Source of Emergent Contributors in IBM Collaborative Lifecycle Management Ecosystem

by

Navpreet Kaur
B.Tech. Lovely Professional University 2013

A Master's Project Submitted in Partial Fullfilment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

Source of Emergent Contributors in IBM Collaborative Lifecycle Management Ecosystem

by

Navpreet Kaur
B.Tech. Lovely Professional University 2013

Supervisory Committee

Dr. Daniela Damian, Supervisor
(Department of Computer Science)

Dr. Kelly Blincoe, Committee member
(Department of Electrical and Computer Engineering, UOA New Zealand)

Supervisory Committee

Dr. Daniela Damian, Supervisor

(Department of Computer Science)

Dr. Kelly Blincoe, Committee member

(Department of Electrical and Computer Engineering, UOA New Zealand)

Abstract

The use of online communication tools in today's modern software systems development has created a culture of transparency. With communication tools widely available, information travels across the teams and projects in an ecosystem. Communication and coordination is critical for the success of software systems development. The previous study of IBM's collaborative life-cycle management proved the existence and importance of emergent contributors in the requirement development phase of a work item. These contributors are difficult to find at the ecosystem level. This report discusses the methodology to find the source of emergent contributors in IBM collaborative life-cycle management data using the *Reference Coupling* methodology.

Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank my supervisor, Dr. Daniela Damian, for mentoring, supporting and encouraging throughout the entire project. I thank Dr. Kelly Blincoe, for serving in my oral defence committee.

I would like to express a special word of thanks to my friends Parminder Kaur and Kathy Do who tirelessly listened to my ideas and offered encouragement. Finally, I would like to thank Aminah Yussuf and Francis Harrison who assisted me with this project.

Chapter 1

Introduction

The new culture of transparency adapted by most of the organizations by using online communication tools for software system development made the sharing of knowledge much easier. This sharing of knowledge within and outside the teams and projects causes technical dependencies between software projects within an ecosystem.

In an IT ecosystem, such as IBM Collaborative Lifecycle Management (CLM), the software system does not develop in isolation. There are number of teams and projects working together to accomplish products. The CLM follows the "open commercial" development model [10], which grants more transparency in development. The CLM ecosystem consists a number of products such as the Rational Team Concert (RTC), Rational Quality Manager (RQM), Rational DOORs Next Generation (DNG), Rational Requirement Composer (RRC), Rational Software Architect (RSA), Rational Rhapsody and Rational insight. The teams and projects coordinate using an online communication tool known as Jazz. IBM's Jazz CLM is a set of integrated tools that provides requirements management, quality management, change and configuration management, project planning and tracking on a common unified platform [3]. In Jazz, a work item tracks the planned work for a team or project. The work items can be classified into types such as Defect, Enhancement, Task, Plan Item and

Story.

In a study of the IBM CLM ecosystem by Aminah Yussuf [16], the emergent contributors are the discussants that are not officially part of the development team or required to participate, but contribute to the work item discussions. The author suggests that emergent contributors within the ecosystem can exist anywhere and they contribute towards the requirement negotiation phase of the work item. In a study by Damian and Kawan [12], the authors suggest that experts are difficult to locate in an ecosystem and developers tend to broadcast information to find the right person. The authors also suggest that the experts are not part of a recipient list but contacted by carbon copy(cc) or an email discussion forwarded to them. In the same study of IBM CLM data [16], approximately half of the discussant population are emergent contributors and their contributions are resourceful and should be identified during the requirement phase of a work item. However, the same study opens the question of emergence of these emergent contributors. Do they emerge from the technical dependencies shared between software projects? In a study by Francis Harrison [8], these dependencies are not easily identifiable at the ecosystem level. The study applied the approach of Reference Coupling, technical dependencies based on cross-reference made between projects. The Reference Coupling turns out to be a promising approach for the GitHub ecosystem.

