# An Exploration of Emergent Contributors Within IBM Collaborative Lifecycle Management Ecosystem

By

Aminah Yussuf

BSc., Applied Software Engineering, University of Reading, 2012

A Thesis Submitted in Partial Fulfillment

of the Requirements for the Degree of

Master of Science

in the Department of Computer Science



University of Victoria

# An Exploration of Emergent Contributors Within IBM Collaborative Lifecycle Management Ecosystem

By

Aminah Yussuf

BSc., Applied Software Engineering, University of Reading, 2012

**Supervisory Committee**

Dr. Daniela Damian, (Department of Computer Science)

**Co-Supervisor**

Dr. Eric Knauss, (Department of Computer Science and Engineering, CHALMERS)

**Co-Supervisor**

## Abstract

**Supervisory Committee**
Dr. Daniela Damian, (Department of Computer Science)
**Co-Supervisor**
Dr. Eric Knauss, (Department of Computer Science and Engineering, CHALMERS)
**Co-Supervisor**

Today, software ecosystems have become transparent, allowing users to submit issue reports and new feature requests to the development team. The more permeable boundaries of ecosystems provide an open culture paradigm where stakeholders, customers, developers and other user groups have the access to participate during all phases of requirement development. One example of this open culture in software ecosystems is found in work item discussions, which are aimed to improve how requirements are elicited, analyzed and validated. In this thesis, we investigate who participates in requirements discussions, identifying and focusing on emergent contributors; discussants that are not officially part of the development team or required to participate, but contribute to work item discussions. We report from a case study of online requirement discussion in IBM's collaborative lifecycle management. We find that external contributors emerge frequently during discussions and that they mediate the clarification of requirements. Our results indicate that it is important for emergent contributors to be involved early in the requirements process, otherwise there is a negative effect on the work items' progress. We discuss the implications of our findings for both practitioners and researchers with suggestions for future studies.

## Table of Contents

## List of Tables

## List of Figures

## Chapter 1 - Introduction

Over the years, the scale of modern software systems has steadily increased. During the development of these modern software systems, teams' and projects' composition are dynamic. This has lead to movement of people and their knowledge across requirements within and outside of teams and products.

Communication is especially critical to the success of distributed software teams. Research shows that when coordination is needed but does not occur, task resolution time, software faults, builds failures, redundant work, and schedule slips can all increase [45]-[7]. The nature of evolving modern software systems and dynamic composition of the team and project members has resulted to uncertainty of requirements, interdependency between development tasks and expert seeking.

In modern, large, globally distributed software development projects, team members have difficulties identifying experts and seeking knowledge from remote sites [47]. Tools and methods that assist developers in identifying who to coordinate with already exist. But these tools rely on identifying technical dependencies between tasks or code conflicts to make recommendations on coordination needs. However, developers seek expert knowledge on requirements through communication. This usually occurs before the existence of technical dependencies within the code base.

Studies have found that although technical dependency signals whom to coordinate with this is not always the right person to seek expert knowledge from [17]. Some of this expert knowledge comes from product's domain and market, relevant technology and product management [18]. However developers, testers and other team members are not aware of residing expertise [3].

The dynamic nature of team and projects' members composition further complicates expert identification. This is because when experts are needed within a given team, the qualified member could have been moved to a different team within or across products. Since modern software systems within IBM's ecosystem are not developed in isolation, the probability of experts existing anywhere within the software ecosystem is very high.

This thesis focuses on identifying and investigating the presence and impact of expert participation through online communication. In modern, large and open development environments, online communication is the main mechanism that underlines coordination process. Through online communication team members are able to reach an unambiguous understanding of requirements, seek expert assistance leading to a more effective coordination.

### 1.1 Summary of Problem Statement

Developers are constantly seeking other developers [17]. Also, Damian and Kwan [17] found that when developers do not know the right individuals to seek knowledge from, they send email broadcasts that cause information overload which results in communication breakdown. Although the research community has provided solutions that help software projects identify related team members through interdependencies within source code and tasks [7], this does not provide a desirable solution for developers who seek expertise through online communication before getting to the implementation stages.

### 1.2 Research goal

Our research goal is to study emergent contributions and investigate the impact of their participation during the development of requirements. We define emergent contributors as discussants from a software product's ecosystem who are not responsible for the development of a requirement but contributed to the requirement in some way. Based on existing literature that supports the notion that developers do constantly seek other developers, we decided to find out if this was true within a software ecosystem. Also, we were interested in exploring the impact of emergent presence and contribution.

*RQ1. Do emergent contributors exist within a software ecosystem?*

*RQ2. How do they contribute towards the development of requirements?*

*RQ3. What is the impact of emergent participation towards the development of a requirement?*

### 1.3 Research methodology

To address the problem described in this thesis, we conducted a study on an open commercial software ecosystem known as IBM's Jazz Collaborative Lifecycle Management [43]. We used a mixed methods approach of investigation, which includes qualitative and quantitative method for data analysis.

For qualitative analysis, we carried out manual coding by employing the rules of grounded theory approach on the content of requirement discussions to understand the contributions of stakeholders to requirements [48]. We used a statistical analysis approach to understand the quantity of emergent stakeholders and their distribution accordingly.

## 1.4 Research Contribution

Our findings from this study provide better understanding of the impact of emergent contributors. We also recommend to both researchers and practitioners to take advantage of the presence of emergent stakeholders.

To software engineering researchers, our study provides more insight to an already existing trend in the literature. First, we provide evidence that given a software development domain that embraces openness as a culture (a software ecosystem) emergent contribution do take place. We also provide empirical evidence about the impact of emergent people towards the development of requirements. To practitioners, our study demonstrates the importance of identifying these people and involving them at the early stages of requirement development.

## 1.5 Thesis Structure

In chapter two, we provide the background information necessary to understand our research process. We explain the problem our investigation was set out to solve, our goal and overall contribution. This chapter describes current literature that is related to our research.

In chapter three, we explain the procedures carried out during our research analysis. We describe the methodologies that we used to answer our initial research questions. We present the findings that correspond to each of our research questions. We also discuss the implications of our research for both industry practitioners and researchers.

In chapter four we revisit our findings and their implications for both practitioners and researchers. We also highlight and discuss the threats to validity our research faces.

Finally in chapter 5, we revisit our research goals, questions and outline our overall contribution along with possible future work.

## Chapter 2 – Background and Related Work

In this chapter we review our literature investigation that is related to our research study. We also identify how our research contributes to existing literature.

Our literature search first focused on challenges within global software development. We concentrated on literature that investigated awareness as a challenge within globally distributed large software projects. Our selected literature suggests that awareness is a challenge in globally distributed large software project and is mostly found in a domain where openness is a culture [2], therefore we embarked on a literature review of software ecosystems. Our literature includes work from two main research bodies namely a) challenges found in global software development and b) software ecosystems.

### 2.1 Challenges in Global Software Development

Globally distributed large software projects face many challenges as their member's co-ordinate and collaborate over long distance. Much research has focused on investigating the issues faced by GSD teams and provides possible mechanisms and tools that help software developers and managers to mitigate these challenges. There are six identified categories of challenges faced by globally distributed software development teams one of which includes experts seeking and that most times when experts are needed they are not found, hence cannot be exploited.

In software engineering research, awareness has been given a lot of attention. As pointed out by Layzell and French [45] globally distributed teams have special needs for awareness. According to a four-month study carried out at IBM Ottawa software lab on studying the need for awareness within GSD team [6], it was found that due to the

dynamic nature of a work item throughout its development, awareness needs to be closely maintained. Also, they reported that communication breakdowns do occur due to information overload. Another study carried out by Damian and Kwan [6] found that one of the causes of information overload within largely distributed software team is due to developers broadcasting emails seeking awareness of other knowledgeable developers.

Evidently, awareness is a challenge faced by developers within largely distributed software projects. Also, the lack of mechanisms to support this, results in communication breakdowns.

According to a study by Kwan et al, experts quietly exist in software projects, and are almost hidden [17]. They discovered that hidden experts emerge within email discussion under four main conditions, which are a) when crisis occurs b) when they respond to an explicit request c) when they are forwarded an announcement d) when discussants follow up on a previous event i.e. a meeting. The same study showed that these experts do not only respond to situations when other users request them but also execute routine tasks where they are resourceful. This research emphasises that a good expert seeking mechanism will reduce the time required to resolve issues. Existing research also suggests awareness is needed between developers but also identifies the direct importance of expert detection.

