Sharing awareness during distributed collaborative software development

Inah Omoronyia

Department of Computer and Information Sciences
University of Strathclyde
PhD
November 2008
'The copyright of this thesis belongs to the author under the terms of the United Kingdom Copyright Acts as qualified by University of Strathclyde Regulation 3.49. Due acknowledgement must always be made of the use of any material contained in, or derived from, this thesis.'

@copyright 2008
Abstract

Software development is a global activity unconstrained by the bounds of time and space. A major effect of this increasing scale and distribution is that the shared understanding that developers previously acquired by formal and informal face-to-face meetings is difficult to obtain. This thesis proposes and evaluates a shared entity model (called CRI) that uses information gathered automatically from developer IDE interactions to make explicit orderings of tasks, artefacts and developers that are relevant to particular work contexts in a distributed software development project. It provides a detailed description of literature related to awareness in collaborative software engineering, a thorough description of the CRI model, and the results of a qualitative empirical evaluation in a realistic development scenario.

The research findings suggest that the CRI model can be used to: identify entities (developers, tasks, artefacts) most associated with a particular work context in a software development project; identify relevance relationships amongst tasks, developers and artefacts e.g. which developers and artefacts are currently most relevant to a task or which developers have contributed to a task over time; and, can be used to identify potential bottlenecks in a project through a ‘social network’ view. Furthermore, this awareness information is captured and provided to developers who may be working in different locations and at different times.
Publications


Acknowledgements

I am delighted that I have managed to put together this thesis - I will not consider this work as being perfect, but when I started this PhD I did not imagine coming this far. I cannot believe that I have enjoyed it on day to day basis.

I owe great gratitude to many people that have showed love, help and support through the course of this thesis. Firstly I have to thank my supervisors Dr John Ferguson, Dr Marc Roper and Dr Murray Wood. They have been so unimaginably supportive and patient with me even when I was getting it all wrong and thinking I was right. I am really thankful for all that constructive advice and criticism.

I would like to thank every member of the advanced software engineering class from the 2006/2007 and 2007/2008 sessions that participated in the evaluation studies. This work would have been much more difficult without them. I also appreciate the help of Kenny Forte and Ian Gordon, I cannot count the number of times I knocked on their door regarding systems support issues. I would also like to thank Dr Douglas Kirk for all the suggestions; talking with you in the lab those early days of this research was really encouraging. Thanks also to my fellow PhD colleagues - Mark Meiklejohn, Sukumar Letchmunan and Konstantinos Liaskos for their company and friendship.

I would like to give my deepest gratitude to my mother; she has shown so much faith in me and provided me with all the support both financially and morally without which this research would not have started never mind completing it. I would also love to mention the support from my brothers Ogban and Ezukwa and those four great queens - Amanda, Amabel, Amy and Obongha. Finally, my love to my dear wife Theresa for all her belief in me and the sacrifices she made so I would have time to work on this thesis.

Inah Omoronyia

November 2008

Possibilities appear to be bound by the limitations of our minds; with God all things are possible…

Anonymous
Contents

Abstract ii
Publications iii
Acknowledgement iv
Contents v

Chapter 1 Introduction 1
1.1. Overview 1
1.2. Approach and Methodology 2
1.3. Contributions 3
1.4. Thesis Structure 4

Chapter 2 Collaboration in distributed Software Engineering 6
2.1. Introduction 6
2.2. Review of awareness concepts within the framework of collaborative work 8
2.3. How to disseminate awareness information during collaborative work 11
2.4. Awareness information needs and dissemination in software development teams 12
2.5. Enhancing context awareness in distributed software development environments 17
2.6. Research Motivation: dynamic dependencies in distributed context awareness information 29
2.7. Research Hypotheses 35

Chapter 3 An awareness model for encoding collaboration work context 36
3.1. Introduction 36
3.2. Articulation and representation of a work context within virtual distributed collaboration space 37
3.3. Building an awareness perception of work effort that is dependent on interaction events that represent a selected entity work context 45
3.4. Building an awareness perception of work effort that is dependent on a varying sphere of influence that represents a selected entity work context 47
3.5. Using sphere of influence as a factor in obtaining a notion of relevance of entities to a selected work context 51
3.6. An awareness model for the relevance ranking of entities representing overall and recent work effort over a selected work context in a collaboration space 53
3.7. The CRI social graph  
3.8. Conclusion  

**Chapter 4 Implementation**  
4.1. Introduction  
4.2. Overview of the CRI architecture  
4.3. CRI tool integration  
4.4. Conclusion  

**Chapter 5 Study 1- Initial proof of concept study**  
5.1. Introduction  
5.2. Study Format  
5.3. Results  
5.4. Discussion of findings and resultant CRI system modifications  
5.5. Conclusion  

**Chapter 6 Study 2- Detailed analytical and subjective validation of CRI**  
6.1. Introduction  
6.2. Study Format  
6.3. Interview session questionnaire design  
6.4. Results  
6.5. Significance of CRI model constructs (RQ1)  
6.6. Comparison of model theoretical design with behaviour in practice (RQ2)  
6.7. Correlation between CRI relevance list rankings and developer perception of work effort (RQ3)  
6.8. Enhancing awareness during collaborative work processes (RQ4)  
6.9. Possible threats to validity  
6.10. Conclusions  

**Chapter 7 Conclusions and future work**  

References  

**Appendix A1: CRI History mode formal description**  
**Appendix A2: CRI Recent mode formal description**  
**Appendix A3: Details of an example project**  
**Appendix B1: CRI User Manual**  
**Appendix C1: CRI Subjective Evaluation Questionnaire (Study 1)**
Appendix C2: Analysis of Collaboration Data (Study 1) 201
Appendix D1: Questionnaire design (Study 2) 206
Appendix D2: Analysis of Collaboration Data (Study 2) 213
Appendix D3: Snippet of interview feedback (study 2) 223
Chapter 1 Introduction

1.1. Overview

Software development is a collaborative effort where groups of developers work together within a global time/space matrix. During such collaboration developers need to keep up their awareness of how a particular task or project artefact is progressing, what fellow developers are (or have been) doing and the current state of resources associated with the project. In collocated settings the awareness information that concerns developers directly or tangentially is achieved through the use of instant messaging clients, emails, scrum meetings and developers stopping at co-workers’ offices to update them on problems or to see what problems they are facing [KDV07]. Developing and maintaining such awareness is more difficult in distributed software teams than collocated ones [Cra01]. This is because the awareness information required in such settings is tacit, inherent, dynamic and contextual. Tacit since most of what developers do in collaboration spaces builds from experience, skills, heuristics and interactions that can hardly be documented [HF01, BR00], and inherent since this knowledge is deeply bound to these developers. Its dynamic nature stems from the ever changing state of software projects. Finally, the relevance of such information varies across differing contexts.

A number of studies have revealed the problems caused by these peculiar attributes of distributed teams. They include poor visibility and control of remote resources; inadequate communication, collaboration and coordination across distributed teams; diminishing trust; and lack of shared contextual awareness [HD04, CD04, BF04]. An industrial experience report on distributed software teams located over ten sites by Kommeren and Parviainen [KP07] identified shared contextual awareness of work carried out by different team members as a major issue. Herbsleb [Her07] suggested that this lack of contextual awareness information makes it difficult to initiate contact and often leads to misunderstanding of communication content and motivation. The lack of context information limits the capability to track the effect of changes in distributed, collaboration space [KP07].

Collaboration techniques spanning the whole cycle of software engineering process have evolved to address the problem of context awareness information. For instance, agile development methods encourage developers to regularly work with their stakeholders and users at every stage of the development cycle [Som07, Bec99]. Whitehead [Whi07] mentioned other strategies that have evolved to enhance shared understanding during
collaborative software engineering. These include, engaging domain experts to ensure convergence on a single system architecture and design; management of dependencies among activities, artefacts and organizations; reduced dependencies among software engineers; and recording organizational memory. Irrespective of these evolutions in software engineering processes, few successes have been recorded in distributed collaborative software development compared to collocated development.

On the other hand, distributed software development offers a number of expected benefits, including shortening time-to-market cycles, more rapid response to customer needs since collaborations are independent of time and space and more effective resource pooling [SCS06, KP07]. The goal of this research is to bridge the gap between the reality of distributed software development and its expected benefits by investigating systems that can be used to emulate collocation in distributed settings.

In this thesis, it is proposed that the benefits of collocation in virtual and distributed collaboration spaces can be achieved by capturing the interaction activity trails that occur within these spaces. These trails are built up as developers go about their daily development tasks leaving historical traces behind. An empirical study carried out by Fritz et al. [FMH07] suggests that these developer interactions can be used to build a model of awareness about a code base.

### 1.2. Approach and Methodology

This thesis starts by providing a thorough discussion of related literature on awareness in a general setting before focusing on previous research that has provided support for increased awareness in a collaborative and/or distributed software engineering environment. From this review it becomes clear that awareness information needs, and the mechanism for dissemination in software development teams, are more easily obtained in collocated than distributed scenarios. To achieve the potential benefits of distributed, collaborative development further work is required that can focus attention on who and what is relevant to particular work contexts within a software project. In contrast to previous work the approach proposed in this thesis aims to build an awareness model without relying on developers ‘tagging’ particular artefacts or on the limitations of the underlying configuration management system or relying on a single developers’ perception of task without considering the influence of other developers on the state of such task (that is analogous to token passing of work context).
Instead, this thesis introduces a ‘Continuum of Relevance Index’ (CRI) model based on developer interactions in a shared collaboration space. The basis of this model is the monitoring of key interactions, such as project views, updates and creates, made by any developer while working in a distributed collaborative space. The model is then used to provide relevance rankings that enhance awareness depending on the particular work context of interest.

Two qualitative evaluation studies including an initial proof of concept study and a subjective and analytical validation of CRI were carried out. The results of the first study provided an initial insight into the feasibility of CRI and aspects of the model that needed to be further fine-tuned and also understand the challenges and risks involved. The outcome of the second study carried out after the initial proof of concept study demonstrated that the model can provide accurate relevance rankings, and thereby increased developer awareness of artefacts, developers and tasks, and the extent of their interrelationships, over a range of distributed, collaborative project contexts. It also showed that the relevance rankings from the different work contexts within a project can be combined to generate a social network graph void of context. This network graphs provided useful information related to the state of tasks, code artefacts and developers that contributed to a distributed project.

1.3. Contributions

The work presented in this thesis makes the following contributions:

- A model that provides a perception of the relevance and impact of tasks, developers and artefacts associated with a distributed software project in a selected work context. For a selected task instance, awareness is provided of the relative impact of project developers and code artefacts. For a selected code artefact, awareness is provided of the relative impact of project tasks and developers on the state of the code artefact. Similarly, for a selected developer, awareness is provided of the relative impact of tasks and code artefacts on the work context of the developer.

- The provision of contextual awareness that is independent of configuration management systems or the need for tagging.

- The approach is addressed from a collaborative perspective rather than that of an individual developer that is analogous to token passing of a single developers work context.
• The provision of context based awareness in the proposed model is not limited by time or space and continually changes to keep pace with software project dynamics.

• The approach is based on real-time seamless gathering and analysis of dependencies based on core interaction events that directly or indirectly affect the state of shared entities such as views, updates and creates.

1.4. Thesis Structure

Chapter 2: Collaboration in distributed software engineering. This chapter presents a review of literature on the different concepts of contextual awareness in general Computer Supported Collaborative Work and collaborative software development. It first presents a review of general awareness concepts within the framework of collaborative work; this is followed by a discussion on how awareness information is disseminated during collaborative processes. Awareness information needs and dissemination mechanisms in software development teams are then presented, followed by a discussion on enhancing awareness in distributed software development environments. This review concludes with the thesis research motivation.

Chapter 3: An Awareness model for encoding collaborative work context. This chapter discusses the details of the awareness model. This describes the proposed technique for articulating and representing work context within a virtual distributed collaboration space. This is followed by a discussion on building an awareness perception of work effort that is dependent on interaction events that represent a selected work context. Building an awareness perception of work effort that is dependent on varying sphere of influence that represents a selected work context is also discussed. An awareness model for the relevance ranking of entities representing overall and recent work effort over a selected work context is then presented. Finally, this chapter presents a technique for building social network graphs from individual context graphs that exist within shared collaboration spaces.

Chapter 4: Implementation. This chapter discusses the perceived challenge in implementing CRI in a real distributed software development environment. A general overview of the CRI architecture is given, along with the core components that consists the different layers of the CRI architecture.

Chapter 5: Study 1- initial proof of concept study. This chapter presents the first study which was required to ascertain the feasibility and also understand the challenges and risks involved in achieving a relatively complex model such as CRI. On the whole this initial study aims to provide insight into four main research questions that assess the feasibility of
achieving CRI. These are: 1) Is it possible to capture and integrate interaction data for developers working outwith the bounds of time and space? 2) How do the formal constraints on task definition, activation and deactivation affect developer collaboration? 3) What is the validity of the assumptions used in building CRI relevance ranking of entities in a collaboration space? 4) Can CRI relevance list visualisation be used to provide a measure of work effort dissipated across a collaboration space?

Chapter 6: Study 2- Further subjective and analytical validation of CRI. This chapter builds on the initial study in chapter 5 and presents a second deeper study. The motivation behind this second study is to build on the results obtained from the first proof of concept study. The research questions addressed here fall into two general categories: analytical validation and subjective validation/usefulness. Core questions investigated are 1) What impact/significance does each of the CRI constructs have on overall CRI behaviour? 2) Does the model behave according to the design? 3) Does CRI relevance ordering correlate with developers’ perception of activities within a collaboration space? 4) Do developers feel that the different relevance list perspectives and social graph visualisations are useful in providing awareness during distributed collaborative work?

Chapter 7: Conclusion and further work. This chapter presents a brief summary of the work carried out in this research. The limitations of the CRI model are presented and avenues for further research are discussed.
Chapter 2 Collaboration in distributed Software Engineering

2.1. Introduction

Software engineering projects are inherently cooperative and require many software developers to coordinate their effort to produce large software systems. An integral part of this cooperative effort is developing shared understanding surrounding different software project tasks and artefacts and even the expertise of other developers on the shared project. Each of these task, artefact or developer entities embodies its own model over the entire development process. Whitehead [Whi07] suggested that such focus on model oriented collaboration embedded within a larger process is what distinguishes collaboration research in software engineering from broader collaboration research which tends to address artefact-neutral coordination technologies and toolkits.

Systems that enable appropriate attainment of awareness information are a core requirement to realising the shared understanding surrounding software project entities in collaborative software engineering. During such collaboration, developers need to maintain their awareness of how a particular task or project artefact is progressing, what fellow developers are (or have been) doing, and the state of the resources and tasks that they are actively using. In collocated settings, the awareness information that concerns developers directly or tangentially is typically achieved by using instant messaging clients, emails, scrum meetings and informal personal interruptions [KDV07]. Developing and maintaining such awareness is more difficult in distributed software teams, where developers are separated by time and/or space, than collocated ones [Cra01]. This is because the awareness information required is naturally tacit, inherent, dynamic and contextual. Tacit since most of what developers do in collaboration spaces builds from experience, skills, heuristics and interactions that can hardly be documented [BR00, HF01], and inherent since this knowledge is deeply bound to these developers. Its dynamic nature stems from the ever changing state of software projects. Finally, the relevance of such information varies across differing contexts.

Collaboration techniques spanning the whole cycle of software engineering process have evolved to address the problem of information awareness and hence shared understanding. For instance, agile development methods encourage developers to regularly work with their stakeholders and users at every stage of the development cycle [Som07, Bec99]. Whitehead
[Whi07] mentioned other strategies that have evolved to enhance shared understanding during collaborative software engineering. These include, engaging domain experts to ensure convergence on a single system architecture and design; management of dependencies among activities, artefacts and organizations; reduced dependencies among software engineers; and recording organizational memory.

Irrespective of these evolutions in software engineering processes, few successes have been recorded in distributed collaborative software development compared to collocated development. A number of studies have revealed the problems caused by the peculiar attributes of distributed teams [, HD04, CD04, BF04].

One core problem has been the lack of shared contextual awareness during distributed collaborative processes. An industrial experience report on distributed software teams located in over ten sites [KP07] point out that shared contextual awareness of work carried out by different team members is still a problem. Herbsleb [Her07] has also pointed out that developers at different sites share relatively little context awareness. Distributed developers tend to have little knowledge of what people at other sites are doing day to day. Herbsleb also suggested that this lack of contextual awareness information makes it difficult to initiate contact and often leads to misunderstanding of communication content and motivations. Furthermore, lack of context information inhibits tracking the effects of changes as they propagate across the distributed collaboration space.

This lack of shared contextual awareness has also had a ripple effect on a number of other key issues needed for successful distributed development. These include team coordination and communication, managing requirements and architecture, integration, and configuration management [KP07].

On the other hand distributed software development offers a number of potential theoretical benefits, including shortening time-to-market cycles and rapid response to customer needs since collaborations are independent of time and space. Furthermore, a potentially positive impact of distributed development is the mixing of developers with different cultural background which can trigger new ideas. Finally, distributed software development also allows organisations to benefit from access to a larger, qualified resource pool with the promise of reduced development cost [KP07, HD04, ED01].

The goal of this research is to contribute in bridging the gap between the reality of distributed software development and its theoretical benefits by further investigating the concept of contextual awareness and its improvement during distributed collaborative software development. Next a review of general awareness concepts within the framework of
collaborative work is presented; this is followed by a discussion on how awareness information is disseminated during collaborative processes. After that, awareness information needs and dissemination mechanisms in software development teams are presented, followed by a discussion on how awareness might be enhanced in distributed software development environments. This review concludes with the thesis research motivation.

2.2. Review of awareness concepts within the framework of collaborative work
The available research literature on the notion of awareness in general settings shows that there is no single meaning of awareness. Awareness is used in very different situations and domains to mean varying forms of consciousness gained by collaborators’ perception of information derived from their environment. Dourish and Bellotti [DB92] defined awareness as understanding of the activities of others, which provides a context for your own activity. Gutwin and Greenberg [GG98] presented awareness as a technique for enhancing coordination and efficiency when people work together. Furthermore, work carried out by Gaver et al. [GSO91] and also that of Williamson and Shneiderman [WS92] referred to awareness as a technique used to enhance usability by providing users with feedback about the state of a system and information related to a task.

Research literature also demonstrates that there exist different types of awareness. Thus, consciousness of individual collaborators in a group with regards to “affective behaviours, collective orientation, particularism or diffuseness” as viewed by Totter et al. [TGS98] is termed group awareness. Also, collaborators’ consciousness of a current process situation and projection of the current process state into the near future is termed situation awareness [Wel93].

Acknowledging that there are other forms of awareness that exist, this work focuses on four specific awareness types drawn from experiences in face-to-face interactions and addresses the needs of group work dynamics that exist during collaborative work [GST05, GGR96]. This thesis also presents how contextual awareness can be inferred from each of the awareness types discussed.

Informal Awareness
This type of awareness information is associated with a pervasive sense of who is around, what they are doing, and what they are going to do. This is the kind of knowledge people have when they work together in the same office. Informal awareness can been used to facilitate casual interactions and initiation of appropriate modes of communication [GST05].
This form of awareness provides information on the presence, location and absence of collaborators.

**Group-structural awareness**

This constitutes information about group members such as their roles, status, and position on certain issues [Gut97]. Behavioural science describes a role as a prescribed pattern of behaviour expected of a person in a given situation by virtue of the person’s position in that situation [HBC83]. In Information Science, roles are viewed as independent objects within the system that can be defined separately; these roles can then be performed by a set of one or more human users at the same time [Zhu03, GW00]. Having such formal roles defined, assigned and monitored during collaborative work provides collaborators with information related to hierarchies and dependencies that exist amongst them. Group-structural awareness is essential to obtaining cognitive knowledge of other collaborators’ expertise based on the roles they assume, this knowledge can prove important in choosing who to initiate an interaction with for mentoring on project related activities.

**Social awareness**

This is the type of information collaborators have about each other in a conversational or task context, including information such as the attention, interest and emotional state of a collaborative task or conversation partners [GST05]. Providing such awareness information helps minimise interruptions and disturbances when engaging in collaborative processes or rather ‘appropriate obtrusiveness’ as described by Schmidt [Sch02a]. Such awareness also helps in articulating complex structures that exist among collaborating entities.

A collaborator X is socially aware of collaborator Y if X has access to the interaction context of Y, this relationship is not necessarily symmetric, that is Y may not also be socially aware of X. Also Y does not necessarily need to know about such a relationship. This concept is referred to as ‘monitoring’ and ‘displaying’ methods of enabling social awareness [HSJ+02]. Monitoring is used to describe the situation where each collaborator in a collaborative process monitors the task of others; this act of monitoring does not disturb the collaborator being monitored. An important characteristic of monitoring is that it is focused on selective task processes of relevance to a particular collaborator. On the other hand, displaying describes the situation where collaborators make their task, which may be of relevance to their colleagues, publicly available. Displaying is also a selective task process since it is the displaying collaborator that chooses the task to publicise. Schmidt [Sch02a] suggested that displaying and monitoring are complementary to each other. The act of monitoring is facilitated by the display of the relevant task aspects to be monitored. Displaying aspects of
task processes to other collaborators presupposes that there is monitoring of other collaborators tasks and thereby an awareness of their concerns, expectations, and intentions.

**Workspace awareness**

This concerns information about other collaborators’ interactions with a shared project workspace and the artefacts it contains. Gutwin et al. [GGR96] described a set of elements that collaborators may keep track of during a collaborative process in a shared space and the relevant questions associated with these elements as shown in table 2-1.

<table>
<thead>
<tr>
<th>Element</th>
<th>Relevant Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity</td>
<td>Who is participating in the activity?</td>
</tr>
<tr>
<td>Location</td>
<td>Where are they?</td>
</tr>
<tr>
<td>Activity level</td>
<td>Are they active in the workspace?</td>
</tr>
<tr>
<td>Actions</td>
<td>What are they doing?</td>
</tr>
<tr>
<td>Intentions</td>
<td>What are they going to do?</td>
</tr>
<tr>
<td></td>
<td>Where are they going to be?</td>
</tr>
<tr>
<td>Changes</td>
<td>What changes are they making?</td>
</tr>
<tr>
<td></td>
<td>Where are the changes being made?</td>
</tr>
<tr>
<td>Objects</td>
<td>What objects are they using?</td>
</tr>
<tr>
<td>Extents</td>
<td>What can they see?</td>
</tr>
<tr>
<td>Abilities</td>
<td>What can they do?</td>
</tr>
<tr>
<td>Sphere of Influence</td>
<td>Where can they have effects?</td>
</tr>
<tr>
<td>Expectations</td>
<td>What do they need me to do?</td>
</tr>
</tbody>
</table>

Koichi et al. [KTT99] has extended the concept of workspace awareness to define an individual activity model for tracking activities, determining awareness scope as a structure of activities, and generating awareness information within the scope. Koichi et al.’s view of workspace awareness indicates perspectives and progress of tasks performed in collaborators’ individual workspaces.

**Contextual awareness**

The general notion of context refers to the set of circumstances or facts that surround a particular event or situation [Una06]. This form of awareness has also had different interpretations depending on the community of practice. From a Computer Science perspective, context awareness was initially perceived as referring to the location of an entity [Dey01]. The notion has now evolved to not just a location but part of a process with different state transitions [BCQ+07]. For instance, when collaborating on a task, the work context of such task does not only consist of the current state that defines such task but will also include all the different states that the task has transitioned through. Similarly, the work context of a collaborator within a project will not only be determined by the resources that
the collaborator is currently associated with but also by every other resource that they have been associated with.

Each of the awareness types discussed above is associated with the notion of context. For instance, informal awareness that provides information on the presence, location and absence of collaborators is highly dynamic. Similar conclusions can be made about the changing states of group-structural, social and workspace awareness. Furthermore, workspace awareness within a collaboration space is highly contextual and cannot be generalised. This is so because collaborators work on different tasks and different resources and hence form different perceptions of their workspace. Context aware systems in collaborative work are concerned with the acquisition, abstraction, understanding and presentation of context during team work [Sch02]

In the remainder of this thesis, contextual awareness is inferred each time the word awareness is used.

**2.3. How to disseminate awareness information during collaborative work**

Researchers have pursued several strategies to address the problem of how to keep people aware of important information. This is the strategy by which notifications are sent and represented to users of an awareness tool and the strategy by which the user is made aware of the arrival of a new notification. Cadiz et al. [CGJ+01] suggested that these strategies generally fall into one of three categories: polling, alerts, and peripheral awareness.

Polling involves making information accessible and allows people to repeatedly check—or “poll”—the information. This is advantageous when an individual has knowledge of where to repeatedly check for updates on information. This can be considered as an effective form to disseminate awareness when such information is required on a ‘as needed' basis.

Alerts involve intentionally interrupting an individual to provide awareness information. Examples include alerts reminder windows or pop ups in the middle of a screen informing of a timely activity. Alerts address the drawback of polling if a critical event occurs, since in polling users find out only when they poll the information source. Alerts can be delivered via audio or visual cues. This can also be done in both highly or minimally intrusive ways, and can be delivered using intelligent algorithms to determine if the cost of interrupting the user with an alert is worth the benefit [HJH99]. The main disadvantage is that alerts require that users get interrupted from their primary task. Studies carried out by Cutrell et al. [CCH01] and McFarlane [Mcf99] have highlighted the potential harm of these interruptions. Alerts
Peripheral awareness works by filling user’s peripheral attention with information such that it envelops them without distracting them. With this method, the goal is to present the information such that it works its way into a user’s mind without intentional interruptions. An example is knowing what the weather is like outside. If you work in a windowed office you are likely to have an idea about the current weather conditions. You are also likely to acquire this information without consciously looking out the window and checking the state of the weather every few minutes. Rather, such information persistently resides in your peripheral attention, it works its way into your knowledge and understanding of the environment.

A peripheral form of disseminating awareness has been studied using peripheral audio in the work presented by Alexanderson [Ale04] and that presented by Pacey and MacGregor [PM01]. Other forms of dissemination have focused on the use of peripheral vision or a mix of peripheral visual and audio cues [CGJ+01, HHT99, WB96]. The disadvantage of peripheral awareness is that it is much easier not to give attention to important information which is available only at the periphery. Furthermore, the challenge is to figure out what would be important to convey remotely and to strike a balance between too much and too little [Ped98].

Kantor and Redmiles [KR01] pointed out that there is no optimal strategy to provide awareness information in collaboration spaces because of their associated advantages and disadvantages. For example, if one author is changing a document, a co-author and an end-user of the document are likely to benefit from different awareness strategies. Specifically, a co-author may want to be aware that changes are currently being made, and later that the document has been checked back into a version control system. On the other hand, end users of the document may only want to receive awareness information as a digest of all changes rather than being notified each time a change is made.

2.4. Awareness information needs and dissemination in software development teams

A study conducted by Ko et al. [KDV07] to understand the information needs in collocated software developments teams showed that the most frequently sought information included awareness about tasks, artefacts and co-workers. Developers frequently sought information about how the artefact resources they depended on had changed; what their fellow team mates have been doing; and what information was relevant to their task.
In collocated teams, such awareness information has been disseminated via polling and alert mechanisms such as email and instant messaging clients. Development teams also have frequent brief meetings throughout the day to keep up to date of work effort and problems team mates are going through. Furthermore, developers would stop by co-workers’ offices, in the hallway or chatting over coffee to update them on problems or to see what problem they were facing [CKI88]. On a whole, research studies demonstrate that the social nature of software development work is also driven by awareness information seeking needs of developers. For instance, Perry et al. [PSV94] reported that over half of developers’ time was spend interacting with teammates, of which much of the communication was to maintain awareness. The study carried out by Ko et al. also demonstrated that co-workers were the most frequent source for information seeking.

Furthermore, simply physically watching another developer carry out their task [Seg95] and feedthrough, where observation is made of changes to project artefacts [DFA+04], have also been used as a source of obtaining awareness in collocation.

Awareness information needs in distributed software development teams are not distinctively different from collocated needs. In distributed teams, developers also seek to maintain awareness of other developers including the code artefact resources and the tasks they are working on. A study presented by Gutwin et al. [GPS04] asked distributed developers what kind of information about others they tracked. The results showed that developers sought to maintain a broad awareness of who are the main people working on their project, and what their expertise is. Furthermore, when a developer wished to work on a selected task, they needed to gain more detailed knowledge about the developers with experience in related tasks and developers that had previous experience with the code artefact they are currently working on.

For distributed teams, obtaining and disseminating of awareness information is rather more challenging and mostly dependent on electronic means compared to collocated teams where face to face communication under different scenarios is core in obtaining awareness. Open-source software development projects are typical examples of collaborative work where almost all entities that are involved in the collaboration are distributed. The open-source community have managed to produce large, complex and successful systems such as the OpenOffice application suite\(^1\), the Linux Operating system\(^2\) and the widely used Apache web

\(^1\) [www.openoffice.org](http://www.openoffice.org) (verified 24/08/2008)

\(^2\) [http://www.linux.org](http://www.linux.org) (verified 24/08/2008)
server. Studies carried out on these successful projects showed that awareness was mainly maintained using text-based communication such as mailing lists and online chat [GPS04].

The wiki is another text-based channel which has been enhanced to support awareness in distributed development processes such as code reviewing, bug tracking and functional testing. Trac is an enhanced wiki and issue tracking system for software development projects. It allows for creating links and references between bugs, tasks, change sets, files and other wiki pages. It also provides an interface to Subversion, which is a configuration management system. TWiki has also been developed as a text-based collaboration platform and a knowledge and document management system [XCY07].

The main advantage of text-based channels in sharing awareness in distributed software development stems from its simplicity of use. In a wiki, users write simple text following a small number of conventions. The system creates the necessary links automatically. Detailed knowledge of programming is normally not required to update such systems [Lou06]. Thus, use of textual channels is expected to reduce the effort required for collaborating developers to get on with the core development task and still maintain awareness of the collaboration environment. But while textual channels have several characteristics that help support the maintenance of awareness, its success is dependent on developers’ commitment to reading the list and making their project communications public [BCS+07].

Version check-in logs into configuration management systems have also been reported to be used in obtaining awareness of work in both distributed and collocated software development. Study carried out by de Souza et al. [DRM+03] on collocated developers showed that when determining the appropriate person to ask about code related issues, developers often gauge expertise by inspecting check-in logs. Similarly, Gutwin et al. [GPS04] reported that distributed developers obtained awareness through commit logs (records of changes made to the source code). The record of each change includes information such as the developer committing the change, the files affected, the number of

---

3 http://httpd.apache.org (verified 24/08/2008)

4 http://trac.edgewall.org

5 Subversion is an open source version control system used to maintain current and historical versions of files such as source code, web pages, and documentation. http://subversion.tigris.org/

6 http://twiki.org

7 Expertise assumes an embodiment of knowledge and skills within an individual. The definition used here is similar to the view presented by McDonald and Ackerman [MA98] which distinguishes expertise as a range. An individual can have different levels of expertise about different topics. Expertise can be topical or procedural.
changes and difference between the old and new versions. Such changes can be sent automatically to a mailing list that developers can subscribe to. The advantage of commit logs is that they are an awareness source that is based on actual manipulations of the project artefacts.

However, awareness information obtained from check-in logs or commit logs is not always accurate. While these logs can help find other developers, they generally do not distinguish levels of expertise [MA98]. This problem is made more complex if the piece of code artefact a developer wants to seek awareness information about has been worked on by several developers who may have different levels of expertise related to the code artefact. While strict partitioning and code ownership rules can be enforced to avoid this problem, ideal distributed software development such as the open source development, code ownership is not always obtainable. Gutwin et al.’s study showed that partitioning was not so strongly applied on open source projects, developers were free to work where they saw fit. Similarly, Mockus et al.’s. [MFH02] study findings also showed the lack of clear ownership and partitioning in distributed open source projects and concluded that the structure of a project is not sufficient to enable developers to obtain the awareness necessary to coordinate their actions. Other coordination mechanisms based on non-structural relations within a shared collaboration project such as those based on developer navigation patterns within a shared collaboration space can be used to attain awareness in distributed software. Furthermore, a study presented by de Souza et al. [DQT+07] also suggested that technical dependencies among software components create social dependencies among the software developers implementing those components.

The closest imitation of collocation in distributed software development has been in the use of media spaces. A typical media space consists of permanent video and audio connections between geographically distributed sites. It has also been reported that media spaces can help in the attainment of high awareness because the video and audio connections transmit natural real-time audio and video pictures of remote sites. The permanent connection of media spaces have been shown to reduce the cost of initiating collaboration and contributes to the creation of a common social space irrespective of distance [Far00, Sin99, BHI93]. While media spaces cannot replace face to face awareness attained in collocated software development, it provides an opportunity for obtaining awareness information that is not possible without ‘being there’ [BHI93]. The use of video conferencing, video phone and desktop video and audio capabilities in distributed development has also been studied [TTH08].
The use of video and audio channels for awareness is limited to synchronous processes alone. The use of such channels is highly challenging for distribution characterised by different time zones where work hours are likely to be different. Furthermore, such a mode of awareness lacks flexibility compared to other electronic means such as email and check-in logs where information can more easily be searched and referred to when convenient. Furthermore, an empirical study conducted by Seaman and Basili [SB98] on software inspection meetings found that collocation of the workspace of inspection participants enhanced awareness in ways that is unavailable in distributed situations. Gutwin et al. also pointed out that coordination problems can be exaggerated in distributed development due to a lack of spontaneous localised communication channels.

On the whole, the use of mixed media- a combination of synchronous and asynchronous awareness channels- has been proposed during distributed collaboration. This is achieved by matching collaboration task with media characteristics. This concept has formed the basis of the Time-Interaction-Performance (TIP) theory developed by McGrath and Hollingshead [MH94] and the theory of Task/Technology Fit (TTF) by Zigurs and Buckland. [ZB98]. Work related to TTF has also been presented by Goodhue and Thompson [GT95]. Both theories account for the nature of task and technology characteristics in the evaluation of media for electronically-mediated group collaboration. In requirements engineering during distributed software development, it has been proposed that a combination of lean (example email) and rich (example video conferencing) media is needed to improve the effectiveness of distributed requirements negotiations [DLM08].

Distributed software processes in high process maturity organisations have also not shown convincing results. Studies carried out by Ramasubbu and Balan [RB07] reveals that even in high process maturity environments, distributed software practice reduces development productivity and conformance to quality. Companies that institute high quality software processes are far more likely to overcome the negative effect resulting from distribution of software development than companies that are not. Another study conducted by Herbsleb and Mockus [HM03] showed that, compared to collocated software development, distributed development takes much longer, and requires more people for work of equal size and complexity. The study also showed that this outcome was largely due to the lack of physical presence of colleagues when workload is heavy in distributed development.

On a whole, the review of research literature does show that collocated software development is attributed with some awareness information benefits that are not frequently obtained in distributed scenarios. Collocated teams are likely to achieve higher productivity,
shorter schedules, and higher satisfaction among stakeholders [TCK+06]. To leverage on the advantages of both collocation and distributed development, it is also important that tool support for distributed teams endeavour to emulate the attributes of collocation awareness in their design.

Conversely, research studies show that tool support for distributed software development teams are still inadequate in enhancing distributed awareness. In Sillito et al.’s [SMD08] empirical study, it was revealed that most tools are designed to answer a specific kind of question and targeted on only a particular type of code artefact. Also, most tools treat information seeking questions as if they were asked in isolation. The study further suggests that such assumptions make it extremely difficult for a distributed development team to obtain the contextual awareness that is necessary to enhance the understanding of distributed project processes. This is because awareness questions asked by developers are often not in isolation but part of a larger process involving multiple questions.

The next section provides a review of concepts, methodologies, models and systems in software engineering development environments that have the capability of enhancing contextual awareness in distributed collaboration.

2.5. Enhancing context awareness in distributed software development environments

Today software development environments are enhanced with facilities which are themselves software applications used to make software development easier. Such Integrated Development Environments (IDEs) are normally designed to maximise developers’ productivity by the integration of comprehensive facilities. Such facilities include source code editors, compilers, interpreters, debuggers, visualisations and generators. IDE can be a standalone application or may be included as part of one or more existing and compatible applications. Typical examples of IDEs include NetBeans\(^8\), Microsoft Visual Studio\(^9\) and Eclipse\(^10\). The sheer complexity, size and distributed nature of today’s software projects also comes with a demand for more tools to enhance their potential. This section discusses the different concepts that have been used to enhance contextual awareness during distributed software development within IDEs.

---

8. [http://www.netbeans.org](http://www.netbeans.org)
Attaining context by social tagging

A tag is generally referred to as a keyword assigned to a piece of information. Such keywords can then be used to search or describe the information. Social tagging describes the collaborative activity of marking shared content with tags as a way to organize content for future navigation, filtering, or search [LL08, YGT06, GH06]. An important feature of tagging to an individual developer is the ability to annotate the information in a way that is easier for them to recall at a later date. This concept has been introduced into a number of IDE components which can essentially be used to enhance contextual awareness in distributed collaborative software project.

The concept of social tagging can be found in Jazz\(^{11}\), which is a real-time team collaboration platform for the Eclipse IDE to help integrate work across the different phases of a collaborative software development lifecycle. One of the aims of Jazz is to introduce contextual awareness into the collaboration environment [HCR+04]. With Jazz developers can initiate chats, which can be saved as code annotations on the section of the code artefact involved in the discussion [CSH+04].

Storey et al. [CDS07, SCB+06] have presented TagSEA (Tags for Software Engineering Activities in Eclipse) based on the concept of Waypoints (locations of interest) and Social Tagging (social bookmarking). This tool enhances contextual awareness by supporting collaborative annotation in software development. As shown in figure 2-1 labelled X, the developer Chris has annotated the program code related to `getSelectWaypoint()` java method with awareness information describing its state. The waypoint analogy corresponds to marking specific locations in the software such as Java source code elements (classes, methods, packages etc). The social tagging concept is based on the view that these waypoints are supplied by programmers. TagSEA is motivated by interest in supporting collaborative annotation, navigation, and coordination activities through inline comments. Programmer-created annotations written as comments embedded in the code result in very explicit landmarks for readers to support navigation and coordination. Preliminary feedback suggest that implicitly captured meta-data combined with the lightweight nature of tagging results is a promising technique for supporting contextual awareness in distributed software development. The risk of adding annotations to source code is that it can become unwieldy and outdated over time.

\(^{11}\) [http://www.jazz.net](http://www.jazz.net)
Figure 2-1 TagSEA plug-in for Eclipse IDE (Source: http://tagsea.sourceforge.net/)

19
The modelling of tagging can also be derived from Kantor and Redmiles [KR01] work where they developed a software development awareness infrastructure called CASS (Cross Application Subscription Services). CASS was developed based on the premise that any aspect of a software system can be an information source to collaborating developers. There are developers who may be affected by changes to the state of the software, and could perform their work more efficiently if they were aware of those changes. CASS provides a notification server for the distribution of awareness information. The server enables developers to subscribe to types of information that they believe will affect them, and to specify which types of awareness tool the information should be sent through. The goal of CASS is to link an arbitrary number of awareness information sources with an arbitrary number of ways such awareness information can be disseminated.

Appropriate use of CASS requires that a developer identifies the aspect of the software system that can change and which is also of interest to them. Next, the developer needs to build monitors into the software system that are dependent on their specific needs. Finally, the developer needs to provide information such as which email to send the awareness notification information. The outcome of such configuration is that developers get contextual awareness information of the state of different aspects of a software system that directly affect their work. The main challenge with using CASS to attain contextual awareness lies with the configuration that is needed to be carried out by the developer to get the tool running. It is expected here that this might be asking too much from a developer that is not familiar with the development space, or from developers who see carrying out such steps as too demanding or secondary compared to the primary task of actual coding.

Furthermore, Augur [FD04] is a visualisation tool that provides a line-oriented view for supporting distributed software development processes. These views are formed by tagging developers to the different aspects of an artefact they have been associated with. Initial evaluation of Augur with open source software developers suggests that generating views based on tagging the activities of developers to subsections of artefacts is both meaningful and valuable to software developers.

**Attaining context by mining relational properties among software project entities**

During software development processes, different forms of relationships are formed. Such relationship can be structural - relations derived based on direct and indirect links among artefacts that constitute a project. Such relationship can also be social – relations derived based on direct and indirect links among developers collaborating over a project. A hybrid of such relations can also be attained that involve relations brought about by associations
between developers and artefacts that are associated with a shared software project. The use of such project structural and social relational properties that exist during software development collaborations has also been modelled into context awareness mechanisms within IDE components.

The Rational Team Concert, a plug-in to Jazz, enables contextual awareness by mining relational properties of entities constituting shared software projects. A relational view of the Rational Team Concert model is as shown in figure 2-2. In Rational Team Concert, project artefacts are stored in a **Repository**, which further contains a number of **Project Areas**. Each project area contains the artefacts for a project and has an associated process which governs how the project is run and the way Jazz behaves. A process is defined by its specification, state and description. The process specification defines the project's **Iterations** and how it behaves during these iterations. The process state defines the current iteration. The process description corresponds to a web site explaining the process. Project areas are decomposed into a set of **Team Areas**, which describe the teams that work on the project. Each team area has a list of team members and the **Process Role** they play within the team. A user can be a member of more than one team. Each team area can define **Process Customizations** of the process to tailor Jazz for the team and its sub-teams. Finally, the planned work is described by **Work Items**. The types of work items used in a project area are defined by the process [Fro07].

The benefit that the rigorous relational view of collaborative processes in Rational Team Concert model offers is that it can enhance traceability and the process of obtaining contextual awareness of the state of different entities that exist within a project. For instance, information about the state of a work item can be derived by viewing the code artefact resources and developers that such a work item is associated with. Similarly, contextual information about a project team can be derived by navigating the different team members and process associated with the project team. On a whole, Rational Team Concert enforces rigorous dependency constraints that enable different aspects of system artefacts to relate to their abstract model representations.
An evaluation case study conducted by Cheng et al. [CSH+04] using Jazz pointed out that privacy is a concern among collaborating software developers. Participants in the study expressed apprehension that the resource-centred nature of Jazz might be used by unethical managers to monitor their work, instead of being used as a coordination aid to enhance awareness. The case study demonstrated that Jazz capabilities reduce possible development conflicts by its integration of different collaborative capabilities within a coding environment. The case study also demonstrates its ability to provide context and traceability by enabling developers to converse around code and link messages and status information with code artefacts. In a related empirical study carried out by Nguyen et al. [NWD08] showed that the effect of distance was mitigated using Jazz in which communication in large distributed work teams was facilitated by an ability to asynchronously comment on and tracking activity of work items.

The work of de Souza et al. [DQT+07] in developing Ariadne also demonstrates the use of relational properties among entities to derive contextual awareness within IDEs. Ariadne is an endeavour to reduce the difficulty of communication and coordination in large-scale software development projects. De Souza et al. postulated that this difficulty was based on the fact that the number of dependencies that software developers have to face increases with the size of the project. Ariadne is a plug-in for the Eclipse IDE and analyses software projects for dependencies and collects authorship information about projects relying on configuration management repositories. The tool translates technical dependencies among
components into social dependencies among developers and creates a visualisation to convey the dependency information. Ariadne is used to identify developers who are more likely to be communicating by assuming that developers with similar dependencies make them likely to collaborate.

Ariadne identifies technical dependencies in source code by constructing static call-graphs (technical matrix) which are further annotated with social information. The association between authors (creator or owner of a code artefact) and code is represented as a sociotechnical matrix where entries represent a value for the strength of connection between author and code. The technical and sociotechnical matrix are then combined to generate sociograms describing the dependency relationship among software developers. Figure 2-3 shows a sociogram view of a project in Ariadne.

Ariadne aims at bridging the gap between dependencies and communication and coordination needs during software development, and is able to identify developers who are more likely to be communicating as well as developers whose similar dependencies make them likely to collaborate. Visualisation approaches used highlight code modules that are more highly depended on than others by displaying them in larger size. This relative strength of dependency and hence its size representation is determined via structural static analysis of code artefacts.
Figure 2-3 Sociogram of the Sourceforge.net project Tyrant, created using Ariadne (Source de Souza et al. [DQT+07])

On the whole, Ariadne can help provide the awareness that reduces redundant work and helps identify relevant developers to ask for help with particular code modules. Developers can achieve this by identifying other developers that share similar socio-technical dependencies with them. The accuracy of Ariadne is dependent on the accurate state of the versioning system from which it derives its relations. Furthermore, while Ariadne derives sociograms based on authorship or creators of codes, one would also expect that capturing other events such as updates by developers and artefacts they have viewed to derive understanding of software projects can further strengthen such sociograms.

A number of other tools and models have been investigated with the potential to enhance an individual’s contextual awareness in distributed development based on mining relational properties that exist among software project entities. Bruegge et al. [BDW06] have developed Sisyphus, a tool that supports the creation and subsequent browsing of a graph created by linking artefacts, as well as annotations and comments on those artefacts. It achieves this by encouraging collaborators to make communication and issues (tasks) explicit in the context of system models and also to become aware of relevant stakeholders.
Finally, Cubranic et al. [CMS+05] describe the design and evaluation of Hipikat, a tool that draws on information retrieval techniques to help developers identify artefacts that are related to an initial artefact used to generate a query. Information retrieval mechanisms are exploited to derive relative dependencies among abstract model representations such as bug reports and their tangible artefact representations. Evaluation shows that the tool finds useful starting points for exploring the code.