Our research focuses on identifying the source of emergent contributors in the IBM CLM data using Reference Coupling methodology. The emergent contributors act as a catalyst to the requirement discussion process, making the requirement clarification and decision-making process fast and more efficient [16]. Given the above fact, it is very important to look for the emergence of these contributors so that they can be involved in early phases of the work item. Considering the development process in IBM CLM, we used the online discussions on Jazz as one of our data source to see if emergent contributors come from the work items cross-referenced in those discussions. We have also looked at what type of dependencies were shared between the cross-referenced work items and the work items on which the cross-reference link was created. The

other data source used to look for the source of emergent contributors was the work items mentioned on IBM's Jazz link tab. As mentioned above, both data sources are link to other work items; it is worth studying the overlap between the two data sets.

The remainder of this report is organized as follows:

Chapter 2 provides a literature review from other researchers.

Chapter 3 describes the methodology for studying source of emergent contributors in IBM's ecosystem.

Chapter 4 presents the analysis and results.

Chapter 5 discusses results to draw conclusions

Chapter 2

Literature Review

Today, modern software systems are developed in a distributed environment. There are various factors responsible for the use of distributed environment such as client site support, travel or relocation costs for project members, project members being unable to relocate, technical resources only available at certain places, and shortage of office space. [6] Communication is the critical factor for the success of distributed software systems. Almost all organizations use online communication tools for co-ordination among teams and projects. The communications within and outside the teams and projects help in understanding the requirements and achieving the deadlines [16]. Studies have shown that team members seek specific technical information or administrative help from people outside their software teams [5]. Experts are not part of initial email discussions and emerge only after a number of initial messages have been sent [12]. In Yussuf's study of IBM CLM ecosystem [16], emergent participants play a major role in requirement discussion yet they contribute later. The study further proved that the early contribution of emergent contributors helped in requirement clarification of work items. Now the question is if these contributors do exist, should they be contacted earlier? To contact emergent contributors earlier, one should be able to locate them in an ecosystem and for that their origin should be known. As the software systems do not develop in isolation, do

emergent contributors emerge from the technically dependent software systems?

In a study of the GitHub ecosystem [4], developers make use of other projects in their own work and watch the projects they depend on. In GitHub, a link is automatically created to another repository when it is cross referenced in a pull request, issue or commit comment. Due to the transparency of ecosystem, developers view actions of other developers. In a Study of code reuse in Open Source Software [15], the development of software systems involve reusability of existing systems instead of creating them from scratch. Reuse of artifacts from all stages of software life cycle can be highly profitable [9, 11]. It is challenging to identify dependencies on the ecosystem level due to the magnitude and number of projects involved. The methods proposed [13][14], based on source code analysis requires large amounts of memory and computation time. The other proposed method of determining technical dependencies using configuration settings of projects [7]; however, this information is not always available.

The Jazz reporting services provide real time collaboration among teams and projects; the users make cross-references to other work items in online discussions and the Link tab. These cross references are established on the basis of various relations [2] such as parent-child, duplicates, sub-tasks, plans and iterations. The cross-references to other work items in the IBM ecosystem made the ecosystem ideal to apply reference coupling methodology [8].

Chapter 3

Methodology

This chapter describes the methodology and source of data used to answer our research on finding the source of emergent contributors.

3.1 Data Analysis Methods

The research analysis was divided into four different sets of questions. Below is a description of the analysis methods used for each research question:

Research Question 1:
If emergent users do exist within the IBM software ecosystem, what is the source of their emergence?

To answer the above question, we started analyzing the work items having emergent contributors from our previous set of data [16]. There were two ways to look into the IBM CLM data. The first way was to analyze the online discussions on the work items. In this study of online discussions, the contributors on work items made links to other work items i.e., a cross-reference to other work items in their discussions. The second method was to look at the cross-references explicitly defined by IBM i.e. system dependencies on the IBM Jazz Link tab. On the Link tab, the work items included a set of predefined link

types for traceability. The contributors made links to other work items manually according to predefined types. There were 26 system-defined dependencies available on the Link Tab. In our research, we have found 14 widely used dependencies in IBM. Out of these 14, some popular dependencies were Add Related, Set Parent, Set Children, and Add affected by defect.