**2.2 Software Ecosystem**

Jansen et al. define software ecosystems as follows [21]

> *"A software ecosystem is a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them. These relationships are frequently underpinned by a*

*common technological platform or market and operate through the exchange of information, resources and artefacts".*

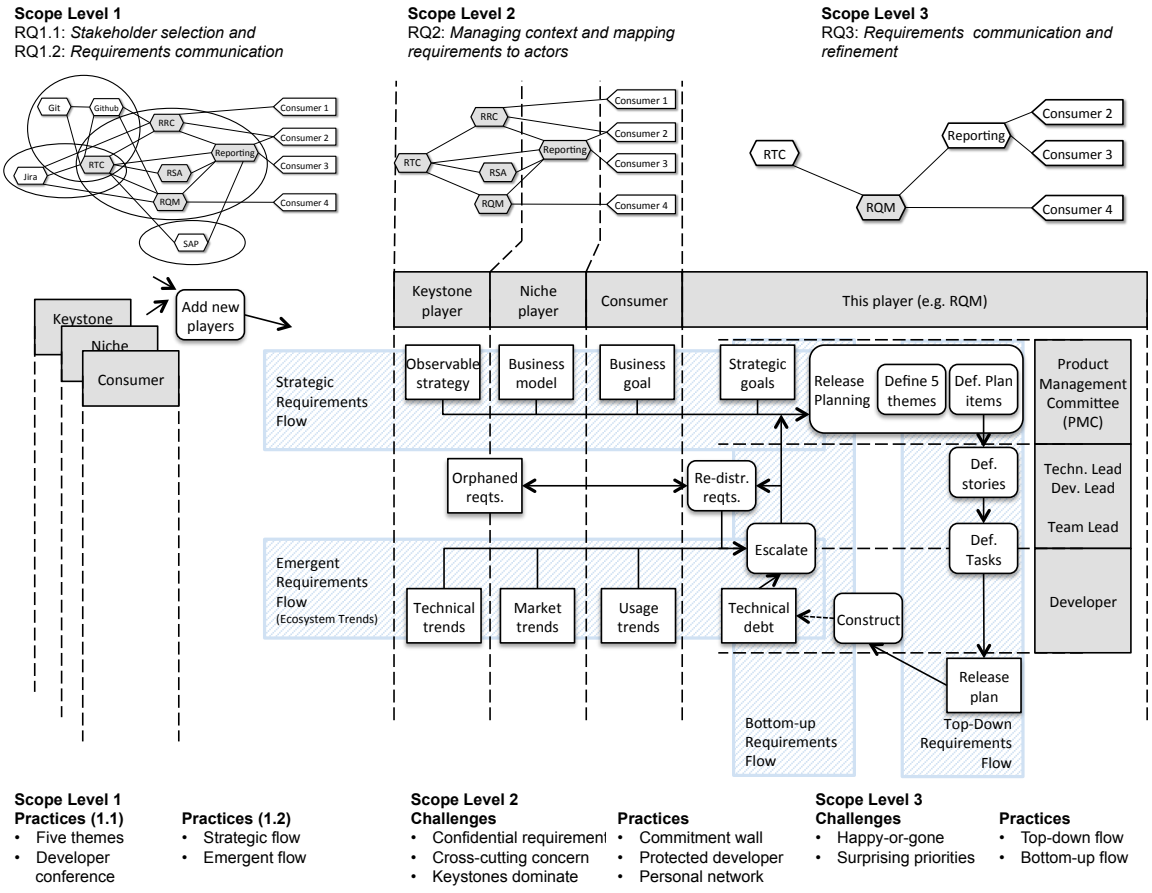
Knauss et al. [20] applied Jasen et al's definition and conceptualisation of a software ecosystem [21]. Using Jasen's framework [21], Knauss et al. conceptualised IBM's Collaborative Lifecycle Management ecosystem and analysed the flow of requirement discussion. They defined three scope levels:

**2.2.1** **Scope Level 1** – A high level of abstraction that offers a view of the ecosystem externally. The definition of the scope at this level is based on the technological platform, target market and third parties stakeholders. This level has an impact on requirements by facilitating integration or attraction of new actors into the ecosystem as well as approaching new markets.

**2.2.2** **Scope Level 2** – This level offers an internal view of the ecosystem where customers' requirements evolve from interaction with internal actors to the ecosystem.

**2.2.3** **Scope Level 3** – A view of the ecosystem from an organisational-centric perspective. This level affects requirements from a more operational and tactical prospective.

**Figure 2.1 Scope level view of IBM Jazz ecosystem [20]**




While each of these three levels [20] has an impact on requirement engineering, the first two levels have the most significant impact. The open commercial paradigm found in IBM CLM ecosystem provides a means for end users to articulate change or design requirements, but on the other hand it challenges end users to send requirement requests to the correct recipient.

Studies have identified software ecosystems as platforms that facilitate communication between software teams and help the flow of requirements from stakeholders to developers [20]. However, according to Souza and Redmiles, [7] the complexity and scale of stakeholder relationships creates an evident problem in managing

dynamic and diverse flow of requirement sources. Also, responding to market demands requires the ability to globally define requirements on the strategic level, while the scale and complexity of software ecosystems require self-organized teams' ability to locally and timely address context specific customer need 'just-in-time' [20]. Hence, from current research although ecosystems facilitate requirements engineering they also face the constraint of emerging requirements and delivering requirements in "real time" to customers. The fact that software ecosystems constantly evolve causes the ideal recipient for a requirement to change over time, hence, it is relevant to re-think the way requirements travel through the software organization and which roles are responsible to seek alignments of goals.

Related work focuses on identifying dependent tasks through technical dependencies, which can only be identified during implementation. But, requirement discussion usually exists much earlier before implementation occurs and dependencies can be identified. In order to identify the alignment of goals and understand how requirements travel across a software organization, we need to study requirements in their context, which can be captured by investigating end users interactions with their systems [6] [26].

In Manikas and Hansen's literature review it was found that there is an increase in research on software ecosystems in recent years [25]. They report that current literature does not address organization of and decision-making in software ecosystems. In our study we argue that in order to make a decision about development of a feature, the right stakeholders need to be involved at the right time, which is very challenging.

Research on the behaviour of developers and other software engineers in software ecosystems is somewhat limited. In free and open source software, the *onion* model is well established [26] [27]. This model represents the involvement of stakeholders similar to the layers of an onion: a member starts from the external layers having tasks with low responsibility, e.g., translation, and slowly moves to the inner layers as the member gains responsibilities.

Another study [28] investigated the interaction of developers within the Ruby Github software ecosystem and found three notable roles: the *lone wolf* who works mainly alone to develop a large part of the system which then is used by the rest of the users, the *networker* who focuses on being connected to other developers, and finally those who contributed (parts of) one component and did not show significant activity apart from that.

With respect to the flow of knowledge and information within software ecosystems, Fricker proposes the model of requirements value chains [29]. This study continues this line of research by investigating who participates in these requirement discussions across the ecosystem, focusing on stakeholder interactions that are emergent and not anticipated during the planning stages of the project.

Despite these recent studies into ecosystems and large-scale requirements engineering, we lack systematic investigations into the stakeholder interaction around product requirements in software ecosystems. In our study we explore the existence, characteristics of emergent contributors and their impact on requirements within software ecosystems.

## Chapter 3 - Research Methodology

### 3.1 A case study

It is a general belief that software engineering research that is related to tools and methods should be based on empirical data. Hence we decided to conduct an empirical study. Also, we based our choice of case study on the population and diversity of a modern software engineering project and targeted a large globally distributed software system. Globally distributed software systems mostly rely on online communication tools which provide us with a significant quantity of data to mine and investigate.

This study focuses on identifying the presence and analysing the impact of emergent contributors from their social interactions during the development of requirements. Given our goal to investigate emergent participants through their social contributions, we targeted software systems that promote and enable communication within and across teams, projects, stakeholders and other entities that where directly or indirectly involved.