**Attaining context by monitoring developer interactions**

Interactions that are carried out in collaboration spaces can be viewed as having different levels of impact on the state of a project. By associating different weightings to interactions based on their perceived levels of impact on the state of a shared project it is possible to obtain contextual awareness.

This concept of weighting severity of developer interactions can be seen in the modelling of Palantír [SNH03, SH02]. Palantír is a workspace awareness tool that complements existing configuration management systems such as CVS\(^\text{12}\). It enhances awareness by continuously sharing information regarding operations performed by all developers. The tool specifically informs a developer of which other developers change which other artefacts. Furthermore, Palantír provides a measure of the severity of those changes and graphically displays the information in a configurable manner.

One of the concepts behind Palantír is that simply knowing which artefacts are changing in other workspaces is useful, but having an associated indication of severity conveys more information. While small changes typically will be easy to reconcile, large changes may signify a regression towards potentially difficult conflicts in an eventual integration. A trivial severity measure is binary and simply indicates whether or not any kind of change has occurred. A slightly more complicated measure of severity is calculated by dividing the number of lines that has been added, removed, and changed by the total number of lines in an artefact. Palantír provides each developer with a graphical display that not only shows which remote artefacts are changing, but also presents them with a measure of both the severity and the impact of the changes. The main visualisation presented by Palantír is as shown in figure 2-4; the two vertical bars indicate severity and change impact. Artefacts that have not been checked in yet do not have any severity or change impact since the potential changes are still hidden in the workspace.

\(^{12}\) CVS (Concurrent Versions System) is a configuration management or version control system that keeps track of all work and all changes in a set of files
The key concern here was to disseminate awareness information that helps in avoiding direct and indirect coding conflict for developers depending on the same piece of code artefact. The accuracy of Palantír is highly dependent on collaborating developers’ consistent check in of the work they are carrying out to a version management system. Also, while it is expected that non structural relations among collaborating software developers with artefacts can further enhance the attainment of awareness, it is not considered in Palantír. Finally, it is expected that there are cases in distributed scenarios where more than one developer has influenced the state of an artefact. Appropriate context in such scenarios will require that the relevance of all developers to the shared artefact is represented.

Furthermore, this conceptual view of obtaining contextual awareness is shown in Biehl et al.’s [BCS+07] work in the development of FASTDash (Fostering Awareness for Software Teams Dashboard). FASTDash is a visualisation tool that seeks to improve team activity awareness using spatial representation of the shared code base that highlights team members’ current activity. FASTDash is designed for project teams of 3-8 collaborating programmers and can enable obtainment of contextual awareness information such as which code files are changing, who is changing them and how they are being used. Furthermore, the visualisations can be annotated, allowing collaborating programmers to supplement context information with additional status details.

FASTDash is optimised for developers that are working at relatively close proximity and time, and will be less useful across different time zones or in cases where collaborators are
given the freedom of choice of place and time of work. The evaluation of FASTDash design provided initial evidence that it makes a significant contribution to improve collaborating developers’ sense of awareness and their feelings of stability in their environment.

A number of other tools such as Community Bar [TGG06], SeeSoft [ESS92] and SEAPort [BB06] have also been created to convey the context status of different aspects or sub aspects of a shared artefact while collaborators are working on them in real-time.

*Attaining context by monitoring developer interactions and derived relational properties*

In addition to attaining context by inferring that different interactions have different levels of impact on the state of the project, the project structural and social relational properties that exist during the collaboration process could have a consequence for the formation of a perception of context. For instance, a developer that has updated an artefact a number of times could consider such an artefact as more significant within their work context compared to an artefact they updated just once.

This notion of context can be seen in the development of the Team Tracks project [DKC+05]. This utilises the notion of the relational property of an interaction and the frequency of such an interaction. Team Tracks helps developers understand unfamiliar source code by mining the navigation data as development teams go about their daily programming activities. Team Tracks is based on two insights: the more often developers visit a part of the code the more important it is; and, the more often developers visit two parts of the code in succession the more related they are. To help a newcomer quickly find the most important parts of the code, Team Tracks limits the code overviews to the most frequently visited items (favourite classes view). To help a newcomer find code related to the module currently being worked upon, Team Tracks recommends parts of the code visited just before or after that module (related items view).

A controlled lab study has shown that Team Tracks significantly improves a developer's ability to perform updates to unfamiliar code. On the other hand, Team Tracks provides no indication of contextual relevance of developers with respect to code.

Other similar tools such as Expertise Browser [MH02] and Expertise Recommender [MA00] exist in this category of deriving and sharing awareness information.

*Attaining context by combining developer interactions and derived relational properties with the notion of time and expiration*

In addition to attaining context by inferring that different interactions combined with relational properties have different levels of impact on the state of the project, the time at
which such interaction occurred can also affect on the way the formation of context is perceived. For instance a developer’s perception of work context can be more influenced by recent interactions to achieve a software project endeavour compared to interactions carried out at the earlier stages of the project.

This notion of context formation can be seen in the modelling of Mylyn [Ker07] as a task-focused interface for Eclipse. The main objective of the tool is to reduce information overload and make multi-tasking easier. Mylyn monitors a developer’s work activity to identify information relevant to the task-at-hand, it then uses the task context to focus the Eclipse user interface on recent activities.

A conceptual overview of Mylar model is as shown in figure 2-5. This demonstrates how interaction history is used to form a task context model which is then projected into different visualisations. The nodes and edges in this graph reference concrete elements and relations in the target information system, in this case different kinds of Java declarations (M=method, C=class, I=interface). As the interaction history is captured from a developer, a degree of interest (DOI) function assigns a real number weighting to each element and relation.

Figure 2-5 Constructing and projecting a task context in Mylar (Source Kersten [Ker07])

A conceptual overview of Mylar model is as shown in figure 2-5. This demonstrates how interaction history is used to form a task context model which is then projected into different visualisations. The nodes and edges in this graph reference concrete elements and relations in the target information system, in this case different kinds of Java declarations (M=method, C=class, I=interface). As the interaction history is captured from a developer, a degree of interest (DOI) function assigns a real number weighting to each element and relation.
corresponding to the difference between the frequency of access to the element or relation and a decay factor that corresponds to the total number of interaction events captured. Accessing an element increases its weight, while accessing other elements decays the weight of infrequently accessed elements. Value ranges on the DOI specify which elements and relationships are interesting and uninteresting. Interesting elements or relationships are those with a positive DOI value. Uninteresting elements or relationships are those with a negative or zero DOI value, which occurs either through decay or because the element or relation has never been the target of an interaction. Description of the core interaction events monitored by Mylyn is as shown in table 2-2.

<table>
<thead>
<tr>
<th>Event kind</th>
<th>Interaction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection</td>
<td>direct</td>
<td>Editor and view selection via mouse or keyboard</td>
</tr>
<tr>
<td>Edit</td>
<td></td>
<td>Textual and graphical edits</td>
</tr>
<tr>
<td>Command</td>
<td></td>
<td>Operations such as saving, building, preference setting and interest manipulation</td>
</tr>
<tr>
<td>Propagation</td>
<td>indirect</td>
<td>Interaction propagates to structurally related elements</td>
</tr>
<tr>
<td>Prediction</td>
<td></td>
<td>Capture of potential future interaction events</td>
</tr>
</tbody>
</table>

Empirical evaluation of Mylyn, in which professional programmers used this tool for their daily work on enterprise-scale Java systems, shows developers spending more time working on code than they spend navigating it (as opposed to the other way round). While Mylyn encodes a degree of interest based on developer interactions, and possesses support for shared task contexts with a notion of time and its expiration from an individual developer perspective, ideally a degree of interest model for distributed software development should also take collaborative effort into account.

2.6. Research Motivation: dynamic dependencies in distributed context awareness information

This review has been centred on the general notion of context-based awareness and related dissemination mechanisms, in combination with the understanding of the concepts and related tools that can be used to enhance contextual awareness during distributed software development within an IDE. It has exposed five general concepts that can be used to enhance contextual awareness during distributed software development within an IDE. These concepts include obtaining context by social tagging; obtaining context by mining relational
properties among software project entities; obtaining context by monitoring developer interactions; obtaining context by monitoring developer interactions and derived relational properties; and obtaining context by combining developer interactions and derived relational properties with the notion of time and expiration.

The initial insights that have been obtained from this review of literature is that context awareness information needs and the mechanisms for dissemination in software development teams are more easily attained in collocated than distributed scenarios. Related work has shown this challenge increases as the distributed nature of software development teams becomes more complex. Figure 2-6 shows the increasing nature of this complexity. A collocated (single organization) team is the least complex, and a distributed, multi-organization team is the most complex.

![Figure 2-6 Complexity scale for distributed projects (Source: Simons [Sim04])](image)

Furthermore, contextual awareness has been elucidated using different techniques for the abstract representation of system models in collaborative software development. More formal representations such as UML use case diagrams, class diagrams, activity diagrams have been used; other formal axiomatic, abstract, algebraic and state machine model specification languages such as VDM, Z and Obj have also been utilised. Informal representations such as tasks, problem statements, test case definition and bug reports are also common. Each of these models aims to abstract and represent the software system from different process, behavioural or structural perspectives. While distributed teams have their associated advantages, to achieve them necessitates that more is still needed to be done in the way context based awareness is modelled and disseminated.

Studies that have investigated the overall breakdown of a developer’s time, report that maintaining awareness of a software project’s evolving states still consumes a significant proportion of a developer’s time. Some have reported developers spend as much as 40% of the work day communicating about code. These review findings suggest that to further enhance context based awareness in distributed collaborative software development there also needs to be a more rigorous elucidation of the dynamic
nature of dependencies that exist among entities that are involved in achieving a project
everend. De Souza et al. [DQT+07] have already pointed out some core awareness
problems that can arise in distributed collaborative software development including lack of
awareness of evolving social dependencies, lack of awareness of evolving code
dependencies and lack of awareness of finding the right or similar developer.

Finally, software projects that are complex and distributed also require that more resources
are allocated to them to be achieved with more rigorous abstract system models representing
them. The negative consequence of such increased resources and associated complex
abstract system models in open source software development have been reported by Ellis et
al. [EWD+01] and Mockus et al. [MFH02]. One of the consequence of this is the
introduction of a new level of intricacy where a number of resources such as developers,
artefacts and tasks might only have marginal effect on project state. Furthermore, such a
marginal developer might not be attuned to the realities of the distributed project. These
complex dependency amongst entities in a collaboration space makes it more difficult to
attain awareness of the relevance of entities responsible for the state of a particular work
context.

Within such complex dependency settings, one will expect that a code artefact can be
associated with a number of developers and used to achieve a number of project tasks such
as use cases and bug fixes. Similarly, a project task can be associated with a number of
collaborating developers and a number of code artefacts. Finally, it is expected that a
developer can be working on a number of project tasks and using a number of code artefacts
to achieve the task [GPS04, MFH02]. Such dependencies simply increase the number of
entities that can be used to describe a work context and hence make it more difficult to
derive meaning from awareness information extracted from the context.

For instance in a distributed setting, without any pre-knowledge about a measure of
developer work effort or the amount of effort put into achieving a task or implementing a
code artefact then vital awareness information required during effective collaboration will be
missing. For instance selecting an appropriate developer to seek for help on an artefact that is
associated with a number of developers will be much more difficult compared to if it was
just a single developer associated with the artefact. Similarly, determining which code
artefacts have affected the state of a task will be easier if the task was achieved with a few
artefacts compared to using large a number of artefacts. Finally, a developer gaining a grasp
of a work context or that of colleagues becomes more difficult as developers are associated
with more project tasks and code artefacts. While some awareness can be easily obtained in
collocated settings by informal means, it will be much more difficult to obtain this in a distributed setting without any tool assistance.

The approach taken by systems such as TagSEA [SCB+06], CASS [KR01], Augur [FD04], FASTDash [BCS+07] and some aspects of Jazz [CSH+04] as previously discussed are not based on the actual manipulation of code and are thus subject to a level of misrepresentation of context as the tagging developer deems fit. A system that builds context awareness based on the actual manipulation of code artefacts by developers as they work on different tasks that constitute a software system will have a reduced chance of context misrepresentation. Again, tag based systems can become outdated over time especially when developers fail to update tags as different aspects of the system change. A real-time model that seamlessly gathers and builds context awareness as developers carry out their development task will be more appropriate for the elucidation of complex dependencies amongst entities. On a whole, tag based systems can easily become unwieldy and are not optimised for enforcing dependency constraints that are essential for the generation and representation of more complex contextual awareness information.

Also, the challenge of alternative approaches adopted by systems such as Sisyphus [BDW06], Ariadne [DQT+07], Hipikat [CMS+05] and the Rational Team Concert [Fro07] arises as the relational dependencies amongst entities become increasingly complex. Such complex dependencies arise over time as more entities are used to achieve a selected work context. Sisyphus enforces dependency constraints and presents contextual awareness through a participant’s ability to directly attach collaboration artefacts to system models such as tasks, problem statements, requirements, architecture, detailed design or test cases thus elucidating dependencies that will otherwise be left implicit. Sisyphus does not focus on addressing the relative impact and relevance of a formal or informal abstract representation of system model on the actual state of collaboration artefacts it is associated with. In this research, the assertion is that such focus will provide further insight into the relative strength of each dependency constraint that has been enforced and thus enable collaborators achieve awareness of the impact and relevance of tangible artefacts on the realization of an abstract model.

Ariadne addresses issues of dependencies amongst entities with a social call-graph describing which software developers depend on which other software developers for a given piece of code. Ariadne does not exploit dependencies generated resulting from artefacts that are not structurally related such as dependencies created by developer’s
frequent viewing of code artefacts to attain insight for the development of another code artefact. The aim in this research is to exploit the different interaction events that directly or indirectly affect the state of shared entities such as views, creates and updates. Furthermore, Ariadne does not focus on a representation of abstract system models such as tasks and how different aspects of the system model impact on its tangible artefact representations.

Hipikat and Rational Team Concert address contextual awareness resulting from different levels of dependencies amongst entities using information retrieval techniques and rigorous enforcement of dependency constraints that enable traceability between system models and code artefacts. However, they have not focused on exploiting the relative strength of the dependency among these different formal or informal abstract representations of system models and the related tangible artefacts representations.

Palantír focuses on developers that are working in parallel on the same code artefact. Dependencies and their weighted severity are articulated based on change updates captured by Configuration Management (CM) systems during check-out and check-ins. The reliance of Palantír on CM limits the nature and amount of dependencies it represents. For instance the dependency created by developers’ frequent view of code artefacts to obtain insight for the development of another code artefact is not captured by CM systems. Palantír is mainly focused on resolving conflicts and less focused on elucidating contextual awareness amid complex entity dependencies. For instance, Palantír does not seek to enhance awareness of knowledge of the appropriate developer to seek for help amid a number of other developers that have been dependent on a code artefact, nor does it incorporate dependencies between formal or informal abstract representation of system models and the tangible artefacts that reside in the CM systems.

Again, Team Tracks is limited in the nature and amount of dependencies it represents and the related weighted severity of the dependency since only developer view interaction events are captured amid other possible interaction events such as updates and creates that can directly or indirectly affect the state of project artefacts. Team Tracks does not explicitly associate a formal or informal abstract representation of system model with their tangible artefact representations.

Finally, Mylyn does not focus on distributed software development. Its notion of task context for collaborative software development is analogous to passing tokens of context generated by one developer to another to continue building upon. Such process will not adequately represent all the varying influences an artefact has on the different aspects of a system model.
over its history. Thus, assuming an artefact has been associated with two or more tasks collaborated with by a number of developers, then a task context token related to the artefact does not hold awareness of the influence of the same artefact on other task tokens. It is asserted that while it is useful to achieve awareness of the impact of a task on the state of an artefact, it will be more useful for developers to obtain awareness of the impact of all other tasks that also had influence on the state of a code artefact over a project history.

The motivation for this work is to extend and strengthen the awareness capabilities of existing distributed collaboration systems. This thesis will examine ways that such complex dependencies can be elucidated to enable simpler and more intuitive representation of contextual awareness in distributed software development projects. The approach is as follows:

- For a selected task perspective, monitor the developers and code artefacts that constitute the task work context and the relevance of these to the selected task.
- For a selected artefact perspective monitor the tasks and developers that constitute the artefact work context and the relevance of these to the selected artefact.
- For a selected developer perspective, monitor the artefacts and tasks that constitute the developer work context and the relevance of these to the selected developer.

The outcome of this approach is a context based relevance model that provides a perception of the relevance and impact of tasks, developers and artefacts associated with a software project in a selected work context. The goal is then to investigate the use of this relevance model to derive accurate and real-time contextual awareness of the overall work effort of individual developers as well as their current work.

The novelty of the proposed approach compared to previous work is a model that provides a perception of the relevance and impact of tasks, developers and artefacts associated with a distributed software project in a selected work context. For a selected task instance, awareness is provided of the relative impact of project developers and code artefacts. For a selected code artefact, awareness is provided of the relative impact of project tasks and developers on the state of the code artefact. Similarly, for a selected developer, awareness is provided of the relative impact of tasks and code artefacts on the work context of the developer. The provision of such awareness is independent of configuration management systems or the need for tagging, and is from a collaborative perspective rather than that of an individual developer. Furthermore, the provision of this context based awareness is not limited by time or space and continually changes to keep pace with software project
dynamics. The approach is based on real-time, seamless gathering and analysis of dependencies based on core interaction events that directly or indirectly affect the state of share entities such as views, updates and creates.

2.7. Research Hypotheses

The two main research hypotheses investigated in this thesis are:

1. Contextual awareness can be built from a model that is based on real-time monitoring of IDE interactions such as create, edits and views by developers on related tasks and code artefacts.

2. Such a model can enhance contextual awareness during distributed collaborative software development.
Chapter 3 An awareness model for encoding collaboration work context

3.1. Introduction

This research proposes that the awareness of development context in virtual and distributed collaboration spaces can be enhanced by capturing and analysing dependencies amongst entities based on developer navigation cues that occur within these spaces. An empirical study carried out by Fritz et al. [FMH07] suggest that these navigation cues can be used to build a model of awareness about a codebase. The particular enhanced awareness attributes investigated in this work consist of: an accurate, real-time perception of the overall work effort of individual developers as well as their current work; an indication of which tasks and artefacts have consumed most effort over all developers; and a reasonable insight into the difficulty and error-proneness of different facets of a project, highlighting those facets that require more care. Navigation cues are built up as developers go about their daily development tasks using code artefacts, and leaving historical traces behind. A range of cues that can be obtained in a collaboration space have been presented by Gutwin et al. [GGR96]. These include the identity, location, actions, activity level, abilities, changes and sphere of influence of entities that exist in such collaboration spaces.

A ‘Continuum of Relevance Index’ (CRI) model based on developer interactions in a shared collaboration space is introduced. The basis of this model is the monitoring of core interactions such as project views, updates, creates and deletes. The model is used to provide relevance rankings that depend on the context of work being carried out by a developer. Rankings are then provided of tasks, developers and artefacts that exist in a shared collaboration space.

The main contribution of CRI stems from its provision of contextual relevance cues about the tasks, developers and artefacts that exist in a shared collaboration space. These cues are different for every work context. Thus a developer can be identified as being highly relevant to the current state of a particular task or artefact instance, but not in anyway relevant to the state of another task or artefact instance, though all such instances exist in the same shared collaboration space. Also, for any selected task instance, relevance ranking cues can be obtained on the developers and artefacts that have contributed to the task over time. Finally, for any selected artefact instance, a ranking cue can be obtained reflecting its relevance to
each task that exists in the collaboration space and a relevance ranking of developers that have worked with the artefact.

The remaining part of this chapter discusses the details of the CRI awareness model. The next session details the technique of articulating and representing a work context within a virtual distributed collaboration space. This is followed by a discussion on building an awareness perception of work effort that is dependent on interaction events that represent a selected work context. Building an awareness perception of work effort that is dependent on a varying sphere of influence that represents a selected work context is also discussed. An awareness model for the relevance ranking of entities representing overall and recent work effort over a selected work context is then presented. It is expected that this the perception of overall work effort can also be used to provide a reasonable insight into the difficulty and error-proneness of different facets of a shared collaboration project. Finally, this chapter presents a technique for building social network graphs from individual context graphs that exist within shared collaboration spaces.

3.2. Articulation and representation of a work context within virtual distributed collaboration space

Context can be understood from many perspectives, for instance the concept of context in linguistics is not necessarily the same as in human computer dialogue. Dey et al. [DSA01] described context as any information that can be used to characterise the situation of entities involved in achieving a job specification. This emphasises the notion of formulating context based on observable or readily discoverable attributes of elements (entities) in the environment. From a collaborative point of view, these attributes include the location and identity of collaborators and objects they interact with, the activity status of collaborators and their related objects, and the time period the interaction was carried out. In this thesis, the articulation and representation of work context is achieved by further exploring the notion of context as presented by Dey et al. from a relational perspective during collaborative software development.

To enable the articulation and representation of work context in this research, a subset of entities in a collaboration space is chosen to be explored. The choice of these entities has been informed by a review of related literature on the way contextual awareness about system models has been elucidated. Context based awareness has been examined using different formal representations such as UML use case and class diagrams, and informal representations such as task problem statements, test case definitions and bug reports. In each of these representation techniques, one of the objectives has been to present different
levels of understanding of software system models either at a more abstract level or at a
more granular level referencing the tangible entities that represent a system model. For
instance, a bug report can be associated with a set of code artefacts and the sequence of
actions carried out that resulted in the bug. The associated code artefacts and sequence of
actions define the context of the bug.

The view taken in this research is that during collaborative software development a number
of entities constitute the core components that define a shared system model. These include
the project that defines the objective of collaboration, the different tasks that are carried out
to achieve a project objective, the developers that execute these tasks, and the tangible
artefacts representing a project objective. Other entities such as project cost constraints do
exist in software collaboration spaces but are not investigated because it is not felt that they
can contribute to a shared understanding of a software project. The subset of entities
investigated in this research is formally defined as follows:

- A project is an endeavour embarked on to create a software product or service and
  serves to define the bound of the collaboration space.

- A task is viewed as a component of a software project abstract model representation that
  is required to be accomplished in order to achieve a software project. Tasks can be use
cases, user stories in agile processes, bug reports, etc. An interaction event is an action
  that results in the transformation of a task.

- Developers within the virtual collaborative software engineering environment are the
  individual team members that interact over a project. A developer can exact presence on
  a set of artefacts and tasks at the same time.

- Artefacts are project components such as software modules and documents that are
  manipulated by developers during interactions. These are tangible representations of the
  outcome of a software project.

In this research different forms of relations among this subset of identified entities are
enforced. A project relates to a set of tasks, developers and artefacts. A task relates to a set of
developers and artefacts, and a developer relates to a set of tasks and artefacts. Finally, an
artefact relates to a set of developers and tasks. No relations are enforced for entity instances
of the same type. Enforcing such same entity type relations is considered more complex and
beyond the scope of this research. Also, further analysis of the nature of defined tasks such
as their scheduling and dependency properties need to be carried out. Furthermore, insight
into the dependencies and expertise that exist among collaborating developers needs to be highlighted.

Every interaction event is associated with an active task, developer and artefact instance. While an event relates to an artefact directly, it is indirectly related to the task and developer that motivated the event. Furthermore, the subset of interactions associated with these entities explored in a collaboration space include create, update, view and delete events. Although, other events such as menu selection or button presses also comprise the sub-set of interactions that collaborators will normally carry out to achieve a software project, it is expected that these four events are the core interaction types that directly or indirectly determine the changing state of a software project. The following assertions are made about these core interaction events monitored in this research:

- ‘Create’ interaction events affect the state of an entity instance directly. This event is responsible for making the representation of artefact instances tangible within a collaboration space. This is also the first interaction event associated with an artefact and occurs once in the lifetime of an entity instance. Update, view and delete events can only occur on tangible entities.

- ‘Update’ interaction events also affect the state of an entity instance directly. This event is responsible for the different tangible state transitions an artefact evolves through from an initial to a final state. Update is measured based on the size of an update delta, where an update delta is the absolute difference in the number of characters associated with a code artefact before and after an update interaction event.

- ‘View’ interaction events indirectly affect the state of entity instances within a collaboration space. It is expected that a developers’ view of an artefact instance can enhance understanding used to further update the same artefact or other artefact instances that exist within a collaboration space. However, Zou and Godfrey [ZG06] has recorded cases of random view events that are irrelevant to on-going development work. Further insights into the identification of these random or spurious views can be carried out by measuring their duration. Here it can be assumed that a random view will last for a shorter duration compared to a relevant view event.

- ‘Delete’ interaction events directly affect the state of an entity within a collaboration space. This event is responsible for transforming an entity to an intangible state. Update, view and create events cannot occur on intangible entities. This interaction type occurs once in the lifetime of an entity instance, and is considered less
significant since developers will normally not always create artefacts to subsequently delete them.

During a collaborative software development project, different work contexts are formed that can be used to characterise the situation of entities in a collaboration space. This is as developers collaborate over different tasks using the same or different code artefacts by creating, updating or viewing them.

The constraint that every interaction event is associated with only one active task, developer and artefact instance, limits the number relations that a developer can establish with shared tasks over a single event. Thus, while a developer is carrying out an update, view or a create event on an artefact, it cannot be assumed that such an interaction event is being used to achieve more than one task. Furthermore, for a developer to be working on more than one task or artefacts requires that one task or artefact is active while the others remain inactive. This constraint reduces the computational complexity of an awareness model that determines the relative relevance multiple tasks will attain within the active developer’s work context.

The activation and deactivation of artefacts by developers during a collaboration process is not expected to be obtrusive since this is automatically carried out in the background as developers switch between artefact instances in the process of viewing, updating or creating them. Conversely, a level of obtrusion is expected while developers explicitly activate or deactivate the current task they are working on. This is so because such activation and deactivation of tasks can be considered as a secondary requirement during a software development process. Sections 5.3 and 6.7 each provide insight into the obtrusive effect that the activation and deactivation of tasks have on collaborating developers and the outcome of the CRI relevance ranking.

An entity instance can be associated with more than one interaction event at any given time. This also asserts that an entity instance can be active across a number of work contexts at the same time. This assertion corresponds to a scenario where multiple developers are synchronously collaborating over the same task or artefact; or an artefact instance is actively being used by multiple developers to accomplish different tasks at the same time.

To further establish this standpoint, a simple scenario of collaboration during a software development project is described below.

---

13 The task, developer and artefact associated with an interaction event are active, others not associated are inactive within a selected work context.
Scenario:
Assuming Bill, Amy and Ruben are members of a team collaborating to achieve an online cinema ticketing system called TickX. A number of tasks (use cases) is required to accomplish TickX including ‘Purchase Ticket’ and ‘Browse Movies’. A use case diagram for TickX is as shown in figure 3-1. Furthermore, a number of code artefacts are being developed to achieve TickX and include Ticket.java, Customer.java, Account.java, Booking.java, Movie.java, MovieCatalog.java and Cinema.java. A class diagram for TickX is as shown in figure 3-2.

![Figure 3-1 Use case diagram for TickX](image1)

While Amy and Bill have been collaborating to implement the ‘Purchase tickets’ use case task, Ruben has been responsible for the ‘Browse movies’ use case task. The following interaction trails were observed as these collaborators worked on their associated tasks:

- While Amy was collaborating on ‘Purchase tickets’ she created and updated Account.java and Customer.java code artefacts. She viewed and updated Booking.java a number of times. She also viewed MovieCatalog.java and Cinema.java.

- In the initial phase of Bill’s collaboration over the ‘Purchase tickets’ task, he viewed the Account.java and MovieCatalog.java code artefacts. This was subsequently followed by his creation and update of the Ticket.java and Booking.java code artefacts.

- Ruben’s execution of the ‘Browse movies’ task involved the creation and further updating of the MovieCatalog.java, Cinema.java and Movie.java code artefacts. Ruben also viewed Ticket.java a number of times.

The scenario presented above can be used to further represent a notion of work context from a relational perspective. This is achieved by deriving graphs based on the relational
properties of interaction events described in the monitored trails. Each interaction event from the trails of interactions related to an entity can contribute a node to a graph that represents its context - its ‘context graph’. If an interaction being processed refers to an entity instance not yet represented in the graph, a node for the instance is added to the graph. Assuming a delete event aimed at rendering a specified node intangible was subsequently carried out after the node instance had been added during a development process, then the node henceforth ceases to be associated with the context graph. Furthermore, the work context graph of the deleted node freezes and stops its subsequent association with interaction events. The history of interaction events associated with the deleted node is, however, retained in the log of event trails.

For example, the context graph of Amy will consist of every task and code artefact that she has created, updated or viewed. Similarly, the context graph of ‘Purchase tickets’ task will consist of every code artefact that was created, updated or viewed along with the developers that carried out the interaction event while actively working on ‘Purchase tickets’ task. Finally, the context graph of MovieCatalog.java will consist of every task and developer associated with the views, updates and create carried out on MovieCatalog.java. Figures 3-3, 3-4 and 3-5 illustrate the work context graphs of developers, tasks and code artefacts cited in the scenario presented above.

Figure 3-3 Developer work context graphs of example scenario
Thus, from a relational viewpoint, different representations of work context can be attained and represented in a work context graph. A comparison of the generated context graphs can be used to demonstrate near symmetric or symmetric relational properties that these graphs can achieve. Two work context graphs are symmetric if every node instance that exists in one also exists in the other. Conversely, two work context graphs are asymmetric if they do not have any node instances in common. Finally, two work context graphs are near symmetric if they have one or more node instances in common. Comparisons of two context graphs gradually move from near symmetric relation to symmetric as they increasingly have more nodes in common.

For instance, a comparison between Movie.java and Customer.java context graphs shown in figure 3-5 f and g is asymmetric since they have no node instance in common. A comparison between Ticket.java and Movie.java context graphs exhibits some near symmetric property since both artefacts have been worked on by Ruben and are also associated with the ‘Browse
movies’ task. Furthermore, Cinema.java and Ticket.java have more node instances in common and hence a comparison of their context graphs yield a more significant near symmetric measure relative to that between Ticket.java and Movie.java. Totally symmetric work context graphs are not attainable since entity instances that exist in a collaboration space are unique. Insight on how these symmetric properties of work context can be used in determining sphere of influence of entities is discussed in section 3.5.

The advantage provided by context graphs and their relative symmetric properties, is the alternative and unique representations of different work contexts that affect the state of entities in a collaboration space. Thus, the work context of a task collaborated over by many developers using different code artefacts to achieve the task, is shown to be different and unique from the work contexts of code artefacts and individual developers that have influenced the state of the task.

Context graphs also demonstrate crosscutting relationships that exist among entities in a collaboration space. This is shown in cases where a developer is related to more than a single task or artefact, an artefact is related to more than a single developer or task, and a task is associated with more than a single developer or artefact. For instance, it can easily be demonstrated that Amy has worked with MovieCatalog.java and Customer.java. Also, ‘Purchase ticket’ has been worked on by both Amy and Bill. Finally, while the MovieCatalog.java code artefact has been used to achieve ‘Purchase ticket’, it has also been used to achieve ‘Browse movies’ task.

Furthermore, while the ‘Purchase ticket’ task consists of the interaction events of both Amy and Bill while actively working on this task, context graphs of these entity instances are each unique and provide a different perspective of collaborative work. The context of the ‘Purchase ticket’ task consists of the work effort of both Amy and Bill and the code artefacts they have worked with while executing ‘Purchase ticket’. Conversely, the context of Amy and Bill consist of their individual work effort while executing ‘Purchase ticket’.

Each entity that comprises a selected work context attains different levels of relevance to that context. This is so because the amount of work effort, difficulty, or bugs associated with each entity within a selected work context is not necessarily evenly distributed. For instance, although Amy has created and updated Account.java and Customer.java, difficulty or bugs experienced while she was working on these two artefacts are not necessarily the same. Also, the amount of work effort that Amy has put into the ‘Purchase ticket’ task is likely to be different from the effort Bill put into the same task. The aim of this research is to enhance
awareness among distributed software developers by capturing and presenting these different levels of relevance attained by entities in a selected work context. This is based on insights from monitored interaction trails that are generated in a shared collaboration space.

By analysing the interaction trails of the scenario presented above a number of insights can be obtained. For instance the trails illustrate that Cinema.java and Customer.java were part of the ‘Purchase ticket’ work context under different circumstances. While Cinema.java code artefact was part of the ‘Purchase ticket’ work context because it was viewed by Amy and Bill while executing the task, Customer.java was part of the ‘Purchase ticket’ work context because it was created and further updated by Amy while executing the ‘Purchase ticket’ task. Also, the number of entities in the work context of ‘Purchase ticket’ at the time Customer.java was created and updated by Amy is likely to be different from the number of entities in the work context of ‘Purchase ticket’ at the time she viewed and updated Booking.java. Furthermore, the number of entity instances in the collaboration space is expected to increase as the lifetime of a shared project extends. Finally, interactions in a collaboration space are carried out at different times.

This research investigates enhancing awareness of the different levels of relevance entities attain within a selected work context based on three main constructs. These are the type of interaction event, the sphere of influence of the selected entity context, and the time of interaction.

3.3. Building an awareness perception of work effort that is dependent on interaction events that represent a selected entity work context

As hypothesised in section 3.2 the properties of interaction events associated with a selected work context can be used to predict a perception of the level of relevance that will be associated with entities in that work context. In this section, further insight into how this research uses the properties of interactions associated with a selected work context to obtain a relevance ranking of entities consisting the work context is presented.

To further investigate the properties of interaction events, and their weighted influence on the relevance of entities in a collaboration space, a study of CVS records associated with real development projects was performed. These records were derived from a group project software engineering class at the University of Strathclyde and open source Eclipse IDE technology and tools projects. CVS repositories of 200 artefacts from a combination of the
Eclipse Communication Framework (ECF)\textsuperscript{14}, Dash\textsuperscript{15}, Mylar\textsuperscript{16}, Equinox\textsuperscript{17}, and Eclipse Modelling Framework (EMF)\textsuperscript{18} open source projects were analysed. Only artefact check-ins with version repositories associated with more than one project member were considered.

The results showed that for the CVS checked in versions that were analysed, collaborators associated with the first artefact checked in were also associated with 49.6% of subsequent checked in versions. This result implies that, assuming the collaborator associated with the first checked in version is the artefact creator, in these cases, the creator of an artefact is associated with 49.6% of subsequent updates. From a collaborative relevance standpoint, this strongly suggests that granular interaction types that have a direct effect on the state of entity instances, such as create and update, can be used to derive relevance orderings.

The outcome of analysing CVS records does suggest that although a create interaction event occurs once in the lifetime of an entity instance, the creator of an entity, in a significant number of cases (49.6%), is subsequently associated with the greater number of further interaction types. It is assert that this makes the create event particularly important relative to other interaction types.

Furthermore, while it is expected that view events can enhance understanding of project related processes, studies conducted by Zou and Godfrey [ZG06] also suggested that cases of random view events that are irrelevant to an on-going development work can occur. In weighting the influence of view events on the relevance of entities in a collaboration space, it is important that the effects of such irregularities are inhibited.

Finally, this work does not associate a level of influence to delete interactions, expecting that they occur less frequently compared to other interaction event types. An evaluation of this assumption is presented in section 5.3.

Based on the insight obtained from these interaction types, initial influence-based weightings as shown in table 3-1 are assigned to each interaction event type. A view interaction event is equivalent to 10 units of absolute update delta\textsuperscript{19}, while a create interaction is equivalent to

\begin{itemize}
  \item \textsuperscript{14} \url{http://www.eclipse.org/ecf} (Accessed 02/2007)
  \item \textsuperscript{15} \url{http://www.eclipse.org/dash} (Accessed 02/2007)
  \item \textsuperscript{16} \url{http://www.eclipse.org/mylar} (Accessed 02/2007)
  \item \textsuperscript{17} \url{http://www.eclipse.org/equinox} (Accessed 02/2007)
  \item \textsuperscript{18} \url{http://www.eclipse.org/emf} (Accessed 02/2007)
  \item \textsuperscript{19} Absolute update delta is the positive or negative difference in the number of characters associated with a code artefact before and after an update interaction event.
\end{itemize}
100 units of absolute update delta. Related work by Fritz et al. has also suggested importance of create or authors of code artefacts. A similar approach in associating weights to interaction events has been used in the development of the Mylyn degree of interest model.

In Mylyn, a scaling factor of 1 was assigned to selection, propagation and prediction events. Similarly, factors of 0.7 and 0.017 were assigned to edit and decay events respectively. Detailed description of each of these events is as presented in table 2-2 of chapter 2. Each of these values was assigned based on usage statistics during the programming of Mylyn itself and validated based on feedback from other developer’s usage of the tool [Ker07].

<table>
<thead>
<tr>
<th>Interaction type</th>
<th>View</th>
<th>Update</th>
<th>Create</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weighting Factor</td>
<td>0.001</td>
<td>0.0001*Δ</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Δ - Absolute Update delta

The main concern at this stage of the research has been that poorly assigned weighting values can prevent developers in a distributed collaboration space from achieving the right perception of the relevance of an entity to a selected work context. At the same time, letting collaborating developers adjust weightings meant that the perception relevance of entities to a selected work context would be highly approximate, and would further complicate the usability of such an awareness tool. It is expected that these weightings will be further adjusted with insight from usage statistics of the implemented relevance model and after feedback from participants in an initial proof of concept study presented in chapter 5.

3.4. Building an awareness perception of work effort that is dependent on a varying sphere of influence that represents a selected entity work context

During collaborative software development, it is expected that the size of an entity’s work context or the number of other entities that an entity exacts its presence on, is proportional to the relative influence such an entity exacts on the collaboration space. For example, a developer in a collaboration space that has either created, updated or viewed a greater proportion of code artefacts within a shared project will be considered to hold more information about the state of the project compared to a developer that is newly introduced to the collaboration space and has only created, viewed or updated a few artefacts. Similarly, a task that has long existed in a collaboration space with developers implementing it using a number of artefacts is considered to hold more information about the state of the project compared to a task that is newly introduced into the collaboration space and is associated
with fewer developers and artefacts. Finally, the same analogy holds for an artefact that has been associated with a greater number of tasks and developers.

Similarly, it is also expected that the size of the selected work context that an entity is associated with is proportional to the relative influence such an entity exacts on the collaboration space. For example, a developer working on a task consisting of a number of other developers and code artefacts, will be considered to hold more information about the state of the shared project compared to a developer working on a task that consists of just a single developer and few code artefacts. Furthermore, a task that is associated with a developer that has worked on many other tasks and artefacts, is considered to hold more information about the state of a shared project compared to a task that is associated with a developer that has worked on fewer tasks and code artefacts.

It is necessary that within a shared collaboration space, a model that appropriately enhances awareness perception of the relevance of entities to a selected work context reflects this collaboration pattern in its encoding of interaction event trails.

The concept of sphere of influence is used to capture this collaboration pattern within the model of awareness during distributed software development. Sphere of influence is a general concept used to capture both geographic and semantic groupings. Spheres are used to provide well defined boundaries for interactions. Socially, individuals each have their own spheres – the locales they frequent, the organizations they associate with, and then set of friends, family, and acquaintances. The many relationships they participate in affect the others, often in subtle and unseen ways. Research carried out by Gutwin et al. [GGR96] on workspace awareness for groupware systems refers to sphere of influence as where collaborators can make changes within a shared artefact.

Sphere of influence in this research refers to a region over which an entity exacts some kind of usefulness. This usefulness stems from the interaction events an entity is associated with within a collaboration space. The sphere of influence of an entity is defined by its work context. The number of entities that comprises a selected work context is directly proportional to its sphere of influence. Figure 3-6 a, b and c are sphere of influence representations of developers, artefacts and tasks comprising the TickX project scenario presented in section 3.2. As shown in figures 3-6, the sphere of influence of Amy consists of all the tasks and code artefacts that she has interacted with. Similarly, the sphere of influence of Ticket.java consists of the tasks and developers that have influenced the state of the artefact. Finally, the sphere of influence of ‘Purchase ticket’ task consists of all the developers and artefacts involved in achieving the task.
Entities that comprise a defined sphere of influence can be characterised with overlapping properties. For instance as shown in figure 3-6a, Customer.java falls within only the sphere of influence of Amy, while MovieCatalog.java falls within the sphere of influence of Amy, Bill and Ruben respectively. Figure 3-6c illustrates that Cinema.java, MovieCatalog.java and Ticket.java fall within the sphere of influence of both the ‘Purchase ticket’ and ‘Browse movies’ tasks.

**Figure 3-6 Sphere of influence representations for entities consisting TickX collaboration project**

Sphere of influence ratio is used to represent the relative influence an entity exacts on the collaboration space. The sphere of influence of a task $t_m$, developer $d_y$ or artefact $a_z$ instances are rigorously defined as below:

If

$$T = \{t_1, t_2, t_3 \ldots t_x\}, \quad D = \{d_1, d_2, d_3 \ldots d_y\} \quad \text{and} \quad A = \{a_1, a_2, a_3 \ldots a_z\}$$

Where,

$T$ is the set of all possible tasks, $D$ is the set of all possible collaborating developers and $A$ is the set of all possible code artefacts.
And,

\[ t_{i\text{context}} = D'_{t_i} \cup A'_{t_i} \text{ where, } 1 \leq i \leq x \]

\[ d_{i\text{context}} = T'_{d_i} \cup A'_{d_i} \text{ where, } 1 \leq i \leq y \]

\[ a_{i\text{context}} = T'_{a_i} \cup D'_{a_i} \text{ where, } 1 \leq i \leq z \]

\[ D'_{t_i} \subseteq D, \ A'_{t_i} \subseteq A, \ T'_{d_i} \subseteq T, \ A'_{d_i} \subseteq A, \ T'_{a_i} \subseteq T, \ \text{and } D'_{a_i} \subseteq D \]

Where,

\( D'_{t_m} \) is the set of developers in the work context of \( t_m \),

\( A'_{t_m} \) is the set of artefacts in the work context of \( t_m \),

\( T'_{d_y} \) is the set of tasks in the work context of \( d_y \),

\( A'_{d_y} \) is the set of artefacts in the work context of \( d_y \),

\( T'_{a_z} \) is the set of tasks in the work context of \( a_z \),

\( D'_{a_z} \) is the set of developers in the work context of \( a_z \)

Then

\[ t_{i\text{sol}} = \frac{|t_{m\text{context}}|}{|D| + |A|} \iff D \neq \emptyset \land A \neq \emptyset \text{ where, } 1 \leq i \leq x \]

\[ d_{i\text{sol}} = \frac{|d_{y\text{context}}|}{|T| + |A|} \iff T \neq \emptyset \land A \neq \emptyset \text{ where, } 1 \leq i \leq y \]

\[ a_{i\text{sol}} = \frac{|a_{z\text{context}}|}{|T| + |D|} \iff D \neq \emptyset \land T \neq \emptyset \text{ where, } 1 \leq i \leq z \]

Where,

\( t_{m\text{sol}} \) is the sphere of influence ratio of task \( t_m \) and \( t_{m\text{sol}} \leq 1 \)

\( d_{y\text{sol}} \) is the sphere of influence ratio of developer \( d_y \) and \( d_{y\text{sol}} \leq 1 \)

\( a_{z\text{sol}} \) is the sphere of influence ratio of artefact \( a_z \) and \( a_{z\text{sol}} \leq 1 \)

More simply, the sphere of influence ratio of an entity instance \( E \) is given as:

Sphere of influence ratio of \( E \)

\[ = \frac{\text{number of entity instances that consists } E\text{'s work context}}{\text{number of entity instances in the collaboration space}} \]

The maximum sphere of influence ratio that \( E \) can attain is 1. This is for a case where \( E \) is associated with every other entity in the collaboration space. This is typical for scenarios where the collaboration space consists of a single artefact, task or developer. A minimum value of 0 is attained if the work context of \( E \) is an empty set; this is typical for scenarios where, for example, a developer in a collaboration space has not interacted with any task or artefact or a task has not been associated with any developer and hence artefact. On the
whole, as the number of entities that constitutes an entity’s work context relative to the number of entities in collaboration space increases, the sphere of influence ratio of the entity also increases.

Finally, given a collaboration space consisting of only a single developer $d_1$, task $t_1$ or artefact $a_1$, then a summary of the expected cases for the conditions for attaining the maximum sphere of influence are stated below:

\[
\begin{align*}
\cdot \quad d_1_{\text{soi}} = 1 \iff & \begin{cases} |D| = 1 \\ |A| \geq 1 \\ |T| \geq 1 \end{cases} \\
\cdot \quad a_1_{\text{soi}} = 1 \iff & \begin{cases} |D| \geq 1 \\ |A| = 1 \\ |T| \geq 1 \end{cases} \\
\cdot \quad t_1_{\text{soi}} = 1 \iff & \begin{cases} |D| \geq 1 \\ |A| \geq 1 \\ |T| = 1 \end{cases}
\end{align*}
\]

where $d_1 \in D$, where $a_1 \in A$, where $t_1 \in T$

3.5. Using sphere of influence as a factor in obtaining a notion of relevance of entities to a selected work context

The articulation of sphere of influence in this research leaves open two options from which it can be used to provide a notion of relevance of entities to a selected work context. These options are discussed below:

- The first option is based on the expectation that during collaborative software development, the size of an entity’s work context or the number of other entities that an entity exacts its presence on, is proportional to the relative influence such an entity exacts on the collaboration space.