The *Add Related* dependency is used to capture a general relationship between the CCM (Change and configuration Management) work item and another item. The *Set Parent* and *Set Children* dependencies are used to show hierarchical structure of work items and capture a parent-child relationship between two work items. The child work item, use the set parent link type to set the parent work item and vice-versa. The fourth popularly used dependency that we found in our dataset was *Add Affected by defect* and which is used to capture a relationship in which a work item is affected by a defect item.

Following are the definitions used for the concept of emergent users and their source:

Emergent users: The emergent users are the discussants that are not officially part of the development team or required to participate, but contribute to work item discussions [16]. To find these emergent users, the idea is to check these users in the internal teams assigned to the work item. As the team's composition is dynamic and can change over time, we used the current composition of members, admins and team history. If the given user was not present neither the internal team's history nor the current composition of team members and admins, it implies that user is emergent to this work item. There can be some other reasons for a user to not be present in data such as incomplete team history, missing user id, and the inability to find internal teams. Table 3.5 describes all the codes used.

Source of emergent users: The method for finding the source of emergent users is very similar to the method used to find the emergent users. If a user is emergent in a work item discussion, we checked if the emergent user was part of the team's history, member composition, or admin composition for any cross-

referenced work item. If the user is present in any team's history or current composition of members and admins, then the cross-referenced work item is the source for this user, otherwise not. Other factors such as unavailability of an internal teams list, incomplete team history and missing user id are also considered and have codes assigned accordingly shown in Table 4.1.

Research Question 2: Research question 2 is sub divided into two different sets of data.

Research Question 2.1:
Do the emergent contributors come from the cross-referenced work items that are mentioned in online discussions?

The text of online discussions from Table 3.7 was parsed using regular expressions in Java shown in Table 3.1. A cross-reference to other work items in online discussions results in a link pattern in API data.

1	item ([0-9]+)
2	(< synthetic >)(·∗)(< /synthetic >)
3	(·∗)(Extracted from work item ([0-9]+))(·∗)
4	(·∗)(Copied from work item ([0-9]+))(·∗)
5	(task ([0-9]+))
6	(story ([0-9]+))
7	(defect ([0-9]+))
8	(<a href="https://jazz.net/jazz0([0-9])/resource /itemName/com.ibm.team.workitem.WorkItem/([0-9]+)">)
9	(< a href="https://jazz.net/jazz/resource/itemName /com.ibm.team.workitem.WorkItem/([0-9]+)">)

Table 3.1: Regular Expressions

Research Question 2.2:
Do the emergent users come from the link associations defined by IBM system dependencies on Link tab?

Work items include a set of predefined link types for traceability. There are 26 system-defined link dependencies on IBM Link tab. Creating a link type in one work item automatically creates a backlink in the other work item. In both items, the links are displayed in the link section on the Link tab. The link Set Parent in one work item creates a backlink Add child in other work item. The Table 3.2, shows the system defined dependencies.

Research Question 3:
What is the overlap between cross-referenced work items in online discussions and work item found on the Link tab?

We started our analysis using bottom up approach. We first concentrated on question 3 of our analysis to see if the cross-references in online discussions intersect with IBM system dependencies.

Research Question 4:
If the Reference Coupling technique finds new cross-references in online discussions, what type of dependencies are captured in online discussions that are not in overlap with link tab?

We addressed this question by manually studying 300 random comments where a cross-reference link was created. We categorized referenced work items as per IBM defined dependencies. We used the category "unknown" for some of the referenced work items as they were difficult to fit into the predefined categories.