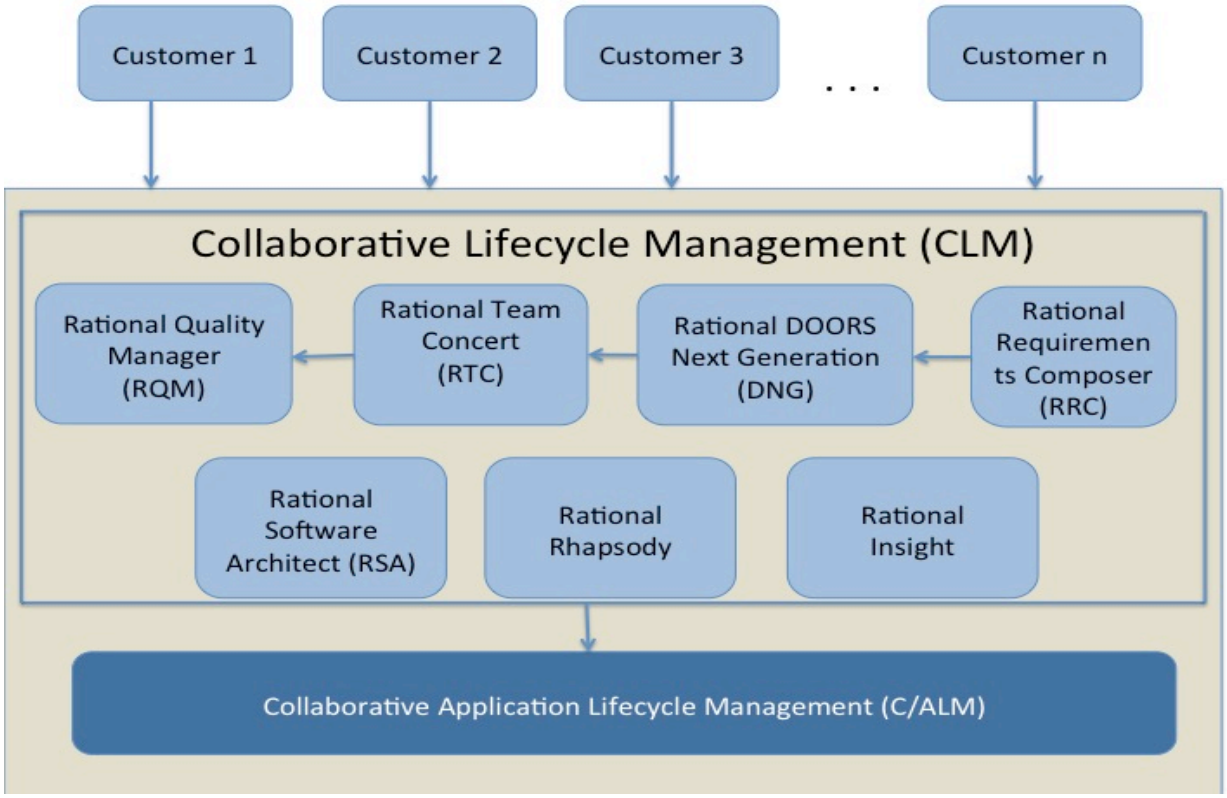
### 3.2 Study Setting

Having identified our key-deciding factor in selecting a study setting, we decided to investigate IBM's CLM ecosystem and we choose to study the Rational Team Concert (RTC) Project which is a large globally distributed software project (43). RTC is one of IBM's rational solution products situated in the midst of other co-located products in IBM's Jazz ecosystem referred to as CLM. 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. The CLM ecosystem consists of independent integrated products designed to interoperate on a

technological platform known as Jazz. IBM Rational, a brand of IBM that is responsible for the coordination of the ecosystem, also provides external integration to third party related software. The ability to integrate CLM's internal residing products to third-party products, provide customers with a flexible and customizable service, this encourages interaction within and across actors, stakeholders, customers and market masses of the ecosystem.

The CLM ecosystem consists of a number of products including 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. Rational Team Concert (RTC) products enhance collaboration across teams with integrated features in an agile development environment, this tool kit provides visibility into residing projects activities, requirement implementation process and team progress [43]. Figure 2, shows the various actors in the CLM ecosystem. It shows the relationship between the ecosystem actors, which includes its integrated products, residing developers, customers, and interacting third party software.

**Figure 3.1 IBM Jazz Ecosystem Software Product and Market Masses s**




At the time of our investigation, the Jazz CLM ecosystem had an archive of six development projects, which includes Rational Team Concert (RTC), Jazz Collaborative ALM (C/ALM), Smart-cloud Continuous Delivery (SCD), Rational AMC, Jazz Foundation and Testing Only.

Rational Team Concert project is responsible for the development of the RTC product. The RTC project is responsible for delivering software team collaborative features for which includes work items tracking, planning, builds, source control and reporting. It provides those features through a variety of clients, which includes Web UI, Visual studio Integration, windows Explorer Integration and a command Line.

The Rational AMC project is responsible for the development of 'The Eclipse Way' feature. The Eclipse way is an agile iteration-based process with a focus on consistent, on-time delivery of quality software [43]

The Jazz Collaborative ALM (C/ALM) project delivers features that integrate all the rational tools residing on the Jazz ecosystem, CLM. It provides integration between RTC, RQM, RRC, RSA and DOORS [44].

The Jazz Foundation project develops frameworks and services used as a platform for collaborative application development [44].

The CLM products are developed by a globally distributed large software team and rely on online project repositories and communication channels for all coordination, communication, collaboration and project management. The CLM has an open commercial culture where internal information about CLM products, its development processes and its communication is available to all its stakeholders including its market masses. The openness found in the CLM ecosystem allows developer-customer communication through the following means:

3.1.1 An issue tracker is publicly available to masses of CLM ecosystem. The issue trackers host many discussion forums where rich discussions about product requirements are held. This issue tracker attracts a variety of stakeholders. End-users are able to contribute during the development of a particular requirement through this discussion forum, they also can keep track of the requirement development progress and updates. This forum is also used for bug reports about a particular feature.

3.1.2 A Wiki, which contains technical documentation written by developers of the various ecosystem products. The Wiki represents a knowledge base of technical features, Developers usually write documentation in the Wiki, and also enter other forms of information for customers or other co-developers.

3.1.3 A requirement online discussion forum

The CLM projects are developed in large distributed software development teams, which rely on online project repositories for all coordination, communication and management activities. The Jazz development team uses continuous delivery methodology. They release a feature every cycle, which is eight weeks. Each cycle consist of two sprints. A sprint is usually four weeks long. Work Items are used to manage tasks and issues that need to be addressed during a software development lifecycle [48]. There are different types of work items, which are generally defined by a project standard. End user values are represented as Plan Items, Stories or bug reports.

Our study investigates the project that develops the RTC product, our dataset consist of user stories whose development iteration was between 2006 and 2012.

The rationale behind this decision is that user stories are generally known as requirements as they are feature requests that come directly from market masses.

The aim of Rational Team Concert project is to provide an open mechanism for developers, stakeholders and other actors of the ecosystem to create, track, collaborate, manage and enhance source control and report on work items. It also provides importers for other residing products within CLM.

The RTC project consists of 91 members and 8 administrators. The RTC project has two main types of teams, an umbrella and functional teams. An umbrella team is a higher hierarchy of teams, which is further decomposed into smaller functional teams. The RTC project is composed of five main umbrella teams, which include PMC, RTC maintenance, RTC development, Source control, and the Report team. The project management team oversees all current work in development and is responsible for their management. Other umbrella teams are larger unit of functional teams.

**Table 3.1 Official Members of RTC Project**

| | |
|---|---|
| Number Project Members | 99 |
| Number Of Teams | 8 |
| Number Of Team Members | 976 |

### 3.3 Data Collection Methods

The Jazz central repository is a web application server that communicates directly to a rational database backend. IBM's team provided our research data, we also gathered extra information as required from the Jazz repository through their publicly available REST APIs.

The Jazz development team keeps a Wiki. The Jazz Wiki is a work area used by development teams to plan and discuss technical designs and operational procedures related to the development projects at Jazz.net. From this Wiki, we found technical documents that provided guides and tutorials on how to navigate and query the Jazz central repository. We gathered three main sets of data from two Jazz REST APIs.

The first set of data retrieved from the reportable REST API is used to gather foundation schema information. From this API, we retrieved five main monolithic resource of interest to our research study, which includes: - project areas, project members, team areas and members, project iterations and links. We downloaded this set of data as an XML file manually through a browser. We downloaded six jazz projects, with their respective iterations project and team members.

The second and third set of data was gathered from the Change Management REST API. The change management resources provides a set of work item information, activities, tasks, release and plan of a given project delivery lifecycle. The change management resources return an XML file that includes a snapshot of RTC issue tracking system between 2006 and 2012. We downloaded communication data from each work items' forum. Also, we retrieved information about members, including their official affiliation, which we were able to retrieve a member's official geographical location. For the RTC project we retrieved 97 requirements. Also, for each work item, we gathered its history data.

The history data of a work item is a record of past events and changes that has taken place during its life span. For all the three sets of data, we manually downloaded XML files from their respective API. We wrote a script to import this data from the XML file into our own PSQL database.

Our research interest is studying the impact of emergent contributors during the development of requirements, hence we decided to investigate work items that are relatively similar in definition to a requirement. Therefore, we decided to concentrate on

a specific type of work item, which are user stories. As a user story represents a feature request directly from stakeholders, these work items represent requirements.