This first option relates to the assertion that the number of other entities that an entity’s work context has been associated with is a factor in determining its relevance to other work contexts it subsequently gets associated with during a collaboration process. This is analogous to assuming that the number of tasks and developers an artefact has been associated with should constitute a determining factor in deriving the relevance of such an artefact to a task or developer work context that it subsequently gets associated with. Similarly, such an assertion would suggest that the number of artefacts and developers a task has been associated with should constitute a determining factor in deriving the relevance of the task to an artefact or developer work context that it subsequently gets associated with. Finally, the number of tasks and artefacts a developer has been associated with should constitute a determining factor in deriving the relevance of the developer to a task or artefact work context he/she subsequently gets associated with.

There are a number of challenges associated with this option. For instance, given that a developer has worked on a greater number of tasks associated with a project, it
cannot directly be assumed that this attribute should be factored into determining their relevance to a task they have never worked on. This is so because a developer associated with more tasks and artefacts might end up making little contribution to the status of a new task but is made more relevant to the task compared to other developers because he had been associated with greater number of other tasks and artefacts. Also, if a developer has worked on fewer tasks or artefacts, this does not directly necessitate that this should be a factor in determining the relevance position of the developer to another task work context that they are subsequently associated with. This is because such an assumption will be invalidated if the developer ends up putting more effort into the new task but is not reflected in the relevance ranking of developers associated with said task because he had previously worked with fewer tasks or artefacts.

This option will be less risky and appropriate for deriving a perception of relevance of entities to a selected work context for cases where it is possible to obtain the degree of similarity or variance between each of the diverse work contexts that exist in a collaboration space. This similarity or variance can be achieved by determining symmetric and near symmetric relational properties amongst context graphs previously discussed in section 3.2. Thus, given the degree of similarity or variance between the artefacts and tasks that a developer D has been associated with compared to the artefacts and other developers that have been associated with a task T which is subsequently worked on by D, one can then determine the relative impact that D’s sphere of influence would have on determining their relevance value to the work context of T. In general, the greater the similarity in entities between two different work contexts, the more impact the sphere of influence should have in determining the relevance value of the associated entity to the selected work context.

- The second option is based on the expectation that the size of the selected work context that an entity is associated with is proportional to the relative influence such an entity exacts on the collaboration space.

This second option relates to the assertion that the rate at which an entity gains relevance within an associated work context is also dependent on the sphere of influence ratio of the selected work context in a collaboration space. For instance, the rate at which an artefact gains relevance within an associated task work context will increase as the sphere of influence ratio of the task increases. Likewise for the relevance of a given developer to the work context of a selected task. Also, the rate at which a task gains relevance within an associated artefact work context will increase
as the sphere of influence ratio of the artefact increases. Likewise for the relevance of a given developer to the work context of the selected artefact. The rationale behind such reasoning is that a developer is more conversant with the status of a project and hence more relevant when working on a task or artefact that is being accomplished with more project resources compared to when working on a task or artefact that has been executed with fewer resources.

There are also risks associated with this option. For instance, this option assumes that if a developer is achieving a task by carrying out an update on an artefact that has also influenced the status of other tasks, then the developer has insight into which aspects of the artefact has also impacted on other tasks.

The relevance model discussed in this thesis implements the second option in deriving the notion of relevance of entities to a selected work context. While both options have their advantages and disadvantages, with a possibility of combining both, the choice of the second option is based on its simplicity of implementation. The complexity of the first option is based mainly on the requirement to accurately determine the degree of similarity or variance between each of the diverse work contexts that exist in a collaboration space.

3.6. An awareness model for the relevance ranking of entities representing overall and recent work effort over a selected work context in a collaboration space

The definition of development work context, interaction event trails associated with a selected entity work context, and the variation of its sphere of influence ratio forms the basis of our ‘Continuum of Relevance Index’ (CRI) model. This model is intended to provide an accurate, real-time perception of the overall work effort of individual developers as well as their recent work; an indication of which tasks and artefacts have consumed most effort over all developers; and a reasonable insight into the difficulty and error-proneness of different facets of a project.

CRI is a linear model that cumulatively builds the relevance values of entity instances as they are associated with interaction events and as their sphere of influence ratios vary. These cumulative relevance values are derived for two modes; history and recent. This section further discusses these modes and how they are used to generate a relevance ranking of entities within a selected work context.

**CRI History mode**

The history mode aims to provide real-time awareness perception of the overall dissipation of work effort across entities that constitute a selected task, developer and artefact work
context respectively. This mode cumulatively associates numeric values (relevance values) with entities within a selected work context that are representative of the overall work effort dissipated by each entity within the work context. These relevance values are cumulative because they are reprocessed for every interaction event that is carried out within the selected work context and incorporates previous values in this calculation.

Each time an interaction event occurs, the history mode computes the relevance values for all entities that constitute the work context of every active task, developer and artefact that defines the event. This is carried out by linearly combining the relevance value associated with an entity in a selected work context before an interaction event, with the relevance gained as a result of the interaction event. The relevance gained as a result of an interaction event is dependent on the type of interaction event and the sphere of influence ratio of the selected entity work context. Only entities which belong to the work context and are also defined in the current interaction event gain in relevance value. The entities belonging to the work context but are not defined in the current interaction event are considered inactive within the selected work context and hence their relevance values are unaffected.

The description of the operation of history mode is as follows:

In general, the work context of a task $t_m$ associated with a developer $d_1$ and artefacts $a_1$, $a_z$ is represented as:

$$t_{\text{context1}} = \{d_1, a_1, a_z\}$$

If given a subsequent event $e_n$ defined by developer $d_y$, task $t_m$ and artefact $a_z$, the following conclusions that impact on the work context $t_m$ and the relevance values of entities that consists the work context in the history mode can be made:

1. $t_{\text{context2}} = \{d_1, a_1, a_z, d_y\}$ where $d_y$ is a new developer introduced into the work context of $t_m$

2. The relevance value $x_{(n)d_y}$ gained by $d_y$ within $t_m$ work context is represented as follows:

$$x_{(n)d_y} = e_n \text{weight} \times t_m \text{soi}$$

$e_n \text{weight}$ - Weight associated with the type of interaction event as shown in table 3-1.

It is important to note here that $d_y$ is a newly introduced developer into the work context of $t_m$ as demonstrated by comparing the set $t_{\text{context1}}$ and $t_{\text{context2}}$. This means that $d_y$ had no previous relevance value to $t_m$ until event $e_n$ was carried.
out. This new relevance value is a factor of the type of interaction and the current sphere of influence ratio of task $t_m$.

3. It is also necessary to note that the sphere of influence ratio of task $t_m$ referenced as $t_{m \text{ soi}}$ was used to determine the relevance value $x_{(n)d_y}$ gained by $d_y$ within $t_m$ work context. This is opposed to using the sphere of influence ratio of $d_y$ in determining its relevance to $t_m$ work context as discussed in section 3.5. The relevance value of $d_4$ within the work context of $t_m$ remains unchanged given that the developer was not involved in the event $e_n$. That is:

$$x_{(n)d_4} = x_{(n-1)d_4}$$

Similarly, the relevance value of $a_1$ within the work context of $t_m$ remains unchanged. That is:

$$x_{(n)a_1} = x_{(n-1)a_1}$$

4. The relevance value $x_{(n)a_2}$ gained by $a_2$ within $t_m$ work context is represented as follows:

$$x_{(n)a_2} = x_{(n-1)a_2} + e_n \text{weight} \ast t_{m \text{ soi}}$$

The current relevance of $a_2$ is the addition of its previous relevance value and a new relevance value that is a function of the type of interaction and the current sphere of influence ratio of task $t_m$.

A more rigorous and formal description of history mode may be found in appendix A1.

**CRI recent mode**

The recent mode aims to provide a real-time awareness perception of the *current* dissipation of work effort across entities that constitute a selected task, developer and artefact work context. This mode cumulatively associates numeric values (relevance values) to entities within a selected work context that are representative of current development effort of entities constituting the work context.

Similar to history mode, relevance values attained by entities in the recent mode are cumulative within a selected work context. Each time an interaction event occurs, the recent mode computes the relevance values for all entities that comprise the work context of every active task, developer and artefact that defines the event. This is also carried out by linearly combining the relevance value associated with an entity in a selected work context before an interaction event with the relevance gained as a result of the interaction event. The relevance gained as a result of an interaction event is dependent on the type of interaction event and the
sphere of influence ratio of the selected entity work context. Only entities belonging to the work context and are also defined in the current interaction event gain in relevance value. Entities that belong to a work context but are not defined in a current interaction event are considered as inactive within the selected work context.

The core difference between history and recent mode is how the relevance values of inactive entities are computed. In the history mode, relevance values of inactive entities remain unaffected. While in the recent mode, relevance values of inactive entities context continue to decay for every interaction event that impacts the work context. Thus, the longer the duration of inactivity associated with an entity within a selected work context, the more the relevance of the inactive entity decays. This process of decay in relevance is represented using the notion of periodic decay and is dynamically determined by the sphere of influence ratio of the selected entity work context and interaction event type. Relevance lost due to the effect of periodic decay represents the negation of the relevance gained by the active entities defined in an event that impacts a selected work context. A related notion of modelling recent work is demonstrated in the development of Mylyn [Ker07].

The description of the operation of recent mode is as follows:

In general, the work context of a task \( t_m \) that consists of a developer \( d_1 \) and artefacts \( a_1, a_z \) is represented as:

\[
\text{tm}_{\text{context1}} = \{d_1, a_1, a_z\}
\]

If given a subsequent event \( e_n \) defined by developer \( d_y \), task \( t_m \) and artefact \( a_z \), the following impact on the work context \( t_m \) and the relevance values of entities that consists the work context in the recent mode can be observed:

1. \( t_m_{\text{context2}} = \{d_1, a_1, a_z, d_y\} \) where \( d_y \) is a new developer introduced into the work context of \( t_m \)
2. The relevance value \( x_{(n)d_y} \) gained by \( d_y \) within \( t_m \) work context is represented as follows:

\[
x_{(n)d_y} = e_n \text{weight} \times t_m \text{soi}
\]

The reasons discussed for the history mode are also applicable for \( d_y \), since \( d_y \) does not have a previous relevance value.

3. The relevance value of \( d_1 \) within the work context of \( t_m \) decays by a periodic decay factor which is determined by the weight associated with the type of event defined by \( e_n \) and the sphere of influence of \( t_m \). This is because \( d_1 \) is in the work context of \( t_m \) but not defined in event \( e_n \). That is:
Similarly, the relevance value \( x_{(n)}a_{i} \) of \( a_{i} \) within the work context of \( t_{m} \) will decay as shown below:

\[
x_{(n)}a_{i} = x_{(n-1)}a_{i} - e_{n} \text{ weight} \times t_{m} \text{ soi}
\]

4. The relevance value \( x_{(n)}a_{2} \) gained by \( a_{2} \) within \( t_{m} \) work context is represented as follows:

\[
x_{(n)}a_{2} = x_{(n-1)}a_{2} + e_{n} \text{ weight} \times t_{m} \text{ soi}
\]

As in the history mode, the current relevance of \( a_{2} \) is the addition of its previous relevance value and a new relevance value that is a function of the type of interaction and the current sphere of influence ratio of task \( t_{m} \).

A more rigorous and formal description of recent mode may be found in appendix A2.

**Relevance list of entities based on relevance value**

On the whole, the outcome generated by both history and recent mode is the association of numeric values to the relevance of entities that constitute a selected work context. A ranking of entities based on their relevance values within the work context forms the relevance list for that context. In such ranking, entities with the highest relevance values are positioned at the top of the list while entities with the lowest relevance value are positioned at the bottom of the list.

Using the relevance lists, collaborating developers can then obtain awareness of overall and recent work effort that has impacted the different work context of distributed entities bound by a software project in a collaboration space.

This section presents an example of the application of the CRI model in a project development scenario.

**Example:**

To illustrate how the CRI model can be used to obtain a perception of relevance of an entity instance to a selected work context it is assumed that the interaction trails shown in table 3-2 were the events used to achieve TickX, an online ticketing system presented in the collaboration scenario in section 3.2. Each row in table 3-2 corresponds to an event carried out to achieve TickX. Each event is associated with an interaction type, an active artefact, task and developer. The project started with the creation of Account.java code artefact by Amy while accomplishing the ‘Purchase ticket’ task, this is subsequently followed by her updating Account.java that generated an update delta size of 300. Furthermore, Ruben
created and subsequently updated MovieCatalog.java (update delta size of 650) while accomplishing ‘Browse movies’ task.

Table 3-2 Monitored interaction trails used to achieve TickX across 25 timelines

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>Interaction type</th>
<th>Artefact</th>
<th>Task</th>
<th>Developer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create</td>
<td>Account.java</td>
<td>Purchase ticket</td>
<td>Amy</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Update delta = 300</td>
<td>Account.java</td>
<td>Purchase ticket</td>
<td>Amy</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Create</td>
<td>MovieCatalog.java</td>
<td>Browse movies</td>
<td>Ruben</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Update delta = 650</td>
<td>MovieCatalog.java</td>
<td>Browse movies</td>
<td>Ruben</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Update delta = 150</td>
<td>Account.java</td>
<td>Purchase ticket</td>
<td>Amy</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Create</td>
<td>Cinema.java</td>
<td>Browse movies</td>
<td>Ruben</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Update delta = 50</td>
<td>Cinema.java</td>
<td>Browse movies</td>
<td>Ruben</td>
<td></td>
</tr>
<tr>
<td></td>
<td>View</td>
<td>Account.java</td>
<td>Purchase ticket</td>
<td>Bill</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>View</td>
<td>MovieCatalog.java</td>
<td>Purchase ticket</td>
<td>Bill</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Create</td>
<td>Movie.java</td>
<td>Browse movies</td>
<td>Ruben</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Update delta = 75</td>
<td>Movie.java</td>
<td>Browse movies</td>
<td>Ruben</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Create</td>
<td>Ticket.java</td>
<td>Purchase ticket</td>
<td>Bill</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Update delta = 175</td>
<td>Ticket.java</td>
<td>Purchase ticket</td>
<td>Bill</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>View</td>
<td>MovieCatalog.java</td>
<td>Purchase ticket</td>
<td>Bill</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Create</td>
<td>Booking.java</td>
<td>Purchase ticket</td>
<td>Bill</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Update delta = 70</td>
<td>Booking.java</td>
<td>Purchase ticket</td>
<td>Bill</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>View</td>
<td>Cinema.java</td>
<td>Purchase ticket</td>
<td>Amy</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>View</td>
<td>Booking.java</td>
<td>Purchase ticket</td>
<td>Amy</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Create</td>
<td>Customer.java</td>
<td>Purchase ticket</td>
<td>Amy</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Update delta = 84</td>
<td>Customer.java</td>
<td>Purchase ticket</td>
<td>Amy</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Update delta = 25</td>
<td>Booking.java</td>
<td>Purchase ticket</td>
<td>Amy</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Update delta = 10</td>
<td>Cinema.java</td>
<td>Browse movies</td>
<td>Ruben</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>View</td>
<td>Ticket.java</td>
<td>Browse movies</td>
<td>Ruben</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Update delta = 5</td>
<td>Movie.java</td>
<td>Browse movies</td>
<td>Ruben</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>View</td>
<td>MovieCatalog.java</td>
<td>Purchase ticket</td>
<td>Amy</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Update delta = 60</td>
<td>MovieCatalog.java</td>
<td>Purchase ticket</td>
<td>Amy</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Update delta = 90</td>
<td>MovieCatalog.java</td>
<td>Purchase ticket</td>
<td>Bill</td>
<td></td>
</tr>
</tbody>
</table>

The TickX project space assumes a collaboration environment where developers can work synchronously or asynchronously. Thus, a number of concurrent events can actively be executed at the same time. This is demonstrated in row 7, 21 and 25 of table 3-1. Ruben updated Cinema.java and Bill viewed Account.java at the same time in row 7. Row 21 demonstrates that both Ruben and Amy concurrently updated Cinema.java and Booking.java respectively. Row 25 demonstrates a scenario where Amy and Bill were actively working on MovieCatalog.java and Booking.java respectively at the same time to achieve ‘Purchase ticket’ task.

The application of the CRI model to each monitored interaction trail used to achieve TickX generates the relevance lists shown in figures 3-7 to 3-12. Further details on the interaction events and SOI ratio variation for a selected work context and accumulated relevance values are shown in appendix A3.
Figures 3-7 to 3-12 represent the history and recent mode relevance list outcomes for the entities that constitute the work context of ‘Purchase ticket’, ‘Browse movies’, Amy, Bill and MoviesCatalog.java. Entity instances with greater relevance values are positioned at the top of the relevance list. Also the relative differences in relevance values of entities are proportional to the relative distance between representations of instances in the list. Each list consists of entity instances of the same type that constitute a selected work context. For instance, figures 3-7a and b represent the relevance list of artefacts and developers that comprise the ‘Purchase ticket’ work context. Account.java is at the top of the list because it has attained the highest relevance value among artefact instances comprising ‘Purchase ticket’. Similarly, the relative difference in relevance value between Account.java and Booking.java is greater compared to the relative difference in cumulative relevance value between Booking.java and Ticket.java.

<table>
<thead>
<tr>
<th>History mode</th>
<th>b) Developers relevance list</th>
<th>Recent mode</th>
<th>d) Developers relevance list</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) Artefacts relevance list</td>
<td>Account.java</td>
<td>c) Artefacts relevance list</td>
<td>Booking.java</td>
</tr>
<tr>
<td></td>
<td>booking.java</td>
<td></td>
<td>Customer.java</td>
</tr>
<tr>
<td></td>
<td>Ticket.java</td>
<td></td>
<td>Account.java</td>
</tr>
<tr>
<td></td>
<td>Customer.java</td>
<td></td>
<td>Ticket.java</td>
</tr>
<tr>
<td></td>
<td>MovieCatalog.java</td>
<td></td>
<td>Cinema.java</td>
</tr>
<tr>
<td></td>
<td>Cinema.java</td>
<td></td>
<td>MovieCatalog.java</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Amy</td>
<td>Bill</td>
</tr>
</tbody>
</table>

Figure 3-7 History and recent mode artefact and developer relevance lists for the ‘Purchase ticket’ work context
The relative positions of entity instances on the history and recent mode relevance lists representing a selected work context can be used in attaining insight into overall and recent work effort dissipated over entity instances constituting the context. Figure 3-7 shows that within the ‘Purchase ticket’ task work context, Account.java has had greater overall influence on the state of the task, while Booking.java is associated with the most recent coding effort on ‘Purchase ticket’. Also, the figure demonstrates that Amy is attributed with both the greatest overall and recent coding effort in the task.

Furthermore, entity instances achieve different levels of relevance across the range of work contexts they are associated with. Although MovieCatalog.java has influenced the state of both the ‘Purchase ticket’ and ‘Browse movies’ tasks, its relative relevance to the two tasks is different. As shown in figures 3-7 and 3-8, while MovieCatalog.java is at the top of artefact relevance ranking for both the recent and history mode relevance lists representing ‘Browse movies’, the same artefact is lower in the artefact relevance lists representing ‘Purchase ticket’ in both modes. This feature is further represented in the relevance list of the MovieCatalog.java work context shown in figure 3-9.
### Figure 3-9 History and recent mode task and developer relevance lists for MovieCatalog.java work context

<table>
<thead>
<tr>
<th>History mode</th>
<th>Recent mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) Tasks relevance list</td>
<td>c) Tasks relevance list</td>
</tr>
<tr>
<td>Browse movies</td>
<td>Browse movies</td>
</tr>
<tr>
<td>Purchase ticket</td>
<td>Purchase ticket</td>
</tr>
<tr>
<td>Ruben</td>
<td>Ruben</td>
</tr>
<tr>
<td>Amy</td>
<td>Amy</td>
</tr>
<tr>
<td>Bill</td>
<td>Bill</td>
</tr>
</tbody>
</table>

### Figure 3-10 History and recent mode task and developer relevance lists for Amy’s work context

<table>
<thead>
<tr>
<th>History mode</th>
<th>Recent mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) Artefact relevance list</td>
<td>c) Artefacts relevance list</td>
</tr>
<tr>
<td>Account.java</td>
<td>Account.java</td>
</tr>
<tr>
<td>Customer.java</td>
<td>MovieCatalog.java</td>
</tr>
<tr>
<td>MovieCatalog.java</td>
<td>Customer.java</td>
</tr>
<tr>
<td>Booking.java</td>
<td>Booking.java</td>
</tr>
<tr>
<td>Cinema.java</td>
<td>Cinema.java</td>
</tr>
<tr>
<td>Purchase ticket</td>
<td>Purchase ticket</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Recent mode</th>
<th>d)Tasks relevance list</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purchase ticket</td>
<td>Purchase ticket</td>
</tr>
<tr>
<td>MovieCatalog.java</td>
<td>Booking.java</td>
</tr>
<tr>
<td>Customer.java</td>
<td>Cinema.java</td>
</tr>
</tbody>
</table>
Also, the relevance lists representing the work context of Bill and Amy show that although both collaborators have been collaborating over the same task their focus of work within the task is different. This is illustrated in the different artefact instances and their positions on the relevance list shown in the figures. While Amy’s overall and recent work effort is centred on Account.java, Bill’s work effort has been centred on Ticket.java and Booking.java.

Two evaluation studies are presented in this research and presented in chapters 5 and 6. One main purpose of these studies was to ascertain if there is reasonable correspondence between the developers’ perception of overall and recent work effort with the CRI relevance ranking. The second study was also aimed at exploring the level of correspondence between the history mode relevance ranking and a developer’s perception of difficulty and error-proneness. Furthermore, this study was carried out to obtain insight into the form of awareness information that developers can achieve using CRI relevance list representations of the different work contexts that constitute a collaboration space. The main advantage this model provides is that developers can acquire this context based awareness about their collaboration space irrespective of their distribution in time and space.

### 3.7. The CRI social graph

This section describes how a social network graph of entity instances can be attained by combining work context graphs of all entity instances that exist in a collaboration space. Thus, the network graph of a shared project consists of a combination of all context graphs...
of task, artefact and developer instances associated with the project. A centrality measure in Social Network Analysis (SNA), based on network efficiency, is then presented.

**Deriving social network graphs**

The basic context graphs shown in figures 3-3 to 3-5 are examples of a graph $G$ consisting of two sets of information: a set of nodes $E$ that each represent an entity instance represented as $E = \{e_1, e_2, \ldots, e_n\}$ (where $n$ is the number of entities in a context graph) and a set of arcs $L$ (lines where the direction is specified) that represent entity interactions, also represented as $L = \{l_1, l_2, \ldots, l_m\}$ (where $m$ is the number of lines in a context graph). Arcs are ordered pairs of distinct entities $l_k = \langle e_i, e_j \rangle$. The arc $\langle e_i, e_j \rangle$ is directed from $e_i$ (the entity that forms the perspective of work context) to $e_j$ (the entity that is relevant to the work context of $e_i$). Thus $(e_i, e_j) \neq (e_j, e_i)$. The entity $e_i$ is adjacent to $e_j$ if $\langle e_i, e_j \rangle \in L$. Finally, an entity is *incident* with an arc if the entity is one of the ordered pairs of entities defining the line, and the *degree* of an entity is the number of lines that are incident with it [WF07].

The attributes of each arc in $L$ can further be specified using the CRI value of entities in context to determine their strength or intensity. This results in an additional set of values $V = \{v_1, v_2, \ldots, v_m\}$ in $G$. Thus, a CRI valued context graph can be denoted by $G(E, L, V)$.

$G$ is a 3-partite graph since its entities $E$ can be partitioned into three subsets $E_t$, $E_s$ and $E_a$ that each represent the set of tasks, developer and artefact instances that form the graph. All lines are between entities in different subsets and no entities in the same subset are adjacent. For example, there is no direct relationship between two task instances $t_1$ and $t_2$, and they can only be indirectly related if an artefact or a developer instance is related to both $t_1$ and $t_2$, thus creating an indirect path from $t_1$ and $t_2$. Figure 3-12 shows an indirect relationship between $t_1$ and $t_2$ via $a_1$ each belonging to different subsets but related by $a_1$. The arc label $v_{a_1t_1}$ represents the relevance value of $t_1$ to the work context of $a_1$. Conversely, $v_{t_1a_1}$ represents the relevance value of $a_1$ to the work context of $t_1$.

![Figure 3-12 A 3-partite context graph](image)

The total number of context graphs in a collaboration space depends on the unique number of tasks, artefacts and developers that are committed to a shared project. Furthermore, a task
instance cuts across a number of artefact and developer context graphs, an artefact instance cuts across a number of task and developer context graphs, and a developer instance cuts across a number of task and artefact context graphs. This produces a complex network that presents another level of abstraction that combines the results of the CRI task, artefact and developer perspectives. A typical example of such a network for TickX project is as shown in figure 3-13. The weights shown on the graph are history mode entity relevance values for the different work contexts in TickX project collaboration space. The network graph demonstrates that the relevance value of Amy to Customer.java work context is 0.0074 while the relevance value of Customer.java to Amy work context is 0.0102. The same explanation holds for every other relation amongst entities in the graph.

The CRI valued work context graph of any entity $e$ can be represented by $G_e (E, L, V)$. The combination of every task, developer and artefact work context graphs in a collaboration space denoted as $G_T, G_A$ and $G_D$ respectively can be represented as follows:

$$G_T = \{G_{t_1, \text{context}} \cup G_{t_2, \text{context}} \ldots \cup G_{t_m, \text{context}}\} \quad \text{Where } G_{t_m, \text{context}} \text{ represents the context graph of task } t_m.$$  
$$G_A = \{G_{a_1, \text{context}} \cup G_{a_2, \text{context}} \ldots \cup G_{a_n, \text{context}}\} \quad \text{Where } G_{a_n, \text{context}} \text{ represents the context graph of artefact } a_n.$$  
$$G_D = \{G_{d_1, \text{context}} \cup G_{d_2, \text{context}} \ldots \cup G_{d_y, \text{context}}\} \quad \text{Where } G_{d_y, \text{context}} \text{ represents the context graph of developer } d_y.$$
And m, n and y represent the total number of tasks, artefacts and developers respectively in the collaboration space.

Finally,

\[ G_e = \{G_T \cup G_A \cup G_F\} \]

**Centrality measure of entities in CRI networks**

This section employs SNA measures to enhance the visualisation and usefulness of CRI network graphs during distributed collaborative software development. In SNA, centrality indices are normally used to convey an intuitive feeling that in most networks some vertices or edges are more central than others [KLP+05, LM07]. A typical example of a centrality index which suits the CRI network graph definition is the Markov centrality. This is because Markov centrality can be applied to directed and weighted graphs. This centrality algorithm has also been provided within the JUNG framework\(^{20}\) which was used in the CRI implementation architecture discussed in the next chapter.

To attain the centrality of entities in a collaboration space, this research models a CRI valued network graph as a Markov chain. White and Smyth [WS03] described a Markov chain as a single ‘token’ traversing a graph in a stochastic manner for an infinitely long time, and the next node (state) that the token moves to is a stochastic function of the properties of the current node. White and Smyth also interpreted the fraction of time that the token spends at any single node as being proportional to an estimate of the global importance or centrality of the node relative to all other nodes in the graph.

The modelling of a CRI valued network graph as a Markov chain so as to determine the centrality of entities in a collaboration space lies in the ability to derive a transition matrix from the CRI valued network graph. This is achieved by assuming that the likelihood of a token traversal between two nodes is proportional to the relevance weight associated with the arc linking the nodes. Thus, higher transition probabilities are associated with arcs with higher relevance values. The weights on a network graph are then converted to transition probability weights by normalising the relevance value of an entity associated with a work context to one. The transition probability is dependent on the relevance value of an entity and the total number of entities associated with the work context.

Figures 3-14 and 3-15 both represent the transition diagram and transition matrix derived from the CRI network graph of TickX project previously shown in figure 13-3. Figure 3-14

\(^{20}\) http://jung.sourceforge.net/
shows that the transition probability of a token from Ticket to Browse movies task is 0.0339 while the reverse of a token traversal from Browse movie to Ticket is 0.0044. Similarly, the transition probability of a token traversal from Ticket to Purchase ticket task is 0.4661. These probabilities are represented in the transition matrix as shown in figure 3-15. It is important to note that each of the rows in the transition matrix sums to one. Finally, this transition matrix is then the input into the JUNG graph framework. JUNG generates the centrality values of each entity and also a network graph in which the size of each entity is proportional to its Markov centrality.

Figure 3-14 TickX project space transition diagram
Figure 3-15 TickX project space transition matrix

Figure 3-16 is a graphical representation of a CRI social network graph for TickX project where the size of each entity is proportional to its Markov centrality in the network. This figure shows the relatively higher centrality that MovieCatalog.java has achieved in the collaboration space. This centrality is based on the relevance value that MovieCatalog.java has attained and the number of entities it is associated with. The advantage of such a visualisation is that it is based on indirect links amongst entities and hence generalised and void of contextual properties.

Figure 3-16 A Social graph based on centrality of entities in a virtual collaboration space.

A potential social cue that can be obtained from such a social graph is the ability to easily determine bottleneck entity instances in a collaboration space. For instance deleting artefacts, updating tasks descriptions, or removing developers that are all attributed with high information centrality in a shared project collaboration space could be detrimental to the information/knowledge flow for that project.
3.8. Conclusion

This chapter has presented a CRI model used in deriving relevance cues of entities associated with a selected work context. The relevance cues provided by CRI are built by first identifying a subset of entities that exist in a collaboration space. The relevance of these entities is then modelled based on three main concepts that include the type of interaction events they are associated with; the ‘sphere of influence’ of the selected work context; and, the relative time the interaction takes place in the project cycle represented using the notion of ‘periodic decay’.

Different development work contexts and the relative importance of entities to these contexts are built using combinations of these concepts. The relevance rankings derived from the model are then extracted to provide different forms of visualisations in a shared collaboration space to provide awareness of activities that are constantly changing the state of a team project. Two main forms of visualisations have been demonstrated in this chapter. They include the CRI relevance lists and social network graphs. These visualisations are intended to provide an accurate, real-time awareness perception to developers distributed in time and / or space. Such awareness perceptions include insight into the overall work effort of individual developers as well as their recent work; an indication of which tasks and artefacts have consumed most effort over all developers; and an insight into the difficulty and error-proneness of different facets of a project. The next chapter presents the implementation framework of CRI model in a distributed, collaboration space.
Chapter 4 Implementation

4.1. Introduction
This chapter presents a general overview of the CRI architecture (section 4.2) and the core components that comprise the different layers of the CRI architecture (section 4.3).

4.2. Overview of the CRI architecture
There are a few key challenges associated with implementing CRI. The first stems from the complexity of extending the generic nature of the model into a mechanism that understands a programming domain structure (for example Java and C++) and an integrated development environment for working with such a domain structure. This mechanism should be able to monitor and retain a trail of view, update and create interaction events from every collaborating developer’s individual workstation within the collaboration space. Furthermore, the trails of events need to be continuously mapped to an abstract representation of the work context for entities identified in the trails.

In this research a choice of implementing CRI using Java as a programming domain and Eclipse as the integrated development environment was made. Eclipse is optimised for working with Java domain structures. This choice was also based on the insight that all the advanced software engineering students from the Department of Computer and Information Sciences were familiar with programming in Java using Eclipse IDE. The participants in the study carried out to validate CRI came from this class. The main advantage of using Eclipse is that it provides a mechanism for seamlessly integrating external plug-ins into the development platform. While other existing IDEs such as Netbeans and Visual Studio also provide extensions from which the core interaction events of interest in CRI can be monitored, achieving this would require a reimplementation for these environments.

Secondly, the collaborative nature of CRI means that the formation of an entity’s work context is not from a single developer’s workstation or interaction perspective but an integration of all interactions from every developers’ workstation that exist in a collaboration space. For instance, the work context of a shared task cannot be determined from a single developer point of view or by using only the artefacts he/she has used while actively working on the task; such a task work context will rather be determined based on every developer involved, and artefacts used, in achieving the task.
On a whole, the formation of a work context, and hence associated relevance lists of shared entity instances, are generated based on events gathered from all developers, shared tasks and artefacts associated with a project. This factor makes the implementation of CRI also dependent on the interconnection among different computer networks. A realistic implementation needs to be robust enough to reasonably cope with pragmatic scenarios where computer networks are slow or nonexistent.

The CRI tool can be viewed as implementing a client-server architecture in which every developer’s Eclipse IDE workstation is a client while the model processing logic and interaction events trails storage is carried out on the server. The client monitors each trail of view, update, create and delete interaction events executed within Eclipse. When a network connection exists, the monitored developer interaction events data is offloaded to the server that in turn synchronises the data with that of other collaborating developers. For cases where a network connection does not exist or connection is slow, the client temporarily stores monitored event trails locally. For such a scenario, then the client acts also as the server by implementing model processing logic. The server also stores the data and provides feedback to the client on processed relevance value of entity instances related to a developer selected work context.

The data messaging between a developer’s Eclipse IDE and the server is carried out synchronously and asynchronously. Generally, in synchronous messaging architectures, the client and server are coupled in time because resources are dedicated in one party while the other party and/or network are processing the message. In asynchronous messaging, the coupling in time is relaxed but adds a further coupling in space because the state of the system is spread across the client and server. To re-use the resources on the client, the session attributes of the client must somehow be serialised, so that it can be reified when the response comes back. Figure 4-1 is an overview of CRI implementation architecture.

In this research, asynchronous data messaging is used to transfer interaction event data from the Eclipse IDE client end to the server. This is achieved using the Apache ActiveMQ message broker framework\(^\text{21}\). Given that resources are decoupled and not explicitly dedicated by the client during event data transfer over the network and processing by the server, the monitoring process is more scalable with improved performance. This makes the capturing of event data from collaborating developers less distracting and more seamless.

\(^{21}\) \url{http://activemq.apache.org/} (verified 20/08/2008)
A request for the relevant entities to a selected work context explicitly made by a developer via Eclipse is fulfilled using synchronous messaging. This is achieved using the Apache Axis SOAP web services framework\textsuperscript{22}. Experience in attempting to implement this process asynchronously was problematic and challenging. The problem occurs when a developer explicitly makes a request for relevant entities within a selected work context while generating and transferring interaction event data at the same time. This resulted in the corruption of monitored event data and inaccurate relevance values associated with entities associated with the affected work context.

![Diagram of CRI implementation architecture]

**Figure 4-1 An overview of the CRI implementation architecture**

### 4.3. CRI tool integration

The CRI tool is an integration of an eclipse plug-in client end that represents the different collaborating developers’ workstation and a server end responsible for the model processing logic and archiving of interaction event trails. This section details the four core layers that spread across the client and server ends as shown in figure 4-2. This includes RCP, messaging, event and model layers\textsuperscript{23}. The client section of each layer is plugged into the Eclipse platform while the server section resides on an Apache Tomcat web application server.

\textsuperscript{22} \url{http://ws.apache.org/axis/} (verified 20/08/2008)

\textsuperscript{23} CRI version 1.2 implementation is made up of 207,351 Test Lines of Code (TLOC and is computed as non-blank, non-comment test lines of code), 469 classes, and 29 packages.
**RCP layer**

The RCP (Rich Client Platform) is the minimal set of components needed to build a rich client application within the Eclipse platform. This layer consists of the CRI visualisation and editor components. Both reside only on the client end of the CRI tool. Explicit request/response messages from this layer to the server are processed synchronously via the messaging client shown in the messaging layer of figure 4-2.

The visualisation.rpc component is responsible for processing requests explicitly made by a developer including logging into CRI to enable monitoring; creation, activation or deactivation of a collaboration task; switching between history and recent model processing modes; and switching between offline and online modes during collaboration. Furthermore, the editor.rpc component enables developers to make requests related to updating their identity within the collaboration space and the attributes of collaboration tasks. Figure 4-3 and 4-4 are snapshots of Eclipse views and editors generated by visualisation and editor components.
1. Upload offline interaction events
2. Switch offline/online mode
3. Switch history/recent mode
4. Generate social network graph
5. Collaboration group attributes
6. Active collaborator
7. Active task
8. Pop up menu generated with a right click on selected task

Figure 4-3 CRI Eclipse view

The visualisation.rpc component is also responsible for making a request to the model layer via the messaging client (figure 4-2) for entities and their respective relevance values associated with a selected work context. These relevance values are then used in generating and presenting the social graph and relevance lists associated with the selected entity work context. As previously demonstrated in the model description (chapter 3), entity instances with greater relevance values are positioned at the top of the relevance list. The relative difference in the relevance values of entities is proportional to the relative distance between entities on the relevance list and is depicted using varying colour intensity. Entities at the top of the relevance list are represented with greater colour intensity while entities at the bottom of the relevance list are depicted with less colour intensity. Closely related entities show the same relative colour intensity. Figure 4-5 shows the history and recent mode artefacts relevance list for the ‘Purchase ticket’ task previously presented in the example TickX project scenario in chapter 3.

Figure 4-4 CRI task and group properties within the Eclipse editor

Social graph visualisations are generated using JUNG (Java Universal Network/Graph Framework). The JUNG framework is used to support visualisation of data that can be
represented as a graph or network. The size of each entity in a social graph is determined by its Markov centrality within the social graph.

Further details on the views and editors generated by the RCP layer is demonstrated in the CRI tool quick user guide shown in appendix B1.

Figure 4-5 History and recent mode artefacts relevance list for ‘Purchase ticket’ work context

**Messaging layer**

This layer is responsible for asynchronous and synchronous data communication processes between the client and server. Components of this layer reside in both the client and server end of CRI tool. The client.messaging and server.messaging components act as a hub where generated interaction events and requests/responses related to the relevance of entity instances to a selected work context are transmitted.

This layer also constitutes the offline.sync and collaboration.sync components. The CRI tool enables developers to work offline and still generate a trail of interaction events which they can upload to the server for resynchronisation with events generated by other developers within the collaboration group. The offline.sync component is responsible for temporary storage and subsequent uploading of offline interaction events while collaboration.sync is responsible for synchronisation of such offline events with events generated by other collaborators based on a timestamp identifying each interaction event.

The offline.emulator emulates the server end functions of the model and event layers while a developer is generating interaction events in the offline mode. This component keeps a record of the latest work context and relevance status of entities in the collaboration space while the developer was in online mode. This status is subsequently used to locally update the context and relevance state of entities that are associated with the developer’s interaction events. This component enables the use of the CRI tool by individual developers who either
have no access to network connections or are working on personal non-collaborating projects.

Event layer

This layer is responsible for the capturing and archiving of interaction event trails generated by every developer within a collaboration space. The layer consists of three main components including monitor.event and registrar.event that reside at the client end while log.event resides at the server end.

The monitor.event component is responsible for monitoring every code artefact resource viewed, created, deleted and updated by a developer within the Eclipse IDE. For an update event, the monitor.event component further generates the associated absolute update delta and other attributes such as the statement line number, the method, and class where the update was carried out. This captured event is then passed over to the registrar.event component that associates it with other essential attributes. These attributes include a timestamp identifying when the event occurred, the identity of the active developer instance at the time the event occurred. The event.registrar then transfers the event object to the log.event component via the messaging layer (figure 4-2). The transfer process is carried out asynchronously. The log.event component is the clearing centre and data warehouse of all events generated within a collaboration space. Each event is archived as XML and parsed using DOM and SAX parsers. Figure 5-6 is an example of a typical XML representation of CRI event.

```
<event
  collaborationId="6XX65798295"
  collaborationName="null"
  creationTime="1553435796"
  creator="null"
  eventId="8972431524"
  modificationTime="1553435796"
  name="null"
  rootId="null"
  rootName="null"
  sessionId="null"
  type="null"
  version="null">
  <projectArtifact>
    <projectArtifact type="null">
      <projectArtifact type="null"/>
      <projectArtifact type="null"/>
      <projectArtifact type="null"/>
      <projectArtifact type="null"/>
    </projectArtifact>
  </projectArtifact>
</event>
```

Figure 4-6 XML representation of a CRI event

Model layer

This layer is considered as the main event processing unit of the CRI tool. It consists of three main components that all reside in the server end of the CRI tool. The context.model component is responsible for keeping track of the formation of an entity work context and its
related sphere of influence ratio. The recent.model and history.model components are responsible for generating the relevance values of entities for a selected entity work context for CRI recent and history modes respectively.

4.4. Conclusion
This chapter presents a general overview of the CRI the tool that was implemented as an Eclipse plug-in client and a web application server end. Data communication between the client and server ends was enabled synchronously and asynchronously using Apache Axis web service and ActiveMQ framework respectively.

The CRI architecture consists of four core layers. They include the RCP layer responsible for generation of CRI related views and editors; the messaging layer that acts as a hub between the client and server ends of the CRI architecture; the event layer that is responsible for monitoring and archiving of interaction events carried out by collaborating developers; and the model layer that generates and keeps track of the formation of an entity’s work context and the relevance of entities comprising such a context in the history and recent modes.
Chapter 5  Study 1- Initial proof of concept study

5.1. Introduction
An initial proof of concept study was required to ascertain the feasibility and also understand the challenges and risks involved in achieving a relatively complex model such as CRI in practice. This initial study aimed to provide insight into four main research questions that enable the feasibility of achieving CRI to be judged. These are:

1. Is it possible to capture and integrate interaction data for developers working without the limitations of time and space?

The complexity of achieving such a model is because of its distributed nature. Thus, developers are not restricted in time and place of work. An efficient implementation will require that the system can capture and integrate developer interaction data in a seamless manner from their different individual workstations, and also provide relevance list or social graph visualisation feedback that depends on the context of work of collaborating developers. Also, the model is based on building a collaborative context in parallel, compared to building context by token passing as demonstrated in Mylyn [KM06] where individual context is built and subsequently shared with other developers to build upon. Such a system requires that developers’ IDE workstations are, aware in real time of the project tasks and artefacts that are being created, updated, and viewed or are affecting the state of shared projects in a collaboration space.

Such an awareness system should be able to provide an accurate, real-time perception of the overall work effort of individual developers as well as their current work; an indication of which tasks and artefacts have consumed most effort over all developers; and a reasonable insight into the difficulty and error-proneness of different facets of a project, highlighting those facets that require more care. Such awareness is built using a combination of the core interactions types studied in this work, the sphere of influence of entities involved in interactions and the notion of time based on periodic decay of inactive entities. The first aim of this proof of concept study is to investigate and understand the viability and challenges of building such an awareness system for collaborating software developers that are distributed in time and/or space.

2. How do the formal constraints on task definition, activation and deactivation affect developer collaboration?
An exploratory research study on global software development projects has indicated that a well-defined task structure influences positively the efficiency, effectiveness and satisfaction level of global virtual teams [ES03]. CRI imposes a further level of restriction on the manner developers will normally develop systems. For instance CRI requires that developers explicitly think of the task, and formally activate or deactivate such tasks as their work context changes. The second aim of this proof of concept study is to understand the benefits as well as the difficulty and challenges that developers will encounter when they have to work within such formal constraints on task activation and deactivation.

3. What is the validity of the assumptions used in building CRI relevance rankings of entities in a collaboration space?

Furthermore, CRI is built on a number of assumptions that have influenced the weighting factors that are assigned to interaction types shown in table 3-1. These core assumptions include:

1. “Create” interaction types occur once in the lifetime of an entity instance. Related work by Fritz et al. [FMH07] on how programmer activity indicate knowledge of code had pointed out that the developers are likely to have more knowledge about program artefacts they created or authored. Thus, although create occurs once, this does not make this interaction type less important than other types. For instance, it is assumed that if a developer $D1$ creates a code artefact $A$ while executing task $T1$, it is more likely that a substantial proportion of subsequent updates and views on $A$ will be associated with $D1$. In order to capture this assumption in deriving the amount of code effort $D1$ has on $A$, the create interaction type is considered more significant compared to other interactions that affect the state of $A$.

This assumption does not invalidate monitoring other interaction types. For example if other developers say $D2$, $D3$ and $D4$ have updated or viewed $A$ while also executing $T1$, or another task say $T2$ or $T3$, it is also important to know the amount of work effort that that $D2$ has put into $A$ relative to $D1$, $D3$ and $D4$. It is also important to know which of $T1$, $T2$ or $T3$ has had more impact on the state of $A$.

2. The relevance of an “Update” interaction type can be measured based on the size of the update delta. This is because the update delta can be assumed to be a direct representation of the amount of effort expended on a piece of code.

3. “View” interaction types are considered less significant than create and update types. This is because development processes can sometimes be associated with random views
that are irrelevant to the context of selected work. Also, views are considered to be a passive interaction type and contribute less to changing the state of a system compared to update and create interaction types. However, the more an entity instance is viewed within a work context, the more relevant such an entity is to the work that is being carried out.

4. “Delete” interaction types occur once in the lifetime of an entity instance, and occur less frequently than the other types. Delete interaction types have a null effect and thus maintain the present state of relevance of related entity instances.