1	Add Related
2	Add Blocks
3	Add Depends on
4	Add Resolves
5	Add Resolved By
6	Add SVN Revisions
7	Add Related Artifacts
8	Set Parent
9	Add Children
10	Add Duplicated By
11	Set Duplicate of
12	Add Affected By Defect
13	Add Contributes To
14	Add Related Change Request
15	Add Tracks
16	Add Affects Plan Item
17	Add Tracks Requirement
18	Add Related Test Script
19	Add Related Test Execution Record
20	Add Related Test Plan
21	Add Related Test Case
22	Add Implements Requirement
23	Add Affects Requirement
24	Add Tested By Test Case
25	Add Affects Test Result
26	Add Blocks Test Execution

Table 3.2: IBM system-defined dependencies

3.2 Data Source

IBM provides public REST APIs [1] to collect the data required from the Jazz repository. The data used in this research is from IBM CLM, our data collection had an archive of 16 projects. Which includes SilverCity1, JUnit Project, Jazz Collaborative ALM, PLE Extensions, DevOps Platform, Jazz Foundation, Test Scrum, SmartCloud Continuous Delivery, Z testing Purposes only – CLM 2012 (Change Management), Rational AMC, Rational Team Concert, Jazz Collaborative ALM(private), Community [Development], Jazz Business Partners, FabProject, Junit.

3.3 Data Processing

Across all 16 projects, we have collected information in three data sets. Our dataset consists of work items whose development iteration was between 2005 and 2015. The first set contains the team's information, the second set contains work item information and the third set contains the links' information from Link Tab. This team information includes the data for team admins, roles, members and history. The data for the work items includes work item history and communication data for each work item i.e., comments. The history files for work items and teams include records of changes that took place during the entire life span of work item.

Number of projects	16
Number of teams	324
Number of work items	3009
Number of comments	17708

Table 3.3: Total numbers for the data collected

The data from the REST API was collected in XML and converted into JSON files using python scripts. The enormous amount of data was stored in a MongoDB database. The required data was converted into a Postgres relational

database from JSON files stored in MongoDB. The tables in next section show data fields gathered for each data set.

3.3.1 Team Information

The team_history information, shown in Table 3.4, was collected as team compositions are dynamic and change over time. The team history information was required to answer our research question regarding the source of emergent contributors when they emerge from referenced work items. The team history information provided the data regarding Add or Remove of a user with date-time information. It is possible that a member was part of the team at one point in time and not later, which means that current composition of the team does not include data for that user. Table 3.5 and Table 3.6 represent the current team members and current admins assigned to the team, respectively.

Team_id	_4FG5EKWjEeGhUIY5Avv5pw
Change_at	2013-01-28 19:13:43
Change	Add/Remove
Member_name	XYZ
Project_id	_Q2fMII8EEd2Q-OW8dr3S5w

Table 3.4: Example of team_history

User_id	XYZ
Team_id	_7d-KkArbEd-2dIN2nAqZ3A
Project_id	_0rSUcMixEd6A25wBGCmItw

Table 3.5: Example of team_members

3.3.2 Work Item Information

Table 3.7 shows the data gathered for each work item in online discussions. It also includes the information regarding weather or not the comment is made by

User_id	XYZ
Team_id	_s3ihQFZyEdyNxbmpykGIFA
Project_id	_1w8aQEmJEduIY7C8B09Hyw

Table 3.6: Example of team_admins

an emergent contributor. The column Is_emergent_by_team is populated using the methodology used by Aminah Yussuf [16]. For the Is_emergent_by_team field, the values 0, 1, -1, -2, -3, represent "No", "Yes", "Data outside history range", "User ID missing from user table", and "unable to identify internal team for this work item", respectively.

The data in Table 3.8 was collected from the work item history. It included list of all the teams assigned to a work item throughout its lifetime.

A link is created on a work item when it is mentioned in another work item's online discussions. These work items mentioned in online discussions are cross-referenced work items and were extracted using regular expressions from the comment column in Table 3.7. In Table 3.9, ref_workitem represents the referenced work item.

