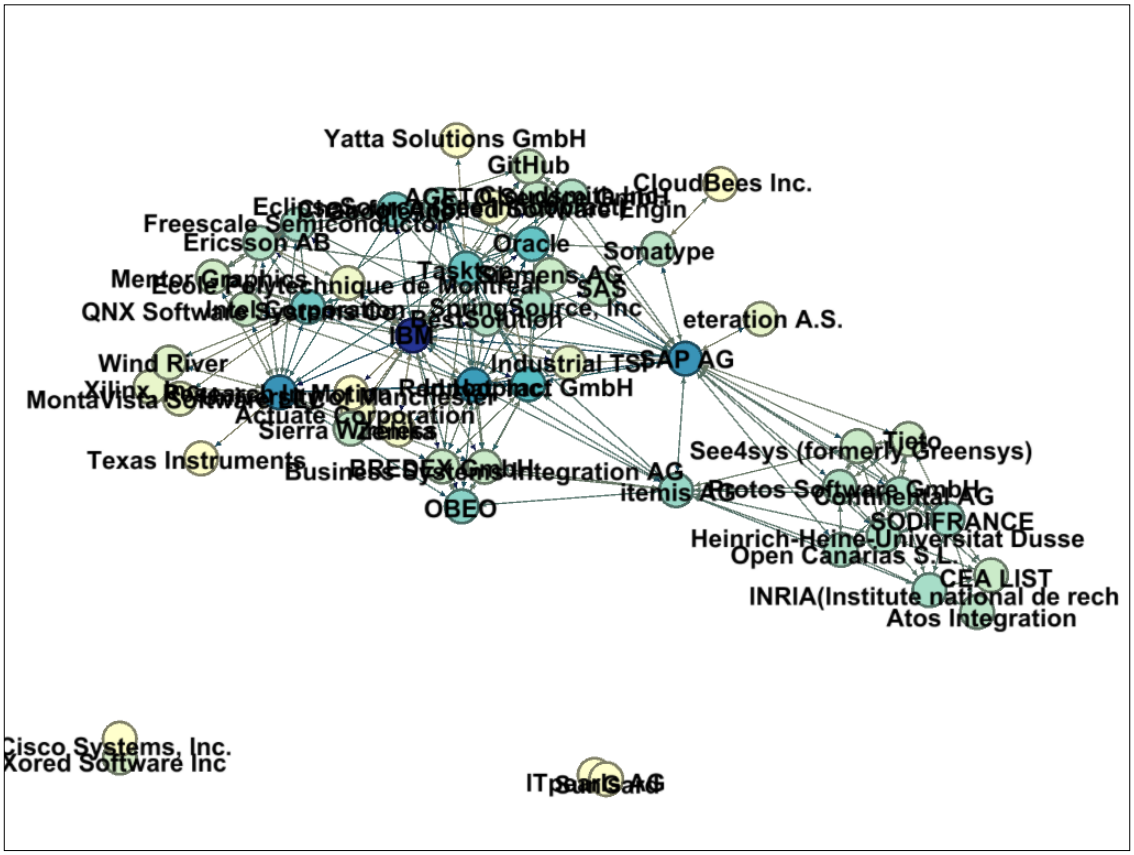
Eclipse company network in 2009



Eclipse company network in 2010



Eclipse company network in 2011



Eclipse company network in 2012

Appendix 3 Company network converter (A piece of code)