The third aim of this proof of concept study is to validate and further understand the implications of the basic assumptions underlying the relative weighting factors given to each interaction type. This study addresses the following research questions:

- Is the creator of an artefact more likely to be associated with a greater proportion of subsequent updates and views on the same artefact?
- How high is the frequency of view interactions compared to updates and deletes?

4. Can the CRI relevance list visualisation be used to provide a measure of work effort dissipated across a collaboration space?

Finally, the implementation of the CRI tool uses varying colour intensity to represent the different CRI values of entities in each selected perspective of the relevance list visualisation. Cumulative CRI values are mapped to different colour intensities. Increasing intensity associated with an entity depicts its increasing relevance. The relevance of entities is ranked in a decreasing order with the most relevant at the top and associated with highest intensity. The fourth aim of this proof of concept study is to generate usage statistics of how the colour intensity matched with developer perception of the collaboration space. Such usage statistics can then be used for fine tuning of the interaction type weighting factors used in the model and hence the intensity of colours used to represent the relevance of entities. The initial interaction type factors used during this proof of concept study are as shown in table 3-1.

The remainder of this section presents the proof of concept study that provided insight into these four questions.
5.2. Study Format

This study involved nine advanced software engineering students from the department of Computer and Information Sciences, University of Strathclyde. The study was carried out in the spring of 2007 and lasted for a period of four weeks, during which they were required to implement, test and demonstrate a program that plays ‘Gizmoball’. The size of a completed gizmoball project is typically 50-100 java classes.

Gizmoball is an editor for pinball simulations in which the objective is to keep a ball moving around in the game, without falling off the bottom of the playing area. Gizmoball allows users to construct their own machine layout by placing gizmos (such as bumpers, flippers, and absorbers) on the playing field\textsuperscript{24} \textsuperscript{25}.

Every developer that was involved in this study belonged to a group of three members. Participants were required to log into the CRI tool each time they worked on the project within the Eclipse IDE. They then either activated a shared task of interest that existed in their collaboration space or created a new one. Tasks include use case definitions, identified bugs and or any other set of coherent activities that are performed within the collaborative space to achieve Gizmoball requirements. Participants were allowed to work from anywhere and at any time.

CRI then monitored and kept records of the core interaction type of interest (views, updates, adds, and deletes) and presented relevance information depending on the selected task, artefact or developer representation. Each entity relevance ranking presented was calculated based on a linear combination of associated gain factors resulting from its interaction context as well as its sphere of influence. Every interaction at a defined time had a decay effect on other entities within its domain. Gain/decay weighting factors were not altered at all through the period of this study.

Background of Participants

Participation in this study was open and voluntary to any individual taking part in the advanced software engineering gizmoball project. They were required to have expertise in the Java programming language, and experience in the use of the Eclipse IDE as a development environment. Of the nine participants that volunteered, eight were male and one female. The real names of participants used in this report have been made anonymous.

\textsuperscript{24} \url{http://www.cis.strath.ac.uk/teaching/ug/classes/52.361/Gizmoball/Gizmoball_spec.htm}

\textsuperscript{25} \url{http://www.mit.edu/~6.170/assignments/gizmoball/gizmoball.html}
Though the selection of participants was left open and voluntary, the sample represented a spread with a mixture of programming expertise from average to excellent such as would be obtained in a real development scenario. Individuals were allocated to groups in this project based on their overall software engineering class performance, thus the best three belonged to one group and the next best belonged to another group, in that order. The reason for this was to obtain groups of equal ability and enable every individual to participate fully in the development process without necessarily leaving the programming task to only the capable ones in their groups. This structure encouraged equal participation among group members. All groups that participated in this study were at least of average performance for the class and also included the best performing group in the class.

**Study Setup**

A week prior to beginning the study each participant was given a three page quick start guide on CRI; this was done to give them an initial familiarisation with the usage of CRI tool and the objective of the study. They were also given a 30 minutes tutorial and demonstration on how to use CRI within the Eclipse IDE before commencing development.

There was also no particular preference as regards time and place of work. Participants were encouraged to carry out the development task at their most suitable time and place. Six of the participants used a versioning system, though the operation of CRI tool is not dependent on its use. They were not required to manage their code using CVS but were encouraged to do so since it is believed to enhance collaboration.

Participants were also allowed to spend time on other tasks that were not related to Gizmoball within the Eclipse IDE.

It was not mandatory that every member of a group needed to volunteer to participate in the study. This is because awareness information of overall and recent work effort is a three faceted mapping of a set of developers to a set of tasks and a set of artefacts used to achieve the task set. Thus, from a standalone point of view, feedback related to a single developer experiences of how CRI represents overall and recent work effort dissipated across an individual’s tasks and code artefacts will also contribute to CRI model validation.

**Questionnaire Design**

The questionnaire was designed to obtain preliminary feedback on the CRI model and its implementation. It contained both open and closed questions and was designed to measure the effect of the Eclipse CRI plugin on subjects’ development experience. This was administered to participants at the end of the study. Questions were asked to find out how
beneficial it was to work within the context of a defined task. Questions also explored if CRI depicted their perception of relevance.

Participants were required to respond to closed statements on a Likert scale where they needed to specify their level of agreement ranging from the response “Strongly Agree” = 5 to the response “Strongly Disagree” = 1. Other questions were related to how they utilised the relevance rankings of entities that existed in their collaboration space. They were also asked to suggest other features they would prefer within the collaboration space. Appendix C1 shows a sample of the questionnaire used during the study.

5.3. Results

Anonymised names and groups of participants are shown in table 5-2. All the three members of group B volunteered to take part in the study, two members out of three in groups A and D volunteered. Finally, only one of the three members of groups C and E participated in the study. At the end of the four weeks, participants were asked to complete a questionnaire to subjectively evaluate their experience during this period. Six of the nine participants returned their questionnaire.

By the end of the study the system had archived 8904 CRI interaction events cutting across 32 tasks. Appendix C2 table C2-1 shows the view, create, update and delete interaction events associated with the 32 task instances; figure 5-1 is a pie chart of the overall distribution of these interaction types over the observation period. Table 5-1 shows the number of CRI interaction events associated with each participant.

![Figure 5-1 Overall distribution of interaction types](image)

82
<table>
<thead>
<tr>
<th>Group</th>
<th>Collaborator</th>
<th>Updates</th>
<th>Deletes</th>
<th>Creates</th>
<th>Views</th>
<th>Total Interaction Events per Collaborator</th>
<th>% of overall event</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Morris</td>
<td>68</td>
<td>3</td>
<td>9</td>
<td>178</td>
<td>258</td>
<td>2.90</td>
</tr>
<tr>
<td>B</td>
<td>Clark</td>
<td>230</td>
<td>0</td>
<td>41</td>
<td>697</td>
<td>968</td>
<td>10.87</td>
</tr>
<tr>
<td>C</td>
<td>Jean</td>
<td>36</td>
<td>1</td>
<td>4</td>
<td>94</td>
<td>135</td>
<td>1.52</td>
</tr>
<tr>
<td>B</td>
<td>Roan</td>
<td>228</td>
<td>1</td>
<td>26</td>
<td>397</td>
<td>652</td>
<td>7.32</td>
</tr>
<tr>
<td>D</td>
<td>Ian</td>
<td>821</td>
<td>2</td>
<td>62</td>
<td>1676</td>
<td>2561</td>
<td>28.76</td>
</tr>
<tr>
<td>A</td>
<td>Pedro</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>21</td>
<td>21</td>
<td>0.24</td>
</tr>
<tr>
<td>B</td>
<td>Evans</td>
<td>619</td>
<td>2</td>
<td>43</td>
<td>936</td>
<td>1600</td>
<td>17.97</td>
</tr>
<tr>
<td>E</td>
<td>Cole</td>
<td>298</td>
<td>12</td>
<td>52</td>
<td>1430</td>
<td>1792</td>
<td>20.12</td>
</tr>
<tr>
<td>D</td>
<td>Andy</td>
<td>404</td>
<td>6</td>
<td>53</td>
<td>454</td>
<td>917</td>
<td>10.30</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>2704</td>
<td>27</td>
<td>290</td>
<td>5883</td>
<td>8904</td>
<td></td>
</tr>
</tbody>
</table>

34.4% of tasks were executed by two or more collaborating developers. On average, 278 CRI interaction events were registered and analysed per task with a minimum of 2 and a maximum of 976. Also, an average of 989 CRI interaction events were registered and analysed per participant with a minimum of 21 and a maximum of 2561. Two view-only\(^{26}\) tasks were recorded.

Questionnaires were not obtained from Morris, Pedro and Jean. Their total number of associated interactions in table 5-2 shows that their use of CRI during development of Gizmoball was minimal, and hence they may not have obtained sufficient usage experience to provide useful responses to the questionnaire.

The number of participants that provided responses is considered not enough to extract statistically significant meaning, especially if compared to the relatively large amount of data that was captured during the study period. It is believed that this was so because at the end of the four weeks evaluation period coincided with the beginning of a semester break. But since this was just a study to provide initial direction on how CRI might possibly enhance awareness during collaborative software engineering, it was felt that it was still important to analyse and report the findings. A summary of the level of respondent agreement to closed questions in this study are as shown in figure 5-2.

---

\(^{26}\) Entities that have only been viewed and not updated within a work context
Figure 5-2 Initial subjective feedback results on the effect of CRI during collaborative software development processes

Study findings: feasibility of building an awareness system for collaborating software developers that are distributed in time and/or space

The implementation of the CRI model suggests that it is feasible to build a system that enables distributed, collaborating software developers to obtain awareness of how work effort is dissipated across the collaboration space. This has been achieved by capturing, integrating and building different work contexts into a collaboration space. This work context has been built using data captured in parallel and related to the type of interaction associated with an entity in a collaboration space, sphere of influence and relative time that interaction event occurred. Furthermore, using the CRI relevance model it was possible to obtain the degree of relevance (hence the relative amount of work effort) of each entity in a collaboration space to a selected work context. At the time of this proof of concept study only CRI recent mode had been implemented. Figure 5-3 shows CRI artefact perspective relevance list visualisations across different timelines to assist participants in the study obtain a perception of what has recently been done within a selected work context. The figure represents four timelines in a sequence that spreads across the study period.

The relevance lists across timelines show at each stage the most recent work effort from associated tasks and developers that has affected the state of the selected artefact OverviewFrame.java within the Gizmoball project. Figure 5-3a shows developers’ relevance ranking across selected timelines 1, 2, 3 and 4. The initial phase of Gizmoball shows that Roan was solely responsible and hence relevant to the state of OverviewFrame.java. Over time, the relevance ranking changed with Evans and Roan collaborating over the same artefact. The final state of the artefact as depicted in the fourth timeline shows Clark as an added collaborator with Evans having recently put in more work effort compared to Clark and Roan. This information can be interpreted to mean that, at the fourth timeline, Evans has
been more responsible for the state of OverviewFrame.java; this is based on the recent work
carried out on OverviewFrame.java.

Figure 5-3 Artefact perspective relevance list views across different timelines (Recent
mode)

Figure 5-3b shows task relevance ranking across the same selected timelines 1, 2, 3 and 4. This shows that OverviewFrame.java was initially only used to achieve Create Build Mode
GUI task. Over time, OverviewFrame.java was used to achieve a number of other task
instances with varying amount of work effort (see timelines 2, 3 and 4). The final state of
OverviewFrame.java (timeline 4) shows that it had been used to achieve five task instances;
also, recently Create Build Mode GUI task has been more responsible for the state of
OverviewFrame.java. GUI Restructure task is ranked low, in terms of the most recent work
that has affected the state of OverviewFrame.java.

In general, figure 5-3 shows how work effort from Gizmoball project tasks and developers
has impacted on the state of OverviewFrame.java. Other similar representations of how
OverviewFrame.java has affected the recent state of ‘Create Build Mode GUI’ task, the
recent work effort of Rory, and the recent state of Gizmoball project over the same timeline
boundary are shown in Appendix C2 figures C2-1 to C2-3.

Feedback from open ended questions administered at the end of the study period helped
understand the feasibility of the approach. Feedback from the study showed that the
approach had the potential to provide participants with awareness information to identify
which entity was most recently affecting the state of their shared project. Snippets of
feedback from participants when asked to list the features of the tool they found useful are
shown below:
… It was useful to see what classes had been used most regularly…

… It was interesting and useful having the ability to view which class/developer was most active in a task.

This showed that CRI did provide participants with current updates of how the state of the collaborative space was changing based on their current activities. The similar representations in CRI relevance list visualisations were also commented on as represented in a snippet of participant feedback below:

… I found it useful to have many different views to see data. Once it is set up it requires low interaction.

Feedback from participants did also show that certain factors had affected the performance of the approach to implementing CRI. The main concern here was that of network latency. Such latency will result in poor usage of CRI within slow networks. Development was carried out on most occasions within the fast University of Strathclyde Computer Science departmental network. All the nine participants reported working from home or using slow networks as well as working from the university network all through the study, two refrained from using CRI over slow networks five days towards the end of the study.

Network latency did affect participants obtaining accurate and real-time awareness information on development work effort during the study. There was also a recorded case where CRI did slow the working pace of a participant. Participants’ feedback when asked to list features that they found difficult to understand or use are stated below:

…CRI suffers with network latency. It would be better if updates were sent as a background task after data is computed, to avoid blocking other tasks…

…CRI should update after a save or asynchronously so that Eclipse does not block important tasks thus slowing down work.

Privacy is another factor that might have also affected the study. There are ethical issues that can arise in providing a view of relevance of developers. CRI could be abused as the basis of capability judgment and reward structuring. To prevent or minimise this, the current implementation of the CRI model allows developers to switch CRI monitors on and off.

Below is a snippet of feedback from a participant when asked if they had any preference for how the relative importance of artefacts, tasks and developers should be viewed in the CRI tool:

…I would say for the “developers” view – it can be used to pass judgement on marks in peer evaluation, for example – developer that is of highest usefulness.
Study findings: CRI and task based development

One of the requirements for the effective use of CRI was that developers had to always formally think of the programming task they wanted to carry out before commencing coding in the collaboration space. This required them to activate a previously existing task or create a new one where necessary. A total of 32 tasks were created across the participants involved in this study, and each of these tasks experienced phases of activation and deactivation (see appendix C2 table C2-1). Responses to the first statement in the subjective evaluation questionnaire (figure 5-2. label 1) showed that no one thought that it was disadvantageous to the collaboration process or the final project output. Two participants thought it was a major advantage while one participant thought it was a minor advantage; three participants were neutral (that is neither advantageous nor disadvantageous).

This result suggests that there may have been occasions where developers refused to create a new development task or switch to an existing one within CRI as their programming task changed. They may not have switched because they felt neutral about the advantage it would bring and thus an added overhead to their programming work.

Feedback from open ended questions helped to further understand why participants felt neutral about formal definition and switching of development tasks. Feedback did indicate that in the long run, participants did find it useful to formally define the tasks they were working on. Snippets of a participant’s response, when asked to list those features of the tool they found useful, are stated below:

… It was useful having to define the task you were working on…

… I feel that overall, the ability to check which classes were integral to a task and which developers were relevant to that task was an intriguing feature which seemed to accurately reflect the work that went into the project…

… It was interesting and useful having the ability to view which class/developer was most active in a task.

…The relevance indexing of artefacts in a task was useful…

But though the formal definition of tasks was useful, open ended feedback also indicated that it was initially a difficult process to formally define these tasks. This was so because they either had forgotten or that it was annoying to do so. Insight into snippets of participants’ responses when asked to list features that they found difficult to understand or use are stated below:
...I often forgot to change task or I would do a bug fix out of the current task when asked by a team member...

...I think it should be quicker to change between activated tasks, as it is very hard to work on one task, and is annoying having to change between tasks...

The first snippet suggest that rather than switching or creating a new task that reflects the bug fix, they may rather do the fix within the current active task. The second suggests that participants would have preferred a quicker way of switching or creating tasks rather than a formal definition in the CRI plugin tool. Zou and Godfrey [ZG06] have also reported that formal task switching often causes extra work to developers within an IDE.

**Study findings: significance of create interaction type**

104 artefact instances were created and monitored by CRI, 19.23% of the artefacts were associated with two or more collaborators. Appendix C2 table C2-2 shows a table of artefacts that were updated by two or more collaborators. Analysing artefacts associated with two or more collaborators showed that in 60% of cases, creators of artefacts were associated with a greater proportion of updates on artefacts they created. The percentage average of total absolute update delta associated with artefact creators was 51.3%. This outcome does suggest that during the study, although the creators of artefacts in significant number of cases do play relatively more role in their further updates, it is also important to capture updates that are associated with other developers on the same artefact since they also play a role that impacts considerably on the state of shared artefacts.

While this result does not rule out the possibility of a reverse scenario arising where there is a drop in the number of cases where the creator of artefacts is associated with a greater proportion of updates on artefacts they created, it is expected that the percentage average of total absolute update delta associated with artefact creators will still be significant.

This result is considered to support the assumption that if a developer \( D \) creates a code artefact \( A \), it is more likely that a greater proportion of subsequent updates on \( A \) will be associated with \( D \). For cases where \( A \) was shared among collaborators, although there is a likelihood that the creator will be associated with a significant percentage of subsequent interactions, the results also show that to obtain an accurate relevance model, it is vital to also depict the work effort of other collaborators that have worked on \( A \) by monitoring their associated updates and views.
Study findings: relative frequency of interaction types

A spread of interaction types over the study period is shown in figure 5-1. Figures 5-4 to 5-7 are bar chart plots of the amount of views, updates, creates and deletes associated with each task instance. A box plot showing the variation of means among interaction types is as shown in figure 5-8. Delete consists of just 0.3% of total interaction types. Create consists of 3.26% of total interaction types, and analysis also shows 30.37% and 66.07% for update and view interaction types respectively.

This results show a dominance of views over other interaction types with delete being the least frequent. A closely related industrial case study carried out by Zou and Godfrey [ZG06] showed that the software artefacts that were viewed but not updated out-numbered the updated artefacts 70% of the time. Zou and Godfrey also found that there are also occasions when the artefacts that are viewed are not directly related to the task at hand. For example, it may suddenly occur to a programmer that some recently written code is incorrect. In such a case, they may examine that code while ostensibly within the current task (i.e., without defining a new task, as they should).

These particular attributes of view interaction type suggest that a system that correctly captures the relevance of viewed entities to a work context will be built on the premise that: ‘the more an entity instance is viewed within a work context, the more relevant such an entity is to the work that is being carried out’. This result also justifies a view interaction weighting factor that is less than the weighting factors that are associated with updates and creates.

The result also shows the relatively minimal occurrence of delete interactions. It is believed here that this is reasonable since developers do not generally create code artefacts in a system to later delete the artefact. The CRI model does not consider the effect of deleting an entity on the relevance ranking of other entities that forms a work context.
Figure 5-4 Bar chart of number views vs task instances

Figure 5-5 Bar chart of number of updates vs task instances

Figure 5-6 Bar chart of number of creates vs task instances
Study findings: Correspondence between developer perception of work effort and CRI relevance list ranking

Closed questions 2 to 6 in the questionnaire administered at the end of the study were aimed at measuring the extent to which the relevance list visualisation associated with increasing colour intensity, corresponded with participants perception of recent work effort involved in achieving the Gizmoball project. Each statement was associated with either obtaining participants viewpoint of task, developer or artefact relevance list visualisations.

Since the sample size involved in this study was too small to consider the statistically significant effect of participants’ level of agreement with each of the statements, the overall shift in viewpoint of participants that is expected to lie at any point on the likert scale is evaluated. On the whole 30 responses ranging between disagree, neutral, agree and strongly agree were received from participants for questions 2-6. A breakdown of the set of responses showed that while there was no strongly disagree responses (0 %), it contained 2 disagree (6.7 %), 10 neutral (30 %), 14 agree (46.7 %) and 4 strongly agree (13.3%). This result show a relative shift from left to right, and tends to suggest a level of correspondence between relevance list visualisation of recent work effort and participants perception of recent work.
effort to achieve Gizmoball. An improvement in this shift will require increasing the percentage of agree and strongly agree responses, this may be achieved by increasing or decreasing values of interaction type weighting factors.

Feedback suggests that there were instances where the relevance values and hence colour intensity increased very slowly or that the ordering of entities did not exactly match the perception of participants. Two snippets of responses below are from open ended questions asking participants to provide feedback on features they found difficult to understand or use, and also make general comments related to their development experience with CRI:

…Sometimes it did not pick up that I was using certain classes, as in a few tasks, the classes used were very low down the order, when they were the only classes used.

…Sometimes CRI did not pick up which tasks are currently the main focus of activity in a collaborative environment.

The two participants that provided the snippets of responses were Roan and Andy from groups B and D. As shown in table 5-1, Roan was associated with 7.32% and Andy with 10.3% of overall interaction events respectively. This amount of associated interaction events from the two participants are considered substantial when compared to the interaction events other developers also generated. Hence, suggesting a need for a deeper insight into how participants formed perception of relevance of tasks, developers and code artefacts to their work context and how this conflicts or conforms to CRI relevance list rankings. Analysis of the second study presented in section 6.7 provides this necessary insight.

5.4. Discussion of findings and resultant CRI system modifications

On the whole, results of this initial proof of concept study are indicative of obtaining awareness of the dissipation of work effort across a collaboration workspace that is distributed in time and space. This section discusses the strengths and weakness of the CRI system based on the findings of this initial study. Resultant modifications that were carried out on the CRI system to try to correct these weaknesses are also presented.

Feasibility of building CRI awareness system

The outcome of this study has shown that using the current implementation approach, it is practicable to build a system that enables distributed collaborating software developers to obtain awareness of how work effort is dissipated across the collaboration space. Of interest in this approach is the capturing and integration of interaction data in a seamless manner from different developer workstations. This meant that interaction data was built from the
viewpoint of every collaborator within a shared space rather than building and sharing the viewpoint of a single developer.

The study revealed that it was possible to capture the different development work contexts that characterise a shared workspace and also provide awareness of the dissipated work effort within such work contexts. This is achieved using a combination of CRI constructs which include the type of interaction associated with an entity, the sphere of influence of such entity and the relative time such interaction took place.

It was pointed out in this study that potential factors related to network latency and privacy could arise based on the current approach. To minimise the effect of these factors, the following modifications were made to the next version of the CRI Eclipse plugin implementation:

- Collaborating and building interaction data offline: In cases where the internet connection is slow, or there is no connection at all, collaborating developers were given the option of working offline and synchronizing interaction events with the CRI server any time an internet connection is detected or at a developer’s convenience. Thus, developers can switch online/offline depending on the availability of networks or the state of existing network.

- To further enhance the performance of CRI at real-time, the next version of the CRI plugin incorporated asynchronous messaging (using Apache ActiveMQ) for developer workstations to retrieve awareness information or synchronise interaction data with the CRI server. Thus, the time collaborators spent waiting to get feedback from CRI server over slow networks, could be used in carrying out other development work within the Eclipse IDE and reducing the effect of network latency.

- The current CRI implementation enables developers to log out/login to Eclipse CRI. As has been done in this study, there is a need for subsequent studies and usage to consider the potential dangers of the use of CRI in capability judgement and reward structuring. The next version of the CRI plugin allowed collaborators to switch off/on their visibility online.

*Constraints associated with CRI task based development*

This study has shown the relative benefits and challenges developers face in formally thinking about and explicitly defining development task, and activating or deactivating development tasks as their work context changes. Findings from the study suggested that in
the long run, participants do find it useful to formally define tasks they are working on. Findings also suggested that there are cases where developers would fail to create a new development task or switch to an existing one within CRI as their programming task changed. In this study this was because participants forgot to do so or because they found it was annoying.

The challenge of overcoming such constraints and overheads lies in the ability to dynamically and correctly capture (without, or with minimal developer intervention) new formal descriptions of tasks and changes in development task as developers go about their work within a collaboration space. CRI as well as other related models such as Kersten and Murphy [KM06] relies on developers formally describing a task. The context and hence the structure of such a defined task is then formed based on monitoring and processing certain types of interactions by developers while executing the defined task.

Relative weighting factors associated with interaction types

Throughout this study, the aim was also to empirically validate the basic assumptions that lay in the choice of weighting factors associated with core interaction types monitored in this research. Analysis of interactions generated from participants in this study did show that the create interaction type was a one-off interaction type, and that the creator of an artefact was more likely to be associated with a greater proportion of subsequent updates on the same artefact. For cases where an artefact was shared among collaborators, the likelihood of the creator being associated with a greater proportion of subsequent updates is slimmer but still considerable. Analysis of results presented in a subsequent study (see chapter 6) further supports this finding.

The outcome of this study also suggests that to achieve an accurate relevance model it is vital to also depict the work effort of other collaborators that have worked on the shared artefact by monitoring their associated updates and views. This finding validates the relative importance of create and hence the assigning of a relatively higher weighting factor compared to the update interaction type. This is so because for artefacts associated with two or more collaborators, in most cases a greater proportion of updates were associated with the creator of the artefact (60%). This suggests that an artefact creator is more likely to possess greater knowledge of the state of an artefact compared to other collaborators over a shared artefact.

The one-off nature of the create interaction type also means that relevance gained by an entity resulting from an associated create interaction type was minimal compared to the relevance gained by the same entity based on associated views and updates. This is
demonstrated in the relatively low frequency of create interactions compared to updates and views during the study.

The findings in this study showed a significantly higher frequency of view interaction types. Studies have suggested that this is so because of the passive nature of the view interaction type and hence it contributes less to the changing state of a shared project compared to update and create interaction types. Thus, it is more accurate to assign a relatively higher weighting factor to the update interaction type compared to the view interaction. Finally, findings from the study also show a significantly low frequency of delete interactions.

In this research the delete interaction type is assigned no weighting factor and considered of null effect on other entities that share the same work context with the deleted entity in the collaboration space. This step is considered reasonable since although a deleted entity is considered intangible its previous influence on the state of a project can normally not be made inconsequential. For instance, removing a developer that has considerably influenced the state of a task or artefact from a collaboration space does not necessitate that all the work effort of the developer on such task or artefact is henceforth null and void. It is considered reasonable to preserve such relevance status of the intangible developer so as to obtain awareness of their previous influence on the collaboration space.

An artefact in the relevance list ceases to exist once it is deleted. The history of interaction events associated with the deleted artefact are retained in the log of event trails. A history sliding or play back functionality implemented in the subsequent version of CRI Eclipse plugin enabled the visualisation of the impact of an artefact on different work contexts before it was deleted.

In general, this study validates that it is more accurate to assign the highest weighting factor to create interaction type followed by update, view and delete respectively.

Improving correspondence between developer perception of work effort and CRI relevance ranking:

The findings of this study show a reasonable level of correspondence between relevance list visualisation of recent work effort and developer perception of recent work effort to achieve a project. Feedback from two participants also suggested that there are instances where the relevance values and hence colour intensity increased very slowly or that the ordering of entities did not exactly match the perception of participants.

Low values for weighting factors meant that relevance values and hence colour intensity was going to increase or decrease slowly, the converse also holds. Colour intensity can be
enhanced by increasing the interaction weighting values by a scaling factor. Weighting factors used in the next version of the CRI Eclipse plugin implementation were increased by a factor of 10. This choice was based on usage statistics generated during the use of CRI in the implementation of the next version of the plugin compared to the output generated during the proof of concept study. This does not affect the relevance ordering of entities and only seeks to enhance the perception of intensity of work effort associated with entities on a relevance list by increasing the rate at which they change. A change in the relevance ordering will require increasing or decreasing relative differences between the different values of interaction type weighting factors. This is because, of the three constructs that constitute CRI, only interaction type weighting factors can be statically determined. The effect of sphere of influence or periodic decay depends on the dynamics of the shared project such as the interaction a developer is involved in, the number of project entities that constitutes a work context relative to the entities in the collaboration space, and the time the interaction occurred. These factors cannot be fixed statically. Increasing or decreasing the relative differences between the values of interaction type weighting factors that corresponds more to developers perception requires a greater insight into the dynamics of shared projects in collaboration spaces. For instance, a more appropriate ranking of tasks, artefacts and developers will require understanding of attributes such as type, lifecycle, and nature of task; complexity and stability of code artefacts; programming domain and developer profiling; project schedules and other risks that can potentially impact the nature of interactions in a shared collaboration space. Studies of these attributes are considered beyond the scope of this current work.

5.5. Conclusion
A proof of concept study required to understand the viability and challenges associated with achieving a CRI model is presented in this chapter. The CRI model is aimed at enabling developers distributed in time and space to obtain awareness of varying shared project work effort that is being dissipated across a collaboration space. A CRI Eclipse plugin implementation was used in this study and only the relevance list visualisation of CRI recent mode was implemented and presented to participants.

The outcome of the study showed that it was feasible to build a CRI awareness system that will aid distributed developers obtain awareness of varying work effort dissipated across a shared collaboration space. It was also realised that network latency and privacy issues were critical to the success of such a system.
Since CRI is a task based model, the study has also revealed the relative benefits and challenges that developers can experience when constrained to work in an environment that requires them to always formally think and explicitly define a development task, or activate/deactivate a development task as their work context changes. Findings from the study suggest that participants did benefit from formally defining, activating or deactivating tasks they were working on. Results also suggest that the process of explicitly creating, activating and deactivating such tasks was considered to be secondary by developers. The challenge of overcoming such constraints and overheads lies in the ability to dynamically and correctly capture new formal descriptions of tasks, and changes in development tasks as developers go about their development work within a collaboration space. This is considered to be beyond the scope of this current research.

This study has also validated the relative weighting factors that were assigned to different interaction types monitored by CRI within a collaboration space. The outcome of this study suggests that it is more accurate to assign the highest weighting factor to the create interaction type and followed by update, view and delete respectively.

Finally, this study also aimed to understand to what extent developer perception of work effort corresponded with CRI relevance list visualisation. Findings from the study suggested a reasonable level of correspondence between participant’s perception of work effort and CRI relevance list visualisation. The study also revealed that this correspondence can be influenced by the slow increasing colour intensity and relevance ordering that sometimes will not match collaborating developers’ perceptions.

Slowly increasing colour intensity in the next version of CRI plugin implementation is improved by carefully scaling up weighting values associated with interaction types. Further improvement on the relevance ordering requires more insight into the attributes and dynamics of a shared project and is beyond the scope of this work. This also involves understanding other risks that can potentially impact on the nature of interactions in a shared collaboration space.

The next chapter presents a second study that gives a more in-depth subjective and analytical validation of CRI. This study was carried out after making modifications to CRI based on the outcome from this proof of concept study. Core modifications addressed included network latency, privacy and scaling to interaction type weighting factors. The history mode and social graph representations were also implemented.
Chapter 6 Study 2- Detailed analytical and subjective validation of CRI

6.1. Introduction
In the previous chapter, an initial proof of concept study was presented to ascertain the feasibility and also understand the challenges and risks involved in achieving a relatively complex model such as CRI. The study revealed the initial viability of the model and modifications that were needed to be carried out to make CRI a more robust and practical system. This chapter presents a second study carried out after system modifications had been implemented. The motivation behind this second study is to build on the results obtained from the first proof of concept study. The research questions addressed here fall into two categories including analytical validation and subjective validation and usefulness.

Analytical validation
Using data generated from participants interactions in this study; this research seeks to address the following research questions analytically:

1) What impact/significance does each of the CRI constructs have on the overall model?

The CRI model consists of three main constructs including interaction type, sphere of influence and periodic decay. The aim of this study is to address the impact and importance that each of these CRI constructs has on the relative relevance of entities using real collaboration data. Each of the constructs can be used to filter relevance and context-related information on its own. Is it possible to build a development work context and obtain useful (perhaps better) relevance of entities, using one or other combination of each of these constructs? To evaluate this set of questions, raw historical event data (views, creates, deletes and updates – see figure 6-2) from this study were extracted. Each of these events defines the developer, task and code artefact they were associated with.

2) Does the model behave according to the design?

Validating if the CRI model behaviour is in accordance with its design involves checking that the results produced by CRI are consistent with embedded formula, sensible and appropriate. Similar to the research question above, using raw historical event data generated from the study, two approaches are used to address this question. Approaches used in addressing this question include building the expected relevance outcome for both CRI recent and history modes using the set of entities that are associated with group historical
events. The second approach involves injecting some known level of activity into existing historical event data and observing how the model behaves from that point on.

Subjective validation and usefulness of the CRI model
Based on participants’ perception and feedback this research seeks to address the following research questions:

3) Does CRI relevance ordering correlate with developers’ perception of activities within a collaboration space?

Do the users of CRI believe that the different perspectives and social graphs, based on relative relevance values of tasks, developers and code artefacts give them results that provide a degree of agreement with their perception of activities within a collaboration space. This question complements the related research question addressed in the first study in understanding the effect of interaction type factors and colour intensity on developer perception of work effort.

4) Do developers believe that the different relevance list perspectives and social graph visualisations are useful in providing awareness during distributed collaborative work?

This will be established by individual debriefing and interview sessions with participants at the end of the study. It is expected that such individual sessions can provide an unbiased view where participants are free to express their group collaboration experience in the absence of their colleagues.

This chapter presents the format of the study, background of participants and study setup. This is followed by a discussion on the design of the interview and questionnaire session administered at the end of the study. Finally, the results generated from the study are presented.

6.2. Study Format
The format of this study is similar to the initial proof of concept study. The study involved ten advanced software engineering students, working in groups of three, from the department of Computer and Information Sciences, at the University of Strathclyde. The set of students involved in this study were different from that used in the previous proof of concept study. The study lasted for a period of six weeks, during which they were to implement and test a system that plays Gizmoball.
Again, every developer that was involved in this study belonged to a group of three members. Participants were asked to log into the CRI tool each time they worked on the project within the Eclipse IDE. They then either activated a shared task of interest that existed in their collaboration space or created a new one.

CRI then monitored and kept records of the core interaction type of interest and presented to them relevance information depending on selected task, artefact or developer representation. Each entity relevance ranking presented was calculated based on a linear combination of associated gain factors resulting from its interaction context as well as its sphere of influence. Every interaction at a defined time had a decay effect on other entities within its domain.

At the end of the six week study period, individual and personalised interviews and debriefing sessions were then conducted with each of the ten participants.

**Background of participants**

Participation in this study was open and voluntary to any individual taking part in the advanced software engineering gizmoball project. All subjects had at least 2.5 years experience of Java programming and were comfortable with the use of Eclipse as a development environment. All the ten participants that volunteered were male. Of the ten participants two groups of three were all from the best performing groups in the class, the remaining three came from groups that were also amongst the strongest and a group that was of average performance. The groups were designed to consist of individuals of similar academic ability to try and encourage equal participation. As in the first study, overall software engineering class performance was used in allocating individuals to groups. The names of the participants in this study have been made anonymous.

**Study setup**

A week prior to beginning the study each participant was given a three page user guide on CRI; this was done to give them an initial familiarisation with the usage of CRI tool and the objective of the study. They were also given a 30 minute tutorial and demonstration on how to use CRI within the Eclipse IDE before commencing development.

There was also no preference as regards time and place of work. Participants were encouraged to carry out development tasks at their most suitable time and place. All participants recorded instances of working from home or within the university campus. All participants used either the CVS or Subversion versioning system. Participants were also
allowed to spend time on other tasks that were not related to Gizmoball within the Eclipse IDE.

6.3. Interview session questionnaire design
At the end of the study interview and debriefing sessions were held separately for each participant in the study. This is in contrast to the way subjective experience was captured in the first study where a questionnaire with a fixed set of questions was used for all the participants. Such personalised interviews and debriefing sessions are essential because a more rigorous study of CRI required that individual contextual experiences of each participant during the study period were captured.

It is expected that obtaining feedback from participants during an interview was more likely to yield useful results in extracting and understanding participants experience. Each of the interview and debriefing sessions was recorded and then further analysed. Participants were encouraged to express and verbalise their thoughts related to each question they were asked during the session. Finally, none of the participants objected to the recording of the session.

The aim in the design of the questionnaire administered during the interview and debriefing sessions with participants at the end of the study was to subjectively validate if CRI relevance lists and social graph matched developer perception of activities within a collaboration space. Perception measures here include overall work effort, recent work effort, difficulty and bugs associated with the different entities that constitute their collaboration space. Furthermore, the interview session was aimed at determining if developers felt that the different relevance list perspectives and social graph visualisations were useful in providing awareness during distributed collaborative work.

For each question, its contribution to the study goal was determined as well as the necessary preconditions, such as the order in which it should be asked and activities needed to be carried out before the question was asked. The method by which the acquired data would be quantitatively and qualitatively analysed was also determined. Details of each of the steps taken for the interview questions administered in this study are as shown in appendix D1.

At the end of each debriefing session, participants were also presented with a questionnaire similar to the questionnaire used in the initial study (Appendix C1 question 7-11). Participants were allowed to fill and return their response to these set of questions at a suitable time. This was to enable participants to express views that may not have been captured during the debriefing session.
6.4. Results

Ten participants volunteered to take part in the study and use the CRI tool during Gizmoball development. Anonymous names and groups of participants are as shown in table 6-1. All the three members of groups G1, G2, and G3 volunteered to take part in the study. Although all the three members in group G4 volunteered for the study, analysis of interaction data captured by CRI showed that only one of the members actually did use the tool. Eight of the ten participants that used the CRI tool also participated in the interview and debriefing sessions administered at the end of the study period.

By the end of the study the system had archived 7166 CRI interaction processes cutting across 16 tasks. Appendix D2 table D2-1 shows a table of view, create, update and delete interaction processes associated with the 16 created task instances; figure 6-1 is a pie chart of the overall distribution of these interaction types over the observation period. Table 6-1 shows the number of CRI interaction events associated with each participant.

Table 6-1 Total of interaction events associated with each participant in study 2

<table>
<thead>
<tr>
<th>Group</th>
<th>Collaborator</th>
<th>Updates</th>
<th>Deletes</th>
<th>Creates</th>
<th>Views</th>
<th>Total Interaction events per Collaborator</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>Alex</td>
<td>550</td>
<td>1</td>
<td>23</td>
<td>740</td>
<td>1314</td>
</tr>
<tr>
<td>G1</td>
<td>Tony</td>
<td>567</td>
<td>1</td>
<td>18</td>
<td>937</td>
<td>1523</td>
</tr>
<tr>
<td>G1</td>
<td>Luke</td>
<td>232</td>
<td>1</td>
<td>11</td>
<td>210</td>
<td>454</td>
</tr>
<tr>
<td>G2</td>
<td>James</td>
<td>1016</td>
<td>1</td>
<td>54</td>
<td>1157</td>
<td>2228</td>
</tr>
<tr>
<td>G2</td>
<td>Paul</td>
<td>42</td>
<td>0</td>
<td>9</td>
<td>222</td>
<td>273</td>
</tr>
<tr>
<td>G2</td>
<td>Tracy</td>
<td>778</td>
<td>1</td>
<td>8</td>
<td>223</td>
<td>1010</td>
</tr>
<tr>
<td>G3</td>
<td>Blair</td>
<td>12</td>
<td>0</td>
<td>5</td>
<td>43</td>
<td>60</td>
</tr>
<tr>
<td>G3</td>
<td>Greg</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>G3</td>
<td>Boris</td>
<td>57</td>
<td>2</td>
<td>12</td>
<td>134</td>
<td>205</td>
</tr>
<tr>
<td>G4</td>
<td>Smith</td>
<td>22</td>
<td>0</td>
<td>2</td>
<td>71</td>
<td>95</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>3276</td>
<td>8</td>
<td>142</td>
<td>3740</td>
<td>7166</td>
</tr>
</tbody>
</table>

Figure 6-1 Overall distribution of interaction types for study 2
50% of tasks were executed by two or more collaborating developers. On average, 448 CRI interaction processes were registered and analysed per task with a minimum of 3 and a maximum of 2479. Also, an average of 717 CRI interaction processes were registered and analysed per participant with a minimum of 4 and a maximum of 2228. While there were no view-only tasks, two update-only tasks were recorded.

Figure 6-2 is a bar chart plot of the amount of views, updates, creates and deletes respectively associated with each task instance. This result is similar to the first study that showed the dominance of views over other interaction types with delete being the least frequent.

In this study, 136 artefact instances were created and monitored by CRI, 18.38% of the artefacts were associated with two or more collaborators. Appendix D2 table D2-2 shows a table of artefacts that were updated by two or more collaborators. Analysing artefacts associated with two or more collaborators showed that in 40% of cases, creators of artefacts were associated with a greater proportion of updates on artefacts they created. The percentage average of total absolute update delta associated with artefact creators was 62.4%.

![Figure 6-2 Bar Chart showing frequency of Interaction types vs Task instances](image)

**Figure 6-2 Bar Chart showing frequency of Interaction types vs Task instances**

The two participants that opted out of interviews were Blair and Greg, both from group G3. The total number of associated interactions in table 6-1 shows that Blair and Greg’s use of CRI during development of Gizmoball was minimal based on the amount of interactions.
captured. Thus it is expected that they would not have obtained sufficient usage experience required to provide useful responses during the interview and debriefing sessions. Transcripts of feedback from the debriefing and interview sessions are also presented in appendix D3. Responses provided by participants in the second questionnaire they filled in at their own time yielded no new insights. Participants simply supported the initial points they had made during the interview and debriefing sessions.

6.5. Significance of CRI model constructs (RQ1)

The aim of this section is to address the first research question on the impact and significance that each CRI variable construct (see section 3-5) has on the relative relevance of entities that constitute a selected work context. Table 6-2 shows six different sets of treatments to the linear representation of the CRI model. Each treatment set simulates the effect of one or a combination of defined constructs on CRI relevance values using the sequence of historical events. The first three sets of treatments explores the single and combined effect of sphere of influence and interaction type constructs on CRI value without periodic decay of inactive entities (history mode). The last three simulates the effect of all the constructs on CRI value including periodic decay of inactive entities (recent mode). Interaction type weighting factors used in these treatments are similar to that used during the Gizmoball project study shown in appendix D2 table D2-3.

Treatment set 1 simulates a scenario where the relevance of entities to a selected work context is only evaluated based on their sphere of influence. The controlled variables in this treatment include interaction type and periodic decay, and do not contribute to gain or decay in relevance of tasks that constitute a selected entity work context.

Treatment set 2 simulates a scenario where the relevance of entities to a selected work context is only evaluated based on the type of interactions being carried out in a collaboration space. The controlled variables in this treatment include sphere of influence and periodic decay. To obtain insight into the relative contribution of each interaction type to the relevance values of entities, seven sub-treatments (2a-2g) on collaboration data are performed. Set 2a represents a treatment where all interaction events contributed to the relevance value of tasks associated with a selected entity work context while 2b represents a treatment where create interaction events are controlled and only view and update interaction event types contributed to the relevance value of tasks associated with the work context. Furthermore, 2c-2d represents treatments where one or two interaction event types were controlled and only the remaining interaction type contributed to the relevance value of tasks associated with an entity work context. Similar sub-treatments were also carried out for
Sub-treatment 3a corresponds to the history mode in the CRI model. This sub-treatment explores the effect of sphere of influence and all the interaction types on the relevance value of tasks that constitute an entity work context. Sub-treatment 6a corresponds to the recent mode of the CRI model. This sub-treatment explores the effect of sphere of influence and all interaction types with periodic decay of inactive tasks that constitute the selected work context. Together, the six different sets of treatments provided 30 possible combinations of

<table>
<thead>
<tr>
<th>Set</th>
<th>SOI</th>
<th>Interaction Type</th>
<th>Periodic Decay</th>
<th>Treatment Active Entities</th>
<th>Inactive Entities</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>2a</td>
<td>X</td>
<td></td>
<td>X</td>
<td>S_{0(t-1)c} + s_{0(c)}</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
<td></td>
<td>T = { t_i, t_j, t_k }</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
<td>T = { t_i }</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td>T = { t_j }</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
<td></td>
<td></td>
<td>T = { t_k }</td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
<td></td>
<td></td>
<td>T = { t_i }</td>
<td></td>
<td></td>
</tr>
<tr>
<td>g</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 3a  |     |                  | X              | S_{0(t-1)c} + t_{0s} + s_{0(s)} | X                 |   |
| b   |     |                  |                | T = \{ t_i, t_j, t_k \}   |                   |   |
| c   |     |                  |                | T = \{ t_i \}             |                   |   |
| d   |     |                  |                | T = \{ t_j \}             |                   |   |
| e   |     |                  |                | T = \{ t_k \}             |                   |   |
| f   |     |                  |                | T = \{ t_i \}             |                   |   |
| g   |     |                  |                | X                         |                   |   |

| 4a  |     |                  | X              | S_{0(t-1)c} + t_{0s} + s_{0(s)} | X                 |   |
| b   |     |                  |                | T = \{ t_i, t_j, t_k \}   |                   |   |
| c   |     |                  |                | T = \{ t_i \}             |                   |   |
| d   |     |                  |                | T = \{ t_j \}             |                   |   |
| e   |     |                  |                | T = \{ t_k \}             |                   |   |
| f   |     |                  |                | T = \{ t_i \}             |                   |   |
| g   |     |                  |                | X                         |                   |   |

| 5   |     |                  | X              | S_{0(t-1)c} + s_{0(c)} + t_{0s} | X                 |   |
| 6a  |     |                  |                 | X                         |                   | T = \{ t_i \} |
CRI constructs that can be used in generating relevance lists based on the selected work context.