Workitem_id	183390
Comment	@XYZ should we change past iteration?
Comment_by	user id
Comment_at	2011-11-07 15:31:00
Id	Unique comment id for each comment
Is_emergent_by_team	0, 1, -1, -2, -3

Table 3.7: Example of workitem_comment

Workitem_id	228054
Iternal_teams	List of all internal teams to this work item

Table 3.8: Example of an internal team for work item

Workitem_id	183390
ref_workitem	190390
Comment	@XYZ should we change past iteration?
Comment_by	user id
Comment_at	2011-11-07 15:31:00
Id	Unique comment id for each comment
Is_emergent_by_team	0, 1, -1, -2, -3

Table 3.9: Example of cross referenced work item

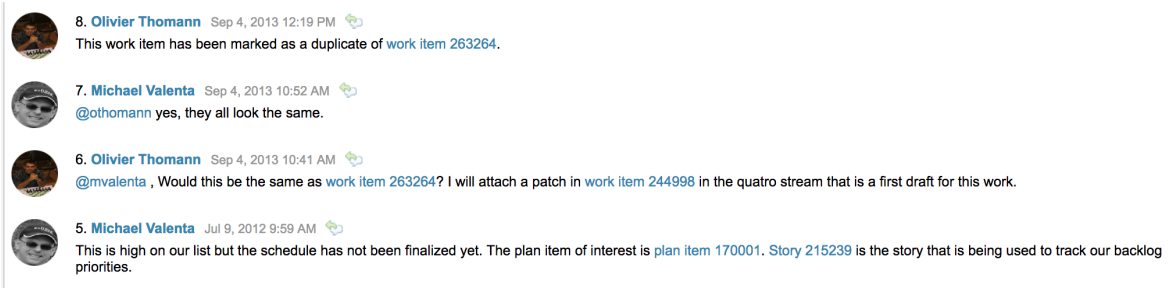

Figure 3.1: Links created in online discussions

3.3.3 Link Tab Information

IBM Jazz provides link types to customize the linking relationships between work item. Links can show relationship properties such as derivation, dependency, hierarchy, associations, development, and test. Link relationships can be captured using 26 system-defined dependencies.

