

# **An Investigation into Collaborative Requirements Analysis Using a Wiki**

**Garret O'Kelly**  
*School of Computing*  
*Dublin Institute of Technology*  
*Kevin St. Dublin 8, Ireland.*

MSc. in Computing (Information Technology)  
DT210

**Dublin Institute of Technology, School of Computing**  
**September 2008**



## **Declaration**

I hereby certify that this dissertation which I now submit for assessment by the School of Computing, Dublin Institute of Technology on the programme of study leading to the award of Master in Science Degree in Computing (Information Technology) is entirely my own work and has not been submitted for assessment for any academic purpose other than in particular fulfilment for the stated above.

Signed

---

Garret O'Kelly

Date

---

## **Abstract**

*During the software development process one serious problem still met by many projects is that the developers don't build the product that the users actually want and need. Requirements gathering is a process conducted by a business analyst early in the life of a project to elicit requirements from concerned parties regarding what needs to be built. This process results in a requirements specification which normally comes in the form of one or more text documents.*

*A specification document should ensure that the right product is built. Even then, such documents can suffer from inconsistencies, ambiguity, omissions, and contradictions. A wrong turn so early in the project can be disastrous downstream. The quality of a requirements specification has considerable implications for the future success of a project. To resolve these problems, interested parties need to discuss, clarify, and arrive at a shared understanding of their needs.*

*Collaborative editing is increasing in prominence on the internet and even in enterprise thanks to wikis, a type of browser based software for collaboratively composing documents over the network. This dissertation aims to consider whether writing the specification collaboratively will alleviate some of the inconsistencies, omissions and ambiguities that cause such problems to projects today. If so, then the increased likelihood of project success will be a very beneficial outcome.*

**Keywords: software requirements collaborative authoring wiki tacit knowledge**

## Acknowledgements

I extend my grateful thanks to my supervisor, Mary Ronner, whose clear thinking and sound judgement regularly brought coherence from confusion. Her generous and patient guidance was invaluable.

I would like to thank the development teams at Curam and Meteor for their valuable input and their valuable time.

The MLPSI team's participation in a pilot study helped me greatly, for which I thank them sincerely. It was a most novel and enjoyable project.

I would like to recognise the lecturers of the Dublin Institute of Technology for making the learning experience there an enjoyable and rewarding one.

To Lorenza for her selfless support and endless enthusiasm.

To my parents for encouraging curiosity.

# Table Of Contents

1	Introduction.....	1
1.1	Research Problem.....	1
1.2	Research Objectives.....	1
1.3	Organisation of The Dissertation.....	2
2	Background.....	4
2.1	Requirements.....	4
2.1.1	What Are Requirements? .....	4
2.1.2	Importance to Project Success.....	4
2.1.3	Must be Multi-disciplinary.....	5
2.1.4	Elicitation.....	6
2.1.5	Specification .....	7
2.1.6	Validation.....	8
2.2	Groupware / Tools.....	9
2.2.1	Wiki.....	9
2.3	Social Space.....	10
2.3.1	Benefits of Group Work.....	11
2.3.2	Problems with Group Work.....	12
2.3.3	Emergence of Moderators.....	14
2.4	Motivation to Participate.....	15
2.4.1	Expectancy/Value.....	15
2.4.2	Social Norms.....	16
2.4.3	Rewards / Valence.....	16
2.4.4	Instrumentality.....	17
2.4.5	Self-Efficacy.....	17
2.4.6	Trust.....	17
2.5	Documentation as Knowledge Management.....	17
2.5.1	Externalisations.....	18
2.5.2	Group Memory.....	19
2.6	Conceptual Model of Co-authorship.....	20
3	Research Method.....	22
3.1	Curam Wiki Users.....	22
3.2	Business Analyst Interviews.....	22
3.3	The MLPSI Group.....	22
3.4	Structuring the Pilot.....	23
3.4.1	Seeding with Supporting Content.....	23
3.4.2	Training the Users.....	23
3.4.3	Addressing Motivations.....	24
3.5	Taking a Baseline.....	24
3.6	Briefing Stakeholders.....	24
4	Consideration of Findings.....	26
4.1	Reflection on the Group.....	26
4.2	Reflection on Conceptual Model.....	26
4.2.1	Wiki as Tool.....	27
4.2.2	Social Space and Moderation.....	28
4.2.3	Conversational Artefacts.....	30
4.2.4	Requirements as Refined Artefacts.....	32
4.2.5	Summary of Findings.....	35
4.3	Revised Conceptual Model of Co-authorship.....	36

4.4 Critique of Methodology.....	37
5 Future Directions.....	39
5.1 Addendum: Facilitators & Thinklets.....	39
6 Appendices.....	41
6.1 Log Of Activity.....	42
6.2 Transcript: Curam Wiki Users Interview.....	43
6.3 Transcript: Meteor Project Manager Interview.....	45
6.4 Transcript: Meteor Business Analyst Interview.....	47
6.5 Transcript: MLPSI Post-Pilot Discussion.....	50
6.6 Requirement Template.....	53
6.7 Requirements Review Checklist.....	54
7 Bibliography.....	55

## Illustration Index

Illustration 1: Bug created/detected phases Source: McConnell (1998).....	5
Illustration 2: Stages of Adoption; Source: Author.....	20
Illustration 3: Revised Conceptual Model; Source: Author.....	36



# **1 Introduction**

## **1.1 Research Problem**

During software development, one serious problem still encountered by many projects is that the developers fail to build the product that the users actually want and need (McConnell 1998). This can lead to expensive rework during development and poor user adoption on delivery.