Appendix D2 figures D2-1, D2-2, D2-3 and D2-4 represent a task relevance list outcome of the Ball.java work context in group G1 for each of the treatments carried out in the study. Ball.java was associated with the highest number of tasks (6 out of 7), and each developer in G1 collaborated on the artefact; it was also worked on throughout the development period (appendix D2 table D2-4 shows a table of tasks and developers associated with artefacts identified with two or more collaborators, furthermore, appendix D2 figure D2-5 illustrates the relative time that artefacts were introduced into the G1 collaboration space). On the whole the outcome of the controlled treatments on Ball.java as shown in both figures D2-2 and D2-4 of appendix D2 illustrates that any one or more combination of variable constructs could be used to generate a relevance list that represents overall or recent work effort expended by entity instances that constitute a selected work context. The main challenge here is understanding how representative a partial combination of constructs will be of the nature of collaboration is a shared project space. The remaining part of this section discusses the significance of each of the CRI model constructs and its effect on relevance list outcome if any of the constructs is eliminated from the model.

Significance of sphere of influence

The concept of sphere of influence in CRI represents the relative levels of influence that entities can exact in collaboration spaces. In this research it is asserted that the size of an entity’s work context or the number of other entities that an entity exacts its presence on is proportional to the relative influence such an entity exacts on the collaboration space. Also, the size of the selected work context an entity is associated with is proportional to the relative influence such an entity exacts on the collaboration space.

Here, the significance and impact of this definition of sphere of influence is examined by analyzing how Alex’s varying sphere of influence ratio impacted on the relevance of entity instances in his work context. Figure 6-3 illustrates the sphere of influence ratio variation and activity time graph of interaction events carried out by Alex on AnimationWindow.java and ObjectHandler.java. Throughout the history of Alex is collaboration over Gizmoball, his sphere of influence varied between 0.25 and 0.43 with frequent upward and downward spikes.
Figure 6-3  Sphere of influence ratio variation and activity time graph of interaction events carried out by Alex on AnimationWindow.java and ObjectHandler.Java
The shaded regions of the graph labelled X and Y is used to illustrate the range of sphere of influence ratios attained by Alex while he worked on AnimationWindow.java and ObjectHandler.java. The average SOI ratio attained by Alex while working on AnimationWindow.java was 0.29 while the average attained while working on ObjectHandler.java was 0.4. Alex viewed ObjectHandler.java 22 times and AnimationWindow.java 56 times. He also carried out 37 update events during which he generated a total absolute update delta of 4910 on ObjectHandler.java. In addition to creating AnimationWindow.java, Alex carried out 12 update events on AnimationWindow.java during which he generated a total absolute update delta of 4513. In order to analyse these events regardless of their type, these events were standardised using the weighting associated with each event type. The result of the standardisation shows that Alex carried out 5173 interaction events on AnimationWindow.java while he carried 5130 interaction events on ObjectHandler.java.

While these amount of interaction events are considered within close range, the average SOI ratio while Alex worked on ObjectHandler.java was higher (0.4) compared to when he was working on AnimationWindow.java (0.29). This outcome suggests that the relevance value of AnimationWindow.java to Alex’s work context was dampened by the lower SOI ratio he was associated with while working on the artefact.

The artefact relevance list in figure 6-3 treatments 2a and 3a illustrates a snapshot of the resulting artefacts relevance list based on Alex’s work context for two treatments carried out on his interactions over the study duration. Treatment 3a illustrates the generation of relevance list using all the interaction types in combination with sphere of influence. The outcome shows that the relevance of AnimationWindow.java (labelled a) had been dampened by the SOI ratio of Alex although the amount of interaction events he associated with AnimationWindow.java is higher than those associated with ObjectHandler.java positioned at the top of the relevance list. The converse is seen in treatment 2a with AnimationWindow.java at the top of the relevance list. In this treatment the artefact relevance list was generated using only the different interaction types without any damping effect of SOI.

In general, the consequence of omitting SOI in the determination of relevance of entities to a selected work context is that the perception of the pace of development work that will normally be experienced as the lifetime of a software project extends will not be accurately depicted.
This example illustrates that SOI works as intended. On the whole, SOI is important if it is an accepted notion that the size of a selected work context that an entity is associated with is proportional to the relative influence such an entity exacts on the collaboration space. Thus, this notion should be accounted for in determining the relevance of entities to a selected work context.

*Significance of the create event*

The implementation of CRI using create interaction events alone or in combination with sphere of influence or periodic decay results in an inaccurate representation of the relevance of entities to the selected work context. This is demonstrated in snapshots of treatments 2g, 3g, 4g and 6g shown in figure 6-4. Although Ball.java had been used to achieve six different tasks, figure 6-4 shows that only ‘Flipper demo’ task constituted the relevance list of Ball.java over the whole of its development period.

Treatments where the create event was not implemented also demonstrated cases of inaccuracy. Figure 6-5 shows snapshots of the relevance positions of developers that constitutes Ball.java’s work context at the early stages of its development for treatment 3a and 3b. In 3a, only the create event is monitored while the converse is true in 3b. The outcome shows that the absence of the create event in 3b resulted in inaccurate representation of the relevance of developers in the early stages of developing Ball.java. This is demonstrated in label i and ii in the figure with the absence of Alex on the relevance list in the early stages of development work on Ball.java. Similar relevance list outcomes were obtained when the same comparison was made for other treatment sets.

The impact of create on the cumulative relevance of an entity to a selected work context is expected to reduce as the duration of the work context extends. This is because the create event occurs once over the lifetime of an entity. This is demonstrated in label iii and iv of figure 6-5 with no conspicuous difference in the positioning of developer or colour intensity representation on the relevance list.
Figure 6-4 The final states of task relevance representations of Ball.java development for treatments where create event alone was used or in combination with SOI and periodic decay

Figure 6-5 Snapshot of developer relevance lists during the initial phases of Ball.java development for treatments 3a and b
Significance of the view event

The significance of view events is gradually obvious in entities that are more frequently associated with views compared to update or create events. While the impact of a single view event is not expected to contribute significantly, accumulation of such views does have a significant impact. Thus, the more a developer views a code artefact instance, the more such an artefact becomes relevant to his work context.

In this study it is illustrated how the omission of view events in CRI can inaccurately affect relevance lists that depict the recent work effort of developers. Figure 6-6 is an activity time graph of interaction events carried out by Alex on PlaceTest.java and MoveCommand.java over the history of his interactions. Analysis of event sequences associated with Alex on these artefacts showed that he was responsible for creating PlaceTest.java which he subsequently viewed 261 times while MoveCommand.java was viewed 11 times. He also updated PlaceTest.java 8 times that yielded a total absolute delta of 242. Similarly, Alex updated MoveCommand.java once that yielded a total absolute delta of 496.

Furthermore, the graph demonstrates that Alex initially viewed and subsequently updated MoveCommand.java. But a greater proportion of the last phase of his work was centred on PlaceTest.java with an increasing sequence of view events and sporadic updates. Such frequent views on PlaceText.java suggest that the artefact forms an integral part of Alex’s most recent work context.

Eliminating the impact of view events in determining the relevance of entities to a selected work context implies that such interaction pattern cannot be captured within CRI. The artefact relevance list for Alex’s work context is shown in figure 6-6 and labelled treatments 6a and c. Treatment 6a represents the artefacts relevance list generated using the sequence of Alex views, updates and create interaction events, in combination with his varying sphere of influence and the measure of recency by decaying the relevance values of entities not involved in the recent interaction event (periodic decay). Conversely, treatment 6c represents artefacts relevance list generated using only the sequence of Alex update and create interaction events with varying sphere of influence and periodic decay.
Figure 6-6 Activity time graph of interaction events carried out by Alex on PlaceTest.java and MoveCommand.java and a snapshot of his generated relevance list for treatment 6a and 6c
The first consequence of eliminating the view event in determining relevance is that PlaceTest.java will only appear in the artefact relevance list for Alex’s work context because he created and updated the artefact instance. If Alex did not create or update PlaceTest.java then PlaceTest.java would not appear in his work context. Treatment 6a accurately represents the recent sequence of view events Alex carried out on PlaceText.java, this is reflected by its position at the top of the relevance list. Conversely, treatment 6c inaccurately represented the relevance of PlaceTest.java with MoveCommand.java at the top of the artefact relevance list. Using only creates and updates carried out by Alex on PlaceTest.java has not reflected the measure of relevance that shows his recent sequence of views on the artefact.

During the interview and debriefing session administered at the end of the Gizmoball study, Alex was presented with the relevance list shown in treatment 6a as representing his most recent work effort. A snippet of comment from his feedback is shown below. This illustrates that although PlaceTest.java was not associated with a significant amount of coding, his continuous viewing and running PlaceTest.java program code did impact his recent work.

…I have spent a lot of time quite recently to develop test cases for the gizmos, particularly on PlaceTest.java…

Though in PlaceTest.java, I don’t think there was that much amount of code in it to keep it in that position, I was always viewing and running the code all the time…

Significance of updates

The significance of update events in determining the relevance of entities to a selected work context is rather more obvious. The omission of update events can result in considerable inaccuracy in the positioning of entity instances on a relevance list.

Figure 6-7 illustrates the activity time graph of events carried out by Alex on BouncingBall.java and ObjectHandler.java. BouncingBall.java and ObjectHandler.java are both associated with an absolute update delta of 4026 and 4910 respectively. BouncingBall.java was viewed 41 times, while ObjectHandler.java was viewed 22 times. Treatments 3a and 3d shown in figure 6-7 illustrate the effect of eliminating updates when determining the relevance of entities to a selected work context. Treatment 3a represents a snapshot of the artefact relevance list of Alex’s work context generated using all the interaction types in combination with SOI. Conversely, treatment 3d represents the artefact
relevance list of Alex’s work context generated using only view and create interaction types in combination with SOI. The outcome in 3a reflects the amount of absolute update delta that Alex has associated with ObjectHandler.java and BouncingBall.java. The relative positions of these artefacts on the relevance list are significantly altered in treatment 3d when update interaction events were omitted. This is because the combination of the amount of view and create events on ObjectHandler.java and BouncingBall.java combined with the damping effect of Alex’s SOI could not equate to the updates that Alex had carried out on these artefacts.
Figure 6-7 Activity time graph of interaction events carried out by Alex on BouncingBall.java and ObjectHandler.java
Significance of Periodic Decay

The main objective of periodic decay is to provide the perception of recent work effort exerted by different entities that exist in a collaboration space. The perception of recent work carried out within a project space can be derived and presented by decaying the relevance values of entities that are not currently associated with any interaction event (inactive entities). This decay is equivalent to a measure of the relative impact an interaction event has on the relevance of active entity (the task, developer and artefact) instances associated with an event. Figure 6-8 is used to illustrate the significance of decaying the relevance of inactive entities by this measure.

Figure 6-8 is an activity time graph of interaction events carried out by Alex on PlaceTest.java and ObjectHandler.Java. While Alex generated a total update delta of 242 over the duration of his work over PlaceTest.java, he generated a total update delta of 4026 for his work on ObjectHandler.java. He also viewed PlaceTest.java and ObjectHandler.java 261 and 22 times respectively. The activity time graph shows that interaction events on ObjectHandler.java were carried out by Alex during an earlier phase of his work while the interactions on PlaceTest.java were carried out in the later phases of his work. The interaction set associated with PlaceTest.java were the latest interaction events that Alex carried out before the end of the Gizmoball study. In order to enable the analysis of these events regardless of their type, interaction events on these two artefacts were standardised using the weighting associated with each event type. After standardisation, ObjectHandler.java and PlaceTest.java are associated with a total of 5130 and 2952 interaction events respectively.

Based on overall work effort and given the amount of associated total interaction events that Alex exerted on these two artefacts, it is expected that ObjectHandler.java will be more relevant compared to PlaceTest.java for Alex. Conversely, based on recent work effort, the interactions on PlaceTest.java are more recent compared to the interactions carried out on ObjectHandler.java and thus PlaceTest.java should attain a higher relevance position. The relevance position of ObjectHandler.java will decay by a measure of every interaction event that Alex carried out after his work on ObjectHandler.java. Analysis showed that artefacts such as AddTest.java, SaveTagTest.java, SaveLabelTest.java, and BuildWindow.java all contributed substantially to the decay in the relevance of ObjectHandler.java.
Figure 6-8 Activity time graph of interaction events carried out by Alex on PlaceTest.java and ObjectHandler.java and related relevance lists generated for treatment 3a and 6a
This expected outcome is illustrated in the artefacts relevance list represented in treatment 3a and 6a as shown in figure 6-8. Treatment 3a represents the relevance list of overall work effort by Alex. Relevance values are generated using view, update and create interaction events with sphere of influence, and without the decay in the relevance values of inactive entities. Treatment 6a on the other hand, uses all the interaction event types with sphere of influence and decays the relevance values of inactive entities. The high relevance of ObjectHandler.java is depicted in treatment 3a since this treatment does not include periodic decay, the position of PlaceTest.java is low down (24\textsuperscript{th} position) in the relevance list. The high relevance of PlaceTest.java is in treatment 6a given that this treatment now decays the relevance of inactive entities.

On the whole, in order to capture each of these observed collaboration patterns, it is essential that a relevance model that accurately represent a work context implements each of the model constructs discussed in this work. In this research, a combination of sphere of influence and interaction type construct consisting of view, update and create is considered more appropriate in representing the overall effort dissipated by entities in a selected work context. In representing recent work effort, periodic decay is used in addition to these constructs.

6.6. Comparison of model theoretical design with behaviour in practice (RQ2)

The previous section investigated the relative contribution of each component of CRI. To address the second research question, this section builds on that, assuming a CRI model constructed from all its components, and investigates the behaviour of the complete model. The methodology used is an analysis of CRI outputs during real development to determine if the final CRI relevance value behaves in accordance with its design.

Firstly sphere of influence is analysed more closely under real development conditions. The artefact Ball.java from group G1 is used as the basis of this analysis. Figure 6-9 shows the variation in SOI as new tasks and developers were associated with the G1 collaboration space and/or the Ball.java work context. The dots show the point at which an entity is added, label a on the right shows the times at which entities are added to the G1 collaboration space, label b shows when they are added to the Ball.java context, and label shows the corresponding SOI value (not the CRI value).

Entities in the blue shaded area on the left existed prior to the creation of Ball.java. Alex was responsible for the creation of Ball.java whilst working on the Flipper.Demo task. The initial SOI value is 0.5 because there are 4 entities in the collaboration space, 2 of which are in the
work context of Ball.java. The subsequent addition of Tony to the Ball.java work context leads to an SOI value of 0.75. Similarly, the SOI value drops to 0.6 when Absorber.java is added to the G1 collaboration space, but not to the Ball.java work context. The SOI behaviour is characterised by growth with occasional downward spikes as entities are added to the G1 space. In this case, the behaviour is appropriate because of the fact that Ball.java is associated with most (6 out of 7) tasks and all collaborating developers in group G1. Further investigation involved constructing similar plots for the six tasks associated with Ball.java and the three developers. While most of them showed a similar profile of upward growth with occasional downward spikes, ‘Flipper Demo’ showed significant downward movement as artefact entities were added to the collaboration space without being added to the ‘Flipper Demo’ work context. An interesting contrast occurred in the plots and data analysis of ‘File Demo’ and ‘Collision Demo’. The sum of entities in their collaboration space was the same (both 81) and remained relatively constant. However, ‘Collision Demo’ was developed by ‘Alex’ using only one artefact, while ‘File Demo’ was developed by ‘Tony’ using 13 artefacts and hence had a corresponding greater SOI (13/81 v 1/81) in keeping with the sphere of influence philosophy embedded in CRI. This evidence indicates that the SOI component of CRI is behaving as expected in practice.

Figure 6-9 Sphere of influence ratio variation as new entities are associated with the G1 collaboration space or the Ball.java work context
To investigate the overall behaviour of CRI all the interactions associated with Ball.java were investigated. Table 6-3 shows the distribution of these interactions across developers and artefacts associated with Ball.java. Corresponding to the data in Table 6-3 the time series of events associated with Ball.java were also analysed. These are shown in Figure 6-10. The figure has four main sections. The top section (S1 creates) shows the occurrence of create interactions associated with Ball.java – it was created by Alex within task Flipper Demo. Below that (S2 views) shows the occurrence of view interactions associated with tasks and developers. At the bottom left (S3 updates) the update interactions are shown. To the right of this (S4 AUD) shows the corresponding absolute update delta for every task and developer appearing to the left in S3. For S1, S2 and S3 the y axis plots at 1 represent an active event while plots at 0 represent a period of inactivity. For S4 the y-axis scale is different for each entity instance associated with Ball.java and is determined by the maximum absolute update delta associated with the entity instance.

The first insight this data provides is the importance of capturing the create interaction in CRI. This immediately associates Alex and Flipper Demo with the context of Ball.java. Again, this is in keeping with the CRI design philosophy that the creator of an entity is likely to have useful knowledge of that entity and should therefore be captured in a model that provides awareness associated with that entity.

The next observation concerns the apparent close association between view interactions and update interactions - reading vertically between S2 and S3 tasks and developers it is seen that there is often a matching burst of views and update interactions associated with the Ball.java artefact. This behaviour again is to be expected, developers will often view artefacts to ensure they are relevant prior to updating or view related artefacts before making a change in an associated artefact. Both of these actions lead to increasing relevance within CRI. On the other hand, at the bottom of S3 Luke is associated with Ball.java around time frame 11 and updates Ball.java without viewing it at all. It is still important that Luke is thereafter added to Ball.java’s work context with a relevance value based on the size of his update and his current SOI within in the collaboration space. Finally, there are expected case where views occur without corresponding updates. This presents a dilemma for the CRI model – are these views really relevant or just spurious. The possibility that views may be spurious is dealt with by the reduced weight attributed to views - typically a factor of 10 less than the average update delta (based on data captured to date). As a result, the odd, random spurious view will have little cumulative impact on CRI rankings whereas regular viewing of an artefact (which is presumed to mean that it is not spurious) will accumulate true relevance.
Table 6-3 The distribution of interactions among entities in Ball.java work context

<table>
<thead>
<tr>
<th>Entity Instance</th>
<th>Views</th>
<th>Updates</th>
<th>Absolute Update Delta</th>
<th>Creates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tasks</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Absorber Demo</td>
<td>18</td>
<td>13</td>
<td>1233</td>
<td>0</td>
</tr>
<tr>
<td>Collision Demo</td>
<td>1</td>
<td>1</td>
<td>55</td>
<td>0</td>
</tr>
<tr>
<td>Build Mode</td>
<td>29</td>
<td>34</td>
<td>3818</td>
<td>0</td>
</tr>
<tr>
<td>File Demo</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Play Mode</td>
<td>17</td>
<td>50</td>
<td>1755</td>
<td>0</td>
</tr>
<tr>
<td>Flipper Demo</td>
<td>87</td>
<td>50</td>
<td>12058</td>
<td>1</td>
</tr>
<tr>
<td>Developers</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alex</td>
<td>75</td>
<td>29</td>
<td>2239</td>
<td>1</td>
</tr>
<tr>
<td>Tony</td>
<td>78</td>
<td>110</td>
<td>14925</td>
<td>0</td>
</tr>
<tr>
<td>Luke</td>
<td>0</td>
<td>10</td>
<td>1757</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 6-10 Activity-time graph of developer and task interactions associated with Ball.java
Relevance list behaviour in history and recent modes

To further investigate the expected behaviour of CRI a known level of activity was injected to the model for Ball.java and corresponding changes in relevance ranking were analysed. Table 6-3 shows the starting state for the CRI model prior to this controlled injection and Figure 6-9 shows the corresponding starting SOI value of 0.9. At this stage Flipper Demo is at the top of the task relevance ranking.

In this study the theoretical estimated effort that should carry Absorber Demo to the top of the task relevance ranking for Ball.java in the history mode was estimated. Knowing that at the end of the study, Ball.java had attained a sphere of influence ratio of 0.9, and given the amount of interaction events Flipper Demo task was associated with to be at the top of the relevance list, a minimum additional standardised interaction events of 10364 is required for Absorber Demo task to be at the top of the task relevance list of Ball.java history mode relevance list. This minimum standardized interaction events are also equivalent to 100 view events and an update delta of 9364. This outcome is based on the assumption that over the duration of injection, only Absorber Demo task was active, and the sphere of influence remained constant.

A controlled injection at a constant sphere of influence ratio of 0.9 was then carried out. During this 100 views and updates generating a total update delta of 9364 was added to the collaboration space associated with developer Luke, task Absorber Demo and artefact Ball.java. The updates were carried out as a series of 139 update events that generated a total update of 9364. A series of CRI history and recent mode snapshots during the injection are shown in figure 6-11.

The section labelled snapshot 1 shows the relevance rankings of Ball.java before the injection of interaction events, while the section labelled snapshot 2 shows the relevance list of Ball.java after injecting 30 view events and 12 update events that generated a total update delta of 863. Finally, the section labelled snapshot 7 shows the relevance list of Ball.java at the end of injecting 100 view events and update events that had generated a total update delta of 9364. This demonstrates that CRI behaved as expected for this particular scenario.

Furthermore, it demonstrates the difference in behaviour between history and recent mode. Snapshot 7 shows Absorber Demo reaching the top of the relevance rankings as expected after the estimated amount of simulated interactions but in recent mode it had reached the top

27 minimum additional standardised interaction events for Absorber Demo to be at the top of Ball.java relevance list =(((view events for Flipper Demo) - (view events for Absorber Demo))*10 + (total update delta for Flipper Demo) - (total update delta for Absorber Demo))*SOI
by snapshot 6. The reason for this is that in recent mode the ranking of inactive entities such as the Flipper Demo task will decrease as other tasks are active to reflect awareness of recent activity – therefore it should be expected in recent mode that changes such as this are more quickly reflected. Thus, as expected, in recent mode a reduced amount of interaction was required to propagate Absorber Demo to the top of the rankings – only 100 views and a total update delta of 3925.

Furthermore, the contrasting affect on colour intensities reflecting the relative significance of entities can be seen by comparing the history and recent snapshots in Figure 6-11. In history mode the colour intensity representing inactive tasks remained unchanged while the colour representing Absorber Demo increased as interactions were injected. However, in recent mode the colour representing inactive entities progressively reduces as interactions are injected, representing their reducing relevance.

This research question has investigated whether CRI behaves as expected when used in real development. To investigate this real data was captured from six weeks of development by advanced Software Engineering students. This has been achieved via a detailed analysis performed on the behaviour of the model for the Ball.java. The Ball.java artefact was the focus of significant development for a group of three developers and had a major influence on project tasks. The analysis shows that: the sphere of influence component behaved as expected, growing and shrinking as entities were added to Ball.java work context or associated with the collaboration space but did not affect the Ball.java work context; it showed how CRI captured the different interactions types, their relative frequency and the justification for their weightings in the CRI model; and, finally through a controlled injection of activity into a known CRI model state showed that both the history and recent modes responded as predicted.
Figure 6-11 Snapshots of Ball.java task relevance ranking for history and recent mode while injecting interaction events
6.7. Correlation between CRI relevance list rankings and developer perception of work effort (RQ3)

This section addresses the third research questions in this study. This is performed by seeking more insight from participants’ feedback during the interview and debriefing sessions administered at the end of the Gizmoball project study. To address this research question it was necessary to understand participants’ experiences while carrying out development tasks using the CRI tool. For instance: Were participants using the tool appropriately and consistently over the collaboration period? Does CRI impact the way that participants will normally carry out a programming task? One main aim during the debriefing and interview session administered to participants at the end of the Gizmoball project study was to obtain their experience while using the CRI eclipse plugin tool. This was to provide insight whether participants in the study used the CRI tool correctly and consistently over the collaboration period and if the tool impacted the way they carry out programming tasks.

To address this third research question empirical probabilities of subject relevance ranking of entities matching those presented by CRI relevance lists were calculated. Furthermore, this study sought to provide insight into factors that can influence developers’ perception of relevant entities constituting their work context. Understanding whether CRI was useful in enhancing the awareness and relevance perception formation processes of participants on their interactions within the collaboration space was also essential.

Correct and consistent use of CRI tool over a collaboration period

The appropriate and consistent use of the CRI eclipse plugin required that participants to log on with their user details each time they needed to undertake a programming task. During the interview and debriefing session, participants were asked to express how frequently they remembered to log into CRI. Feedback on the percentage frequency participants remembered to log into CRI is shown in figure 6-12. This provides an initial understanding that the total interactions captured by CRI model tool represented between 60-90% of individual participant interactions used in achieving Gizmoball.
Transcripts of participants’ responses to the first question during debriefing sessions are shown in the below in Q1 (also see appendix D3). These snippets suggest that requiring developers to log into CRI each time they needed to carry out a programming task might impair developers’ appropriate and consistent use of the Eclipse CRI plugin during collaborative software development. This is because, in general, participants confirmed there were occasions they forgot to log into CRI.

**Q1: How frequently did you remember to log into CRI?**

*Tracy:* The problem was that sometimes when you start up eclipse, you just forget that there is a panel in the right…

*Luke:* Not as often as I probably should have…
I found that sometimes when carrying out some work, I would have already gone half way through before remembering that I needed to activate CRI…

*Alex:* Recently I have been forgetting quite a lot to log into CRI because there is so much work to do…

*Boris:* It’s a pain to remember to do it…
Because you want to get work done CRI is not the top thing in your mind to be logging into…

*James:* Basically, it’s that I was eager to do something on my project, and because CRI is not always what I am thinking of…
And when I started working…
Then I just remember…
Oh! I forgot to log into CRI…
If it auto logged you in that would have been much better…

*Paul:* Initially, I used to forget to logon since I was not really used to using CRI…
Sometime I only logon when I have already started doing some work, and that is when I remember…

*Tony:* It takes about half an hour before I remember each time I start work…
In the reverse if I was automatically logged on into CRI, I will only oblige to that if we had talked about it before hand…
Usually I will be programming for about six hours, and I do not consider 30 minutes into this time to be much, especially since within that time I am still figuring out what to do…

A number of reasons can contribute to developers forgetting to log into CRI. For instance, snippet from Tracy’s transcript points out that the unfamiliar user interface of the CRI
eclipse plugin was a contributing factor. Luke, Alex, Boris, James, Paul and Tony all pointed out that logging into CRI could be a secondary consideration, especially while trying to meet tight schedules during collaborative software development.

Challenges associated with logging in by collaborating developers can be mitigated where developers are automatically logged into CRI. This can be done each time the system notices developers are carrying out a program coding task. Further studies need to be carried out to determine if such an automatic logging process is intrusive and can be done accurately. For instance, while James did not object to the idea of being automatically logged in, Tony though he would only agree to being automatically logged in if previously informed.

In general, based on participant’s feedback, the total amount of interactions captured by the CRI eclipse plugin tool in this study represents an average of 75% of the overall interactions contributed by each participant to achieve Gizmoball in their respective groups. More insight on the captured interaction events being representative of the actual interaction events from each developer can be carried out by simulating the consequence of loosing a percentage of overall interactions needed to achieve a project. In this thesis, the average of 75% of overall interaction events is considered high enough to obtain accurate representation of participant’s behaviour.

*Constraints associated with CRI task based development*

Findings from the initial proof of concept study suggested that participants did find it useful to formally define tasks they were working on. The initial study also suggested that there were cases where participants failed to create a new development task or switch to an existing one within CRI as their programming task changed. Some reasons participants gave why this might occur was that they either forgot to define a new task or switch to a new task when their work context changed, or that they found the process annoying.

To obtain further insight into this constraint associated with formal task definition and switching, it is necessary to understand if other factors such as the usability of the CRI tool, the nature of defined tasks or privacy concerns are factors. It was also important to understand if relevance ranking of participants that depends on the task or artefact they were working on negatively influenced collaboration dynamics within CRI. The second question administered to each participant was intended to provide insight into the influence of these factors (question 2 appendix D3) on formal task definition and switching.
During the study collaborators in group G1 formally defined and switched between seven tasks, G2 defined and switched between five tasks, G3 defined and switched between three tasks, and G4 defined and worked within the context of a single task. Figure 6-13 below shows the number of tasks associated with each participant as their work context changed. Of the eight participants that were interviewed and took part in the debriefing sessions, two (Boris and Smith) did not switch tasks and worked within the context of only one task all through the development period. The remaining six switched work context between two and six tasks.

Figure 6-13  Tasks associated with participants

Figure 6-14 Percentage frequency participants activated or created tasks as work context changed

Figure 6-15 Difficulty experienced by participants in always working within the context of a task

Figure 6-16 Difficulty experienced by participants when creating a new task using the CRI tool

Figure 6-17 Difficulty experienced by participants when activating an existing task using the CRI tool

Figure 6-14 is a dot plot of subjective feedback on how frequently participants felt they activated or created a new task as their work context changed. None of the participants were absolutely confident that each time their work context changed they had created a new task or formally switched task within CRI. Feedback from participants that switched between two or more tasks showed that frequency of task switching and creating with changing work...
context ranged between 25 and 50%. Figures 6-15, 6-16 and 6-17 are also subjective experience feedback from participants related to their difficulty of working within the context of a task, the difficulty to create a new task and activate an existing task respectively. Comparison of these subjective experiences with the frequency participants created or activated a task as their work context changed show that it is less likely that this frequency resulted from the usability of CRI tool, but more likely that it is a result of difficulty experienced by participants in working within the context of a task.

Analysis of transcripts while participants provided responses to the second question during debriefing sessions provides further insight into how factors such as usability of the CRI tool and the nature of defined tasks affected participants’ creation of new tasks or activation of existing tasks as their work context changed.

Each of these participants in the duration of the study created a task and activated two or more tasks during the study. A few snippets of feedback from participants in response to questions Q2b and Q2c below suggest that participants generally did not experience difficulty while either creating a new task or activating an existing one using the CRI tool. But they did have difficulty identifying the particular task they were working on and logging that within CRI. It is pointed out that the incident of network latency and slow system performance does sometimes still arise.

**Q2b: I found it difficult to always be working within the context of a task?**

*Luke:* I was a lot clearer in the prototype related tasks because it was just a kind of standalone task, for instance I knew I was working on Flipper Demo, and I simply concentrated on that, I did not need to depend or think about other tasks…

This has become less clear as we have shifted to developing the final system that consisted of Build Mode and Play Mode tasks; these two tasks are so closely linked… Also, when I had thoughts of what to do in the system, I always want to get it done so I don’t forget rather than to go into CRI to activate which tasks I was working on…

*Alex:* Yeah…

Because we had other things to do as well, it was quite difficult to stay within one thing…

*Paul:* I think it gets sometimes difficult to know if you are straying into a task that is not directly related to the current task you are working on… Sometimes I sit down and I am coding on a certain task… I suddenly get bored with the task…

*Or* I am not too sure about what I am to do in the task…

*Or* you get something that is not working properly on the task…

Then I feel like I want to work on another task…

*Tony:* Oh yeah…

I struggle with working on other things when attending to a particular task…
I do so because it’s either getting boring, or I am going nowhere with the task or even that I am distracted… But usually I just write those things down as task items and tick them as I finish each of them…

Q2c: How difficult was it to create a new task in CRI?

Luke: I think creating a task is not a problem at all… It’s remembering to do it that is the problem…

Alex: It was quite easy, it behaved exactly as I expected…

James: It’s not difficult, but sometimes the network latency is annoying…

Tony: It’s not difficult at all…

Understanding the nature of tasks participants defined is critical to providing insight into the relative frequency at which participants activate and deactivate tasks with changing work context. A task is considered to be any set of actions that affects the functionality of a system in some way. These include fixing bugs, improving performance or implementing new use case features. The generality or granularity of defined tasks determines the nature of its attributes such as related subtasks and dependency on other tasks. It is expected that more generally defined tasks will have more subtasks and less dependency on other tasks, while tasks defined with higher granularity will have less subtasks and more dependency on other tasks. Similarly, a collaboration space where tasks are generally defined will be associated with less frequency of task switching compared to a collaboration space where tasks are defined with a greater level of granularity.

Direct correspondence between frequency of task switching and granularity/generality of defined task cannot be obtained in this study. Feedback from debriefing sessions does provide insight into why participants felt they probably did not create or switch into tasks as much as they should have during the study. A few snippets of participants’ response to how frequently they activated or created a new task as their work context changed are as shown in the question labelled Q2a below. Q2b also shows a few snippets of participants’ experiences while working within the context of a task. Snippets in Q2a show that participants preferred to create and work within the context of more general tasks that represent different parts of the system. More granular tasks such as specific bug fixes or different sub components of a system part were seldom formally specified within CRI. Practical reasons pointed out by James and Tony on why more granular tasks were not formally created or activated includes avoiding disrupting or losing the flow of thoughts while carrying out development work. On
the whole, it can be concluded that participants were more comfortable with creating general tasks, requiring less activation and deactivation sequences.

Q2a: How frequently did you activate or create a new task as your work context changed?

Alex: I think our tasks list was pretty general…
So each of the demos was a task…
There were not that many tasks to choose from, so we did normally work within a number of classes but on the same task…

James: I thought it was just implementing one part of a system or another part, I thought each of those parts was a task…
So I did switch CRI depending on which part of the system I was working on. Most of the time I either switched either between one part of my task (User Interface) and the other model task I did (Basic Core)…
I think I can have a lot more tasks within these main tasks I was working on…
Yeah, I did switch between these tasks each time I wanted to start work…
Within each task I had active, I did deviate…
For instance if I noticed that there was a problem in the model while I was working on the User Interface I go straight and change it without remembering to explicitly switch on CRI…
Because it was like I didn’t want to lose my thoughts on the problem I wanted to solve…

Tony: We have had a few tasks in general, and the description of these tasks has been quite vague…
For instance for the past three weeks or so we have virtually been on the same task Build Mode…
The tasks could probably be defined to be more specific, but I feel that the process of creating or activating a task while working will be disrupting the flow of my thoughts…

Although participants were more comfortable with creating general tasks requiring less activation and deactivation sequences, this study also reveals a number of scenarios that would have required participants to change work context and hence explicitly create new task or activate existing task. Snippets in Q2b show that most participants found it difficult to be working within the context of a defined task. Reasons for experiencing this difficulty include lack of clearly specified task (Luke); straying into a task that is not directly related to the currently executed task (Alex, Paul); distractions within the collaboration space (Tony); and current boring or difficult tasks (Paul, Tony).

Matching between developers’ work perceptions and CRI relevance list ranking
This research also aimed to obtain further insight on whether the relevance lists generated can be used to enhance awareness of work effort during distributed software development.
To obtain this insight, a comparison was made between the work perceptions of participants in the Gizmoball project with the actual CRI relevance lists generated during the study.

During the interview sessions, participants were presented with a list of artefacts and tasks they had been working on, then asked to rank the top four artefacts based on four different criteria. These criteria included:

i. Ranking based on overall coding effort they had put into each artefact through the duration of the Gizmoball project.

ii. Ranking based on artefacts in Gizmoball they had recently put the most coding effort into.

iii. Ranking based on the most difficult artefact they had worked on.

iv. Ranking based on the artefact that generated the most number of bugs.

Using the data on the ranking of artefacts, it is possible to determine the empirical probabilities associated with CRI history and recent mode relevance list rankings matching with the criteria stated above. Two sets of matching are carried out in this analysis. The first involves the probability of match between CRI history mode relevance list rankings and participants’ perception of overall coding effort, most difficult artefacts and artefacts that generated the most number of bugs. The second involves the probability of match between CRI recent mode relevance list rankings and participants’ perception of artefacts they had recently put the most coding effort. Results of the analysis are as shown in tables 6-4, 6-5 and 6-6. The third column in each of the table corresponds to history mode (overall work effort); the fourth column corresponds to recent mode (recent mode) while difficulty and bugs corresponds to the fifth and sixth columns.

Table 6-4 represent the empirical probability of the first selection from a participant matching with CRI relevance ranking. The first row in the table shows that the probability an artefact at the top of subject rankings of artefacts for overall coding effort matched the first artefact on CRI history mode relevance ranking is 0.625. Similarly, the probability that the artefact at the top of subject rankings of artefacts they had recently put the most coding effort into is a subset of the top two artefacts on CRI recent mode relevance ranking is 0.5 (row 2, table 6-4). The probability that the artefact at the top of subject rankings of artefacts they felt was the most difficult they had worked on was a subset of the top nine artefacts on CRI history mode relevance ranking is 0.75. Likewise, the probability that the artefact at the top of subject rankings of artefacts that generated the most number of bugs is a subset of the top nine artefacts on CRI history mode relevance ranking is 0.875.
Table 6-4 Empirical probability of any first selection from participant matching with CRI relevance ranking

<table>
<thead>
<tr>
<th>Empirical Probability</th>
<th>Overall Work Effort</th>
<th>Recent Work Effort</th>
<th>Difficulty</th>
<th>Bugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Ranking subject(1) (\subseteq) Ranking CRI(1)</td>
<td>0.625</td>
<td>0.375</td>
<td>0.375</td>
<td>0.5</td>
</tr>
<tr>
<td>2 Ranking subject(1) (\subseteq) Ranking CRI(2)</td>
<td>0.75</td>
<td>0.5</td>
<td>0.375</td>
<td>0.5</td>
</tr>
<tr>
<td>3 Ranking subject(1) (\subseteq) Ranking CRI(3)</td>
<td>0.875</td>
<td>0.5</td>
<td>0.625</td>
<td>0.75</td>
</tr>
<tr>
<td>4 Ranking subject(1) (\subseteq) Ranking CRI(4)</td>
<td>0.875</td>
<td>0.5</td>
<td>0.625</td>
<td>0.75</td>
</tr>
<tr>
<td>5 Ranking subject(1) (\subseteq) Ranking CRI(5)</td>
<td>0.875</td>
<td>0.625</td>
<td>0.625</td>
<td>0.75</td>
</tr>
<tr>
<td>6 Ranking subject(1) (\subseteq) Ranking CRI(6)</td>
<td>0.875</td>
<td>0.625</td>
<td>0.625</td>
<td>0.75</td>
</tr>
<tr>
<td>7 Ranking subject(1) (\subseteq) Ranking CRI(7)</td>
<td>0.875</td>
<td>0.75</td>
<td>0.625</td>
<td>0.75</td>
</tr>
<tr>
<td>8 Ranking subject(1) (\subseteq) Ranking CRI(8)</td>
<td>0.875</td>
<td>0.75</td>
<td>0.625</td>
<td>0.75</td>
</tr>
<tr>
<td>9 Ranking subject(1) (\subseteq) Ranking CRI(9)</td>
<td>0.875</td>
<td>0.75</td>
<td>0.75</td>
<td>0.875</td>
</tr>
<tr>
<td>10 Ranking subject(1) (\subseteq) Ranking CRI(10)</td>
<td>0.875</td>
<td>0.75</td>
<td>0.75</td>
<td>0.875</td>
</tr>
</tbody>
</table>

Table 6-5 Empirical probability of any first two selections from participant matching with CRI relevance ranking

<table>
<thead>
<tr>
<th>Empirical Probability</th>
<th>Overall Work Effort</th>
<th>Recent Work Effort</th>
<th>Difficulty</th>
<th>Bugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Ranking subject(2) (\subseteq) Ranking CRI(2)</td>
<td>0.875</td>
<td>0.5</td>
<td>0.875</td>
<td>0.75</td>
</tr>
<tr>
<td>2 Ranking subject(2) (\subseteq) Ranking CRI(3)</td>
<td>1</td>
<td>0.5</td>
<td>0.875</td>
<td>0.75</td>
</tr>
<tr>
<td>3 Ranking subject(2) (\subseteq) Ranking CRI(4)</td>
<td>1</td>
<td>0.625</td>
<td>1</td>
<td>0.75</td>
</tr>
<tr>
<td>4 Ranking subject(2) (\subseteq) Ranking CRI(5)</td>
<td>1</td>
<td>0.75</td>
<td>1</td>
<td>0.75</td>
</tr>
<tr>
<td>5 Ranking subject(2) (\subseteq) Ranking CRI(6)</td>
<td>1</td>
<td>0.75</td>
<td>1</td>
<td>0.875</td>
</tr>
<tr>
<td>6 Ranking subject(2) (\subseteq) Ranking CRI(7)</td>
<td>1</td>
<td>0.875</td>
<td>1</td>
<td>0.875</td>
</tr>
<tr>
<td>7 Ranking subject(2) (\subseteq) Ranking CRI(8)</td>
<td>1</td>
<td>0.875</td>
<td>1</td>
<td>0.875</td>
</tr>
<tr>
<td>8 Ranking subject(2) (\subseteq) Ranking CRI(9)</td>
<td>1</td>
<td>0.875</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9 Ranking subject(2) (\subseteq) Ranking CRI(10)</td>
<td>1</td>
<td>0.875</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6-6 Empirical probability of any first three selections from participant matching with CRI relevance ranking

<table>
<thead>
<tr>
<th>Empirical Probability</th>
<th>Overall Work Effort</th>
<th>Recent Work Effort</th>
<th>Difficulty</th>
<th>Bugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Ranking subject(3) (\subseteq) Ranking CRI(3)</td>
<td>1</td>
<td>0.5</td>
<td>1</td>
<td>0.75</td>
</tr>
<tr>
<td>2 Ranking subject(3) (\subseteq) Ranking CRI(4)</td>
<td>1</td>
<td>0.625</td>
<td>1</td>
<td>0.75</td>
</tr>
<tr>
<td>3 Ranking subject(3) (\subseteq) Ranking CRI(5)</td>
<td>1</td>
<td>0.75</td>
<td>1</td>
<td>0.75</td>
</tr>
<tr>
<td>4 Ranking subject(3) (\subseteq) Ranking CRI(6)</td>
<td>1</td>
<td>0.875</td>
<td>1</td>
<td>0.875</td>
</tr>
<tr>
<td>5 Ranking subject(3) (\subseteq) Ranking CRI(7)</td>
<td>1</td>
<td>0.875</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6 Ranking subject(3) (\subseteq) Ranking CRI(8)</td>
<td>1</td>
<td>0.875</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7 Ranking subject(3) (\subseteq) Ranking CRI(9)</td>
<td>1</td>
<td>0.875</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8 Ranking subject(3) (\subseteq) Ranking CRI(10)</td>
<td>1</td>
<td>0.875</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6-5 represents the probability of either of the first two selections from participants matching with CRI relevance rankings. The first item on 6-5 shows that the probability of
either of the top two artefacts in the subjects’ perception of artefacts with the most overall coding effort is a member of the top two artefacts in the CRI history mode relevance ranking is 0.875. This suggests that it is more likely for either of the top two artefacts that a participant felt represents the most overall coding effort being a subset of the top two artefacts in the CRI history mode relevance list (probability of 0.875 as shown in table 6-5 row 1). This is compared to the likelihood that the top artefact a participant felt represents the most overall coding effort being the top artefact in the CRI history mode artefact relevance list for that participant (probability of 0.625 as shown in table 6-4 row 1).

Similarly, comparing table 6-4 and 6-5 shows a higher probability for either of the top two artefacts that participants felt they had recently put the most coding effort into, being the most difficult or having the highest number of bugs (probability of 0.5, 0.875 and 0.75 respectively as shown in table 6-5 row1). This is compared to the probability that the top artefact participants felt they had recently put the most coding effort, being the most difficult or having the highest number of bugs (probability of 0.375, 0.375 and 0.5 respectively as shown in table 6-4 row 1). Table 6-5 row 8 shows that probabilities of 1 were obtained when considering artefacts consisting criteria i, iii and iv discussed earlier when considering the top nine of CRI artefacts in the relevance list.

Finally, table 6-6 represents the probability of any of the top three selections from participants matching with CRI relevance ranking. The outcome shows higher probability values compared to tables 6-5 and 6-4. The first row on table 6-6 shows that the probability that any of the top three artefacts consisting subjects’ perception of those with the most overall coding effort is a member of the top three artefacts on the CRI history mode relevance ranking is 1. Also, the probability that any of the top three artefacts that participants felt they had recently put the most coding effort into, and had the most difficulty or with the highest number of bugs were represented in the top three of the CRI artefacts relevance list were 0.5, 1 and 0.75 respectively.

Empirical probabilities obtained in this analysis are quite subjective and can be influenced by external factors not controlled during the Gizmoball project study. For instance, recent activities that participants were engaging in before the interview and debriefing sessions can potentially affect participants’ perception of relevance. Also, developers were free to engage in other programming related activities that were not related to the Gizmoball project or monitored by the CRI Eclipse tool. The extent to which captured interaction events are representative of the overall interactions that were actually carried out by participants can also affect the outcome in this study. It is estimated that the CRI eclipse plugin tool captured
only 75% of overall interactions contributed by each participant to achieve Gizmoball in their respective groups.

Based on these factors, not controlled in the study, the probability of any of the first two selections from a participant matching with CRI relevance ranking as shown in table 6-5, and the probability of any of the first three selections from a participant matching with the CRI relevance ranking as shown in table 6-6, are considered more realistic compared to the probability of any of the first selections from a participant matching with the CRI relevance ranking as shown in table 6-4.

These results suggest that the optimal performance of the CRI model in the accurate ranking of entities on the relevance list is obtained when the history mode relevance list is compared with developers perception of overall work effort. A considerable level of accuracy is also obtained when the history mode relevance list is matched against the most difficult artefacts developers have worked on and artefacts that have generated the most number of bugs. The outcome of comparing the recent mode relevance list with artefacts developers have recently put the most coding effort into yielded less accurate results compared to the other measurement criteria. To obtain further insight into the accuracy of the CRI model it is important to understand factors that influenced participants’ perception of the relevant entities that constituted their work context. It is also essential to know how CRI relevance lists were useful in enhancing awareness and the perception formation processes of participants.