Workitem_id	180165
Ref_Workitem	170745
Dependency	Type of dependency

Table 3.10: Example of a link captured from IBM system dependencies

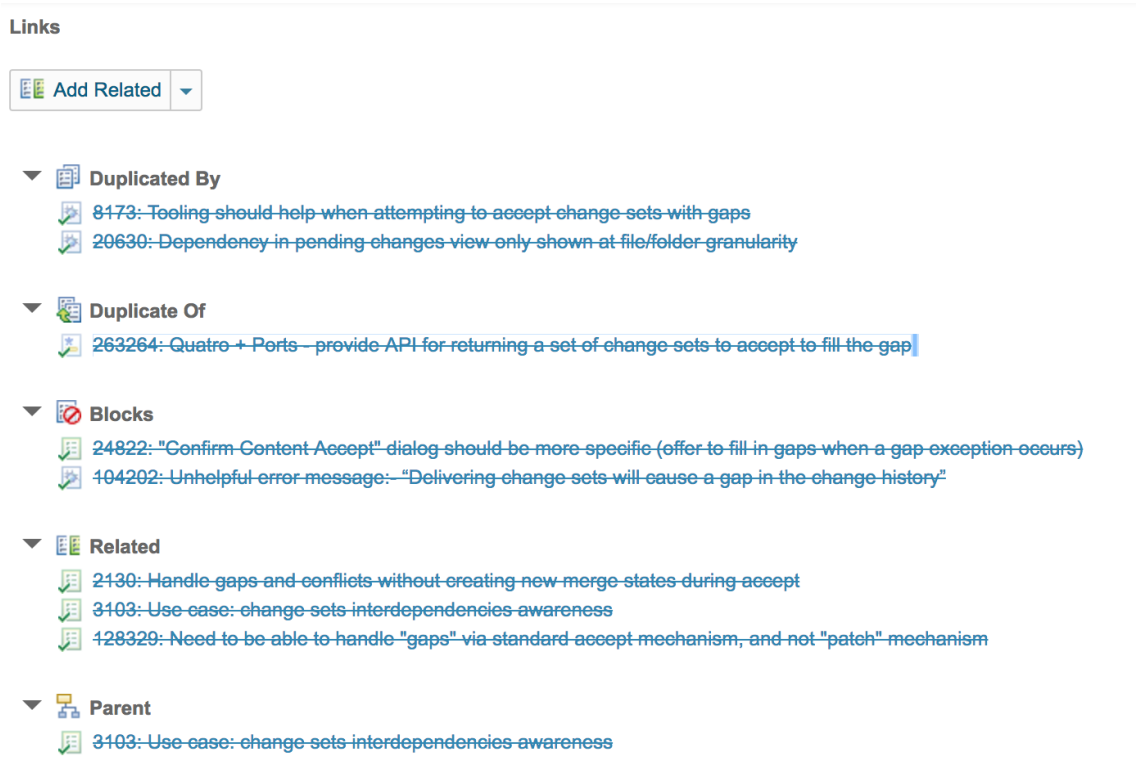

Figure 3.2: The links created on the Link tab

3.3.4 User's Information

The user's information is required to map the comment_by data field from Table 3.9 to Table 3.4, 3.5 and 3.6. The Table 3.9, does not include the official name of the users whereas, the other tables contain the official names.

User_id	abc
Name	Abc xyz
Email	abc@ibm.com

Table 3.11: Example of internal users

Chapter 4

Analysis and Results

Research Question 3:
What is the overlap between cross-referenced work items in online discussions and work items mentioned by IBM system dependencies in the Link Tab?

In our dataset, there were 4506 distinct links explicitly defined by IBM system dependencies and 2108 distinct links from online discussions. The intersection of these datasets contained 682 links which left 1426 links that were not captured by IBM system dependencies.

Research Question 2:

Research question 2 is sub divided into two sub-questions 2.1 and 2.2. The analysis data is available in one table as shown in Table 4.1, the data field IBM_system_dependency represents the referenced work items captured from the Link tab and data field Comment_cross_ref represents the cross-referenced work items in online discussions.

In Table 4.1, all the contributions are made by emergent contributors. A python script was used to find the source of emergent contributors. The Tables 3.2-3.9, were used to create the resulting data analysis table in Table 4.1. The script takes *Comment_by*, *Comment_at* and the referenced work item

Workitem_id	283310
Comment	@XYZ should we change past iteration?
Id	Unique comment id for each comment
Comment_by	user id
Comment_at	2011-11-07 15:31:00
Is_emergent_by_team	1
IBM_system_dependency	79380,283275,283207,96347,283246,96356,79387,283210
comment_cross_ref	283210
source_cross_ref	283207,283275,283246
type_cross_ref	12,12,12
is_officially_assigned_ref_workitem	1

Table 4.1: Example of source_of_emergent_contributor

(*IBM_system_dependency* and *comment_cross_ref*) as input and append the *source_cross_ref* for each work item in Table 4.1. The field *is_officially_assigned _ref_workitem* defines the source of emergent contributors. The value 1 for Is_officially_assigned_ref_workitem means that the emergent contributor comes from a referenced work item, 0 means that the emergent contributor does not come from a referenced work item, -1 means that the data is outside of the history range, -2 means that the user id was missing and -3 indicates that the internal table for a work item was unable to be identified.

In 4054 contribution links, there were 1025 distinct work items. We have successfully tracked the source of emergence for 2836 contribution links out of 4054 contribution links. The rest of the contributions i.e. 1218, were unable to be tracked as some of the contributions were made outside of the history range, some contributions were missing internal teams for referenced work items and one case, the official user name in *comment_by* was not present in the dataset.