Requirements gathering is a process conducted by a business analyst expert early in the life of a project to elicit requirements from concerned parties regarding what needs to be built. This process normally results in a Software Requirements Specification (SRS) document.

A specification document should formally define users' needs and ensure that the right product is built. Even then, such documents can suffer from inconsistencies and ambiguities that are inherent in natural language. A specification needs to be described and checked with a certain rigour to overcome this.

Certain co-authoring efforts, such as Wikipedia and the Linux project, are producing results of quality due largely to the beneficial effects of group authorship.

In that vein, the core undertaking of this dissertation is to compare the effectiveness of the expert analyst versus the effectiveness of the group in defining software requirements.

The challenge is to tap into the implicit knowledge of the group of stakeholders and to focus their resources to produce an improved software requirements specification. It is hoped that this will improve the chances of project success.

It is not guaranteed that people will contribute willingly or productively. It is hoped that the background literature will inform on how to encourage productive participation.

This dissertation draws from several areas of knowledge, each broad in its own right: software requirements gathering, collaborative authoring, groupware, knowledge management, psychology, and team work. This thesis hopes to draw on representative literature from each area in considering the subject.

## **1.2 Research Objectives**

This thesis aims to test whether a collaborative documenting tool will allow more people to explore a requirement in greater depth and breadth, thus producing a more robust, better quality specification than traditional analyst-centric authoring.

In order to achieve this, a conceptual model will be derived from background research. The conceptual model will be tested by conducting a pilot study with a group who are in the process of specifying requirements for a new software project.

The study will compare a set of requirements developed by an analyst with a set of requirements that have been refined by the group.

It is then hoped to establish if a link exists between the quality of a specification and frequent and diverse peer review.

This thesis will subsequently refine the conceptual model based on feedback from the pilot so that it more accurately depicts the key stages of collaborative authoring.

The findings will be presented and suggestions made on how future studies might build on lessons learnt.

The subject of this thesis is of practical commercial benefit as it can lead to the timely prevention of a class of bugs that can be very expensive to remedy. It offers lessons learnt on the early stages of a project where intervention is highly cost-effective. It is believed that this can de-risk a project and improve the chances of project success.

It has academic merit in considering a method of specification and review that differs from the traditional analyst-centric method. It also has academic merit in providing a model of collaborative authoring that specifically addresses the software requirements specification process, and in providing a pilot study in this particular area upon which further work can be based.

Finally this topic is of interest for one further reason. When specifying software requirements, the essential problem is one of communicating complex ideas between people. The problem of achieving shared understanding is perennial. The findings of a study in this area may be more enduring than studies into the transient technical problems of the day. Therefore, the findings of this thesis have the potential to remain relevant and valuable for some time to come.

### ***1.3 Organisation of The Dissertation***

This first chapter states that this dissertation intends to address the problem of poor software requirements specifications. The goals of the research are described, as well as the merits, both commercial and academic.

Chapter 2 presents the background material upon which this thesis is based. It covers four topics: requirements gathering, wiki as groupware tool, software systems as a social space, and documentation as a knowledge management tool.

These four topics are related into ascending levels of a conceptual model of requirements co-authoring.

Software requirements and their importance to the success of a project are described. The elicitation, specification, and validation aspects of the requirements process are introduced.

A wiki, as a form of groupware, is proposed as a suitable collective authoring tool that could be used to accommodate these aspects of the requirements process.

Collaborative software is then described as a “social space”. The benefits and drawbacks of group work are described as well as the emergence of moderators among the group. There is a discussion of peoples' motives for taking part in social software.

From social space, the discussion progresses to documentation as a knowledge management tool. The chapter concludes with a conceptual model that integrates the four progressive levels as they relate to the software requirements process.

Chapter 3 covers the research methods used in this dissertation, both primary and secondary. Two sets of interviews are reported on. There is also a description of a pilot study that was conducted, including details on how the study was prepared and structured.

Chapter 4 presents the findings of the study by reflecting on the pilot group and on the pilot study, and relates this to the information provided by interviewees. Each of the four levels of the conceptual model is considered leading to a revised conceptual model as informed by the findings. Chapter 4 also draws a critique of the methodology used and offers lessons learned in the process of conducting the pilot study.

Chapter 5 offers potential future directions of interest arising from this study. Further investigation into the role of the analyst and the make-up of the team is suggested. Serious Games can offer lessons on fun as a motivator while tagging can give insights into stakeholder perceptions. A set of patterns for group facilitation known as thinklets are also offered as a potential further topic. Appendices and bibliography follow this chapter.