**Table 3.2 Case Study Data**

| | |
|---|---|
| User Requirements | 97 |
| Teams | 20 |
| Comments | 1254 |
| Discussion Participants | 91 |

### 3.4 Data Constructs

We gathered two main types of data sets. The first set of data included project and team information. Project information included project members, administrators and its iteration. Also, we had team information, which included team members and administrators.

The second set of data are artefacts and their related information, we gathered all types of work items defined earlier. For each work item we collected its properties which include its assigned team, creation date, priority, state, a list of its subscribers, a summary of work item development, its contributors (which included owner, modifier, resolver), the project it belongs to, it assigned iteration, when it was created and resolved, discussion from its online forum, the historical data throughout the development timeline.

As a user story represents a feature request directly from stakeholders, these work items represent requirements, we have a work item with the identity number 27096 and it is a user story (requirement). This work item is assigned to the “eclipse UI implementation team”, it was created on May 19th, it has a high priority, this work items

has 20 subscribers, it was created, modified and is owned by user Y. It was completed on June 1st, at 12pm. Work Item 27096 has a discussion forum, we found 40 comments from 18 participants contributing to its online discussion forum. The work item has twenty snapshots, which makes up its history data. A snapshot is a record of a work item at a particular time, when a major change is carried out on a work item, a snapshot of that work item is taken, this means the current state before and after the changes is committed to the work item. For this research we are interested in users who contributed towards the development of a work item from its historical record.

For each set of data gathered, in the table below we show an example of its fields.

**Table 3.3: Example of Work Item Properties**

| | |
|---|---|
| **Work item Id** | 27089 |
| **Assigned Team** | Eclipse UI team |
| **Assigned iteration** | RTC Development 2.1.0 |
| **Creator** | User Y |
| **Owner** | User X |
| **Work item pattern** | Back to draft |
| **Assigned Project** | Rational Team Concert |
| **Work Item Type** | Story |
| **Priority** | High |
| **Severity** | Medium |
| **Subscribers** | A list of its subscribers |

**Table 3.4: Example of Work Item Snapshot**

| | |
|---|---|
| **Work Item Id** | 27089 |
| **Work Item type** | Story |
| **Snapshot creator** | User XXX |
| **Snapshot resolver** | User YYY |
| **Snapshot owner** | User ZZZ |
| **Snapshot taken at** | May 23rd 2008 22:02 pm |

**Table 3.5: Example of Work Comments**

| | |
|---|---|
| **Work Item** | 27089 |
| **User** | YYY |
| **Email** | yyy.Affiliation.com |
| **Affiliation** | Location.ibm.com |
| **Geographical Location** | Derive from Affiliation data |
| **Comment Time** | Date, time and Time zone |
| **User Comment** | I have completed this bug, could someone here test this feature out. |

Our research was centered on a set of discussants called *Emergent Contributors.* We generally define emergent contributors as discussants that are not officially expected to contribute towards the development of a work item but participated in its online discussion forum.

*Emergent contributors are stakeholders who contribute towards the development of a work item but are not officially responsible for its development.*

Our first stage of qualitative analysis was to identify and quantify the existence of emergent users. We derived more data from calculations, which include:

- **Population of Emergent contributors**: In order to understand the population of emergent contributors, we calculated the total number of emergent users across all work items within a product i.e. RTC. For each work item, we add up its number of existing emergent users, we then calculate the total across all work items and derive it percentage. We also calculate the average number of emergent users that existed across all work items within Rational Team Concert project.
- **Timeline position** - Given the time a user participated in a work item's discussion forum, we analysed the time emergent and non-emergent users contributed to the discussion thread with respect to the work item creation date and time.

### 3.4 Data Analysis Methodology

This study employs a mixed research methodology to examine the characteristics and impact of emergent users in an open commercial software ecosystem, CLM. Our quantitative analysis includes calculating when users made a comment relative to its requirement development lifetime, population count including mean, median and mode of emergent users, statistical analysis to calculate a population distribution and its

significant difference. The qualitative analysis included thematic analysis of user discussion and manually coding users comments.

To answer our first research question *RQ1, "Do Emergent Contributions Exist?"* we investigated the characteristics of the ecosystem and seek to understand how users are made responsible for the development of a work item. Our investigation shows that a team is assigned to the development of a work item. This team are officially responsible for the development and delivery of their assigned work item. We also identified that a work item belongs to a particular software development project, users who are part of this project are indirectly responsible for the development and management of the work items. Based on this understanding, we were able to conceptualise and define who these emergent people are. For each definition of emergent users, we investigated the quantity of their contribution using statistical calculations.

In an attempt to answer our second research question *"How do emergent users contribute?"* we carried out qualitative analysis on each comment within a work item discussion forum. We had two researchers separately and collaboratively code these comments to the best of their understanding. The qualitative coding was carried out iteratively.

To answer our third research question, "*what are the impacts of emergent users participation towards the development of requirement?"* We carried out thematic analysis on these user stories. We carefully analysed each user story and tried to find an overall theme that explains the impact of existing emergent users towards the requirement under development. We tried to see if there is a general pattern of how they contribute,

we carefully studied the scenarios and how their participant affects the condition of the requirement development.

### 3.5 Conceptualization Of Emergent Contributors

In a previous work by Damian and Kwan [17], the authors suggest that hidden experts are users who emerge in a discussion thread without being part of the recipient list but are included via a different means of communication such as carbon copy (CC) or the thread was forwarded to them, hence the opportunity for them to contribute aroused. This definition suggests two main attributes that should be found in a group so as to label them as emergent users. The two main characteristics are, firstly, they must emerge and there is an identifiable means in which they contribute towards the development of requirement.

For a set of users to be grouped as emergent contributors, they must emerge from an unexpected source, this means they must not be officially assigned or expected to contribute towards the development of a requirement. Without been responsible for the development of a requirement but appears to be participating to the discussion of the requirement is the first criteria to be labeled an emergent contributor.

The second characteristics is the means in which emergent user participate. If users, developers, managers, stakeholders are not responsible for development of a requirement, hence they also by default do not have the means to contribute to the development. For an emergent user to appear and contribute, there should be a means in which they can achieve this goal.

Based on our defined attributes of emergent contributors earlier, we identified two main types of emergent contributors. ***Emergent by Product*** and ***Emergent by Team.***

### 3.5.1 Emergent by Product:

Every existing project within CLM ecosystem as a group of defined members and administrators who are part of the software's project development. This group of users are recognized to be officially affiliated to its requirement. Similar to team affiliation, we found that a project as two types of affiliation which includes members and administrators. Project administrators are members of project management committee who are responsible for the management, maintenance and overall co-ordination of the project. Project members are users that include developers, testers, stakeholders and other residing actors that are responsible for the project development.

A user who contributed to a requirement during it development by participating in its discussion but is not affiliated to the requirement through project membership is referred to as emergent by product. By definition a user who is emergent by team is also expected to be emergent by product. If a user is emergent by product, they are considered to be outside of a products organisation. This umbrella of users usually consists of members of other products, actors and stakeholders within CLM ecosystem.

### 3.5.2 Emergent by Team:

Every requirement is assigned to a team. This assignment is a way of definition a group of users who are officially responsible for the development of a requirement throughout its lifecycle. A team is composed of two types of users, which include team members and team administrators. The official role of team administrator is to manage, maintain and co-ordinate the requirement throughout it's development cycle whilst team members are usually developers, testers or maintainers, they are officially responsible for the implementation of the requirements.

For each requirement, we identified its assigned team members and administrators. We define contributors who are outside a requirement team as our first set of emergent users referred to as emergent by team.

### 3.6 Qualitative Analysis - Categorising Discussion

To understand how a participant contributed towards the development of a requirement during its lifetime we carried out two types of qualitative analysis, which are qualitative coding by employing the rules of grounded theory and thematic analysis. For the first qualitative analysis, we wanted to understand in details how emergent users contributed. We carried out a qualitative analysis by abiding to the rules the grounded theory coding. This analysis involved two researchers who studied the data individually and collaboratively. There were 1254 comments across 97 work items. The coding process was iterative and each iteration includes two main phases. The first phase, researchers would individually code the comments. The first iteration was carried out on the first 100 comments, and then researchers came together to agree or disagree so as to achieve a baseline code to continue with. This style of qualitative coding iteration continued until all comment was coded and agreed upon. We also kept record of agreement and disagreement ratio, using Kripendorfs alpha measure to obtain an inter-code reliability. The purpose of this further stage of qualitative analysis is to carry out more informative investigation about the contribution of emergent users towards development.

The next stage of thematic analysis was building on the already existing qualitative coding. Based on how we have categorised requirement, we search for common trend of emergent participation, when we found few trend we further analysed this trend in more details i.e. analysed storied individually based on the general trend.