After each participant had provided their rankings on the different criteria above by relying on their memory recall and without the aid of CRI, they were then presented with artefact relevance list rankings produced by CRI history and recent modes. These relevance lists were generated based on each participants’ context of work captured while they were logged into the CRI tool. The aim here was to obtain feedback from participants on their perception of CRI history and recent mode relevance lists, relative to their overall work effort and artefacts they had recently put the most coding effort into.

It is expected that such feedback from participants on their perception of CRI history and recent mode relevance lists will provide insight into factors that influence how participants’ form perception of relevant entities constituting their work context. For cases where relevance rankings of participants based on recall did not closely correspond to that of CRI relevance ranking, feedback may provide insight into whether the CRI relevance ranking was useful in enhancing awareness and perception formation process. This may explain the causes of some of the discrepancies obtained in the empirical probabilities presented above.
Appendix D3 Q4c-Q4e shows snippets of transcriptions from participants on their perceptions of CRI history and recent mode relevance lists relative to their overall work effort and artefacts they had recently put the most coding effort into.

**Factors influencing participants’ perception of relevant entities constituting their work context**

This study has revealed a number of factors that can influence how participants’ form perceptions of relevant entities constituting their work context. Analysis of feedback from participants shows that greater coding effort over a selected artefact while executing a project task, does not consistently correspond to a higher perceived level of difficulty associated with the implementation of the associated task or code artefact. For instance, the snippet of feedback below is a response from James when he was presented with a history mode relevance list of artefacts that constitute his work context.

**James:**

Although I put in much coding effort into GizmoColourScheme.java and did a lot of edits on it, it was not as hard to implement compared to LeftFlipper3D.java…

I think it’s up there because I have made a lot of changes to it…

Yeah, the ranking does actually show the amount of coding effort I have put into each of the classes.

Mostly what I did in GizmoColourScheme.java was getters and setters and some few changes in colour schemes…

I was always going back to GizmoColourScheme quite frequently when I was working on the ‘User Interface’ task…

CRI history mode relevance list representing artefacts that constitute James work context showed that GizmoColourScheme.java was 4th positioned while LeftFlipper3D.java was 6th positioned. The feedback from James reactions to these positions illustrates that although James had put in more coding effort into GizmoColourScheme.java compared to LeftFlipper3D.java, the perceived level of difficulty was greater for LeftFlipper3D.java. Thus, from a subjective viewpoint, the position of GizmoColourScheme.java in the CRI history mode relevance list can be said to more accurately represent the coding effort of James compared to the difficulty he experienced in implementing the artefact.

Analysis of interaction event data associated with James working with these two artefacts shows that while GizmoColourScheme.java was associated with 37 view events and update events that generated an absolute update delta of 8207, LeftFlipper3D.java was associated with 31 view events and update events that generated an absolute update delta of 6877. Further insight into view and update interaction events carried out on GizmoColourScheme.java and LeftFlipper3D.java by James is as shown in figure 6-18.
figure illustrates that the sphere of influence ratio was constant all through the duration that James implemented GizmoColourScheme.java and LeftFlipper3D.java. Furthermore, the figure illustrates that the duration between the first and last recorded active state of GizmoColourScheme.java while James was carrying out the User Interface task is relatively longer than the duration between the first and last recorded active state of LeftFlipper3D.java while James was carrying out the same task. This corresponds with James feedback that he was always going back to GizmoColourScheme.java while executing the User Interface task.

Findings from this study also suggest that participants also form perception of work effort by clustering closely related code artefacts. For instance, the snippet of feedback below is a response from Paul when he was presented with the history mode relevance list of artefacts that constitute his work context.

Paul:
I am not sure of Wall.java, I don’t think I put as much coding effort into Wall.java as I put into the flipper related classes…
I would have also thought that the flipper classes would be further up in the ranking…
I tend to agree with the Geometry classes being low down; I really did not use them much…

This illustrates that Paul considered putting in more coding effort into a cluster of flipper related artefacts compared to Wall.java (the cluster of flipper related artefacts here consists of all the artefacts that are connected to the functionality of Gizmoball flipper). Analysis of interaction events associated with Paul identified three obvious artefacts that were directly related to Gizmoball flipper functionality that Paul had also interacted with. These included LeftFlipper.java, Flipper.java and RightFlipper.java. The CRI history mode relevance list representing artefacts that constitute Paul’s work context showed that Wall.java was 4th positioned while LeftFlipper.java, Flipper.java and RightFlipper.java were 6th, 9th and 17th positioned respectively.
Figure 6-18 Activity-Time graph of interaction events carried out by James on GizmoColourScheme.java and LeftFlipper3D.java while executing the User Interface task
Table 6-7 Total interaction events carried out by Paul on flipper related artefacts and Wall.java

<table>
<thead>
<tr>
<th>Artefact</th>
<th>view events</th>
<th>Absolute update delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wall.java</td>
<td>12</td>
<td>1400</td>
</tr>
<tr>
<td>LeftFlipper.java</td>
<td>7</td>
<td>778</td>
</tr>
<tr>
<td>Flipper.java</td>
<td>8</td>
<td>214</td>
</tr>
<tr>
<td>RightFlipper.java</td>
<td>19</td>
<td>25</td>
</tr>
</tbody>
</table>

Figure 6-19 Activity-Time graph of interaction events carried out by Paul on flipper related artefacts and Wall.java
Further analysis of interaction event data associated with Paul working with these artefacts is shown in table 6-7. Figure 6-19 also shows the activity-time graph of interaction events carried out by Paul on flipper related artefacts and Wall.java. Examination of these interactions shows that the Paul’s sphere of influence ratio varied minimally with occasional spikes between 0.34 and 0.28 while he was interacting with these four artefacts. Furthermore, Wall.java is associated with a greater proportion of standardised interaction events relative to every individual artefact constituting flipper related artefacts. Conversely, when all the interaction events that constitute flipper related artefacts are combined together, a greater proportion of standardised interaction events are rather associated with the cluster of flipper related artefacts compared to Wall.java.

The study also shows that developers can sometimes be influenced by the size of an artefact while forming a perception of work effort. This suggests that developers do not always build a perception of relevance based on the artefacts they have created, the frequency of views or the amount of absolute update delta generated while updating the artefact. The snippet of feedback below is a response from Tracy when he was presented with the history mode relevance list of artefacts that constitute his work context. Irrespective of the number of times Tracy updated or viewed Gizmoball.java while collaborating over the User Interface task, the actual size of the artefact was more paramount while considering overall coding effort he had put into each artefact through the duration of the Gizmoball project.

**Tracy:**
Yep…
I will say that is right at head-on…
The only thing I will say is that I am surprised that Gizmoball.java is even on the ranking, because it’s just a driver consisting of two lines of code…
The only thing I ever remember doing on it was to update and call it every time I was testing the User Interface task…

Furthermore, this study has revealed that simple copying and pasting of program code from alternative sources into an artefact can also influence a developer’s perception of work effort, difficulty and bugs associated with a code artefact. This is illustrated below in the snippet of feedback from Boris when he was presented with the history mode relevance list of artefacts that constitute his work context.

**Boris:**
….I am not sure I accept the position of Stream.java. I did not really put much effort; I simply copied and pasted from an online source, and have not really changed the code much…
More subjective studies need to be carried out to validate the association of this identified collaboration pattern with perception of overall work effort, difficulty, bugs and most recent coding effort experienced by developer. The outcome also suggests that an automated awareness system such as CRI has the potential to maintain more accurate rankings than might be possible by human developers who can be influenced by recent effort or overwhelmed by the amount of work done or the scale of a project.

6.8. Enhancing awareness during collaborative work processes (RQ4)

The final research question addressed in this study seeks to discover whether CRI was useful in enhancing the awareness and relevance perception formation processes of participants on their interactions within the collaboration space. Firstly, insights are presented on how CRI enhanced awareness and the relevance perception process of individual developers using the different generated relevance lists. This is followed by findings on how CRI enables awareness in distributed software development spaces.

Enhancing awareness and the relevance perception process of individual developers using CRI relevance lists

While there were recorded cases where the presentation of CRI relevance rankings did not change the perception of participants on the relevant artefacts that constituted their work context, there were also instances where CRI was useful in enhancing the awareness and the relevance perception formation processes of participants.

The snippet of feedback below is a response from Boris when he was presented with the history mode relevance list of artefacts that constitute his work context.

Boris:
…I know why Utilities.java should be so high…
It’s because I have been creating some static methods…
So instead of creating an instance of Utilities class I just do Utilities.save() or Utilities.load() to load the file name…

Yes I understand why OuterWall.java should be there, I recently was working on it…
I don’t know why BouncingBall.java will be higher than a Flipper.java…,
I know why! …
It’s because I didn’t know if I will use GameModel.java to store MU and MU2 so I also put it in BouncingBall.java so I could go back and change which ever one was going to be changed…
So yeah…
The relevance position of BouncingBall.java is quite right...

This illustrates that before Boris was presented with the history mode relevance list of artefacts that constitute his work context, his relevance perception formation process lacked awareness and proper judgement of the work effort he had put into a number of artefacts. His visualisation of CRI relevance list enhanced his attaining further awareness of previous work he had carried out on Utilities.java, OuterWall.java and BouncingBall.java.

Boris was also presented with a recent mode relevance list of artefacts that constitute his work context. This feedback also demonstrated that the CRI recent mode relevance list was useful in enhancing his awareness and relevance perception of artefacts he had recently put the most coding effort into.

A number of other cases that provide insights into how CRI was useful in enhancing the awareness and relevance perception formation processes of participants on their interactions within the collaboration space were also recorded.

The snippet of feedback below is a response from Alex when he was presented with the history mode relevance list of artefacts that constitute his work context.

**Alex:**
I am a bit surprise that CollidableCircle.java is positioned that high; I thought it would have been a bit lower…
Yeah, giving deeper thought into it, the ranking looks about right, just that sometimes I depend on my recent coding experience…
…based on my recent memory, I would say that BuildWindow.java is higher, but thinking about it, I would have worked much more on other code artefacts…

Also, the snippets of feedback below are responses from Alex, Tracy, Luke and Tony respectively when they were further presented with the CRI recent mode relevance list of artefacts that constitute their work context. In each case, the relevance list provided the participant with useful awareness information that triggered a more appropriate relevance perception formation process.
*Luke:*  
A number of test classes are high up there, yeah…

I accept, just remembering that the test classes had really taken much of my time in the last two days… Yes, and the last data set I uploaded to CRI server contains a lot of work I had done on the test classes…

We all worked on the test classes together… Anybody that creates an artefact makes sure that he writes a test class for that artefact…

As much as possible, if I wrote a test class, then its going to be another member of the group that will run the test class I wrote…

This was to ensure that the testing was rigorous…

*Alex:*  
Right, I will say so… Come to look at it, I have spent a lot of time quite recently to develop test cases for the gizmos, particularly on PlaceTest.java…

Though in PlaceTest.java, I don’t think there was that amount of code in it to keep it in that position, but I was always viewing and running the code all the time…

As I said before, I have also recently been working on the command related classes; and the ranking reflects that…

*Tracy:*  
You know I was just thinking why Drawing3D.java should be there…

Yep…

Can now remember three days ago we were trying to figure out how to do some stuff with it…

Yep, I will say the ranking is spot-on…

*Tony:*  
I think I will agree with every artefact there except for StillEdge.java and StillCircle.java…

I am not too sure of what I would have been doing with StillEdge.java or StillCircle.java, but no doubt I think I would have viewed it a couple of times…

In the last few days, I have actually done a lot of work that I struggle to come to terms with what I have recently put much effort into…

---

*Enhancing awareness cues in distributed software development spaces*

In the literature review, it was revealed that there normally exist complex dependencies among entities that constitute a software project. Within such dependency settings, one will expect that a code artefact can be associated with a number of developers and used to
achieve a number of project tasks such as use cases and bug fixes. Similarly, a project task can be associated with a number of collaborating developers and achieved with a number of code artefacts. Finally, it is expected that a developer can be working on a number of project tasks and using a number of code artefacts to achieve the task [GPS04, MFH02]. Such dependencies simply increase the number of entities that can be used to describe a work context and hence make it more difficult to derive awareness information extracted from a development work context especially in a distributed software development scenario.

One of the objectives of this study was to obtain evidence on whether CRI can enable developers to obtain awareness of development processes in distributed software projects especially where there exist complex dependencies among entities that constitute the shared project. During the Gizmoball group project study, participants were not restricted in time or place of work. Collaborating groups were mandated to have a same time, same place 3 hour meeting at least once every week; during this time they also discussed their progress with the teaching assistant (TA) coordinating the group. Feedback from participants in groups G1, G2 and G3 suggest that besides the mandatory time of meeting they sometimes did organise colocated meetings. All the groups used a version control system (Subversion). Feedback from group G1 suggests occasional pair programming practice, while group G2 also used a wiki system. On the whole, participants spent more time working at different times or places than they spent working together. This study narrates participants’ awareness experience while using the CRI Eclipse plugin tool within these described work settings.

The snippet of feedback from Luke from group G1 shows insights he obtained while sliding through the history of entities constituting their collaboration space:

Luke:

….I had a slide through the relevance positions of developers and java classes for File Demo task… I noticed that it has only been Tony working on that task. TriangleBumper.java and MainProgram.java were the original classes I noticed he started with, and it was so for quite a while…

Currently there are a number of other classes he has used for that task…

If a maintenance task is to be carried out on my system, such information will really be useful too… Since I can see the relative change of relevance that an artefact or a developer would have had associated with a task used to realise the system…

The feedback snippet from Luke suggests that he did not need any formal or informal colocated meeting with other members of his group to obtain awareness insight into the state of File Demo task. Through his use of CRI he was able to understand that it has been
only Tony that has been working on File Demo; from this he also obtained insight into the relative significance of the artefacts Tony was using to accomplish the task.

Analysis of the developer interaction data captured during the study enabled the detailed investigation of this snippet. In particular it was possible to recreate development paths and investigate the validity and details of participants’ comments during the interviews. Analysis of File Demo is as illustrated in table 6-8 showing the percentage of interaction events Tony had associated with each artefact he used in accomplishing the File Demo task. A total of 13 code artefacts were used by Tony to achieve File demo, he interacted significantly with TriangleBumper.java (32.03%), LeftFlipper.java (23.60%) and MainProgram.java (17.55%). Figure 6-20 shows an activity-time plot of File Demo and snapshots of the related artefacts relevance list at different intervals over the history of File Demo. The artefact relevance list in figure 6-20 label 1-6 shows that the initial phase of Tony’s work on File Demo actually involved TriangleBumper.java and MainProgram.java and later progressed to using a number of other code artefacts. The positions of TriangleBumper.java and MainProgram.java have also been consistently high on the relevance list over the history of File Demo as demonstrated by the artefacts labelled a and b on the artefact relevance list. This analysis, confirms the awareness insight obtained by Luke on File Demo.

For a distributed setting, it would have been difficult for Luke to obtain awareness insight into the state of Tony’s work on ‘File Demo’ and the relative importance of the artefacts that Tony used to accomplish the task without the use of a relevance model such as CRI. This is particularly so given that over the history of Tony’s work on File Demo he had different amounts of interaction events with the code artefacts he used to accomplish the task. While artefacts such as Ball.java had minimal significance given that Tony had minimal interaction with it, TriangleBumper.java had greater significance since Tony had substantial interaction with it while executing File Demo.
Table 6-8 Percentage of developer and artefact interaction events associated with File Demo

<table>
<thead>
<tr>
<th>Artefacts</th>
<th>Views</th>
<th>Updates</th>
<th>Absolute Update Delta</th>
<th>% of standardised Interactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>GameModel.java</td>
<td>3</td>
<td>5</td>
<td>184</td>
<td>2.23</td>
</tr>
<tr>
<td>MainProgram.java</td>
<td>21</td>
<td>21</td>
<td>1475</td>
<td>17.55</td>
</tr>
<tr>
<td>TriangleBumper.java</td>
<td>24</td>
<td>21</td>
<td>2836</td>
<td>32.03</td>
</tr>
<tr>
<td>GameObject.java</td>
<td>4</td>
<td>1</td>
<td>32</td>
<td>0.75</td>
</tr>
<tr>
<td>RightFlipper.java</td>
<td>9</td>
<td>5</td>
<td>859</td>
<td>9.88</td>
</tr>
<tr>
<td>LeftFlipper.java</td>
<td>10</td>
<td>4</td>
<td>2166</td>
<td>23.60</td>
</tr>
<tr>
<td>Ball.java</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0.23</td>
</tr>
<tr>
<td>GizmoHandler.java</td>
<td>7</td>
<td>2</td>
<td>563</td>
<td>6.59</td>
</tr>
<tr>
<td>CircleBumper.java</td>
<td>3</td>
<td>2</td>
<td>12</td>
<td>0.44</td>
</tr>
<tr>
<td>ApplicationWindow.java</td>
<td>2</td>
<td>3</td>
<td>67</td>
<td>0.91</td>
</tr>
<tr>
<td>Absorber.java</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>0.37</td>
</tr>
<tr>
<td>FileHandler.java</td>
<td>6</td>
<td>3</td>
<td>144</td>
<td>2.12</td>
</tr>
<tr>
<td>GameWindow.java</td>
<td>3</td>
<td>9</td>
<td>286</td>
<td>3.29</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Developers</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tony</td>
<td>97</td>
<td>79</td>
<td>8632</td>
<td>100.00</td>
</tr>
</tbody>
</table>
Figure 6-20 Activity-time plot and snapshots of related artefacts relevance list over the history of File Demo
Again, a snippet of feedback from Tracy in group G2 showed the insight he obtained while sliding through the history of entities constituting their collaboration space:

**Tracy:**

Sliding through the history of a task or artefact gives you a feel of how things have moved on, especially after sliding through a history of the artefacts I have been associated with…

Having a slide through a task view I can gauge how important an artefact has been to the task over time, I did notice that MainScreen.java has retained high relevance over a long duration now, recently KeyConnectFrame.java has turned out to be high also…

… This gave me the clue that these classes are quite important to the User Interface task…

I got particularly interested in MainScreen.java when I noticed James and Paul have also used this class… I have been the only one working on KeyConnectFrame.java

I believe this information will again be very important to me when carrying out a maintenance task on a system I am not really familiar with…

Tracy’s particular interest had been to be aware of other developers that have been collaborating over the same task he was working on and the code artefacts being used to achieve the task. The feedback from Tracy also supports CRI capability in enhancing contextual awareness of distributed software project processes. Tracy was able to attain awareness cues on the impact of MainScreen.java and KeyConnectFrame.java code artefacts on the User Interface task over which he was collaborating with two other of his colleagues (James and Paul). While he has substantially worked on KeyConnectFrame.java, he had not at any time interacted with MainScreen.java while executing the User Interface task. Irrespective of his not working with MainScreen.java, using CRI he was able to obtain insight on the relative impact of the artefact on the task he was collaborating on. He obtained this awareness cue without having to meet formally or informally with James or Paul to enquire about their extent of work on User Interface.

Figure 6-21 shows the context graph of User Interface demonstrating that its accomplishment was dependent on the three collaborating developers and the use of 71 code artefacts. The obtainment of useful awareness insight into the state of User Interface would have been more difficult without the use of a relevance model such as CRI given the number of developers and the numerous artefacts that have impacted on its state. Each of the developers and code artefacts had different levels of influence on the state of User Interface. Figure 6-22 illustrate six different sequential snapshots of the developers and artefacts relevance lists taken at different intervals over the history of User Interface task. The labelled
entities are the artefacts and developers that Tracy was interested in during his collaboration over User Interface. The screenshots illustrate the relatively high positioning of MainScreen.java over the history of User Interface and the higher positioning of KeyConnectFrame.java at the later phase of achieving the task. The developer relevance list also shows that the early phase of development work on User Interface consisted only of Tracy while James and Paul became associated with the task as the task lifetime progressed. The position of James and Tracy switched on the relevance list at different periods of User Interface development. The illustration of figures 6-21 and 6-22 confirms the awareness cue obtained by Tracy during the development of User Interface.
Figure 6-21 The context graph of User Interface task showing 71 code artefacts and 3 developers involved in achieving the task.
Figure 6-22 Snapshots of developer and artefact relevance list over the history of User Interface task.
Finally, the snippet of feedback from Alex in group G1 on insight he obtained using CRI history mode relevance list is as shown below:

**Alex:**

If there is an artefact that has remained high on the ranking over a considerable time line, it tells me where the main focus or problems has been in the project…

I have been watching the PlayWindow.java and BuildWindow.java classes recently on the Build Mode task…

Although I have never worked much on them, I know they have been important in achieving Build Mode …

I also noticed that classes are high on Luke’s relevance ranking…

He is probably doing a lot of work on it….

The feedback from Alex implies that using CRI he was able to build awareness of the relevance of PlayWindow.java and BuildWindow.java to the Build Mode task. These were the top two artefacts on the Build Mode task artefacts relevance list. Furthermore, he was able to obtain an awareness insight that Luke was more relevant to the state of these two artefacts compared to other developers within the collaboration space. Thus, through the use of CRI Alex was able to achieve an awareness cue of code artefacts that were the main focus of Build Mode task and the developers more relevant to the artefacts that formed this focus. Again, it is believed here that it would have been more difficult for Alex to acquire this awareness cue without a relevance model such as CRI given that Build Mode was dependent on all the developers in the group and 70 code artefacts within the collaboration space (figure 6-23). Each of the developers and code artefacts had different levels of influence on the state of Build Mode.

Figure 6-24 illustrates snapshots of entity relevance lists of the final work context state of Build Mode (label 1a and b), Luke (label 2), PlayWindow.java (label 4a and b) and BuildWindow.java (label 3a and b). Label 1a demonstrates the high relevance positions of PlayWindow.java and BuildWindow.java on the Build Mode artefact relevance list and label 1b shows the high relevance position of Luke compared to other developers to the state of Build Mode. The artefact relevance list for Luke shown in label 2 further demonstrates the relative relevance of PlayWindow.java and BuildWindow.java to Luke’s work context. The developer and task relevance lists for PlayWindow.java and BuildWindow.java work context shown in label 3a, b and 4a, b also confirms the awareness cues stated by Alex in his feedback.
Firstly, the Build Mode task has had significant impact on the state of PlayWindow.java and BuildWindow.java given that they are top positioned on the task relevance list for each of the two code artefacts (label 3b and 4b). Secondly, while Alex had minimal impact on the state of BuildWindow.java (label 3a - bottom in the developer relevance list) Luke has had significant impact on the state of this artefact (label 3a - being top in the developer relevance list). Tony also impacted on the state of BuildWindow.java but not as much as Luke had done. Furthermore, Alex had not at any time worked on PlayWindow.java while Luke again had a more significant impact on the state of PlayWindow.java, Tony also impacted on the state of PlayWindow.java but not as much as Luke had done (label 4a). Amid the numerous artefacts and three developers that had collaborated on Build Mode (figure 6-23), Alex did not need to formally or informally meet with Luke or Tony to obtain insight on their influence on these two artefacts or the Build Mode task or obtain cues on the artefacts that were influencing the achievement of Build Mode.
Figure 6-23 The context graph of Build Mode task showing 70 code artefacts and 3 developers involved in achieving the task.
Figure 6-24 Entity relevance lists used by Alex to obtain awareness of the development work process
The snippet of feedback for Tony shown below again demonstrates the usefulness of CRI for distributed software development. His aim of sliding through the history of entities constituting their collaboration space was to monitor the real-time progress of work on tasks and code artefacts before commits were made to Subversion. It was not possible to obtain concrete evidence of the awareness cue he obtained before his commits to the version management system.

Tony:

It is always good to see over time, how much you have worked on certain code or tasks...

After our group meeting, I will watch to see if there is a certain growth in the task we discussed during the meeting...

I noticed some time delay in processing as the timeline gets longer.

This feature will be more intuitive if timeline definitions are more specifically defined.

Tony’s feedback also hints that sliding through histories of entities could suffer from network latency again as the lifetime of the project extends. Also, the usability of the history slicing feature in CRI could be improved by more intuitive timeline definitions such as the particular date or hour an interaction was carried out.

Identifying bottleneck and risk areas within a shared collaboration space using the social graph visualisation

Social graph visualisation where the size of each entity is proportional to its markov centrality in the network is used to augment awareness information represented in CRI relevance lists. A potential social cue that can be obtained from such graphs is the ability to easily determine bottleneck or high risk entities in a collaboration space. Figures 6-25a and b are the social graph visualisations representing collaborations between developers in different groups over tasks and artefacts in their collaboration space.
Figure 6-25a Social graph visualisations of collaborations between developers in groups G1 and G2 over tasks and artefacts in their collaboration space.
Figure 6-26b Social graph visualisations of collaborations between developers in groups G3 and G4 over tasks and artefacts in their collaboration space.
A dot plot of the subjective level of acceptance on a scale of 1-7 for the outcome of the social graphs in their respective groups is shown in figure 6-26. This shows the general level of acceptance for the relative size representations of entities was within the high spectrum and ranged between 5 and 7 with a greater proportion having an acceptance level of 6.

![Figure 6-27 Participants level of acceptance for the outcome of social graphs during Gizmoball study](image)

The outcome of this study suggest that the social graph visualisations did help improve the awareness of participants and also revealed issues they normally would have not thought of. A snippet of feedback from Tony while presenting his views on the social graph visualisation for group G1 is as shown below. This suggests that he could easily identify problem spots within the graph.

**Tony:**

Will say that is spot on…

It’s easy to see where the biggest problems are… For instance, I can see that the Build Mode task and BuildWindow.java have really been a problem spot…

We have spent a few days really trying to figure some buggy stuff there…

The number of artefacts has made it cluttered and complex…

Similarly, feedback from Alex as shown below illustrates that he used the social graph to get an overall perception of the state of the shared project.

**Alex:**

It’s the fastest way to get all the information from CRI…

I always use graph to get a general state view of the project…

I do check it every few days just to give me a grasp of what is going on with developers in the group and which tasks have had a considerable change recently…

The snippet of feedback from Boris shown below illustrates how he was able to obtain awareness of the true state of the JUnit Test task. Although the responsibility for achieving JUnit Test was not that of Boris, using the social graph he was able to point out that there was still a lot more work to do on the task.
**Boris:**

If we have done JUnit Test how come it only looks at only Gizmo.java, Square.java and GizmoModel.java…?

Because I know that it should be looking at virtually all of the code…

There is something wrong…

This tells that there is more work to be done in JUnit Tests…

One main disadvantage revealed in this study is the complexity of the CRI social graph visualisation as a software system gets larger. As pointed out by Tony and Tracy, it is expected that this would have made it more difficult for developers to obtain awareness information as the number of entities in the collaboration space increased in size. Such a disadvantage may be addressed by building an abstraction mechanism into such graph visualisations where developers can abstract a particular section of the graph for more detailed visualisation.

**6.9. Possible threats to validity**

The appropriate functionality of the CRI model was not dependent on the use of version management systems, wikis or any other collaboration facilitating system and the usage of such systems was not controlled in this study. Of the nine participants that took part in the first study, six utilised a versioning system at one point or another during the study. For the second study, all participants used a versioning system and one of the groups also collaborated using a wiki system. The main threat here is that this research cannot validate if some of the enhanced awareness experiences presented in this study was also influenced by the collaborators use of a versioning system.

A concern in this research has been that of network latency. Such latency will result in poor usage of CRI within slow networks. For the two studies carried out in this research participants worked at different locations including using the fast networks within the university campus and working from home where a few participants complained of having slow network connections. In the first study, two refrained from using CRI over slow networks five days towards the end of the study. Also the CRI tool was designed such that when no connection is detected, it stores CRI events offline which are then offloaded to the server and the relevance of entities recalculated when a connection is detected. In the second study, participants could also switch CRI to an offline mode where slow networks were of major concern. For such cases, participants were able to offload interaction events generated
for resynchronisation with the CRI server at a more suitable time or were prompted each time they logged into the CRI Eclipse plugin.

The final Gizmoball project output involved in the two studies in this research was part of an assessed course; this poses the danger of participants providing positive responses to possibly influence their assessment. Participants were not informed of any reward for taking part in the evaluation. Participation was voluntary and students were clearly informed that being part of the evaluation was not going to give them an edge above others that did not. The lecturer of the module was not involved in the interview and debriefing sessions that were administered to participants in the second study.

The result might have been affected by the impact of the learning process of the CRI tool for participants. In both studies, participants were given a short user manual a week before commencement of the study; this was followed up with a 30 minute tutorial and demonstration on the CRI tool.

Since CRI showed the relevance of developers and tasks in real time, it was also important that developers did not forcibly increase their interaction processes so as to obtain a high relevance or that of their associated tasks within the collaboration space. Therefore participants were not informed of any reward for using the tool and they were free to withdraw from the study at any time.

Though it cannot be claimed that the study samples represented real industrial software developers, the participant sample represented a spread with a mixture of programming expertise from average to excellent, such as may be obtained in a real development scenario. This was necessary as it is difficult to get industrial software developers to be part of such a study.

Another point of concern was the collaborative experience of participants, or rather if participants were already familiar with working with other developers or working in a group scenario. Students of the advanced software engineering class from which the participants were drawn were required to have worked as groups in other small scale development tasks related to previous course modules. They also had experience in collaborative work not related to programming.

This study was carried out in a forward engineering context where participants were mostly required to create new artefacts and update them. It cannot be concluded that the results would be the same in reverse engineering or maintenance setting where developers will be required to update previously existing artefacts.
6.10. Conclusions

This chapter has presented a second study carried out after the proof of concept study was initially performed to investigate the viability of the CRI model. The motivation behind the second study was to investigate four core research questions related to the significance of each CRI model construct; model behaviour in practice compared to theoretical design; correlation between CRI relevance lists and participants’ perception of work effort; and finally, obtain insight on whether CRI can enhance awareness during collaborative work processes.

The outcome of the first research question shows that each combination of the CRI model constructs such as any single or multiple combination of view, update and create interaction types, with or without sphere of influence and periodic decay, or just using sphere of influence alone resulted in a relevance list for a selected work context. The main question was how representative a partial combination of constructs would be of the nature of collaboration that has transpired over a selected work context. In answering this research question, the negative consequence of eliminating any of the constructs from the relevance model was demonstrated. On the whole, a combination of sphere of influence and interaction type construct consisting of view, update and create is considered more appropriate in representing the overall effort dissipated by entities in a selected work context. Furthermore, to represent recent work effort, periodic decay in combination with these constructs appeared most appropriate.

The second research question investigated the CRI model behaviour in practice in comparison to its theoretical design. This was carried out by analysing real collaborative development data captured from six weeks of development effort from advanced software engineering students at the Department of Computer and Information Sciences, University of Strathclyde. The outcome showed that the sphere of influence ratio of entities behaved as expected, growing and shrinking as entities were added to a selected work context or associated with the whole collaboration space. The weightings associated with different interaction types were further justified. Finally, through a controlled injection of a level of activity into a known CRI model state showed that both the history and recent modes responded as predicted.

Thirdly, the correlation between CRI relevance list rankings and participants’ perception of work effort and how CRI impacted on how they would normally carry out their development task was investigated. Feedback from participants suggested that the CRI tool captured between 60-90% of the interaction events carried out over the six week study period.
Participants were also more comfortable with creating general tasks requiring less activation and deactivation sequences. The main insight obtained from investigating this research question was that an automated awareness system such as CRI had better potential to maintain a more accurate ranking of the relevance of entities to a selected work context within a collaboration space than might be possible by human developers. This is because human developers were more likely to be influenced by recent effort or overwhelmed by the amount of work done or the scale of the project. Specific factors that can influence developers’ perception of relevant entities constituting their work context revealed in this study included:

- Greater coding effort over a selected artefact while executing project tasks sometimes does not correspond to a higher perceived level of difficulty associated with the implementation of the associated task or code artefact.
- Developers may form a perception of work effort by clustering closely related code artefacts.
- Developers can sometimes be influenced by the size of an artefact while forming a perception of work effort. This suggest that developers do not always build perception of relevance based on the artefacts they have created, the frequency of views or the amount of absolute update delta generated while updating the artefact.
- Simply copying and pasting of program code from alternative sources into a piece of code artefact can also influence developers’ perception of work effort, difficulty and bugs associated with such code artefacts.

Finally, the fourth research question was intended to obtain insight into how CRI can enhance awareness amongst developers during collaborative work processes. In addressing this research question the study revealed instances where CRI was useful in enhancing the awareness and relevance perception formation processes of participants on their interactions within the collaboration space. This was illustrated in cases where participants lacked adequate awareness and proper judgement of work effort. Participants viewing of CRI relevance lists provided better attainment of awareness of previous work effort.

The fourth research question also address the core objective of this research by obtaining insight into how CRI can enhance contextual awareness cues in distributed software development spaces. Particular attention was given to cases where there exist complex dependencies amongst entities involved in a shared software project. This insight was obtained empirically by narrating developer experiences and recreating development paths to investigate the validity of such developer experience. The outcome of the study showed that
CRI can enhance awareness cues in distributed software development particularly in cases where there exist complex dependencies amongst entities. Particular instances include:

- The study revealed an instance where, with the use of the CRI Eclipse tool, a developer did not need any formal or informal collocated meeting with other members of a development group to obtain awareness insight into the state of a project task. The awareness insight obtained included knowledge of a fellow developer that had been working on a task, and the relative work effort the developer has put into artefacts he is using to accomplish the task.

- The study also narrates an instance where, with the use of the CRI Eclipse tool, a developer was able to build real-time contextual awareness of the relative work effort of other developers that have been collaborating over the same task he was working on and the code artefacts being used to achieve the task.

- Again, the study narrated an instance where, with the use of CRI Eclipse tool, a developer was able to attain contextual awareness cues of code artefacts that were the main focus of a task and the developers most relevant to the artefacts that formed this focus.

- Finally, with the use of CRI eclipse tool, a developer was able to monitor the real-time progress of work on tasks and code artefacts before commits were made to Subversion.

In each of these instances stated, it would have been more difficult for the developers to obtain the awareness cues narrated in a distributed setting without the use of a relevance model such as CRI. This is because, in all the cases narrated, there was a complex web of dependencies amongst developers, tasks and code artefacts that existed in the shared collaboration space.

Furthermore, the study revealed instances of how CRI relevance lists and social graphs can enable developers distributed in time or space to obtain awareness cues of overall and current work effort expended within a selected work context and the extent of relevance of entities to such a selected context. This contextual awareness was also obtained amid the complex dependencies that existed amongst the developers, tasks and artefacts that constituted the workspace.
Chapter 7 Conclusions and future work

This thesis has presented and evaluated a model intended to enhance contextual awareness in distributed, collaborative software engineering spaces where developers are free to work at any time and in any location. Key results demonstrate that it is possible to derive real-time relevance rankings of project entities that exist in collaborative space by monitoring developer interactions. These relevance rankings have been used to derive: an indication of the overall work effort of individual developers in particular work contexts through the history mode, as well as their current work through the recent mode; an indication of which tasks and artefacts have consumed most effort over all developers; a history slicing capability that allows a developer in a particular work context to ‘playback’ the development process; and a social network graph that provides an abstract view, void of context, of the overall state of a project which can help determine potential bottlenecks and the potential implications of deleting or modifying artefacts, updating tasks or adding or removing developers from a project.

Two empirical studies were carried out in the course of this research. Firstly, an initial proof of concept study gave an insight into the viability of the model and understanding of the risks associated with properly carrying out a deeper study. In the first study, the feasibility of the model was demonstrated. After exploring this feasibility, model implementation changes were made. These changes were aimed at properly capturing and integrating developer interaction data generated without the bounds of time and space. Network latency and privacy concerns were the main issues that came up. Insights were also obtained into how CRI may affect the way developers carry out a development task. The results suggested that while developers considered it challenging to always formally define the task they are working on, they benefited from the outcome of doing so.

Finally, insight was obtained from the first study into the awareness capabilities of the CRI relevance list visualisation. Results suggested that there was a level of correspondence between participants’ perception of work effort and the CRI relevance list. There were two instances of slowly increasing colour intensity to match work effort. These instances came from participants considered to have generated substantial interaction events when compared to the overall interaction events from other developers that participated in the study. This suggested that there were cases where the rate of change of the colour intensity representing an entity was not matching the developer’s perception of pace of development in the selected
work context. As a precautionary measure, weights associated with interaction events were
scaled by multiplying initial weighting factors associated with interaction types by 10. This
choice was based on usage statistics generated during the use of CRI in the implementation
of the next version of the plugin compared to the output generated during the first study. On
the whole, the scaling increased the rate of change of colour intensity representing an entity
in a work context and did not affect the relevance position of entities on the relevance list.

The second empirical study involved a deeper subjective and more analytical evaluation of
the CRI model using a small but realistic developer collaboration scenario similar to the first
study. Using this study, it was possible to justify each of the components that constitute the
realisation of the CRI model. In particular, both SOI - representing the importance of an
entity in a collaborative space - and periodic decay – reducing the importance of inactive
elements in recent mode - were shown to have a clear impact on the relevance rankings in
keeping with the CRI model design. The significance of capturing each of the core
interaction event types - create, view and update - was also demonstrated.

Secondly, using the second study, it was possible to validate that the model works according
to its design in real development scenarios. Analysis of developer interaction data showed
that the SOI ratio of entities behaved as expected, growing and shrinking as entities were
added to a selected work context or associated with the collaboration space. This study also
showed how CRI captured the different interaction types, their relative frequency, and the
justification for their weightings in the CRI model. Finally, through a controlled injection of
activity into a known CRI model state, it was demonstrated that both the history and recent
modes responded as predicted.

The second study also investigated whether the model can support awareness during
collaborative development and highlighted three areas of strength: a number of examples
were identified where it appeared that CRI was maintaining more accurate relevance
rankings than individual developers; developers used the history slider to ‘replay’ the project
development to help enhance their understanding of who had contributed what at each stage
of development and what tasks and artefacts were most relevant throughout the project
lifecycle; and, the social graph view of CRI was shown to provide an effective high level
summary of a collaborative project – showing what entities were important and also
highlighting areas where development was less than perhaps it should have been.

On the whole, in contrast to related work, this research has focused on the realisation and
evaluation of a model that can enable collaborators to obtain contextual awareness based on
tasks, developers and artefacts that are being used to achieve a distributed software project.
For a selected task instance, awareness of the relative impact of project developers and code artefacts is provided. Similarly, for a selected code artefact, awareness is also provided of the relative impact of project tasks and developers on the state of the code artefact. Finally, for a selected developer, awareness is provided of the relative impact of tasks and code artefacts on the work context of the developer. The relevance model is based on a collaborative perspective rather than an individual developer perception.

This thesis has also identified some core limitations in the CRI model. These limitations are discussed below and form the basis for future work. They mainly lie in the design, implementation, evaluation and application domain of CRI.

Firstly, in the design of CRI, no dependency constraints were enforced amongst entity instances of the same type. This suggests that CRI cannot be used to derive direct knowledge of the relevance of a selected task to any other task instance existing in a collaboration space. Similarly, CRI cannot be used to derive direct knowledge of the relevance of a selected artefact to any other artefact instance. While there are expected benefits in such constraints, enforcing them would require further static and dynamic analysis of code artefacts to identify dependency relations amongst tasks and sub-tasks, including the articulation of scheduling and precedence of tasks. It also demands an understanding of complex group dynamics of how developers and tasks depend on each other beyond relationships derived from structural and more explicit constraints. A less explicit constraint that has not been considered in this thesis is based on the general notion of work context being a highly structured amalgam of informational, physical, and conceptual resources existing in a collaboration space. This amalgam goes beyond the articulation and representation of work context based on relational properties alone, to include the modelling of developer’s concepts, task state, social relations and work culture.

Secondly, CRI assumes a general view of a task without exploring its nature which can range from maintenance, debugging, refactoring or simply forward engineering. Related work by Fritz et al. [FMH07] pointed out that the nature of a development task could influence the level of knowledge developers achieve related to the code they are collaborating on.

Further work would investigate different modalities of representing and recording tasks. To appropriately model the nature of a development task and its impact on CRI relevance rankings, it is important to gain a proper understanding of which types of task most affect the process of knowledge attainment. For instance, Fritz et al.’s. study pointed out that developers had less knowledge of a code artefact used in achieving a debugging task compared to a forward engineering task where they created or updated code artefacts.
View interactions have been articulated as events which indirectly affect the state of entity instances within a collaboration space. Detecting view events that are random or spurious has been proposed based on the frequency of its occurrence. Further work could investigate using the duration of a view event to determine if it is spurious or relevant. Here it can be assumed that a random view will last for a shorter duration compared to a relevant view event.

One of the findings in the second evaluation study was the effect of ‘copy and paste’ operations on the relevance perception formation process of collaborating developers. Further work could investigate the introduction of another event type representing a ‘copy and paste’ operation. Alternatively, the assignment of weightings to update events can be carried out based on the nature of updates; for instance, updates via keyboards can be modelled with greater relevance compared to updates via ‘copy and paste’ operations. A number of attributes such as the source and target of operation and code ownership rules need to be investigated. This will enable appropriate understanding of the relative impact of a ‘copy and paste’ operations on the relevance of associated entities within a work context.

Finally, the modelling of CRI does not explore the influence of developer experience. There is a need to explore the level of knowledge held by developers with more professional work experience and examine how this knowledge impacts on the relevance ranking of entities to a selected work context.

A possible extension of this work is to integrate the CRI Eclipse plugin with other software development tools such as Bugzilla28, Subversion or CVS. This would enable the monitoring of interaction events that are not captured and represented in CRI when developers are not logged into the tool. Integrating into these external tools will also enable the monitoring of wider scope of collaboration processes. For instance CVS also keeps a log of who, when and why changes occurred. Analysis of such information could be vital in determining the relevance of entities to a selected work context.

Furthermore, while there was an attempt to improve on the challenges related to network latency cited in the first study, sporadic cases of network problems still arose in the second study, some of which were not within the control of the CRI model implementation. Understandably, privacy concerns remain a core limitation to the awareness benefit of the CRI model. Developers are naturally hesitant for their work to be monitored for a number of reasons. This can be partially addressed by an appropriate management attitude and also

---

28 http://www.bugzilla.org/
CRI’s implementation facility which allows developers to switch monitors on and off at any stage. Serious ethical considerations exist in the real world use of CRI as it could be abused as the basis of capability judgment and reward structuring.

The evaluation in this thesis could be further strengthened from a number of viewpoint. Firstly, while CRI endeavours to generate a relevance ranking of developers to different task or artefact work contexts, there is no insight into what kind of knowledge developers have attained regarding these work contexts. It is expected that the kind of knowledge obtained will vary and can be dependent on a number of variables including: the nature of an ongoing task; artefact code patterns; work experience; duration of collaboration; and the type of interaction events developers are associated with.

Secondly, the evaluation presented in this thesis sought to obtain an empirical probability of correlation between the CRI history mode with three main collaboration variables. These variables included: overall developer work effort, bugs associated with artefacts, and difficulty experienced while developing artefacts. The CRI recent mode was also correlated with developer recent work effort. The results did show that it was least likely that CRI history mode relevance rankings corresponded with the level of difficulty developers experienced during collaboration. Future work could investigate how other collaboration dynamics such as the nature of task, developer work experience and artefact structure can affect the level of difficulty experience by developers during collaborative software development. More insight is also required of how these collaboration dynamics affect a developer’s perception of recent and overall work effort and how they can be taken into account in the CRI relevance rankings.

Further studies could investigate the use of CRI in an agile or extreme programming development environment. In such environments every user story (a software system requirement) is viewed as a task. In such an environment the creation, activation and deactivation of tasks as work context changes is more intuitive. This is because a task is created or activated as new user stories are implemented and deactivated when completed.