Research Question 2.1:

Do the emergent contributors come from the cross-referenced work items that were mentioned in online discussions?

In 2836 tracked contribution links, 844 contribution links were made by contributors who were officially assigned from referenced work items. Further, from 844 contribution links, there were 567 links from online discussions. By examining the 567 online discussion links, it turns out that 198 contribution links were in common with the Link Tab links. Therefore, rest of the 369 independent contribution links were from cross-referenced work items made in online discussions. So, 19.2% of emergent users have their source from references made in online discussions.

Research Question 2.2:
Do the emergent contributors come from the link associations explicitly defined by IBM system dependencies on the Link tab?

As mentioned in question 2.1, there were 844 contribution links, out of which 567 links were from online discussions and 475 contributions were made by users whose source of emergence was link tab. Out of the 475 contributions, 198 contributions were present in both sets i.e. online discussions and the Link tab. Independently, 277 contributions were from the Link tab system-defined dependencies. So, 17.7% of contributions were made by IBM system-defined dependencies on Link Tab.

Research Question 1:
If emergent users do exist within IBM software ecosystem, what is the source of their emergence?

From the above research questions 2 and 3, we have found the source of emergent contributors from both the Link tab and Online Discussion datasets. We found 19.2% of emergent contributors were emerging from online discussions and 17.7% of emergent contributors from Link tab with 6.9% of intersection in two datasets. So, we found source of 30% of contributors in Source_of_emergent_contributor dataset.

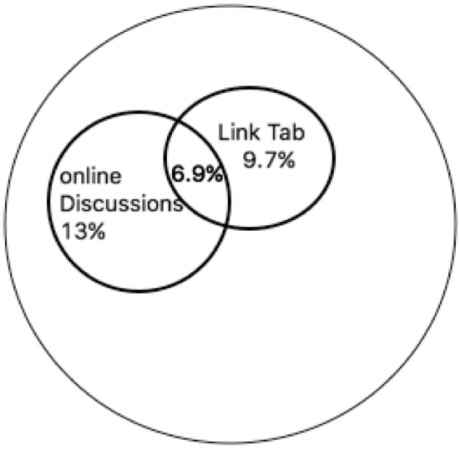

Figure 4.1: Source of Emergent Contributors: 30%

Research Question 4:
If the Reference Coupling technique finds new cross-references in online discussions, what type of dependencies are captured in online discussions that are not in overlap with link tab?

From research question 3, using reference coupling, 1426 distinct links were not captured by IBM's system- defined dependencies on the Link Tab. These 1426 Links were captured from 1770 online discussions. A selection of 300 random links was manually categorized according to the 26 IBM system-defined dependencies as shown in Table 3.1. The description of these dependencies was taken from IBM Jazz website. In table 4.2, the type of dependencies and total count is shown for 300 random links. Some of the discussions were hard to categorize so we used another dependency that we named "unknown".

The most popular dependency that we have found while categorizing was "Add Affected by Defect". There were 137 links out of 300 which fall under this category. The second most popular dependency that we categorized was "Add

Dependency Type	Count	Dependency Name
1	88	Add Related
3	4	Add Depends On
4	1	Add Resolves
7	8	Add Related Artifacts
8	3	Set Parent
11	19	Set Duplicate of
12	137	Add Affected By Defect
15	6	Add Tracks
16	2	Add Affects Plan Item
17	1	Add Tracks Requirement
21	2	Add Related Test Case
27	29	Unknown

Table 4.2: Type of dependencies and count for 300 random comments

Related". As discussed earlier in the methodology section, these results were very similar to the popular dependencies found in the Link Tab dataset.

To validate our results, these 300 links were also categorized by Francis Harrison, Segal Lab Member. We found 95% inter-rated agreement in both results. This 95% agreement suggests that new cross references in online discussions could have been placed under IBM system-defined dependencies.