## 3.7 Quantitative Analysis

### 3.7.1 Population of Contributors

Before investigating emergent users contribution and impact, we counted the total number of participant who contributed towards requirements. We also explored other population count, which includes mean, median and mode. We then investigated the population of the two defined emergent user and non-emergent user (users who participated and are not emergent). Our goal with the population analysis it to have an understanding of emergent users population over a given period of time in respect to the overall population of discussant.

### 3.7.2 Time of Contribution

Our previous research methodology aim to understand how emergent users contributed and when during the discussion forum do they participate. Our next step was to examine if these emergent contributions had an identifiable behavioural pattern. To achieve this, we grouped those who participated in two groups, emergent users and non-emergent users. For each users in both groups, we identified how long since the work item was created they contributed. We then conduced a man Whitney test of distribution to check if their was a statistical difference in the time these two group contributed.

### 3.7.3 Characterizing Contributors

To conclude and enrich our findings, we carried out a thematic analysis on the discussion forums of the 97 requirements we analysed. Given our two groups of users, emergent and non-emergent. We manually looked to study if emergent contributors had a pattern of communication in general.

# Chapter 4 - Results

In this chapter, we present the result from the data analyzed based on our research questions. Our first research question is, do emergent users exist within an online software ecosystem? Earlier, we defined a software ecosystem as an environment that enhances collaboration within and across residing software systems. We also identified the dynamic nature of members within projects and teams. This characteristic of an ecosystem makes the IBM CLM ecosystem a suitable domain to investigate the existence of emergent contributors.

To better guide our first research question that aims to quantify the presence of emergent contributors, we conceptualized emergent contributors based on two types of official affiliation to a requirement. As explained earlier in the methodology section, we identified two types of emergent users, which are ***Emergent by Project*** and ***Emergent by Team.***

## 4.1 Population of Emergent Contributors

***Emergent by Project:***

Of the discussions of 97 user stories that were investigated, we found that discussants categorised as emergent by project existed in 56 of these forums, this means emergent participants existed in 58% of the requirements discussion forum we analysed. Of a total of 431 comments on the 97 user requirements, emergent participants made 54% of comments. On an average, we found, four emergent parcipipants across all requirements.

#### 4.1.2 RQ1. Do emergent contributors exist within a software ecosystem?

Across the 97 requirements we analysed, we found evidence of emergent contributions in 60 requirements, which accounts for more than half of our requirements. We further identified a total of 54% emergent contributions.

**Table 4.1: Population Count of Emergent Contributors**

| | Number Of Requirement | Number Of Contributions |
|---|---|---|
| Emergent by Project | 58% | 54% |

### 4.2 Characterising Emergent Contributions

After understanding the quantity of emergent contributions during requirement development, we wanted to understand their characteristics. We define emergent participants characteristic based on our perception of how they contributed to a requirement discussion forum. As explained earlier, we carried out a manual qualitative analysis on requirement online communication. Based on our qualitative analysis, we identified four main categories of online communication, which includes Requirement Negotiation, Co-ordination, Information and implementation Status.

- **Requirement Negotiation** – This category of our manual codes revolves around requirement discussion, which includes clarification, verification, negotiation and agreement. Before a requirement can be considered for development it needs to be efficiently clear and in line with strategic goals. The need for unambiguous requirements prior to implementation drives online discussion and can result to

ongoing discussion trend with variety of contributors. During our manual coding analysis, we identified that discussants were clarifying, verifying, negotiating and agreeing upon detailed unambiguous definitions and expectations of a given requirement. Clarification and verification usually happen at the early stages of the requirement development lifecycle. In most cases we have stakeholders, customers, managers discussing with the technical team. Sometimes later in a requirement development lifecycle, developers who seek more verification or clarification may trigger requirement negotiation. We also found evidence of new requirements coming up during discussion, this was due to stakeholders and market masses involvement.

- **Coordination** – In most cases requirement co-ordination occurred after requirement clarification but this is not a general rule. We found participants co-ordinating shared resources, skills, task completion, iteration planning. We also grouped as coordination comments cases where we found participants allocating time to a task, scheduling requirement tests, delivery and enhancement plans. Sometimes developers acknowledge the completion of a task and schedule it for a testing or delivery.
- **Information –** During a work item development discussion, users are tend to ask questions requesting for more knowledge about a given subject or task. We categorised comments where users were seeking help and received required assistance as information. Information is different from requirement discussion as information is broader which could include task information requests, technical knowledge, documentation information sharing. Information includes information

status, processing technical request and help, bug reports and responses to bug reports.

- **Implementation Status –** During the discussion of a requirement under development, participants usually update one another about the progress, blockage and current status of the development work. Sometimes the discussion could also be comments on code reviews, information about the development process in general and details.

**Table 4.2 Category of online discussion**

| **Category** | **Subcategory** | **Examples** |
|---|---|---|
| **Requirement Negotiation** | a. Requirement agreement: Participants agreeing to the stated requirement description<br>b. Requirement clarification: This is where discussant try to provide more informative less ambiguous description to a requirement. | A large, UK-based retailer has a heterogeneous environment using Java and .NET. They require a .NET version for take-up of RTC. |

| | | |
|---|---|---|
| | c. Requirement disagreement: This notate a conflict in agreeing upon a detailed definition for a requirement<br>d. New requirement – Sometimes during discussions, stakeholders and other groups do come up with requirements that are not originally planned for but happened spontaneously | |
| **Coordination** | a. Iteration planning - This is when discussants are coordinating<br>b. Process – this is when discussants are | Components X, please add your child work items. Sign-off when work is completed. |

| | | |
|---|---|---|
| | coordinating the development process. | |
| **Information** | Bug report – This is when someone informs the team of an existing bug during development.<br>Bug response / reply – This is when someone responds to the bug in question.<br>Technical Knowledge – We found developers asked and answered technical question that came up during implementation | Bug 103822 comment 23 has a client request for load rules. |
| Implementation Status | This is when developers provide updates about the requirement under development. | Karl, The sdwb folks have started an Open Source project inside IBM; It's currently in early design phase. |

To understand emergent contributions we studied the behaviour of emergent contributors based on our manual coding explained above. Due to the very small quantity

of contributions in the sub categories we decided to only focus on the main categories.

We had 112 comments categorised as requirement negotiation. Of these 112 comments 88 were from emergent participants, hence 79% of requirement negotiation comments were from emergent users whilst the remaining 21% came from non-emergent participants. This indicates to us that emergent users play a major role in requirement discussion and developing new requirements.

We found 111 comments categorised as coordination from participants, of this comments 48 comments came from emergent participants, meaning approximately 43% of the total coordination contribution was by emergent user, whilst 63% was by non-emergent.

Also, out of 65 comments from participants coded as information, we found 48% of these comments where from emergent participants whilst 52% was from non-emergent participants. For comments categorised as implementation we found that 20% came from emergent users whilst 80% came from non-emergent users.

Again, we found slightly strong evidence that suggests that these emergent participants are not majorly contributing towards the implementation or coordination of a requirement during its lifetime.

**Table 4.3 Contributions from Emergent participants**

| | Number of Comments | Emergent | Non-Emergent |
|---|---|---|---|
| Requirement Negotiation | 112 | 88 (79%) | 24 (21%) |
| Coordination | 111 | 48 (43%) | 63 (57%) |
| Information | 65 | 31 (48%) | 34 (52%) |
| Implementation Status | 51 | 10 (20%) | 41 (80%) |

### 4.2.1 RQ2. How do emergent participants contribute towards the development of requirements?

We found that emergent stakeholders play a large role in requirement negotiation and little role in implementation. In comparison to non-emergent users we find the exact opposite where non-emergent participants contribute more to implementation than requirement negotiation. Based on this finding, we decided to investigate if the behaviour is significantly different.

**Table 4.4** Numbers. Of discussant contribution**s**

| Contribution Type | Emergent | Non Emergent |
|---|---|---|
| Requirement Negotiation | 88 | 24 |
| Implementation | 10 | 41 |

**Table 4.5 Chi-Square Result**

| | Requirement Negotiation |
|---|---|
| Implementation | Chi Square = 48.387 P-value = 0.0001 |

Table 4.5 shows the results from our chi square analysis, which includes P value and chi square number. The chi square test is used to test if two groups of population ares significantly independent of one another. In this test, we are seeking to find if the characteristics that emergent users contribution more to requirement negotiation and non-emergent users contribute more toward development is significant within the population. From the result of our P-values, we found that statistically, emergent users significantly contribute towards requirement negotiation and that non-emergent users significantly contribute towards implementation of requirement.