Finally, this thesis has not investigated the potential benefit of applying different software visualisation mechanisms that might use the output of CRI model. It could well have been that the relevance list visualisation and social network graphs used in this work are not the optimal mechanism for enhancing the attainment of contextual awareness information amongst collaborators distributed in time and space, and that other representations may be more effective.
Insight into the scalability of CRI into real industrial projects with more extended lifecycles is required. There is also potential to extend the capability of the CRI model beyond software engineering to other application domains. CRI is a generic model that assumes that, irrespective of domain, collaboration processes always consist of collaborators working in a group over a set of tasks using a number of resources. That is not to say that there are no finer details to investigate. Domain specific processes such as the way interactions are monitored, the nature of tasks, and group dynamics would need to be analysed, but the potential exists for CRI to raise awareness in a range of collaborative projects.
References


174


Appendix A1: CRI History mode formal description

A shared project space consists of a set of developers $D$, tasks $T$ and artefacts $A$. At the start of the project, $D$, $T$ and $A$ are all empty sets. Furthermore, each entity comprising $D$, $T$ and $A$ initially commences with an empty work context set. That is:

$$D = \emptyset, \quad T = \emptyset \quad \text{and} \quad A = \emptyset$$

If an interaction event $e_1$ is defined by an active task $t_1$, artefact $a_1$ and developer $d_1$ respectively at the start of collaboration, then the work context of $d_1$ will be defined by $t_1$ and $a_1$; the work context of $t_1$ is defined by $d_1$ and $a_1$; and the work context of $a_1$ is defined by $t_1$ and $d_1$. That is:

$$D = \{d_1\}, \quad T = \{t_1\} \quad \text{and} \quad A = \{a_1\}$$

$$t_1 \text{ context } = \{d_1, a_1\}$$

$$d_1 \text{ context } = \{t_1, a_1\}$$

$$a_1 \text{ context } = \{d_1, t_1\}$$

The relevance values $x_{(t_1)t_1}$ and $x_{(t_1)a_1}$ gained by $t_1$ and $a_1$ within $d_1$’s work context in CRI history mode is given as:

$$x_{(t_1)t_1(d_1 \text{ context})} = e_1 \text{ weight } \times d_1 \text{ soi} \quad ; \quad x_{(t_1)a_1(d_1 \text{ context})} = e_1 \text{ weight } \times d_1 \text{ soi}$$

$e_1 \text{ weight }$ - Weight associated with the type of interaction as shown in table 4-1

$$d_1 \text{ soi } = \frac{|d_1 \text{ context}|}{|T| + |A|} = \frac{2}{2} = 1$$

Similarly, the relevance values $x_{(t_1)d_1}$ and $x_{(t_1)a_1}$ gained by $d_1$ and $a_1$ within $t_1$ work context; and relevance values $x_{(t_1)t_1}$ and $x_{(t_1)d_1}$ gained by $d_1$ and $t_1$ within $a_1$ work context in CRI history mode is given as:

$$x_{(t_1)d_1(t_1 \text{ context})} = e_1 \text{ weight } \times t_1 \text{ soi} \quad ; \quad x_{(t_1)a_1(t_1 \text{ context})} = e_1 \text{ weight } \times t_1 \text{ soi}$$

$$x_{(t_1)d_1(a_1 \text{ context})} = e_1 \text{ weight } \times a_1 \text{ soi} \quad ; \quad x_{(t_1)t_1(a_1 \text{ context})} = e_1 \text{ weight } \times a_1 \text{ soi}$$

A subsequent interaction event $e_2$ during collaboration is defined by an active task $t_1$, artefact $a_1$ and developer $d_2$ respectively. Then the work context of $d_1$ remains the same while the work context of $d_2$ will be defined by $t_1$ and $a_1$; the work context of $t_1$ is further defined by $d_2$; and the work context of $a_1$ is further defined by $d_2$. That is:

$$D = \{d_1, d_2\}, \quad T = \{t_1\} \quad \text{and} \quad A = \{a_1\}$$

$$t_1 \text{ context } = \{d_1, a_1, d_2\}$$

$$d_1 \text{ context } = \{t_1, a_1\}, \quad d_2 \text{ context } = \{t_1, a_1\}$$

$$a_1 \text{ context } = \{d_1, t_1, d_2\}$$
The relevance values $x_{(2)\mathbf{t}_1}$ and $x_{(2)\mathbf{a}_1}$ gained by $\mathbf{t}_1$ and $\mathbf{a}_1$ within $\mathbf{d}_2$ work context in CRI history mode is given as:

$$x_{(2)\mathbf{t}_1(\mathbf{d}_2\text{ context})} = e_2 \text{ weight} \ast \mathbf{d}_2 \text{ soi}; \quad x_{(2)\mathbf{a}_1(\mathbf{d}_2\text{ context})} = e_2 \text{ weight} \ast \mathbf{d}_2 \text{ soi}$$

The relevance values $x_{(1)\mathbf{t}_1(\mathbf{d}_1\text{ context})}$ and $x_{(1)\mathbf{a}_1(\mathbf{d}_1\text{ context})}$ of $\mathbf{t}_1$ and $\mathbf{a}_1$ respectively within $\mathbf{d}_1$’s work context in CRI history mode remain unchanged since the interaction event $e_2$ does not involve $\mathbf{d}_1$’s work context.

The cumulative relevance value $x_{(2)\mathbf{a}_1}$ gained by $\mathbf{a}_1$ within $\mathbf{t}_1$’s work context is given as:

$$x_{(2)\mathbf{a}_1(\mathbf{t}_1\text{ context})} = x_{(1)\mathbf{a}_1(\mathbf{t}_1\text{ context})} + e_2 \text{ weight} \ast \mathbf{t}_1 \text{ soi}$$

The relevance value $x_{(2)\mathbf{d}_2}$ gained by $\mathbf{d}_2$ within $\mathbf{t}_1$’s work context is given as:

$$x_{(2)\mathbf{d}_2(\mathbf{t}_1\text{ context})} = e_2 \text{ weight} \ast \mathbf{t}_1 \text{ soi}$$

Furthermore, the relevance value of $\mathbf{d}_2$ within $\mathbf{t}_1$’s work context remains unchanged given that $\mathbf{d}_2$ is not defined in $e_2$. That is:

$$x_{(2)\mathbf{d}_2(\mathbf{t}_1\text{ context})} = x_{(1)\mathbf{d}_1(\mathbf{t}_1\text{ context})}$$

Similarly, the cumulative relevance values $x_{(2)\mathbf{t}_1}$, $x_{(2)\mathbf{d}_1}$, and $x_{(2)\mathbf{d}_2}$ attained by $\mathbf{t}_1$, $\mathbf{d}_1$, and $\mathbf{d}_2$ respectively within $\mathbf{a}_1$’s work context is given as:

$$x_{(2)\mathbf{t}_1(\mathbf{a}_1\text{ context})} = x_{(1)\mathbf{t}_1(\mathbf{a}_1\text{ context})} + e_2 \text{ weight} \ast \mathbf{a}_1 \text{ soi}$$

$$x_{(2)\mathbf{d}_1(\mathbf{a}_1\text{ context})} = x_{(1)\mathbf{d}_1(\mathbf{a}_1\text{ context})}$$

$$x_{(2)\mathbf{d}_2(\mathbf{a}_1\text{ context})} = e_2 \text{ weight} \ast \mathbf{a}_1 \text{ soi}$$

In general, the work context of a task $\mathbf{t}_m$ that consists of developer $\mathbf{d}_1$ and artefacts $\mathbf{a}_1$, $\mathbf{a}_2$ is represented as:

$$\mathbf{t}_m\text{ context} = \{\mathbf{d}_1, \mathbf{a}_1, \mathbf{a}_2\}$$

Furthermore, given a subsequent event $e_n$ defined by developer $\mathbf{d}_y$, task $\mathbf{t}_m$ and artefact $\mathbf{a}_2$, the following impact on the work context $\mathbf{t}_m$ and the relevance values of entities that constitute the work context in the history mode are observed:

1. $\mathbf{t}_m\text{ context} = \{\mathbf{d}_1, \mathbf{a}_1, \mathbf{a}_2, \mathbf{d}_y\}$
2. The relevance value $x_{(n)\mathbf{d}_y}$ gained by $\mathbf{d}_y$ within $\mathbf{t}_m$’s work context is represented as follows:

$$x_{(n)\mathbf{d}_y} = e_n \text{ weight} \ast \mathbf{t}_m \text{ soi}$$

3. The relevance value of $\mathbf{d}_1$ within $\mathbf{t}_m$’s work context remains unchanged. That is:

$$x_{(n)\mathbf{d}_1} = x_{(n-1)\mathbf{d}_1}$$

4. The relevance value $x_{(n)\mathbf{a}_2}$ gained by $\mathbf{a}_2$ within $\mathbf{t}_m$’s work context is represented as follows:

$$x_{(n)\mathbf{a}_2} = x_{(n-1)\mathbf{a}_2} + e_n \text{ weight} \ast \mathbf{t}_m \text{ soi}$$
Similar observations can also be made about entities that constitute the work context of $d_y$ and $a_z$. 
Appendix A2: CRI Recent mode formal description

Assuming a shared project space consists of a set of developers \( D \), tasks \( T \) and artefacts \( A \). At the start of project, \( D \), \( T \) and \( A \) are all empty sets. That is:

\[
D = \emptyset, \quad T = \emptyset \quad \text{and} \quad A = \emptyset
\]

Given an interaction event \( e_1 \) defined by an active task \( t_1 \), artefact \( a_1 \) and developer \( d_1 \) respectively at the start of collaboration then the work context of \( d_1 \) will be defined by \( t_1 \) and \( a_1 \); the work context of \( t_1 \) is defined by \( d_1 \) and \( a_1 \); and the work context of \( a_1 \) is defined by \( t_1 \) and \( d_1 \). That is:

\[
D = \{ d_1 \}, \quad T = \{ t_1 \} \quad \text{and} \quad A = \{ a_1 \}
\]

\( t_1 \text{ context} = \{ d_1, a_1 \} \)

\( d_1 \text{ context} = \{ t_1, a_1 \} \)

\( a_1 \text{ context} = \{ d_1, t_1 \} \)

The relevance values \( x_{(1)} t_1 \) and \( x_{(1)} a_1 \) gained by \( t_1 \) and \( a_1 \) within \( d_1 \)'s work context in CRI recent mode is given as:

\[
x_{(1)} t_1(d_1 \text{ context}) = e_1 \text{ weight} \times d_1 \text{ soi} \quad \text{and} \quad x_{(1)} a_1(d_1 \text{ context}) = e_1 \text{ weight} \times d_1 \text{ soi}
\]

Where \( e_1 \text{ weight} \) is the weight associated to the type of interaction event

\[
\text{and} \quad d_1 \text{ soi} = \frac{|d_1 \text{ context}|}{|T|+|A|} = \frac{2}{2} = 1
\]

Similarly, the relevance values \( x_{(1)} d_1 \) and \( x_{(1)} a_1 \) gained by \( d_1 \) and \( a_1 \) within \( t_1 \)'s work context; and \( x_{(1)} t_1 \) and \( x_{(1)} d_1 \) gained by \( d_1 \) and \( t_1 \) within \( a_1 \)'s work context in CRI recent mode is given by:

\[
x_{(1)} d_1(t_1 \text{ context}) = e_1 \text{ weight} \times t_1 \text{ soi} \quad \text{and} \quad x_{(1)} a_1(t_1 \text{ context}) = e_1 \text{ weight} \times t_1 \text{ soi}
\]

\[
x_{(1)} t_1(a_1 \text{ context}) = e_1 \text{ weight} \times a_1 \text{ soi} \quad \text{and} \quad x_{(1)} d_1(a_1 \text{ context}) = e_1 \text{ weight} \times a_1 \text{ soi}
\]

Given a subsequent interaction event \( e_2 \) during collaboration defined by an active task \( t_2 \), artefact \( a_2 \) and developer \( d_2 \) respectively, then the work context of \( d_2 \) remains the same while the work context of \( d_2 \) will be defined by \( t_2 \) and \( a_2 \); the work context of \( t_2 \) is further defined by \( d_2 \); and the work context of \( a_2 \) is further defined by \( d_2 \). That is:

\[
D = \{ d_1, d_2 \}, \quad T = \{ t_1 \} \quad \text{and} \quad A = \{ a_1 \}
\]

\( t_1 \text{ context} = \{ d_1, a_1, d_2 \} \)

\( d_1 \text{ context} = \{ t_1, a_1 \} \quad d_2 \text{ context} = \{ t_1, a_1 \} \)

\( a_1 \text{ context} = \{ d_1, t_1, d_2 \} \)

The relevance values \( x_{(2)} t_1 \) and \( x_{(2)} a_1 \) gained by \( t_1 \) and \( a_1 \) within \( d_2 \)'s work context in CRI recent mode is given as:
The relevance values $x_{(1)t_1(d_1 \text{context})}$ and $x_{(1)a_1(d_1 \text{context})}$ of $t_1$ and $a_1$ respectively within $d_1$ work context in CRI recent mode remains unchanged since the interaction event $e_2$ does not involve $d_1$’s work context.

The cumulative relevance value $x_{(2)a_1}$ gained by $a_1$ within $t_1$’s work context is given as:

$$x_{(2)a_1(t_1 \text{context})} = x_{(1)a_1(t_1 \text{context})} + e_2 \text{ weight } t_1 \text{ soi}$$

The relevance value $x_{(2)d_2}$ gained by $d_2$ within $t_1$’s work context is given as:

$$x_{(2)d_2(t_1 \text{context})} = e_2 \text{ weight } t_1 \text{ soi}$$

Furthermore, the relevance value of $d_1$ within $t_1$’s work context decays by a periodic decay factor which is determined by the weight associated with the type of event defined by $e_2$ and the sphere of influence of $t_1$. This is because $d_1$ is part of the work context of $t_2$ but is not defined in event $e_2$. That is:

$$x_{(2)d_1(t_1 \text{context})} = x_{(1)d_1(t_1 \text{context})} - e_2 \text{ weight } t_1 \text{ soi}$$

Similarly, the cumulative relevance values $x_{(2)t_1}$, $x_{(2)d_1}$ and $x_{(2)d_2}$ attained by $t_1$, $d_1$ and $d_2$ respectively within $a_1$’s work context is given as:

$$x_{(2)t_1(a_1 \text{context})} = x_{(1)t_1(a_1 \text{context})} + e_2 \text{ weight } a_1 \text{ soi}$$

$$x_{(2)d_1(a_1 \text{context})} = x_{(1)d_1(a_1 \text{context})} - e_2 \text{ weight } a_1 \text{ soi}$$

$$x_{(2)d_2(a_1 \text{context})} = e_2 \text{ weight } a_1 \text{ soi}$$

In general, the work context of a task $t_m$ that consists of a developer $d_1$ and artefacts $a_1$, $a_2$ is represented as:

$$t_m \text{ context } = \{d_1, a_1, a_2\}$$

Furthermore, given a subsequent event $e_n$ defined by developer $d_y$, task $t_m$ and artefact $a_x$, the following impact on the work context $t_m$ and the relevance values of entities that constitute the work context in the recent mode are observed:

5. $t_m \text{ context } = \{d_1, a_1, a_2, d_y\}$

6. The relevance value $x_{(n)d_y}$ gained by $d_y$ within $t_m$’s work context is represented as follows:

$$x_{(n)d_y} = e_n \text{ weight } t_m \text{ soi}$$

7. The relevance value of $d_1$ within $t_m$’s work context decays by a periodic decay factor which is determined by the weight associated with the type of event defined by $e_n$ and the sphere of influence of $t_m$. That is:

$$x_{(n)d_1} = x_{(n-1)d_1} - e_n \text{ weight } t_m \text{ soi}$$

8. The relevance value $x_{(n)a_x}$ gained by $a_x$ within $t_m$’s work context is represented as follows:
\[ x(n)a_x = x(n-1)a_x + e_{n, \text{weight}} \cdot t_{m, \text{soi}} \]

Similar observations can also be made about entities that constitute the work context of \( d_y \) and \( a_y \).
Appendix A3: Details of an example project

Table A3-1 shows the interaction events that formed ‘Purchase ticket’ task and the variation of the sphere of influence ratio across the work context. This table shows the number of developers and artefacts that constitutes the work context of ‘Purchase ticket’ relative to the total developers and artefacts that are currently in the collaboration space for every event. Furthermore, table A3-2 and A3-3 shows the history and recent mode cumulative relevance values attained by entities constituting the ‘Purchase ticket’ work context for every interaction event that impacted on the work context of the task.

Figures A3-1 to A3-7 are plots of the CRI history and recent mode cumulative relevance values versus time for entities that constitute the TickX example project.

Details of White and Smyth’s approach to determining entity centrality based on the inverse of mean first-passage time is presented in A3-SNA. This approach was used to derive CRI’s social network graphs.

<table>
<thead>
<tr>
<th>Interaction type</th>
<th>Active artefact</th>
<th>Active developer</th>
<th>Developers in context</th>
<th>Artefacts in context</th>
<th>Total artefacts</th>
<th>Total developers</th>
<th>SOI Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create</td>
<td>Account.java</td>
<td>Amy</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Update delta = 300</td>
<td>Account.java</td>
<td>Amy</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>Update delta = 150</td>
<td>Account.java</td>
<td>Amy</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>View</td>
<td>Account.java</td>
<td>Bill</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>View</td>
<td>MovieCatalog.java</td>
<td>Bill</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>Create</td>
<td>Ticket.java</td>
<td>Bill</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>Update delta = 175</td>
<td>Ticket.java</td>
<td>Bill</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>View</td>
<td>MovieCatalog.java</td>
<td>Bill</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>14</td>
<td>Create</td>
<td>Booking.java</td>
<td>Bill</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>Update delta = 70</td>
<td>Booking.java</td>
<td>Bill</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>16</td>
<td>View</td>
<td>Cinema.java</td>
<td>Amy</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>17</td>
<td>View</td>
<td>Booking.java</td>
<td>Amy</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>18</td>
<td>Create</td>
<td>Customer.java</td>
<td>Amy</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>19</td>
<td>Update delta = 84</td>
<td>Customer.java</td>
<td>Amy</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>20</td>
<td>Update delta = 25</td>
<td>Booking.java</td>
<td>Amy</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>21</td>
<td>Update delta = 5</td>
<td>Booking.java</td>
<td>Amy</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>24</td>
<td>View</td>
<td>MovieCatalog.java</td>
<td>Amy</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>25</td>
<td>Update delta = 60</td>
<td>MovieCatalog.java</td>
<td>Amy</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>25</td>
<td>Update delta = 90</td>
<td>Booking.java</td>
<td>Bill</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>
Table A3-2 History mode cumulative relevance values of entity instances constituting the 'Purchase ticket' work context for every interaction event influencing the context

<table>
<thead>
<tr>
<th>Interaction type</th>
<th>SOI</th>
<th>Relevance value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ratio</td>
<td>Account.java</td>
</tr>
<tr>
<td>1 Create</td>
<td>1.0000</td>
<td>0.0100</td>
</tr>
<tr>
<td>2 Update delta = 300</td>
<td>1.0000</td>
<td>0.0400</td>
</tr>
<tr>
<td>5 Update delta = 150</td>
<td>0.5000</td>
<td>0.0475</td>
</tr>
<tr>
<td>7 View</td>
<td>0.5000</td>
<td>0.0480</td>
</tr>
<tr>
<td>8 View</td>
<td>0.6667</td>
<td>0.0480</td>
</tr>
<tr>
<td>11 Create</td>
<td>0.6250</td>
<td>0.0480</td>
</tr>
<tr>
<td>12 Update delta = 175</td>
<td>0.6250</td>
<td>0.0480</td>
</tr>
<tr>
<td>13 View</td>
<td>0.6250</td>
<td>0.0480</td>
</tr>
<tr>
<td>14 Create</td>
<td>0.6667</td>
<td>0.0480</td>
</tr>
<tr>
<td>15 Update delta = 70</td>
<td>0.6667</td>
<td>0.0480</td>
</tr>
<tr>
<td>16 View</td>
<td>0.7778</td>
<td>0.0480</td>
</tr>
<tr>
<td>17 View</td>
<td>0.7778</td>
<td>0.0480</td>
</tr>
<tr>
<td>18 Create</td>
<td>0.8000</td>
<td>0.0480</td>
</tr>
<tr>
<td>19 Update delta = 84</td>
<td>0.8000</td>
<td>0.0480</td>
</tr>
<tr>
<td>20 Update delta = 25</td>
<td>0.8000</td>
<td>0.0480</td>
</tr>
<tr>
<td>21 Update delta = 5</td>
<td>0.8000</td>
<td>0.0480</td>
</tr>
<tr>
<td>24 View</td>
<td>0.8000</td>
<td>0.0480</td>
</tr>
<tr>
<td>25 Update delta = 60</td>
<td>0.8000</td>
<td>0.0480</td>
</tr>
<tr>
<td>25 Update delta = 90</td>
<td>0.8000</td>
<td>0.0480</td>
</tr>
</tbody>
</table>
Table A3- 3 Recent mode cumulative relevance values of entity instances constituting the 'Purchase ticket' work context for every interaction event influencing the context

<table>
<thead>
<tr>
<th>Interaction type</th>
<th>SOI Ratio</th>
<th>Account.java</th>
<th>MovieCatalog.java</th>
<th>Ticket.java</th>
<th>Booking.java</th>
<th>Cinema.java</th>
<th>Customer.java</th>
<th>Amy</th>
<th>Bill</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create</td>
<td>1.0000</td>
<td>0.0100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Update delta = 300</td>
<td>1.0000</td>
<td>0.0400</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Update delta = 150</td>
<td>0.5000</td>
<td>0.0475</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>View</td>
<td>0.5000</td>
<td>0.0480</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>View</td>
<td>0.6667</td>
<td>0.0473</td>
<td>0.0007</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Create</td>
<td>0.6250</td>
<td>0.0411</td>
<td>-0.0056</td>
<td>0.0063</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Update delta = 175</td>
<td>0.6250</td>
<td>0.0301</td>
<td>-0.0165</td>
<td>0.0172</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>View</td>
<td>0.6250</td>
<td>0.0295</td>
<td>-0.0159</td>
<td>0.0166</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Create</td>
<td>0.6667</td>
<td>0.0229</td>
<td>-0.0226</td>
<td>0.0099</td>
<td>0.0067</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Update delta = 70</td>
<td>0.6667</td>
<td>0.0182</td>
<td>-0.0272</td>
<td>0.0052</td>
<td>0.0113</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>View</td>
<td>0.7778</td>
<td>0.0174</td>
<td>-0.0280</td>
<td>0.0045</td>
<td>0.0106</td>
<td>0.0008</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>View</td>
<td>0.7778</td>
<td>0.0166</td>
<td>-0.0288</td>
<td>0.0037</td>
<td>0.0113</td>
<td>0.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Create</td>
<td>0.8000</td>
<td>0.0086</td>
<td>-0.0368</td>
<td>-0.0043</td>
<td>0.0033</td>
<td>-0.0080</td>
<td>0.0080</td>
<td>0.0267</td>
</tr>
<tr>
<td>19</td>
<td>Update delta = 84</td>
<td>0.8000</td>
<td>0.0019</td>
<td>-0.0435</td>
<td>-0.0110</td>
<td>-0.0034</td>
<td>-0.0147</td>
<td>0.0147</td>
<td>0.0335</td>
</tr>
<tr>
<td>20</td>
<td>Update delta = 25</td>
<td>0.8000</td>
<td>-0.0001</td>
<td>-0.0455</td>
<td>-0.0130</td>
<td>-0.0014</td>
<td>-0.0167</td>
<td>0.0127</td>
<td>0.0355</td>
</tr>
<tr>
<td>21</td>
<td>Update delta = 5</td>
<td>0.8000</td>
<td>-0.0005</td>
<td>-0.0459</td>
<td>-0.0134</td>
<td>-0.0010</td>
<td>-0.0171</td>
<td>0.0123</td>
<td>0.0359</td>
</tr>
<tr>
<td>24</td>
<td>View</td>
<td>0.8000</td>
<td>-0.0013</td>
<td>-0.0451</td>
<td>-0.0142</td>
<td>-0.0018</td>
<td>-0.0179</td>
<td>0.0115</td>
<td>0.0367</td>
</tr>
<tr>
<td>25</td>
<td>Update delta = 60</td>
<td>0.8000</td>
<td>-0.0061</td>
<td>-0.0403</td>
<td>-0.0190</td>
<td>-0.0066</td>
<td>-0.0227</td>
<td>0.0067</td>
<td>0.0415</td>
</tr>
<tr>
<td>25</td>
<td>Update delta = 90</td>
<td>0.8000</td>
<td>-0.0133</td>
<td>-0.0475</td>
<td>-0.0262</td>
<td>0.0006</td>
<td>-0.0299</td>
<td>-0.0005</td>
<td>0.0343</td>
</tr>
</tbody>
</table>

187
Figure A3-1 Plot of CRI recent mode cumulative relevance values versus time for artefacts and developers constituting the 'Purchase ticket' work context

Figure A3-2 Plot of CRI history mode cumulative relevance values versus time for artefacts and developers constituting the 'Purchase ticket' work context

Figure A3-3 Plot of CRI history mode cumulative relevance values versus time for artefacts and tasks constituting the Amy’s work context
Figure A3- 4 Plot of CRI recent mode cumulative relevance values versus time for artefacts and tasks constituting the Amy’s work context

Figure A3- 5 Plot of CRI history mode cumulative relevance values versus time for tasks and developers constituting the MovieCatalog.java work context

Figure A3- 6 Plot of CRI Recent mode cumulative relevance values versus time for tasks and developers constituting the MovieCatalog.java work context
A3-SNA: White and Smyth’s approach to determining entity centrality based on inverse of mean first-passage time.

White and Smyth [WS03] interpreted the fraction of time that a token spends at any single node as being proportional to an estimate of the global importance or centrality of the node relative to all other nodes in a network graph.

This approach examined the inverse of the mean first-passage time in the Markov chain initially presented by Kemeny and Snell [KS76]. Kemeny and Snell defined the mean first-passage time \( m_{rt} \) from \( r \) to \( t \) as the expected number of steps taken until the first arrival at node \( t \) starting at node \( r \), and represented as \( m_{rt} = \sum_{n=1}^{\infty} n f_{rt}(n) \). Where \( n \) denotes the number of steps taken and \( f_{rt}(n) \) denotes the probability that the chain first returns to the state \( t \) in exactly \( n \) steps. Kemeny and Snell represented \( m_{rt} \) in matrix form as \( M = (I + EZ_{\text{diag}})D \) where \( I \) is the identity matrix (a matrix containing ones in the diagonal and zeros elsewhere; \( Z \) the fundamental matrix, \( E \) a matrix containing all ones, \( Z_{\text{diag}} \) is the matrix containing in its diagonal the components of the fundamental matrix (and zeros everywhere else) and finally, \( D \) contains in its diagonal \( 1/d_t \) (1 divided by the components of the limit matrix of the Markov chain). White and Smyth’s approach to determining the centrality of of a node \( t \) given a set of root nodes \( R \) was to then find the inverse of the average mean first-passage time.

The fundamental matrix \( Z = (I - (P - A))^{-1} \) where \( P \) is the Markov transition probability matrix which is used to represent the conditional probability for a token to get to a new node given the current node in a Markov chain. The matrix \( A \) is the limit matrix of the Markov chain. The limit matrix (or stationary distribution) of a Markov chain is a steady-state condition which represents the fundamental theorem that after a sufficiently large number of time steps, the probability of a token being in a node
$n_t$ is the same as being in a node $n_y$. The limit matrix can be determined by letting the transition probability matrix at a constantly greater power until it converges to its long term behaviour (or what is then described as the limit matrix). To summarise, if given the transition matrix for a Markov chain, then its limit matrix, fundamental matrix, mean first-passage time matrix and finally the centrality of nodes in the chain can be determined.
Appendix B1: CRI User Manual

CRI tool is a distributed real-time plugin for Eclipse IDE. It uses information gathered from various collaborating developer IDE interactions to generate orders of relevant tasks, code artefacts and developers specific to a particular work context. CRI assists developers gain more understanding of who and what is important to different aspects of a software system while maintaining awareness of currently developed system state.

CRI does not keep records of actual IDE interaction data, but rather information on the type of interaction (create, update, view and delete). It does not damage/interfere with development work or personal data. The functionality of CRI is not suitable for capability judgement or reward structuring.

Getting Started

Before you get started, please make sure the CRI bundle plugin is downloaded and the content unzipped into your Eclipse plugin directory as shown in figure below. For a one-time download browse to http://www.cis.strath.ac.uk/~inah/download.htm
Opening CRI view
From the Eclipse menu select the following: Window -> Show View -> Other... and select CRI Monitor as shown in figure 1. This opens an Eclipse view (figure 2) that enables developers to create, delete and update a development task. During a project development cycle, this view also presents to collaborators a real-time ordering of relevant developers, code artefacts, and tasks depending on the selected perspective.

Logging into a collaboration group
The use of CRI requires that every developer logs into a collaboration group. To log into CRI simply select your collaborating group from the pull down menu as shown in figure 3 labelled 1 and type in your user name and unique identifier -(figure 3 label 2).

Logging into CRI enables you to create, update and delete tasks. As project related interactions progress (such as creating, updating and viewing of classes), you can also view the relevance ordering of tasks, code artefacts and developer representations based on a selected perspective.

Creating, updating, deleting and activating development task
CRI is a task based tool and relies on developers defining tasks and working within the context of such defined tasks. A task is considered to be any set of actions that affects the functionality of a system in some way. These include fixing bugs, improving performance or implementing new use case features.

Tasks can be created by selecting label 3 on figure 4. Updating an existing task requires that you open the required task by a mouse right click as shown in label 4. At all times, only a single task is active within an Eclipse IDE. You can activate/deactivate tasks as work context changes by a mouse right click on the appropriate task and selecting Activate/Deactivate Task (label 4).

As interactions on a code artefact during an active task by collaborating developers increase, an ordering structure of relevance for related entities emerges.

Increasing colour intensity is used to depict an increasing order of relevance.
Viewing Relevance from the Project perspective

This perspective represents an ordering of relevance of all tasks, developers and code artefacts that are associated with a software project. Figure 5 is a demonstration using a project ‘GizmoDemo’. Analysis of interactions after a number interaction processes show that Build Mode task, BuildCanvas.Java artefact and Ellen developer representation to be high in the relevance ordering.

On the Project tab, select a project from the pull down menu (label 5) and further click related Tasks, Artefacts, or Surrogate View buttons (label 6). These will each present a relevance ordering of the selected type in the project space.

Viewing Relevance from the Task perspective

This perspective presents a perception of the relevance of developers and artefacts [typically Java classes] associated with each project task. This is demonstrated with a hierarchy of relevance of every developer and artefact associated with the project ‘GizmoDemo’ after a number of interaction processes related to the ‘GUI Restructure’ task. Figure 6 shows that after the given number of interaction processes, GameSettingDialog.java and Jamie are more relevant from the perspective of GUI Restructure.

On the Task tab, select a project and task from the pull down menu (label 7) and click on the related Artefacts, or Developers View buttons (label 8) which will each present a relevance ordering of the selected type from the point of view of the selected task.
**Viewing Relevance from the Developer perspective**

This perspective represents a hierarchy of relevance of related artefact and task sets that forms the context of a developer's actions within a project space. Views that are generated by this perspective are based on a single developer’s interaction processes, thus presenting varying artefacts and task relevance depending on the particular developer. This is demonstrated with an ordering of relevance of artefacts and tasks after a number of interaction processes associated with the ‘inah’ developer representation. Figure 8 shows that ‘BuildController.java’ code artefact and the Controllers task instance are more relevant from inah’s perspective.

On the Developer tab, select a project and the required Developer work context from the pull down menu (label 11) and further click Artefacts, or Tasks View buttons (label 12) which will each present a relevance ordering of the selected type from the point of view of the selected developer work context.

**Viewing Relevance from the Code Artefact perspective**

This perspective provides for every artefact within a project space a relevance hierarchy of associated tasks and developers, thus making it possible to derive a shared understanding of the importance of a task or a surrogate to a shared artefact. This is demonstrated with an ordering of relevance of every task and developer after a number of interaction processes associated with ‘PlayController.java’. Figure 7 shows that the Controllers task and developer inah are more relevant from the PlayController.java contextual perspective.

On the Artefact tab, select a project and code artefact from the pull down menu (label 9) and further click Tasks, or Developers View buttons (label 10). These will each present a relevance ordering of the selected type from the point of view of the selected artefact.
T-A-D\textsuperscript{29} relations

For cases where a code artefact is shared amongst a set of tasks or developers, there are situations where developers may desire to understand which particular aspects of code are related to a particular task or developer. CRI enhances this process by further providing information at a more granular level. Figure 9 labels 12 and 13 shows a statement line for the artefact code `PlayController.java` that is associated with the developer ‘Inah’ and tasks ‘Testing’ and ‘Controllers’.

This information can be identified at the method or statement level for any selected artefact by selecting the desired parameters as shown in label 14 of figure 9.

**Social Network view**

A ‘social network’ view can also be generated from the combination of the different perspectives which provides a high level picture of the important entities in the current project. This view can be used to predict potential bottlenecks, risks, system defects or bugs, and understanding of general system properties.

Clicking the button labelled 16 on figure 9 generates a social network view (figure 10) of all the entities that exist in the collaboration space of the GizDemo collaboration group.

\textsuperscript{29} T-A-D: Tasks, artefacts and developers relations
The symbols T, A and D are used to identify tasks, code artefacts and developers in the project group. Each of the nodes in the social network view can be dragged around to visualise its relations. Also, the relative size of an entity is intended to show their relative importance and influence to the project.

Switching between operation modes

CRI can be operated within two modes (Recency and History modes) both aimed at enhancing developer understanding during collaborative software engineering processes. Developers can switch between these modes by clicking the button labelled 17 on figure 11 below.

The ‘recency mode’ shows the current state of a project in terms of the resources that are actively being used.

The ‘history mode’ allows a user to step back in time in the project collaboration space and to see how the relevance of project resources has changed throughout the project lifecycle.

Collaborating with other developers offline

In situations where internet connection speed is slow, or there is no connection at all, collaborating developers have the option of working offline and synchronising interaction events with the CRI server anytime an internet connection is detected or at a developer’s convenience. Developers can switch online/offline and also synchronise with the CRI server by clicking the buttons labelled 18 and 19 on figure 11 respectively.
Certain functionalities such as T-A-D relations and History mode are only optimised for online mode.

**Summary:**

CRI ordering of relevance is depicted in the different views and perspectives provided by this tool. A more granular and abstract perception of collaboration is presented using functions called T-A-D relations and the social network view. This extends the common understanding and tacit knowledge that is a feature of closely co-located project teams to teams that are separated by space and/or time.

This tool aids developers in better understanding and predicting system properties while maintaining awareness of the current state of the system.
Appendix C1: CRI Subjective Evaluation Questionnaire (Study 1)

Thanks for taking time to use the CRI tool. Would also be very grateful for some feedback on how the tool has helped you in your group work.

Please take into account that we are interested in knowing your opinion. Answer questions freely, there are no right or wrong answers.

1) CRI requires you to always work in the context of a defined task. This was found to be:

- Major Advantage
- Minor Advantage
- Neutral
- Minor Disadvantage
- Major Disadvantage

2) The CRI tool identified which tasks are currently the main focus of activity in a collaborative environment.

- Strongly agree
- Agree
- Neutral
- Disagree
- Strongly disagree

3) The CRI tool identified the intensity of collaboration over a range of code artefacts.

- Strongly agree
- Agree
- Neutral
- Disagree
- Strongly disagree

4) The varying colour intensity used by the CRI tool depicted the importance of a code artefact.

- Strongly agree
- Agree
- Neutral
- Disagree
- Strongly disagree

5) In the context of a selected task, the CRI tool identified the relative relevance of individual developers.

- Strongly agree
- Agree
- Neutral
- Disagree
- Strongly disagree

6) The CRI tool can identify the impact of changes to existing code on related tasks and developers.

- Strongly agree
- Agree
- Neutral
- Disagree
- Strongly disagree

Please turn over.
7) Please list those features of the tool you found useful:


8) What features did you feel were of no relevance to your work:


9) Were there features that you found difficult to understand or use? Please list:


10) Were there any features that you thought were missing in the CRI tool and could be usefully added? Please list:


11) Have you any preference for how the relative importance of artefacts, tasks and developers should be viewed in the CRI tool? Please identify:


Once again we appreciate all your help.

Would you mind participating in a brief (max 30 min) interview about your experiences with CRI- at the time that suits you? If so please provide your name:
## Appendix C2: Analysis of Collaboration Data (Study 1)

Table C2-1 View, update, create and delete interaction events relative to defined task instances

<table>
<thead>
<tr>
<th>SN</th>
<th>Group</th>
<th>Task Name</th>
<th>No of Views</th>
<th>No of Updates</th>
<th>No of Creates</th>
<th>No of Deletes</th>
<th>Total Interaction Processes Per task</th>
<th>No of Collaborators per task</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>E</td>
<td>Ball Movement</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>E</td>
<td>Creating a String Metric</td>
<td>285</td>
<td>58</td>
<td>14</td>
<td>5</td>
<td>362</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>E</td>
<td>JavaDoc and Comments</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>E</td>
<td>SaveMode</td>
<td>262</td>
<td>35</td>
<td>9</td>
<td>2</td>
<td>308</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>E</td>
<td>Adding Physics API to Gizmos</td>
<td>278</td>
<td>36</td>
<td>12</td>
<td>3</td>
<td>329</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>E</td>
<td>BuildMode GUI</td>
<td>97</td>
<td>55</td>
<td>2</td>
<td>0</td>
<td>154</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>E</td>
<td>Run Mode</td>
<td>501</td>
<td>112</td>
<td>14</td>
<td>2</td>
<td>629</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>D</td>
<td>GizmoFactory</td>
<td>639</td>
<td>311</td>
<td>26</td>
<td>0</td>
<td>976</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>D</td>
<td>Collisions/Physics</td>
<td>520</td>
<td>314</td>
<td>14</td>
<td>1</td>
<td>849</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>D</td>
<td>Bonus Gizmos</td>
<td>75</td>
<td>29</td>
<td>9</td>
<td>0</td>
<td>113</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>D</td>
<td>MVC Architecture</td>
<td>109</td>
<td>34</td>
<td>11</td>
<td>0</td>
<td>154</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>D</td>
<td>Key Handling</td>
<td>165</td>
<td>55</td>
<td>8</td>
<td>0</td>
<td>228</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>Grid ADT Development</td>
<td>117</td>
<td>59</td>
<td>11</td>
<td>1</td>
<td>188</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>D</td>
<td>Grid ADT</td>
<td>62</td>
<td>45</td>
<td>5</td>
<td>1</td>
<td>113</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>D</td>
<td>GUI Development</td>
<td>203</td>
<td>173</td>
<td>16</td>
<td>2</td>
<td>394</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>D</td>
<td>Gizmoball 3D</td>
<td>240</td>
<td>205</td>
<td>15</td>
<td>3</td>
<td>463</td>
<td>2</td>
</tr>
<tr>
<td>17</td>
<td>B</td>
<td>Create Build Mode GUI</td>
<td>381</td>
<td>151</td>
<td>10</td>
<td>1</td>
<td>543</td>
<td>3</td>
</tr>
<tr>
<td>18</td>
<td>B</td>
<td>Build an XML Loader class.</td>
<td>52</td>
<td>53</td>
<td>2</td>
<td>0</td>
<td>107</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>B</td>
<td>Serialize Gizmos</td>
<td>119</td>
<td>39</td>
<td>9</td>
<td>0</td>
<td>167</td>
<td>3</td>
</tr>
<tr>
<td>20</td>
<td>B</td>
<td>Documentation</td>
<td>354</td>
<td>117</td>
<td>28</td>
<td>0</td>
<td>499</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>B</td>
<td>GUI Restructure</td>
<td>91</td>
<td>33</td>
<td>7</td>
<td>1</td>
<td>132</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>B</td>
<td>Controllers for Run Mode</td>
<td>318</td>
<td>114</td>
<td>13</td>
<td>0</td>
<td>445</td>
<td>3</td>
</tr>
<tr>
<td>23</td>
<td>B</td>
<td>Creating the help screen</td>
<td>42</td>
<td>17</td>
<td>7</td>
<td>1</td>
<td>67</td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>B</td>
<td>Gizmoball Network Play</td>
<td>290</td>
<td>253</td>
<td>12</td>
<td>0</td>
<td>555</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>B</td>
<td>Correct flipper collisions</td>
<td>83</td>
<td>49</td>
<td>7</td>
<td>0</td>
<td>139</td>
<td>2</td>
</tr>
<tr>
<td>26</td>
<td>B</td>
<td>Testing</td>
<td>17</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>19</td>
<td>2</td>
</tr>
<tr>
<td>27</td>
<td>B</td>
<td>Fixing the renderer</td>
<td>142</td>
<td>98</td>
<td>11</td>
<td>0</td>
<td>251</td>
<td>1</td>
</tr>
<tr>
<td>28</td>
<td>B</td>
<td>Trigger Connect Frame</td>
<td>141</td>
<td>152</td>
<td>3</td>
<td>0</td>
<td>296</td>
<td>1</td>
</tr>
<tr>
<td>29</td>
<td>A</td>
<td>Flesh out Gizmo DataTypes</td>
<td>68</td>
<td>26</td>
<td>2</td>
<td>0</td>
<td>97</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>A</td>
<td>Square Gizmos</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>27</td>
<td>2</td>
</tr>
<tr>
<td>31</td>
<td>A</td>
<td>Begin Work on Final Project</td>
<td>104</td>
<td>42</td>
<td>7</td>
<td>3</td>
<td>156</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>C</td>
<td>change flipper drawing</td>
<td>94</td>
<td>36</td>
<td>4</td>
<td>0</td>
<td>134</td>
<td>1</td>
</tr>
</tbody>
</table>

TOTAL: 5883 2704 290 27 8904
<table>
<thead>
<tr>
<th>SN</th>
<th>Group</th>
<th>Artefact name</th>
<th>Creator</th>
<th>Updaters</th>
<th>Total Absolute update</th>
<th>% of total absolute update</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>B</td>
<td>Machine.java</td>
<td>Evans</td>
<td>Evans</td>
<td>51307</td>
<td>97.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Roan</td>
<td>808</td>
<td>3.00</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>Ball.java</td>
<td>Evans</td>
<td>Evans</td>
<td>75177</td>
<td>99.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Roan</td>
<td>275</td>
<td>0.40</td>
</tr>
<tr>
<td>3</td>
<td>B</td>
<td>ColourChangeEvent.java</td>
<td>Clark</td>
<td>Clark</td>
<td>150</td>
<td>84.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Roan</td>
<td>27</td>
<td>16.00</td>
</tr>
<tr>
<td>4</td>
<td>B</td>
<td>Test.java</td>
<td>Evans</td>
<td>Evans</td>
<td>10835</td>
<td>97.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Roan</td>
<td>269</td>
<td>2.40</td>
</tr>
<tr>
<td>5</td>
<td>B</td>
<td>GUIGame.java</td>
<td>Evans</td>
<td>Evans</td>
<td>10333</td>
<td>35.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Roan</td>
<td>18671</td>
<td>64.40</td>
</tr>
<tr>
<td>6</td>
<td>B</td>
<td>OffsetBallEvent.java</td>
<td>Clark</td>
<td>Clark</td>
<td>3568</td>
<td>55.40</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Roan</td>
<td>2870</td>
<td>44.60</td>
</tr>
<tr>
<td>7</td>
<td>B</td>
<td>FlipperTriggerEvent.java</td>
<td>Evans</td>
<td>Evans</td>
<td>1547</td>
<td>98.30</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Clark</td>
<td>27</td>
<td>1.70</td>
</tr>
<tr>
<td>8</td>
<td>B</td>
<td>GizmoBall.java</td>
<td>Evans</td>
<td>Evans</td>
<td>17440</td>
<td>82.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Roan</td>
<td>3875</td>
<td>18.00</td>
</tr>
<tr>
<td>9</td>
<td>B</td>
<td>GUIWindow.java</td>
<td>Evans</td>
<td>Evans</td>
<td>33908</td>
<td>46.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Clark</td>
<td>39395</td>
<td>54.00</td>
</tr>
<tr>
<td>10</td>
<td>B</td>
<td>Circle.java</td>
<td>Evans</td>
<td>Evans</td>
<td>227</td>
<td>99.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Roan</td>
<td>1</td>
<td>0.40</td>
</tr>
<tr>
<td>11</td>
<td>D</td>
<td>Action.java</td>
<td>Andy</td>
<td>Andy</td>
<td>43</td>
<td>9.30</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ian</td>
<td>421</td>
<td>90.70</td>
</tr>
<tr>
<td>12</td>
<td>B</td>
<td>MainController.java</td>
<td>Roan</td>
<td>Roan</td>
<td>3279</td>
<td>8.50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Evans</td>
<td>1888</td>
<td>33.50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Clark</td>
<td>471</td>
<td>58.00</td>
</tr>
<tr>
<td>13</td>
<td>B</td>
<td>PhysicsController.java</td>
<td>Evans</td>
<td>Evans</td>
<td>4881</td>
<td>1.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Roan</td>
<td>2361</td>
<td>32.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Clark</td>
<td>136</td>
<td>66.20</td>
</tr>
<tr>
<td>14</td>
<td>B</td>
<td>OverviewFrame.java</td>
<td>Clark</td>
<td>Clark</td>
<td>10304</td>
<td>3.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Roan</td>
<td>4700</td>
<td>30.20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Evans</td>
<td>563</td>
<td>66.20</td>
</tr>
<tr>
<td>15</td>
<td>B</td>
<td>RunDefaultActionAction.java</td>
<td>Clark</td>
<td>Clark</td>
<td>2</td>
<td>0.70</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Roan</td>
<td>297</td>
<td>99.30</td>
</tr>
<tr>
<td>16</td>
<td>D</td>
<td>Gizmo.java</td>
<td>Andy</td>
<td>Andy</td>
<td>5952</td>
<td>89.30</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ian</td>
<td>715</td>
<td>10.70</td>
</tr>
<tr>
<td>17</td>
<td>B</td>
<td>OverviewFrameController.java</td>
<td>Roan</td>
<td>Roan</td>
<td>553</td>
<td>73.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Clark</td>
<td>2684</td>
<td>15.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Evans</td>
<td>440</td>
<td>12.00</td>
</tr>
<tr>
<td>18</td>
<td>B</td>
<td>XMLLoad.java</td>
<td>Clark</td>
<td>Clark</td>
<td>1339</td>
<td>13.10</td>
</tr>
<tr>
<td>19</td>
<td>D</td>
<td>GameBoardModel.java</td>
<td>Ian</td>
<td>Roan</td>
<td>8918</td>
<td>86.90</td>
</tr>
<tr>
<td>20</td>
<td>D</td>
<td>AnimateFlipperAction.java</td>
<td>Andy</td>
<td>Andy</td>
<td>1125</td>
<td>85.50</td>
</tr>
</tbody>
</table>

Figure C2-1 Artefact Relevance Ranking across selected timelines 1, 2, 3 and 4
Figure C2-2 Task Perspective relevance views across different timelines

a  Developer surrogate Relevance Ranking across selected timelines 1, 2, 3 and 4

b  Artefact Relevance Ranking across selected timelines 1, 2, 3 and 4
Figure C2-3 Project Perspective relevance views across different timelines
Appendix D1: Questionnaire design (Study 2)
Each question stated in the design is associated with its necessary preconditions, its objective or contribution to the study goal, and the method of data analysis.