Company network converter was built for this study. It was used to convert "company to project" network into "company development relationship" network. The Company network represents the real development relationship between companies. It creates ties between companies if they concurrently work on same project. It also has capability to filter out companies with relative lower contributions based on different requirements. This software application was built in C#.

```
private void GenerateExcel(DataTable dataToExcel, string excelSheetName, string fileName)
{
 string currentDirectorypath = Environment.CurrentDirectory;
 string finalFileNameWithPath = string.Empty;
        finalFileNameWithPath = string.Format("{0}\\{1}.xlsx", excelSheetName, fileName);
 var newFile = new FileInfo(finalFileNameWithPath);

 //Step 1: Creating object of ExcelPackage class and pass file path to constructor.
 using (var package = new ExcelPackage(newFile))
 {
 //Step 2: Addingd a new worksheet to store Excel data
 ExcelWorksheet worksheet = package.Workbook.Worksheets.Add(excelSheetName);

 //Step 3: Loading datatable form the first cell of worksheet.
 worksheet.Cells["A1"].LoadFromDataTable(dataToExcel, true);

 //Step 4: Save and create Excel file.
 package.Save();

 MessageBox.Show(string.Format("File name '{0}' generated successfully.",   fileName)
 , "File generated successfully!", MessageBoxButtons.OK,   MessageBoxIcon.Information);
}
}
private void convert_button_Click(object sender, EventArgs e)
{
 int row = 1;
 int column = 0;
 String value;
 newDataTable = new System.Data.DataTable();
 int count = 1;
 //Generate converted data metric (Company development relationship)
 foreach (DataColumn TempColumn in dt.Columns)
```

```
{
   if (count == 1)
   {
     newDataTable.Columns.Add("Company-From");
   }
   else if (count == 2)
   {
     newDataTable.Columns.Add("Company-To");
    }
   else
   {
      newDataTable.Columns.Add(TempColumn.ColumnName);
    }
        count++;
       }
       for (column = 2; column <= newDataTable.Columns.Count - 1; column++)
       {
        for (row = 0; row < dt.Rows.Count; row++)
       {
        value = dt.Rows[row].ItemArray[column].ToString();

        DataRow currentRow = dt.Rows[row];
        DataRow nextRow = null;//dt.Rows[row+1]
        bool has = false;
        double coefficient = 0;

        if (checkBox1.Checked)
        {
         //Set weight filter coefficient
         coefficient = Convert.ToDouble(textBox3.Text);
        }
        if (value != null && Convert.ToDouble(value) <= coefficient)
        {
         continue;
        }
       int next = 1;
       if (row < (dt.Rows.Count - 1))
       {
        nextRow = dt.Rows[row + next];
       }
   while (row < (dt.Rows.Count - 1) && nextRow != null &&
   (currentRow.ItemArray[0].ToString() == nextRow.ItemArray[0].ToString()))
       {
       if (row < dt.Rows.Count && value != "0" && nextRow != null)
       {
        if (nextRow.ItemArray[column].ToString() != "0")//Check the year value
        {
         if (currentRow.ItemArray[0].ToString() ==  nextRow.ItemArray[0].ToString())//Check
      the project string
          {
       int i = 0;
```

```
        //Check if companies' development relationship already exists.
        foreach (DataRow myRow in newDataTable.Rows)
        {
          bool leftToRight = newDataTable.Rows[i].ItemArray[0].ToString() ==
        currentRow.ItemArray[1].ToString() &&
        newDataTable.Rows[i].ItemArray[1].ToString() ==  nextRow.ItemArray[1].ToString();
        bool rightToLeft = newDataTable.Rows[i].ItemArray[1].ToString() ==
        currentRow.ItemArray[1].ToString() && newDataTable.Rows[i].ItemArray[0].ToString()
        == nextRow.ItemArray[1].ToString();
        double currentValue = Convert.ToDouble(currentRow.ItemArray[column]);
        double nextValue = Convert.ToDouble(nextRow.ItemArray[column]);
        if (leftToRight || rightToLeft)
        {
          if (myRow[column] != DBNull.Value)
          {
          int valueTemp = Convert.ToInt32(myRow[column]) + 1;
          myRow[column] = valueTemp;
          has = true;
        }
        else
        {
          if (currentValue >= coefficient && nextValue >= coefficient)
          {
            myRow[column] = 1;
          }
          has = true;
          }
         }
          //add 1 to degree
          i++;
        }
        if (has == false)
        {
          if (Convert.ToDouble(currentRow.ItemArray[column]) >=
        Convert.ToDouble(textBox3.Text) &&
        Convert.ToDouble(nextRow.ItemArray[column]) >=   Convert.ToDouble(textBox3.Text))
        {
          DataRow dr = newDataTable.NewRow();
          string newValue = currentRow.ItemArray[1].ToString();
          dr[0] = newValue;
          dr[1] = nextRow.ItemArray[1].ToString();
          dr[column] = 1;
          newDataTable.Rows.Add(dr);
          }
         }
        }
 }//Check the year number
}
 next++;
if ((row + next) < dt.Rows.Count)
{
```

```
      nextRow = dt.Rows[row + next];
     }
     else
     {
      nextRow = null;
     }
    }// end of while
   }// each row
  }// each column
   dataGridView1.DataSource = newDataTable;
   textBox1.Text = "Completed, " + newDataTable.Rows.Count + " rows" + " and " +
newDataTable.Columns.Count + " columns data has been inserted.";
}
```