### 4.3 Participation Period Of Emergent Contributors

Now that we know emergent users contribute during requirement discussion forums we decided to study when they contribute. Our findings show that these emergent participants play a major role in requirement discussion, we seek to identify at what point during the development of a requirement do they participate. Since they bring a major contribution to shaping requirements, we believe it is important that they contribute and are involved in requirement discussion early. We thus, analyzed the timing of their contribution. We carried out a Mann-Whitney test of difference in distribution. Taking into account the number of days emergent users made a comment after the requirement was created, we found based on Mann-Whitney test that emergent contributors comment

later than earlier during discussion. Table 4.5 shows the figure and result from our calculations

**Table 4.5 Mann Whitney Result: Emergent vs non-Emergent Contribution**

| | Emergent | Non-Emergent | Mann-Whitney Test |
|---|---|---|---|
| All contributions | 29 | 21 | W = 248645, p-value – 0.02914 |
| First Contribution | 28 | 14 | W = 8319, p-value = 0.001712 |

#### 4.3.1 RQ3 - What is the impact of emergent participation towards the development of a requirement?

So far from our results, we have identified that emergent participants play a major role in requirement discussion but yet they contribute later than earlier. To further solidify our findings and conclusions, we seek to identify the context surrounding emergent contributions. Hence, we carried out thematic analysis of a set or requirements, we analysed the context of the requirement, how emergent users participated and the effect of this emergent participation with reference to the context of each individual requirements.

We found two main types of themes. The first theme suggests that early emergent contribution drives requirement clarification whilst late contributions have a variety of effects one of which is wasted effort.

**Early contributions by emergent stakeholders drive the requirement**

We observed in some cases where emergent participants commented early in the work item timeline describing their need for a particular feature implementation or defect fix. The emergent participants, therefore act as catalysts for these changes. From the discussions it appears that emergent stakeholders contribution speed up requirement decision-making and clarification. We also found that when these emergent users participate in requirement discussion, non-emergent users who are responsible for their implementation respond quickly and deliver a desirable requirement.

To further illustrate our example, we selected two requirements discussion forums, shown in Table 4.6 and 4.7. Table 11, shows a scenario where an emergent contributor speed up the requirement clarification process. Table 4.8 displays an occasion where late emergent contribution resulted in disruption.

**Table 4.6 Emergent users act as a catalyst to requirement clarification process.**

| User | Stakeholder Type | Date | Comment |
|---|---|---|---|
| 2 | Emergent | 4 Oct 07 | Don't we need the ability to create x feature, with b and c functionalities as well? |
| 3 | Emergent | 4 Oct 07 | I agree and we really need to define the scope here with a scenario. I've added a P item to the M plan. |
| 2 | Emergent | 24 Jan 08 | Without x functionality, you can't complete the y task, I think we need to include that. If I think about how our IT |

| | | | |
|---|---|---|---|
| | | | folks work, I would really like to see the Web UI cover the “Add a new function k” scenario. Essentially creating a contributor, likely by import from an existing user in the corporate directory, add them to a team, and assign them a role in the team. I don't want to send those guys out to Eclipse to perform those steps. I agree with keeping any process editing in the Web UI out of scope for 1.0, though. |
| 5 | Non-emergent | 30 Jan 08 | We will introduce the notion of KXY feature The web ui should allow to carry out Y function. Team membership and role assignments (which require intimate knowledge about the process, otherwise you don't know which role to assign) should be up to the process area administrators. |
| 6 | Non-emergent | 22 Feb 08 | Primary functionality added in M5... will continue to polish in M6. |
| | | | |

**Table 4.7 Emergent users act as a catalyst to requirement clarification process.**

| User | Stakeholder Type | Date | Comment |
|---|---|---|---|
| 1 | Non-emergent | 31 Mar 08 | Changed the Filed Against field to point to the appropriate place.<br/><br/>Yu Wang - I'm currently working on a CMVC Connector for Jazz. My team is working towards a Beta release on May 2nd |
| 2 | Emergent | 15 Aug 08 | I'd be very interested in this functionality. I am investigating using RTC for my Team (and my Area or Development, Test, and Build). Is this available in RTC 1.0? |
| 3 | Emergent | 1 Oct 08 | Do you need testers for this? We have an immediate need to be able to grab CMVC defects into RTC (only a one-way export/import). |
| 4 | Emergent | 11 Nov 08 | So is this still happening or was it taken off of the board? |
| 3 | Emergent | 12 Nov 08 | The Team X folks have started an Open Source project inside IBM. It's currently in early design phase. |

| | | | |
|---|---|---|---|
| 5 | Emergent | 28 Nov 08 | If this feature is ready, will RQM benefit from this too? Then people using RQM can access defects from CMVC. Really looking forward to it. |
| 6 | Non-emergent | 08 Dec 08 | Dear All, Iteration 1 for CMVC Connector has been released. |
| 6 | Non-emergent | 10 Feb 09 | Dear All, Iteration 2 for CMVC Connector has been released. |

**Late contributions by emergent stakeholders cause disruption and rework in the requirement process.**

On the other hand, we observed cases where emergent stakeholder's contributions come much later in the work item's time line. In many cases, the late involvement of an external stakeholder caused problems. in the work item's time line. In many cases, the late involvement of an external

For example, Table 4.8, we noticed emergent contributions coming much later after a lot of work as been carried out, these late emergent resulted in wasted effort of earlier work and a rework was needed.

In general we noticed that the absence of emergent participants driving requirement discussion leads to unclear implementation of requirements.

**Table 4.8 Absence of emergent user affect requirement development**

| User | Stakeholder Type | Date | Comment |
|---|---|---|---|
| 1 | Emergent user | 3 Oct 08 | One comment on the following sup-topic:<br/>https://jazz.net/wiki/bin/view/Main/ExceptionDesign <br/><br/>This may be obvious to many, but we need a rule that code must never throw a new exception, checked or unchecked, without a message.  This practice is commonly found in unchecked exceptions, like the conditions described in the section<br/>"How the Client API uses unchecked exceptions ".<br/><br/>For example:<br/><br/>if (anArg == null)<br/>	throw new IllegalArgumentException(); //Not helpful, might as well be an NPE.<br/><br/>better:<br/>if (anArg == null)<br/>	throw new IllegalArgumentException("anArg must not be null");  //$NON-NLS-1$  <br/>	<br/>Several times we've been presented with exceptions where there was no associated message which made it difficult for consumers to solve their own problem, and difficult for developers to match up the error with the problem.  |

| | | | |
|---|---|---|---|
| | | | The line numbers often don't line up between the stack trace and the current version of the code, so any message is helpful to the developer.<br/><br/>Another instance is something like this:<br/>if (someCondition)<br/>  //Developer note: should never get here<br/>	throw new IllegalStateException();<br/><br/>Code like the above should have a message. <br/>Just my 5 US cents (inflation adjusted) |
| 2 | Non-emergent | 13 Oct 08 | I've added some screenshots with examples of non-supportive error messages |
| 3 | Non-emergent | 22 Oct 08 | A first draft of the wiki page is available |
| 4 | Emergent | 23 Oct 08 | Looks good :) Couple points I did not see in the wiki<br>1) We have some 'best practices' that came out of a wourkgroup. Let me know if this helps or not: [link]<br>2) I understand and agree we should not 'pollute'& the log with expected errors, especially if we handle them. Could we still 'trace them' in case we need to debug?<br>3) What about the usage or non usage of a subclass of IStatus ? StatusManager ? Do we recommend anything ?<br>4) What are the API each component should provide ? Create Status ? Create Exception ? Log Status and Log String? Show |

| | | | |
|---|---|---|---|
| | | | Status (calling StatusHandler) ? Trace (String ) ?<br>5) What about the best practices about what an error message should look like? What about providing guidance when one review a message so it will be easier for us to externalize when times come?<br>6) What about having an internal feedback mechanism, just for the Jazz dev team so we know what messages we need to focus on first? |
| 3 | Non-emergent | 26 Oct 08 | The draft has been updated. |
| 5 | Emergent | 4 Nov 08 | We currently don't have a story for expressing process problems to REST clients, such as our web UIs. Do we address this gap here or somewhere else?<br/><br/>Also, was it our intent for this work item to also improve how web UIs display error messages or is this work item focused on only the Eclipse-based UIs? |
| 3 | Non-emergent | 4 Nov 08 | So far we only looked at the eclipse UI and I agree that we need to extend the scope to include the Web UI. |

## Chapter 5 – Discussion and Implications of Research Result

In this chapter, we discuss the result of our case study on IBM's Jazz Ecosystem. Our goal was to investigate 1) The presence of emergent contributors 2) Characteristics of emergent contributions 3) Impact of emergent contributors to the requirement development process. We also discuss the impact of our findings to practitioners and researchers.