Study Goal:

1. Determine if CRI relevance orderings reasonably match with developers’ perception of activities within a collaboration space.
2. Determine if developers feel that the different perspectives and social graph visualisations are useful in providing awareness during distributed collaborative work?

Question 1:
How frequently did you remember to log into CRI:

Objective: Determine the extent of participants’ use of CRI over the collaboration period.

Precondition: Should be asked at the start of the questionnaire session.

Method of data analysis:
1. Summarise using dot plots.
2. Display the distribution of observations in a dot-plot or a bar-chart.
3. Examine recorded verbalized reasons of participant for remembering or not remembering to log into CRI.

Question 2:

a) As you changed work context, how frequently did you activate or create a new task:

b) I found it difficult to always be working within the context of an identified task:

c) How difficult was it to create a new task in CRI:

d) How difficult was it to activate an existing task in CRI.
e) YES/NO I found it interesting to see what others were doing. (If yes, why? If No, what bothers you?)

f) Do you feel influenced by your relevance position to a selected artefact or tasks that you were working on? (If yes: How were you influences?)

Objective:
Evaluate if CRI impact the way that participants’ will normally carry out a programming task.
(Results here will further support findings from our initial proof of concept study)

Precondition:
   i. Question 2 should be asked after question 1
   ii. To avoid pre-empting answers for any of the questions, 2a, b, c and d should be asked in sequence.

Method of data analysis:
   i. Display the distribution of observations in a dot-plot.
   ii. Examine recorded verbalized responses to each of the questions.
   iii. The following insight will be used to determine possible reason for 2a:
       1. Correlation between 2a and 2b
       2. Correlation between 2a and 2c
       3. Correlation between 2a and 2d

Question 3:
   a) For illustration purpose: It is assumed that developer Jamie has been working on the following list of tasks to accomplish gizmoball:

Rank the list of first four tasks above based on:
   i. Coding effort you have put into each task over the project duration:

<table>
<thead>
<tr>
<th>Coding Effort over project duration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

   How confident are you of the ranking:
   |  |  |  |  |  |  |
   | Low |  |  |  |  |  |
   |  |  |  |  | High |

   ii. Tasks that you have put in the most coding effort recently:

<table>
<thead>
<tr>
<th>Recent coding effort</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

   How confident are you of the ranking:
   |  |  |  |  |  |  |
   | Low |  |  |  |  |  |
   |  |  |  |  | High |

207
iii. Tasks by their order of difficult:

<table>
<thead>
<tr>
<th>Difficulty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
</tr>
<tr>
<td>High</td>
</tr>
</tbody>
</table>

How confident are you of the ranking:

iv. Tasks by their order of the highest number of bugs generated:

<table>
<thead>
<tr>
<th>Bugs generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
</tr>
<tr>
<td>High</td>
</tr>
</tbody>
</table>

How confident are you of the ranking:

b) For illustration purpose: It is assumed that developer Jamie has been working on the following list of code artefacts to accomplish gizmoball:

5. PlayController.java  6. ... etc

Rank the list of first four artefacts above based on:

i. Coding effort you have put into each artefact over the project duration

<table>
<thead>
<tr>
<th>Coding Effort over project duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
</tr>
<tr>
<td>High</td>
</tr>
</tbody>
</table>

How confident are you of the ranking:

ii. Artefact that you have put in the most coding effort recently

<table>
<thead>
<tr>
<th>Recent coding effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
</tr>
<tr>
<td>High</td>
</tr>
</tbody>
</table>

How confident are you of the ranking:

iii. Artefacts by their order of difficulty:

<table>
<thead>
<tr>
<th>Difficulty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
</tr>
<tr>
<td>High</td>
</tr>
</tbody>
</table>

How confident are you of the ranking:

iv. Artefacts by their order of the highest number of bugs generated:

<table>
<thead>
<tr>
<th>Bugs generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
</tr>
<tr>
<td>High</td>
</tr>
</tbody>
</table>

How confident are you of the ranking:

Objective:

Capture the relevance perception of participants’ prior to being presented with task and artefact rankings from CRI relevance list visualisations.
Precondition:

i. Question 3 should be asked after question 2 and before they are presented with a visualisation of CRI relevance ordering (before question 4). This is to avoid distorting participants’ perception of relevance while administering question 3.

ii. A participant can only rank the list of tasks that he/she has worked on. Thus ‘Jamie’ has worked on ‘Play Mode’, ‘Collision Mgt’, ‘Build Mode’ and ‘Network Play’.

iii. A participant can only rank the list of artefacts that he/she has worked on. Thus ‘Jamie’ has worked on ‘Ball.java’, ‘Circle.java’, ‘Absorber.java’, ‘BuildController.java’, ‘PlayController.java’ etc.

Method of data analysis:

i. Establish empirical probabilities of obtaining a match between user perception overall/recent work effort and CRI recent/history modes. These probabilities will be obtained by correlating results in question 3ai, 3aii, 3bi, 3bii and 4.

ii. Establish empirical probabilities of obtaining a match between user perception of difficulty/bugs generated and history mode/social graphs (using CRI markov centrality values). These probabilities will be obtained by correlating results in question 3aiii, 3aiv, 3biii, 3biv and 4

iii. Examine recorded verbalized feedback as participants ranked the list of tasks and artefacts.

**Question 4:** For illustration purpose: It is assumed that Jamie is the participant being interviewed

**a** The relevance ordering in figure is representative of tasks I have put in most coding effort over the project duration:

- Do you have any comments on this?
- Does this affect your perception of what you initially thought?

**b** The relevance ordering shown in the figure represents both the relevant tasks, and their relative significance, in my recent coding effort:
- Do you have any comments on this?
- Does this affect your perception of what you initially thought?

c | The relevance ordering in figure is representative of artefacts I have put in most coding effort over the project duration:

- Do you have any comments on this?
- Does this affect your perception of what you initially thought?

d | The relevance ordering shown in the figure represents both the relevant artefacts, and their relative significance, in my recent coding effort:

- Do you have any comments on this?
- Does this affect your perception of what you initially thought?

e | For further illustration purpose:

- It is assumed that Runi, Jamie and Jumbo are collaborators
- It is also assumed that these collaborators have been collaborating over Build Mode, Play Mode, File Parsing tasks

The figure represents relevance ordering of developers depending on the tasks they have been working on over the project duration.

Indicate your level of acceptance/rejection of the orderings.

Are there any particular concerns you have about the orderings?
f Are there any benefits you derive from the relevance ordering of tasks, developers and code artefacts compared to a random presentation:

---

g Was it useful to see all the facets (views) of a selected project, task, developer or code artefact perspective:
(Paraphrased: Do you think you will lose any awareness information if for a selected task you are only presented with either related artefacts or related developers. (compared to current presentation of both)

---

h Are there any particular insights you obtain as you slide through the history of your project tasks, developers and code artefacts?

---

i The figure represents a social graph of tasks, code artefacts and developers that have been associated with your project.
The larger the size, the more significant the task, developer or artefact to the project.

Indicate your level of acceptance/rejection of the orderings

![Accept/Reject Scale]

Comments:
Are there any particular comments you have about the size of each representation?
Does it tell you anything useful about tasks and artefacts?

---

Objective:
Capture participant’s perception of CRI relevance ordering visualisation after initially capturing their view of recent and overall work effort as well as difficulty and bug experiences in question 3 above.
Main aim of question 4g is to validate if there are benefits associated with three faceted models compared to two or single faceted ones. While 4h provides further insights into if CRI makes developers develop a perception of project status.

Precondition:

i. Question 4 should be asked after question 3.

ii. Question 4a, b, c and d involve a participant work perspective.

iii. Question 4e involves a number of selected task contexts that the participant has been associated with.

Method of data analysis:

Analysis will be carried out by summarising each interview transcript and tabulating answers for questions 3 and 4 to compare and contrast them.

a) Comparison between question 3ai and 4a will be used to establish developer perception match with CRI task ordering in the history mode.

b) Comparison between question 3aii and 4b will be used to establish developer perception match with CRI task ordering in the recent mode.

c) Comparison between question 3bi and 4c will be used to establish developer perception match with CRI artefacts ordering in the history mode.

d) Comparison between question 3bii and 4d will be used to establish developer perception match with CRI artefacts ordering in the recent mode.

e) It is assumed here that the relative symmetric relationship existing between developer perspective and other perspective makes results here applicable to other perspectives as well.

f) Comments generated from question 4e will be used to validate a match in participants’ perception of developer orderings by CRI.

g) Comments generated from question 4f will be used to validate a match in participants’ perception activity social graph.

Comments generated from question 4h will be used to further validate results in questions 4a and 4c. This will also help in validating subjective usefulness of CRI.
Appendix D2: Analysis of Collaboration Data (Study 2)

Table D2- 1View, update, create and delete interaction events relative to defined task instances

<table>
<thead>
<tr>
<th>SN</th>
<th>Group</th>
<th>Task Name</th>
<th>No of Views</th>
<th>No of Deletes</th>
<th>No of Creates</th>
<th>No of Updates</th>
<th>Total Interaction Processes Per task</th>
<th>No of Collaborators per task</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>G1</td>
<td>Absorber Demo</td>
<td>171</td>
<td>0</td>
<td>11</td>
<td>289</td>
<td>471</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>G1</td>
<td>Build Mode</td>
<td>646</td>
<td>1</td>
<td>24</td>
<td>763</td>
<td>1434</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>G1</td>
<td>Collision Demo</td>
<td>115</td>
<td>0</td>
<td>4</td>
<td>136</td>
<td>255</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>G1</td>
<td>File Demo</td>
<td>113</td>
<td>0</td>
<td>4</td>
<td>177</td>
<td>294</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>G1</td>
<td>Flipper demo</td>
<td>223</td>
<td>1</td>
<td>7</td>
<td>417</td>
<td>648</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>G1</td>
<td>PlayMode</td>
<td>81</td>
<td>0</td>
<td>2</td>
<td>89</td>
<td>172</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>G1</td>
<td>Preliminary release</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>16</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>G2</td>
<td>Basic Core</td>
<td>15</td>
<td>0</td>
<td>7</td>
<td>182</td>
<td>204</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>G2</td>
<td>Core</td>
<td>262</td>
<td>0</td>
<td>5</td>
<td>407</td>
<td>674</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>G2</td>
<td>Geometry</td>
<td>57</td>
<td>0</td>
<td>4</td>
<td>28</td>
<td>89</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>G2</td>
<td>Persistence (File Handling)</td>
<td>53</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>65</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>G2</td>
<td>User Interface</td>
<td>1449</td>
<td>1</td>
<td>54</td>
<td>975</td>
<td>2479</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>G3</td>
<td>Absorber Demo</td>
<td>11</td>
<td>0</td>
<td>4</td>
<td>44</td>
<td>59</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>G3</td>
<td>From Demo to Final</td>
<td>58</td>
<td>2</td>
<td>13</td>
<td>134</td>
<td>207</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>G3</td>
<td>JUnit Tests</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>G4</td>
<td>Gravity and Friction</td>
<td>22</td>
<td>0</td>
<td>2</td>
<td>71</td>
<td>95</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td></td>
<td>3276</td>
<td>8</td>
<td>142</td>
<td>3740</td>
<td>7166</td>
<td></td>
</tr>
<tr>
<td>SN</td>
<td>Group</td>
<td>Artefact name</td>
<td>Creator</td>
<td>Updaters</td>
<td>Total Absolute update Δ</td>
<td>% of total absolute update Δ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>-------</td>
<td>---------------</td>
<td>---------</td>
<td>----------</td>
<td>------------------------</td>
<td>----------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>G2</td>
<td>Ball</td>
<td>James</td>
<td>James</td>
<td>1982</td>
<td>72.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Paul</td>
<td>737</td>
<td>27.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>G2</td>
<td>Flipper</td>
<td>Paul</td>
<td>James</td>
<td>1024</td>
<td>82.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Paul</td>
<td>214</td>
<td>17.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>G2</td>
<td>RightFlipper</td>
<td>Paul</td>
<td>James</td>
<td>872</td>
<td>97.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Paul</td>
<td>23</td>
<td>2.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>G2</td>
<td>CircularBumper</td>
<td>Paul</td>
<td>James</td>
<td>1092</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Paul</td>
<td>1335</td>
<td>55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>G2</td>
<td>LeftFlipper</td>
<td>Paul</td>
<td>James</td>
<td>924</td>
<td>54.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Paul</td>
<td>778</td>
<td>45.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>G2</td>
<td>BuildArea</td>
<td>James</td>
<td>Tracy</td>
<td>348</td>
<td>3.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>James</td>
<td>9857</td>
<td>96.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>G2</td>
<td>Wall</td>
<td>Paul</td>
<td>James</td>
<td>2687</td>
<td>65.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Paul</td>
<td>1400</td>
<td>34.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>G2</td>
<td>GizmoManager</td>
<td>James</td>
<td>Tracy</td>
<td>72</td>
<td>0.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>James</td>
<td>8965</td>
<td>99.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>G2</td>
<td>GameAreaView</td>
<td>James</td>
<td>James</td>
<td>2638</td>
<td>94.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Tracy</td>
<td>147</td>
<td>5.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>G2</td>
<td>SquareBumper</td>
<td>Paul</td>
<td>James</td>
<td>1235</td>
<td>70.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Paul</td>
<td>526</td>
<td>29.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>G2</td>
<td>TriangularBumper</td>
<td>Paul</td>
<td>James</td>
<td>1191</td>
<td>47</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Paul</td>
<td>1344</td>
<td>53</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>G1</td>
<td>MoveCommand</td>
<td>Tony</td>
<td>Alex</td>
<td>231</td>
<td>99.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Tony</td>
<td>1</td>
<td>0.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>G1</td>
<td>AnimationWindow</td>
<td>Alex</td>
<td>Alex</td>
<td>6337</td>
<td>96.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Tony</td>
<td>211</td>
<td>3.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>G1</td>
<td>CircleBumper</td>
<td>Tony</td>
<td>Alex</td>
<td>2698</td>
<td>84.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Tony</td>
<td>492</td>
<td>15.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>G1</td>
<td>BuildWindow</td>
<td>Luke</td>
<td>Luke</td>
<td>18651</td>
<td>74.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Alex</td>
<td>2566</td>
<td>10.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Tony</td>
<td>3867</td>
<td>15.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>G1</td>
<td>LeftFlipper</td>
<td>Tony</td>
<td>Alex</td>
<td>3770</td>
<td>52.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Tony</td>
<td>3419</td>
<td>47.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>G1</td>
<td>ObjectHandler</td>
<td>Tony</td>
<td>Alex</td>
<td>5779</td>
<td>86.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Tony</td>
<td>930</td>
<td>13.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>G1</td>
<td>TriangleBumper</td>
<td>Alex</td>
<td>Alex</td>
<td>1660</td>
<td>37.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Tony</td>
<td>2738</td>
<td>62.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>G1</td>
<td>Ball</td>
<td>Tony</td>
<td>Luke</td>
<td>Alex</td>
<td>Tony</td>
<td>1757</td>
<td>2411</td>
</tr>
<tr>
<td>20</td>
<td>G1</td>
<td>Gizmo</td>
<td>Alex</td>
<td>Luke</td>
<td>Alex</td>
<td>Tony</td>
<td>67</td>
<td>635</td>
</tr>
<tr>
<td>21</td>
<td>G1</td>
<td>RightFlipper</td>
<td>Tony</td>
<td>Alex</td>
<td>Alex</td>
<td>Tony</td>
<td>2247</td>
<td>2034</td>
</tr>
<tr>
<td>22</td>
<td>G1</td>
<td>GameModel</td>
<td>Tony</td>
<td>Luke</td>
<td>Alex</td>
<td>Tony</td>
<td>753</td>
<td>2048</td>
</tr>
<tr>
<td>23</td>
<td>G1</td>
<td>GameObject</td>
<td>Tony</td>
<td>Alex</td>
<td>Alex</td>
<td>Tony</td>
<td>911</td>
<td>216</td>
</tr>
<tr>
<td>24</td>
<td>G1</td>
<td>FileHandler</td>
<td>Alex</td>
<td>Luke</td>
<td>Alex</td>
<td>Tony</td>
<td>318</td>
<td>531</td>
</tr>
<tr>
<td>25</td>
<td>G1</td>
<td>SquareBumper</td>
<td>Tony</td>
<td>Luke</td>
<td>Alex</td>
<td>Tony</td>
<td>9</td>
<td>3858</td>
</tr>
</tbody>
</table>

**Table D2- 3 Interaction type weighting factors used for study 2**

<table>
<thead>
<tr>
<th>Interaction type</th>
<th>View</th>
<th>Update</th>
<th>Create</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weighting Factor</td>
<td>0.01</td>
<td>0.001*Δ</td>
<td>0.1</td>
</tr>
<tr>
<td>Δ - Absolute Update delta</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

215
Figure D2-1 Ball.java related tasks relevance value vs time for treatments 1-3
Figure D2-2 Snapshots of Ball.java task relevance list for treatments 1-3
Figure D2-3 Ball.java related tasks relevance value vs time for treatments 4-6
Figure D2-4 Snapshots of Ball.java task relevance list for treatments 4-6
Table D2-4 Tasks and developers associated with artefacts identified with two or more collaborators

<table>
<thead>
<tr>
<th>SN</th>
<th>Group</th>
<th>Artefact name</th>
<th>Associated Developers</th>
<th>Associated Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>G1</td>
<td>MoveCommand.java</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>G1</td>
<td>AnimationWindow.java</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>G1</td>
<td>CircleBumper.java</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>G1</td>
<td>BuildWindow.java</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>G1</td>
<td>LeftFlipper.java</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>G1</td>
<td>ObjectHandler.java</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>G1</td>
<td>TriangleBumper.java</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>G1</td>
<td>Ball.java</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>G1</td>
<td>Gizmo.java</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>G1</td>
<td>RightFlipper.java</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>G1</td>
<td>GameModel.java</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>G1</td>
<td>GameObject.java</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>G1</td>
<td>FileHandler.java</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>14</td>
<td>G1</td>
<td>SquareBumper.java</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>G2</td>
<td>Ball.java</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>G2</td>
<td>Flipper.java</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>17</td>
<td>G2</td>
<td>RightFlipper.java</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>18</td>
<td>G2</td>
<td>CircularBumper.java</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>19</td>
<td>G2</td>
<td>LeftFlipper.java</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>20</td>
<td>G2</td>
<td>BuildArea.java</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>21</td>
<td>G2</td>
<td>Wall.java</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>22</td>
<td>G2</td>
<td>GizmoManager.java</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>23</td>
<td>G2</td>
<td>GameAreaView.java</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>G2</td>
<td>SquareBumper.java</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>25</td>
<td>G2</td>
<td>TriangularBumper.java</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Figure D2- 5 Relative time artefacts in group G1 were introduced into group G1 collaboration space
Figure D2-6 Sphere of influence variation patterns of tasks and developers
Appendix D3: Snippet of interview feedback (study 2)

**Q1: How frequently did you remember to log into CRI?**

*Tracy:* The problem was that sometimes when you start up eclipse, you just forget that there is a panel in the right…

*Luke:* Not as often as I probably should have… I found that sometimes when carrying out some work, I would have already gone half way through remembering that I needed to activate CRI…

*Alex:* Recently I have been forgetting quite a lot to log into CRI because there is so much work to do…

*Boris:* It’s a pain to remember to do it… Because you want to get work done CRI is not the top thing in your mind to be logging into…

*James:* Basically, it’s that I was eager to do something on my project, and because CRI is not always what I am thinking of… And when I started working… Then I just remember… Oh! I forgot to log into CRI… If it auto logged you in that would have been much better…

*Paul:* Initially, I used to forget to logon since I was not really used to using CRI… Sometime I only logon when I have already started doing some work, and that is when I remember…

*Tony:* It takes about half an hour before I remember each time I start work… In the reverse if I was automatically logged on into CRI, I will only oblige to that if we had talked about it before hand… Usually I will be programming for about six hours, and I do not consider 30 minutes into this time to be much, especially since within that time I am still figuring out what to do…

**Q2a: How frequently did you activate or create a new task as your work context changed?**

*Tracy:* Generally, I have really had few tasks I have been working on the ‘User Interface’, ‘File handling’ and the ‘Basic Core’… So the tasks pretty much stayed static for most of what had been going on… The last few weeks have been ‘User Interface’ struggle… So it has been like ‘User Interface’ all the time… About five weeks ago, I created the ‘File handling’ task, and was just working on that task when I loaded the parser classes and I am back to it now…

*Luke:* It was not that often because we did not really have many different tasks… We initially had the prototype related tasks which were ‘Absorber demo’, Flipper Demo’ and ‘Preliminary release’. Since after then we were rather switching between ‘Build Mode’ and ‘Play Mode’ tasks…

*Alex:* I think our tasks list was pretty general… So each of the demos was a task… There were not that many tasks to choose from, so we did normally work within a number of classes but on the same task…
James: I thought it was just implementing one part of a system or another part, I thought each of those parts was a task…
So I did switch CRI depending on which part of the system I was working on. Most of the time I either switched either between one part of my task (‘User Interface’) and the other model task I did (‘basic core’)…
I think I can have a lot more tasks within these main tasks I was working on…
Yeah, I did switch between these tasks each time I wanted to start work…
Within each task I had active, I did deviate…
For instance if I noticed that there was a problem in the model while I was working on the ‘User Interface’ I go straight and change it without remembering to explicitly switch on CRI…
Because it was like I didn’t want to lose my thoughts on the problem I wanted to solve…

Paul: Initially, I was virtually working on one task at a time, and the next day when you were on you will just switch to another task…
I wouldn’t say that it is a problem; it’s just to accept that it is something that you’ve got to do and to remember…

Tony: We have had a few tasks in general and the description of these tasks has been quite vague…
For instance, for the past three weeks or so we have virtually been on the about same task “Build Mode”…
The tasks could be probably be defined to be more specific, but I feel that the process of creating or activating a task while working will be disrupting the flow of my thoughts…

Q2b: I found it difficult to always be working within the context of a task?

Luke: I was a lot clearer in the prototype related tasks because it was just a kind of standalone tasks, for instance I knew I was working on ‘Flipper Demo’, and I simply concentrated on that, I did not need to depend or think about other tasks…
This has become less clear as we have shifted to developing the final system that consisted of ‘Build Mode’ and ‘Play Mode’ tasks; these two tasks are so closely linked…
Also, when I had thoughts of what to do in the system, I always want to get it done so I don’t forget rather than to go into CRI to activate which tasks I was working on…

Alex: Yeah…
Because we had other things to do as well, it was quite difficult to stay within one thing…

Paul: I think it gets sometimes difficult to know if you are straying into a task that is not directly related to the current task you are working on…
Sometimes I sit down and I am coding on a certain task…
I suddenly get bored with the task…
Or I am not too sure about what I am to do in the task…
Or you get something that is not working properly on the task…
Then I feel like I want to work on another task…

Tony: Oh yeah…
I struggle with working on other things when attending to a particular task…
I do so because it’s either getting boring, or I am going nowhere with the task or even that I am distracted… But usually I just write those things down as task items and tick them as I finish each of them…

Q2c: How difficult was it to create a new task in CRI?

Luke: I think creating a task is not a problem at all…
It’s remembering to do it that is the problem…

Alex: It was quite easy, it behaved exactly as I expected…

James: It’s not difficult, but sometimes the network latency is annoying…

Tony: It’s not difficult at all…
Q2d: How difficult was it to activate an existing task in CRI?

Tracy: Was reasonably intuitive…
Normally I just right click to get the popup menu…

Luke: Again I think activating a task is also not a problem at all…
It’s remembering to do it that is the problem…

Alex: Just that in terms of what I have to do its quite easy…
But the speed on my machine is quite slow; I don’t have the best laptop, so I understand why it is slow…
Even so in CRI offline mode that is normally suppose to be much faster…

James: I thought it was quite easy…

Tony: It’s quite easy; I don’t know how else it could be…

Q2e: YES/ NO I found it interesting to see what others were doing. If No, what bothers you?

Tony: Somehow, CRI made us work more, because it becoming a contest of who was doing the most work…
Though for fun anyway…
But I think if you are working in a small team as in our case, I am not sure that CRI can be used to its full potential…
Especially since communication among individuals reduces as the team size increases…

Luke: It is difficult to say…
I work with my group mostly when we were all sited close to each other, so all I need to do was to look over their shoulder and see what they were doing…
But working from home it was quite useful to know those working on different bits of the system, and asking developers involved in an aspect what they have done…

Alex: Well part of it was that I did not want to be left behind in the work that was going on in the group…
So it was somehow fun to see who was doing more work, or who was not significantly represented in the social graph…
Also, if I was seeing things going a bit funny or out of hand for a particular code artefact, it meant that if I did not have the information presented by CRI, I will had to go through the rigorous process of checking the commit history of the code repository…
A quick look at the graph gives me a quick view of those involved with that particular code artefact…

James: Yeah…
Was like I could see what others were doing before they did make a svn (Version management system) update…
It’s more real-time than updates I will get from svn… it’s was quite an interesting way to keep up with the state of the project…
I could see what my colleagues were more focusing on or what they are struggling with…
That is quite a good thing…

Paul: The network view is probably the most useful…
I can easily see at a glance what other people have been working on…
The relevance ranking of my other colleagues was less useful in terms of what we were doing because we are just three in the group…
The network view provided me enough information on what others were doing…
Tracy: I got rather interested in not just what others were doing but also how much they were doing…
Especially when you load up the network view, I can see proportionately what you have done and what you have been using…
But there is a risk that it makes me want to play catch up at some point especially when I notice that others have been doing more work than me in a task I should be working on…

Boris: I think when you are in a group, you don’t want to be doing other peoples work for them… its good that you can see he is not doing it or maybe there is someone that is too keen on wanting to do things…
You can always get a feel of effort and how the project is going on.
I do think that the potential of CRI is in big industrial projects.

Q2f: Do you feel influenced by your relevance position to a selected artefact or tasks that you were working on?

Luke: I did not feel influenced…
Again, some tasks do take more work than others and we share work in a task based on expertise, so every one was happy…
It was quite good to see how individual effort into a task was represented in CRI…
If seeking information as regards ‘Flipper Demo’ task… and seeing that Tony has done more work on ‘Flipper Demo’ I will rather speak with him than Alex…

Alex: I was not really influenced…

Tony: I am not really sure…

Tracy: I wouldn’t say so…
For the task that I am involved in, I watch where I am on the list because I don’t want to fall behind…
At the same time, if a task that I am not involved in or I know that I am not suppose to be that active in the task, obviously its not a problem to me and I don’t get influenced…

James: I was not really influenced, but I am not sure of my colleagues…
For me since I obviously did more work, and also CRI tend to reflect the work I had done I was ok…
It depends…

Paul: It has not been an issue to me…
Q4c: The relevance ordering in figure is representative of artefacts I have put in most coding effort over the project duration:

**Boris:** I know why Utilities.java should be so high…It’s because I have been creating some static methods…

So instead of creating an instance of Utilities class I just do Utilities.save() or Utilities.load() to load the file name…

I am not sure I accept the position of Stream.java. I did not really put much effort; I simply copied and pasted from an online source, and have not really changed the code much…

Yes I understand why OuterWall.java should be there, I recently was working on it…

I don’t know why BouncingBall.java will be higher than a Flipper.java.…

I know why! … It’s because I didn’t know if I will use GameModel.java to store MU and MU2 so I also put it in BouncingBall.java so I could go back and change whichever one was going to be changed…

So Yeah…

The relevance position of BouncingBall.java is quite right...

**James:** Although I put in much coding effort into GizmoColourScheme.java and doing a lot of edits on it, it was not as hard to implement compared to LeftFlipper3D.java…

I think it’s up there because I have made a lot of changes to it…

Yea, the ranking does actually show amount of coding effort I have put into each of the codes.

Mostly what I did in GizmoColourScheme.java was getters and setters and some few changes in colour schemes…

I was always going back to GizmoColourScheme quite frequently when I was working on the ‘User Interface’ task…
Q4c: The relevance ordering in figure is representative of artefacts I have put in most coding effort over the project duration: (Contd)

Alex: I am a bit surprise that CollidableCircle.java is positioned that high; I thought it would have been a bit lower…

Yeah, giving deeper thoughts into it, the ranking looks about right, just that sometimes I depend quite much on my recent coding experience…

For instance based on my recent memory, I would say that BuildWindow.java is higher, but thinking into it, I would have worked much more on other code artefacts…

Smith: I will say that for the ‘Gizmoball Prototype’ and ‘Gizmoball’ projects it’s quite accurate…

For the ‘GizmoFinal’ project, I did not make any change to Newton.java; I was always making reference to it to figure out what it was doing… I really needed to understand it to implement some of the physics features we have in Gizmoball…
Q4c: The relevance ordering in figure is representative of artefacts I have put in most coding effort over the project duration: (Contd)

**Luke**: Probably I will not say I am sure of the code effort I have put into the test related classes, but again BuildWindow.java and PlayWindow.java are rightly placed…

On a whole I think the ranking is insightful…

**Tony**: Yeah, that is pretty much how it went…

But since we occasionally practice pair programming, I was not too sure it is a direct spot on…
Q4c: The relevance ordering in figure is representative of artefacts I have put in most coding effort over the project duration: (Contd)

Tracy: Yep…

I will say that is right at head-on…

The only thing I will say is that I am surprised that Gizmoball.java is even on the ranking, because it’s just a driver consisting of two lines of code…

The only thing I ever remember doing on it was to update and call it every time I was testing the ‘User Interface’ task…

Paul: I am not sure of Wall.java, I don’t think I put as much coding effort into Wall.java as I put into the flipper related classes…

I would have also thought that the flipper classes would be further up in the ranking…

I tend to agree with the Geometry classes being low down; I really did not use them much…
Q4d: The relevance ordering in figure is representative of artefacts I have put in the most coding effort recently:

**Boris**: I added Stream.java not recently, but not long ago…
Yeah… I will say that is there.

GameModel.java…
I will say that was quite long, yeah… that is right. Right…
Right…,

I have been playing around with Utilities.java quite a lot recently…
OuterWalls.java as I said has just been made.

Stream.java…
Yeah…
Worked on it not long ago.
Just created it, not much work done on it, just copied and pasted it…

I am not too sure about Circle.java…
I know I have been looking at one of the Circle.java… Or maybe I might have been because ball and circle gizmos refer there…
It could have been…
Yeah…
I have been look at the physics package Circle.java …

**James**: I made an update to BuildArea.java, I think it should be the most recent…

I also recently worked on GameAreaView.java, I would have wanted it higher…

I am not too sure that CRI has the most recent data set on the codes I have been working on…

I think it’s partly because I forgot to log in recently…

Recently I have also had a few times that eclipse hanged while uploading CRI offline events to the server…
**Q4d:** The relevance ordering in figure is representative of artefacts I have put in the most coding effort recently:

*Alex:* Right, I will say so... Come to look at it, I have spent a lot of time quite recently to develop test cases for the gizmos, particularly on PlaceTest.java...

Though in PlaceTest.java, I don’t think there was that much amount of code in it to keep it in that position, but I was always viewing and running the code all the time...

As I said before, I have also recently been working on the command related classes; and the ranking reflects that...

*Smith:* I will say that for the ‘Gizmoball Prototype’ and ‘Gizmoball’ projects it’s quite accurate...

For the ‘GizmoFinal’ project, I did not make any change to Newton.java; I was always making reference to it to figure out what it was doing... I really needed to understand it to implement some of the physics features we have in Gizmoball...
Q4d: The relevance ordering in figure is representative of artefacts I have put in the most coding effort recently:

*Paul:* I am not sure I did work much into GizmoTriggerEvent.java or GizmoTriggerListener.java recently…

Though I might have looked into them after working on the flipper classes…

Besides this, I think the rest looks like what I have worked on recently…

*Tracy:* You know I was just thinking why Drawing3D.java should be there…

Yep…

Can now remember three days ago we were trying to figure out how to do some stuff with it…

Yep, I will say the ranking is spot-on…
Q4d: The relevance ordering in figure is representative of artefacts I have put in the most coding effort recently:

**Luke:** A number of test classes are high up there, yea…

I accept, just remembering that the test classes had really taken much of my time in the last two days… Yes, and the last data set I uploaded to CRI server contains a lot of work I had done on the test classes…

We all worked on the test classes together… Anybody that creates an artefact makes sure that he writes a test class for that artefact…

As much as possible, if I wrote a test class, then its going to be another member of the group that will run the test class I wrote…

This was to ensure that the testing was rigorous…

**Tony:** I think I will agree with every artefact there except for StillEdge.java and StillCircle.java…

I am not too sure of what I would have been doing with StillEdge.java or StillCircle.java, but no doubts I think I would have viewed it a couple of times…

In the last few days, I have actually done a lot of work that I struggle to come to terms with what I have recently put much effort into…
Q4e: The figures below represent relevance ordering of developers depending on the tasks they have been working on over the project duration.

Are there any particular concerns you have about the orderings?

*Luke*: I don’t think I have any concerns…

I think this shows that we are all doing our wee bit in the project…

The ranking is quite fair…

*Tony*: 😊 I see I never actually had a part to do with ‘Preliminary release’, that is why I did not remember when I saw it on our group tasklist…

Alex: I think the way we worked as a group, occasionally when we were together, one person was always using another person’s computer, that tends to affect my thinking of the ranking…
Q4e: The figures below represent relevance ordering of developers depending on the tasks they have been working on over the project duration.

Are there any particular concerns you have about the orderings?

Tracy: Yep…

James was responsible for Geometry and ‘Core’;

I will say the ranking is all pretty accurate…

My main fear here is that some developers might want to play catch up on a task they are not necessarily good at. But I think the understanding that some developers are going to be relevant to some tasks while others are not is necessary.

Paul: I am not sure Tracy did more work than James on ‘Basic Core’…

I definitely agree with the rest of it…

James: Yeah, I accept the ranking for ‘Geometry’, ‘Core’ and ‘User Interface’ fully…

I think for ‘basic core’ my name should not be at the bottom…
Q4e: The figures below represent relevance ordering of developers depending on the tasks they have been working on over the project duration.

Are there any particular concerns you have about the orderings?

![Relevance ordering figures]

*Boris*: Is Blair higher than Greg? Yeah… possibly higher, yes at the moment…

I accept that…

No doubt, it’s supposed to be Gordon working on JUnit Tests…

I have never activated JUnit Test before to work on it…

Q4f: Are there any benefits you derive from the relevance ordering of tasks, developers and code artefacts compared to a random presentation:

*Boris*: I believe it is better to give the ranking, but I believe there are two sides to this argument…

If you give the ranking of colleagues on a task, you can see who is working and important to a task…

But that does not mean that those that are not seen or low on the ranking are not doing any work, because they might simply be planning or designing and CRI is overlooking what they are doing…

*Smith*: I believe gather more information from the ordering…

I see information such as time spent, effort put in, and expertise that I will normally not get in a random presentation…

While a random presentation will give me information on every task I have worked on or every code artefact I have looked into or updated, I could just have looked at or updated every task or code artefact, and in that case there is no benefit of random presentation…

*Tracy*: I will go for the ordered ranking, because generally I can look at the ranking and see where I have spent a lot of time if it is at the top…

Or if it low down, I think I probably have a lot of work to do in there…

The way we are working in my group just now is that we have a tick list in the wiki, and once you have finished a task you tick to sigh it off…

CRI information is quite handy because it shows me which part of the system is about complete and which one I need to spend more time on…

*Paul*: It’s more useful when you have a ranking…

Especially when in the group, its better to know who has put in a lot of effort into a particular class…

For instance if you change one line in a class, that does not give you a expert understanding of the
class compared to someone that has put in much more coding to that class…
You cannot get such information in a random ordering…
I think it’s more useful to have a ranking…

James: From the perspective of what I have done, I don’t see any benefit of that…
You know what you have done; I don’t really see any point in being told what you have done…

Alex: I certainly go for the ordered ranking; it always gives me a sense of progression in the system that is being developed…
I would also have preferred it if CRI could also show me my navigation paths from artefact to artefact; a kind of navigation path history…

Tony: If it is for tasks or artefacts, I will prefer it to be ordered, because I know what task or artefact I have been working on most…

Luke: It’s quite difficult to say since I have been working on the project almost everyday…
So I know what I finished doing last night and what I am going to work on today…
But the longer I stay away from the system the more the ranking helps me to know the state of my system when last I left it…
If I was the group leader, I will also want to see a ranking of what my other colleagues have done recently…
Otherwise, I will just be interested in a ranking of what I have been doing rather than what others are doing…

Q4g: Was it useful to see all the facets (views) of a selected project, task, developer or code artefact perspective:
(Paraphrased: Do you think you will loose any awareness information if for a selected task you are only presented with either related artefacts or related developers. (compared to current presentation of both))

Luke: Generally, the more information I get the better…
Within my group, I don’t really want to be left in the dark about what anybody is doing…
Again, when we are sitting next to each other during group work, we do easily get this type of information by simply looking over the shoulder of our colleagues and ask them what they were working on…
When we are not working close by each other, this information is necessary…

Tony: It is important to keep both, because it is much easier to associate which tasks are related to an artefact and which developer was working on that part of the artefact to execute that task…

Alex: I think I did prefer the two…
For instance, even if I name a task descriptively, the same task still takes me to a set of artefacts that are not intuitive from the naming of the task…
So its good to not only see the tasks that I am working on, I did also love to see the code artefacts that are related to that task…

James: From the point of view of my work…
I think the artefacts view shows enough information, rather than having a task view too…
Task view is not necessary because relying on my recent memory…
I can figure out the tasks I have been working on even without a task ranking…
I can look at a code artefact and say yea it’s related to ‘User Interface’ task or ‘Geometry’…
But if I turn around to be interested in what my colleague has been doing; yea I think it is beneficial to have all the facets…
I can see which codes they are working on and which task…
Certainly I get useful information like which artefacts are used for a task by a particular person…
I get a suggestion of how much code effort a particular person has put into a task…
Paul: Where the artefact overlap for doing two tasks, in such case it’s more useful to have both facts… I want to know which developers are working on the artefact and which related task they are working on…

Tracy: I need all the facets… For instance I know that Keith spent a lot of time on ‘Geometry’, but I want to know the artefacts that he has been using to execute the task… And vice versa, Keith has spent all his time on Gizmo.java, I always want to know which task he has been trying to execute using Gizmo.java… In one of the projects I did last semester, bug finding was absolutely a nightmare, and it was simply because we had one person that was not a strong programmer… So CRI would have been handy to know what task this person had been doing and which code artefacts he has been using; that would have made bug finding much easier…

Smith: If I am concerned about a particular task say ‘Gravity and Friction’ I want to see which artefacts are relevant. In addition, I also want to see the developers related to ‘Gravity and Friction’ task… Else I will be loosing information on the developers with expertise in ‘Gravity and Friction’ task or those that are working on the task as well…

Boris: Depends on how you want to keep track of the project… For instance, you want to keep track of a task, you might just ask “is this task being done?”, you don’t really need to know what the developers are doing, you just want to know that the task is progressing… If you want to keep track on different components and their relationship with other components or track timelines, then you would want to know what individual developers are doing…

Q4h: Are there any particular insights you obtain as you slide through the history of your project?

Boris: Yes it tells me how much work I have done and if I am all way behind… It will be easier if I had a manager that tells me the timeline that he wants and what we should have done by then… It makes you not to feel relaxed when you are all way behind… Yeah… it is very useful.

Luke: … I normally slide through the history of tasks, java classes and fellow colleagues, it give me a better understanding of group working style… For instance what tasks me or my colleagues worked on and in what order… Or the way members of my group have developed or used code artefact resources to execute a task… …I had a slide through the relevance positions of developers and java classes for ‘Flie Demo’ task… I noticed that it has only been Tony working on that task… TriangleBumper.java and MainProgram.java were the original classes I noticed he started with, and it was so for quite a while… Currently there are a number of other classes he has used for that task… I do think comparing my work style with other developers in my group will provides a greater mutual understanding of the group coding style, or the style and order used to realise the software system… If a maintenance task is to be carried out on my system, such information will really be useful too… Since I can see the relative change of relevance that an artefact or a developer would have had in associated to a task used to realise the system…

Tracy: Sliding through the history of a task or artefact gives you a feel of how things have moved on, especially after sliding through a history of the artefacts I have been associated with… Having a slide through a task view I can gauge how important an artefact has been to the task over time, I did notice that MainScreen.java has retained high relevance over a long duration now, recently KeyConnectFrame.java has turned out to be high also…
… This gave me the clue that these classes are quite important to the ‘User Interface’ task…
I got particularly interested in MainScreen.java when I noticed James and Paul have used this
class some couple of times… I have been the only one working on KeyConnectFrame.java
I believe this information will again be very important to me when carrying out a maintenance task
on a system I am not really familiar with…

Paul: It’s really good to see how the project has evolved over time…
I believe I will appreciate this more if I have been working on the project for a longer time say like
a year…

Alex: If there is an artefact that has remained high on the ranking over a considerable time line, it
tells me where the main focus or problems has been in the project…
I have been watching the PlayWindow.java and BuildWindow.java classes recently on the ‘Build
Mode’ task…
Although I have never worked much on them, I know they have been important in achieving
‘Build Mode’ …
I also noticed that classes are high on Luke’s relevance ranking…
He is probably doing a lot of work on it.…

Tony: It is always good to see over time, how much you have worked on certain codes or tasks…
After our group meeting, I will watch to see if there is a certain growth in the task we discussed
during the meeting…
I noticed some time delay in processing as the timeline gets longer.
This feature will be more intuitive if timeline definitions are more specifically defined.

Smith: For a long term project, I really see the usefulness of such information, but for the short
term project that we are working on, I can fairly remember what we have done…
Q4i: The figure below represents a social graph of tasks, code artefacts and developers that have been associated with your project.

Are there any particular concerns you have about the size of each representation?

Alex: It’s the fastest way to get all the information from CRI… I always use graph to get a general state view of the project… I do check it every few days just to give me a grasp of what is going on with developers in the group and which tasks have had a considerable change recently…

Luke: I will say that is right because Alex has put in quite a lot of work… It’s quite fair the way he has been represented… It’s quite good to see that our work is being reflected in CRI, I will say this is an accurate reflection of what we have done… Because here are so many classes, it becomes complex as the number of code artefacts increases.

Tony: I will say that is spot on… It’s easy to see where the biggest problems are… For instance, I can see that the ‘Build Mode’ and BuildWindow.java have really been a problem spot… We have spent a few days really trying to figure some buggy stuff there… The number of artefacts has made it cluttered and complex…
Q4i: The figure below represents a social graph of tasks, code artefacts and developers that have been associated with your project.

Are there any particular concerns you have about the size of each representation?

James: I think it quite good…
I thought Paul has not done as much as Tracy…
I did do quite a lot of work on the ‘User Interface’ task, and I think the graph reflects that… This reflects how much code effort that has been put into a task and code artefact…

Paul: It makes the information easier to read compared to having a ranking…

Tracy: The more I look at the graph the more I tend to ask myself a number of questions; for instance why was Mark working on Flipper.java, why was I even looking at Gizmoball.java, or James would have worked much on ‘User Interface’ task…
The graph can really look complex for larger systems.
Q4i: The figure below represents a social graph of tasks, code artefacts and developers that have been associated with your project.

Are there any particular concerns you have about the size of each representation?

Boris: If we have done ‘JUnit Test’ how come it only looks at only Gizmo.java, Square.java and GizmoModel.java…?
Because I know that it should be looking at virtually all of the codes…
There is something wrong…
This tells that there is more work to be done in ‘JUnit Tests’…
I don’t understand why OuterWall.java could get so big when Triangle.java got a lot more coding effort than OuterWall.java…
I think it’s an issue if I am not logged onto CRI since changes are not monitored…
It will be a lot easier if CRI saved login details like svn (Subversion) does, so that developers do not need to remember to login…
I think the history mode can help me a lot more…
Q4i: The figure below represents a social graph of tasks, code artefacts and developers that have been associated with your project.

Are there any particular concerns you have about the size of each representation?

*Smith:* I will say that the network graph is accurate based on the work I have done when I did log into CRI…
The relative variation in sizes of artefact representations gave me a first view of the artefacts I have really worked on…