Chapter 5

Discussion

In this report, we studied the source of emergent users using two different sets of data: Online Discussions and the link associations on Link tab. In a study by Aminah Yussuf [16], the author proved the importance of emergent contributors during the requirement negotiation phase of work items. In our research, Reference coupling methodology results in new cross-references in online discussions which were not present in the Link tab. Furthermore, studying the cross-references and categorizing them under IBM system-defined dependencies with 95% of inter-rated results validated reference coupling methodology.

In the IBM Link tab, creating a link type in one work item automatically creates a back-link in the cross-referenced work item. Using the reference coupling methodology for parsing text in online discussions and then adding the newly found cross-referenced work items under the Link tab will create a back link in cross-referenced work item. As a result, a technical relationship is established in two work items. These technical relationships can be used as input data for a recommender system. The algorithm behind the recommender system will automatically find the technically dependent work items and potential emergent contributors, which are difficult to identify on an ecosystem level. Once these emergent contributors can be automatically identified at the early stages of requirement development, the decision-making process will be more fast and

efficient.

Bibliography

[1] IBM Inc. Jazz Community Site. 2006. https://jazz.net. Accessed: 2017-05-09.

[2] Managing work items in RTC. https://www.slideshare.net/ibmrational/2-rtcworkitems. Accessed: 2017-06-09.

[3] Open Services for Lifecycle Collaboration (OSLC) Wiki. 2008. http://open-services.net/bin/view/Main/ReportingRESTApi. Accessed: 2017-05-09.

[4] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. Social coding in github: Transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 1277–1286, New York, NY, USA, 2012. ACM.

[5] K. Ehrlich and K. Chang. "Leveraging expertise in global software teams: Going outside boundaries". In *2006 IEEE International Conference on Global Software Engineering (ICGSE'06)*, pages 149–158, Oct 2006.

[6] A. French and P. Layzell. "A study of communication and cooperation in distributed software project teams," in Proceedings of the International Conference on Software Maintenance (ICSM). *IEEE*, pages 146–158, 1998.

[7] D. M. German, J. M. Gonzalez-Barahona, and G. Robles. A model to understand the building and running inter-dependencies of software. In

14th Working Conference on Reverse Engineering (WCRE 2007), pages 140–149, Oct 2007.

[8] Francis Harrison. "Reference Coupling A Method for Identifying Software Ecosystems of Technically Dependent Projects," Master's Thesis, University of Victoria. pages 1–78, 2015.

[9] Yongbeom Kim and Edward A. Stohr. Software reuse: Survey and research directions. *Journal of Management Information Systems*, 14(4):113–147, 1998.

[10] A. Knauss, A. Borici, E. Knauss, and D. Damian. Towards understanding requirements engineering in it ecosystems. In *2012 Second IEEE International Workshop on Empirical Requirements Engineering (EmpiRE)*, pages 33–36, Sept 2012.

[11] Charles W. Krueger. Software reuse. *ACM Comput. Surv.*, 24(2):131–183, June 1992.

[12] I. Kwan and D. Damian. "The hidden experts in software-engineering communication (nier track)",in Proc. Intl Conf. Software Engineering (ICSE). *ACM*, page 800–803, 2011.

[13] Mircea Lungu, Romain Robbes, and Michele Lanza. "Recovering Inter-project Dependencies in Software Ecosystems". ASE '10, pages 309–312, New York, NY, USA, 2010. ACM.

[14] A. Mockus. "Amassing and indexing a large sample of version control systems: Towards the census of public source code history". In *2009 6th IEEE International Working Conference on Mining Software Repositories*, pages 11–20, May 2009.

[15] Audris Mockus. Large-scale code reuse in open source software. In *Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development*, FLOSS '07, pages 7–, Washington, DC, USA, 2007. IEEE Computer Society.

[16] Aminah Yussuf. "An Exploration of Emergent Contributors Within IBM Collaborative Lifecycle Management Ecosystem," Master's Thesis, University of Victoria. pages 1–65, 2015.