### 5.1 Emergent Contributors Significantly Exist

Similar to a previous study [17], we found that emergent contributors exist. We know from current research that IBM Jazz ecosystem incorporates openness as a culture [21]. Hence, we expected to find evidence of emergent contributors. But most interestingly is the quantity of emergent participation. Our analysis reports that approximately half of the discussant population - 54% - are emergent contributors. Our findings also show that, across 97 requirements, 58% of them had emergent contributors.

Also, our statistics show that, across 97 requirements, we found emergent contributors in 60 requirements. These sets of result further prove that emergent contributors play a major role to a large quantity of requirement within the IBM Jazz ecosystem.

Our findings here serves as a good indication that we need to further investigate the characteristics and impact of emergent contributors.

### 5.2 Emergent Contributors Participate Mostly in Requirement Negotiation

In our study of emergent contributors, we applied manual coding so as to characterize contributions from emergent participants.

Knauss et al suggest that the open paradigm found in IBMs' software ecosystem yields the involvement of market masses during requirements discussions [21]. We report from our analysis of 97 requirement, 79% of emergent communication accounts for requirement clarification and negotiation. This means majority of emergent communication was towards further clarification and explanation of the requirement in question.

Our result reflects that emergent contributors participate more in requirements negotiation, this implies that the majority of emergent stakeholders play a major role in requirements negotiation. From this investigation we have evidence to show that emergent contributors play a significant role in requirements discussion, but to what extent do they impact requirements?

**5.3 Emergent Contributors Participate Late**

Our above result identified the role and character of emergent contributors. We will now proceed to understand their impact. We first noticed based on our Mann Whitney statistical result, the majority of emergent contributors appeared later than earlier. Given the fact that emergent users play a major role by contributing to requirement discussion, which normally start during the early stages of development, we suggest that emergent users are identified and involved earlier during the development lifecycle.

To support our research contribution, which suggest that emergent user should be identified and involved earlier, we provide and compare scenarios where these emergent contributors appeared early and late.

When emergent contributors appeared early, we notice they drive requirement discussion. They act as catalyst to the requirement discussion process, making the requirement clarification and decision making process faster and efficient. On the other hand, we found scenarios where late participation of emergent contributors caused problems during development. Hence, without emergent contribution at the early stages to drive requirement discussion, requirement clarification and decision-making are prone to face challenges leading to potential unsuccessful outcomes.

## 5.4 Implications For Practitioners

The core advantage of software ecosystems to an organization is the ability for diverse stakeholders to openly interact. This has enhanced communication and coordination, which indirectly sped up the development process. Our research brings more evidence to this existing benefit. From our investigation, we have two main suggestions.

**5.4.1 Identify Emergent Stakeholders For Valuable Contribution** – In chapter 4, we define emergent contributors as participants who contributed towards the development of a requirement despite not officially being required to. We further found evidence that suggests that users play a major role within the ecosystem based on their quantity of participation. We therefore, advise practitioners to take better advantage of the presence of emergent contributors within the software ecosystem. They should automatically identify emergent contributors for each requirement under development.

**5.4.2 Involve Emergent Stakeholders Early** – As mentioned earlier, we noticed emergent stakeholders appear later than earlier during requirement discussion.

Emergent stakeholders drive requirement discussion, hence they should not only be automatically identified but also involved in the early stage of online requirement discussion.

### 5.5 Implications For Researchers

Current literature concentrates on identifying dependencies within tasks and across projects during the implementation stage i.e. code changes, reviews, and commits. Online requirement discussion usually takes place before implementation of requirements begins. Our findings in this study suggest a new means of identifying dependencies during online requirement discussion.

Also, this research provides new evidence to support the existing body of research. Firstly, we confirm that emergent users do exist within large globally distributed software projects. In addition we provide evidence of their characteristics and impact to requirement development. We also- shed more light on software ecosystem research. We provide evidence that the openness culture found within the software ecosystem is extremely valuable. This culture has enhanced the participation of impactful stakeholders (emergent contribution).

### 5.6 Threats to validity

Despite our effort to gather data and conduct a quality analysis our research suffers from several threats to validity. In this section we describe the threats to validity faced by our research to allow for an informed decision about the application of the result and the study itself.

The first threat to validity is ignoring the changes that occurred in team and project membership composition. Team and project membership composition are

dynamic and they change overtime as people move across team and project. The requirements dataset we used for our analysis was developed between 2008 to 2010 but the team and project composition we gathered was between 2011 to 2012. We also noted that it is rare for project membership to change over a period of a year unlike team membership, which changes frequently over time. Hence, for this analysis we decided to concentrate on the less dynamic membership: project affiliation. Although project members composition is less likely to change over a time gap of two years we still cannot guarantee that the membership is completely accurate.

This analysis only concentrates on one type of contribution, which is the online discussion forum for requirements. There are other online communications such as email, comments during snapshot and reviews. Ignoring other means of communication does not dismiss the authenticity of our research, but it does raise a question if this result is applicable to other means of online contribution.

## Chapter 6 - Conclusion

In this thesis we explored the concept of expert seeking within software ecosystems and investigated their impact during the development of requirements. We described our case study using the Rational Team Concert product residing on IBM's Jazz ecosystem. The RTC product is responsible for the delivery of teams collaborative feature within the ecosystem, the RTC product as been under development for a number of years, hence people have moved, and knowledge has passed across the product. Being an older ,large software product, developers face the challenge of finding the right knowledge source to direct queries. Research has proven that awareness is a challenge within large globally distributed software products. Given the nature of software ecosystems, our thesis goal was to find out if emergent stakeholders exist and there impact during the development of requirements.

To address our research questions, we carried out mixed method research analysis on empirical data. We analysed online discussion forums for a given set of requirements. We conducted a population count to find out the quantity and significance of emergent users, manual coding to understand the character of emergent contributions and statistical analysis to investigate emergent contributor's impact during requirement lifecycle.

### 6.1 Contributions

When we started this study, the overall aim was to find a solution to awareness within global large software systems. Our research has made two major contributions. First, it shed more light on current research statement. According to Damian et al, developers within a open source software project are constantly seeking knowledge from experts, our finding in this thesis validates this statement.

Also, Knauss et al, identified openness as a core advantage within software ecosystems. We verify that this statement is true. It improved communication and co-ordination across a variety of large software products residing in the ecosystem.

Secondly, we have suggested a new way of identifying dependencies within and across projects before development gets to the implementation stages. By identifying experts from other sources, we could potentially find dependencies between those tasks.

### 6.2 Future Work

There are a few results claimed here that should be investigated further for validation. Our research suggests that emergent contributors are resourceful and they should be identified and involved early during requirement development. We have only explored the existence and importance of emergent users, however we did not investigate where they could have emerged. The first logical step to further this research is to investigate where these valuable contributors are emerging. Could emergent users exist because of technical dependencies between requirements or across project?

The main purpose of identifying the source of emergent contributors is to design an algorithm that can be used to automatically identify potential emergent contributors. Once these emergent contributors can be automatically identified at the early stages of requirement development, we can involve them to derive more valuable requirement discussion.

Also, knowing that we can identify emergent users before development process gets to the implementation stage, we potentially have a new means of identifying social dependencies across task and project. This will enhance software communication and coordination process.

## Bibliography

[1] *MySQL: The world's most popular open source database.* 1995-2008.

[2] Bass, M. J.D. Herbsleb, and C. Lescher. Collaboration in Global Software Project at Siemens: An Experience Report in *Global Software Engineering, 2007. ICGSE 2007.*

[3] R. E. Kraut and L. A. Streeter, "Coordination in software development," Communications of the ACM, vol. 38, no. 3, pp. 69–81, 1995.

[4] Frost, R., Jazz and the Eclipse Way of Collaboration. Software, IEEE, 2007. 24(6): P. 100 – 250.

[5] M. Cataldo and J. D. Herbsleb, "Coordination breakdowns and their Impact on development productivity and software failures," IEEE Trans. Software Engineering (TSE), vol. 39, no. 3, pp. 343–360, 2013.

[6] D. Damian, L. Izquierdo, J. Singer, and I. Kwan, "Awareness in the wild: Why communication breakdowns occur," in Proc. Intl Conf. Global Software Engineering (ICGSE). IEEE, 2007, pp. 81–90.

[7] C. R. de Souza and D. F. Redmiles, "An empirical study of software developers' management of dependencies and changes," in Proc. Int'l Conf. Software Engineering (ICSE). ACM, 2008, pp. 241–250.

[8] C. R. de Souza, S. Quirk, E. Trainer, and D. F. Redmiles, "Supporting collaborative software development through the visualization of sociotechnical dependencies," in Proc. Int'l ACM Conf. Supporting Group Work. ACM, 2007, pp. 147–156.

[9] C. R. De Souza and D. F. Redmiles, “The awareness network, to whom should i display my actions? and, whose actions should i monitor?” IEEE Trans. Software Engineering (TSE), vol. 37, no. 3, pp. 325–340, 2011.

[10] K. Blincoe, G. Valetto, and D. Damian, “Do all task dependencies require coordination? the role of task properties in identifying critical coordination needs in software projects,” in Proceedings of the 9th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering. ACM, 2013, pp. 213–223.

[11] K. Blincoe, G. Valetto, and S. Goggins, “Proximity: a measure to quantify the need for developers’ coordination,” in Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work. ACM, 2012, pp. 1351–1360.

[12] A. Borici, K. Blincoe, A. Schr¨oter, G. Valetto, and D. Damian, “Proxiscientia: Toward real-time visualization of task and developer dependencies in collaborating software development teams,” in Proceedings of the 5th International Workshop on Co-operative and Human Aspects of Software Engineering. IEEE Press, 2012, pp. 5–11.

[13] P. Dewan and R. Hegde, “Semi-synchronous conflict detection and resolution in asynchronous software development,” in ECSCW 2007. Springer, 2007, pp. 159–178.

[14] S. Minto and G. C. Murphy, “Recommending emergent teams,” in

Mining Software Repositories, 2007. ICSE Workshops MSR'07. Fourth International Workshop on. IEEE, 2007, pp. 5–5.

[15] A. Sarma, L. Maccherone, P. Wagstrom, and J. Herbsleb, "Tesseract: Interactive visual exploration of socio-technical relationships in software development," in Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on. IEEE, 2009, pp. 23–33.

[16] A. Sarma, D. F. Redmiles, and A. Van Der Hoek, "Palantir: Early detection of development conflicts arising from parallel code changes," Software Engineering, IEEE Transactions on, vol. 38, no. 4, pp. 889–908, 2012.

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

[18] D. Damian, R. Helms, I. Kwan, S. Marczak, and B. Koelewijn, "The role of domain knowledge and cross-functional communication in sociotechnical coordination," in Int'l Conf. Software Engineering (ICSE). IEEE, 2013, pp. 442–451.

[19] M. F. Lungu, "Reverse engineering software ecosystems," Ph.D. dissertation, University of Lugano, 2009.

[20] E. Knauss, D. Damian, A. Knauss, and A. Borici, "Openness and requirements: Opportunities and tradeoffs in software ecosystems," in Proc. Intl Conf. Requirements Engineering (RE). IEEE, 2014, pp. 213–222.

[21] S. Jansen, A. Finkelstein, and S. Brinkkemper, "A sense of community: A research agenda for software ecosystems," in Proc. of Int'l Conf. on Softw. Eng., 2009, NIER Track.

[22] A. Boden and G. Avram, "Bridging knowledge distribution-the role of knowledge brokers in distributed software development teams," in Cooperative and Human Aspects on Software Engineering, 2009. CHASE'09. ICSE Workshop on. IEEE, 2009, pp. 8–11.

[23] D. Damian, I. Kwan, and S. Marczak, "Requirements-driven collaboration: Leveraging the invisible relationships between requirements and people," in Collaborative software engineering. Springer, 2010, pp. 57–76.

[24] S. Marczak, I. Kwan, and D. Damian, "Investigating collaboration driven by requirements in cross-functional software teams," in Requirements: Communication, Understanding and Softskills, 2009 Collaboration and Intercultural Issues on. IEEE, 2009, pp. 15–22.

[25] K. Manikas and K. M. Hansen, "Software ecosystems: A systematic literature review," Systems and Software, vol. 86, pp. 1294–1306, 2013.

[26] C. Jergensen, A. Sarma, and P. Wagstrom, "The onion patch: migration in open source ecosystems," in Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering. New York, NY, USA: ACM, 2011, p. 7080.

[27] T. Kilamo, I. Hammouda, T. Mikkonen, and T. Aaltonen, "From proprietary to open source-growing an open source ecosystem," Journal of

Systems and Software, vol. 85, p. 14671478, 2012.

[28] J. Kabbedijk and S. Jansen, “Steering insight: An exploration of the ruby soft- ware ecosystem,” in Software Business, ser. Lecture Notes in Business Information Processing, B. Regnell, I. Weerd, O. Troyer, W. Aalst, J. Mylopoulos, M. Rosemann, M. Shaw, and C. Szyperski, Eds., vol. 80. Springer, Berlin/Heidelberg, 2011, pp. 44–55. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-21544-5 5

[29] S. Fricker, “Requirements Value Chains: Stakeholder Management and Requirements Engineering in Software Ecosystems,” in Proc. of Requir. Eng.: Foundation for Softw. Quality, Essen, Germany, 2010, pp. 60–66.

[30] R. Frost, “Jazz and the eclipse way of collaboration,” IEEE Software, vol. 24, no. 06, pp. 114–117, 2007.

[31] L. Cao and B. Ramesh, “Agile requirements engineering practices: An empirical study,” Software, IEEE, vol. 25, no. 1, pp. 60 –67, jan.-feb. 2008.

[32] E. Knauss and D. Damian, “V: issue: lizer: exploring requirements clarification in online communication over time,” in Proc. Intl Conf. Software Engineering (ICSE). IEEE Press, 2013, pp. 1327–1330.

[33] E. Knauss, D. Damian, G. Poo-Caamano, and J. Cleland-Huang, “Detecting and classifying patterns of requirements clarifications,” in Proc. Int’l Conf. Requirements Engineering (RE). IEEE, 2012, pp. 251–260.

[34] E. Knauss, D. Damian, J. Cleland-Huang, and R. Helms, “Patterns of continuous requirements clarification,” Requirements Engineering

Journal (REEN), pp. 1–21, 2014.

[35] Eclipse Foundation, Eclipse. 2009.

[36] J. Bosch, “From Software Product Lines to Software Ecosystems,” in Proc. of Int’l Conf. on Softw. Product Lines, 2009.

[37] H. van der Schuur, S. Jansen, and S. Brinkkemper, “The power of Propagation: on the role of software operation knowledge within software Ecosystems,” in Proceedings of the International Conference on Management of Emergent Digital EcoSystems. New York, NY, USA: ACM, 2011, p. 7684.

[38] D. Tamburri, P. Lago, and H. van Vliet, “Uncovering latent social communities in software development,” IEEE Software, vol. 30, no. 1, pp. 29–36, 2013.

[39] D. Damian, R. Helms, I. Kwan, S. Marczak, and B. Koelewijn, “The role of domain knowledge and cross-functional communication in sociotechnical coordination,” in ICSE 2013, 2013, pp. 442–451.

[40] C. Castro-Herrera, J. Cleland-Huang, and B. Mobasher, “Enhancing Stakeholder profiles to improve recommendations in online requirements elicitation,” in Requirements Engineering Conference, 2009. RE’09. 17th IEEE International. IEEE, 2009, pp. 37–46.

[41] G. Walsham, “Interpretive case studies in is research: nature and method,” Eur. J. Inf. Syst., vol. 4, pp. 74–81, 1995.

[42] Herbsleb, J.D and D. Moitra, Guest Editors’ Introduction: Global Software Development, in *IEEE* Software. 2001. P. 16 – 20.

[43] IBM Inc. Jazz Community Site. 2006 [cited 2015 May 30th] : Available from: http://jazz.net

[44] IBM Inc. Jazz Team Wiki. 2015[cited 2015 January – May]; Available from http://jazz.net

[45] A. French and P. Layzell, "A study of communication and cooperation in distributed software project teams," in Proc. Intl Conf. Software Maintenance (ICSM). IEEE, 1998, pp. 146–154.

[46] A. Begel, Y. P. Khoo, and T. Zimmermann, "Codebook: discovering and Exploiting relationships in software repositories," in Software Engineering, 2010 ACM/IEEE 32nd International Conference on, vol. 1. IEEE, 2010, pp. 125–134.

[47] J. Herbsleb and A. Mockus. An empirical study of speed and communication in globally distributed software development. Software Engineering, IEEE Transactions on, 29(6): 481-494, 2003.

[48] Kathy Charmaz, Constructing Grounded Theory, A practical guide through qualitative analysis. SAGE Publications, 2006.

[49] IBM Rational Help, 2015 [cited 2015,January – May], https://jazz.net/help-dev/clm/index.jsp?re=1&topic=/com.ibm.team.workitem.doc/topics/t_defining_types.html&scope=null