## **2 Background**

### **2.1 Requirements**

#### **2.1.1 What Are Requirements?**

The SWEBOK describes a software requirement as “a property which must be exhibited by software developed or adapted to solve a particular problem. The problem may be to automate part of a task of someone who will use the software, to support the business processes of the organization that has commissioned the software, to correct shortcomings of existing software, to control a device, and many more.” (SWEBOK 2004)

“the requirements on particular software are typically a complex combination of requirements from different people at different levels of an organization and from the environment in which the software will operate.” (SWEBOK 2004)

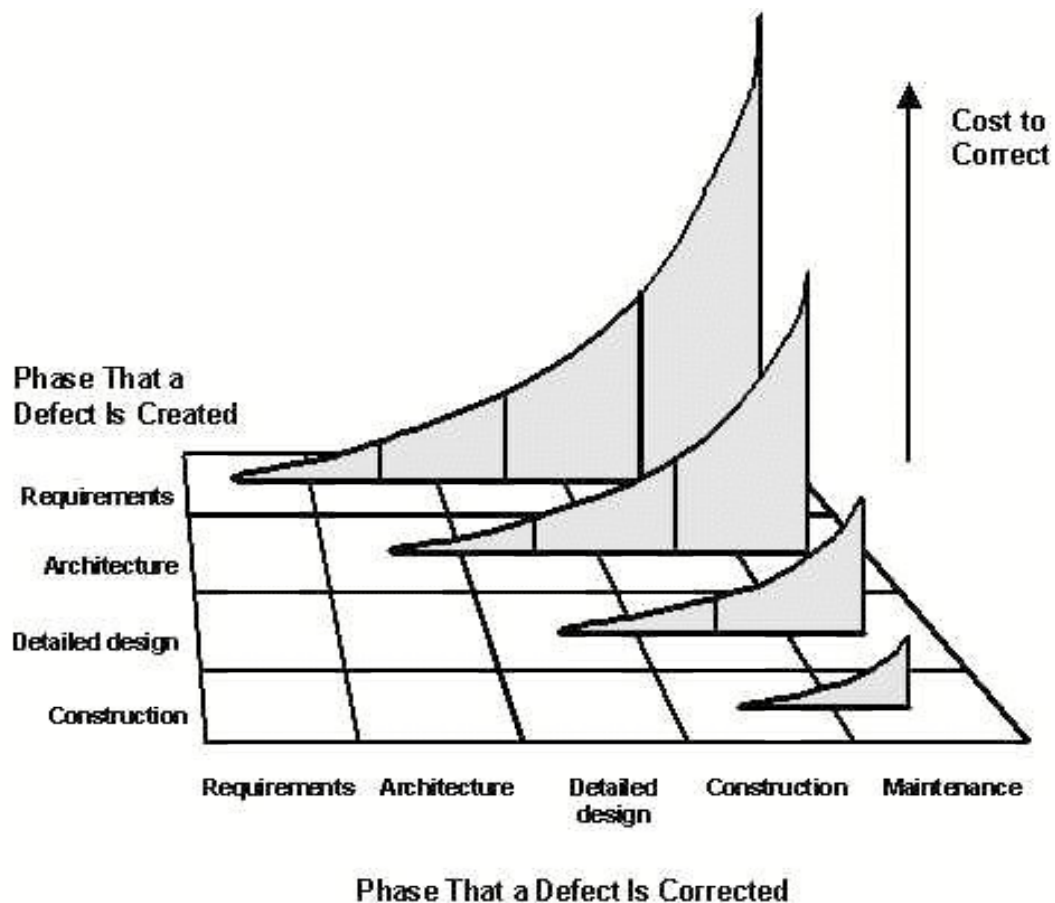
A software requirements specification (SRS) is a set of valid software requirements. An SRS acts as a point of agreement between the customers who commission a software development project and the technical team who will develop the software as to what the delivered product will do.

#### **2.1.2 Importance to Project Success**

Software requirements dictate the implementation of a software system from a very early stage in a software project.

“It is widely acknowledged within the software industry that software engineering projects are critically vulnerable when [software requirements] activities are performed poorly.” (SWEBOK 2004)

In a single diagram, Steve McConnell (1998) impresses the importance of catching bugs as early as possible in the development cycle. The earlier a bug is created, and the later it is found, the more expensive it is to recover from. Requirement bugs that find their way through to production are hugely expensive to rectify.



*Illustration 1: Bug created/detected phases Source: McConnell (1998)*

If tacit stakeholder knowledge remains hidden the following problems occur:

- Requirements remain incomplete because “obvious” ideas are not captured
- The ability to identify conflicts is reduced because not all project-relevant knowledge is explicitly available
- Conflicting interpretations exist due to terminology differences
- Hidden stakeholder expectations and assumptions are not made explicit (Grunbacher & Briggs 2001B)

### **2.1.3 Must be Multi-disciplinary**

When a customer engages a supplier to design and build a software product, both customer and supplier must work together in the specification and design phases of the undertaking.

Typically the customer is not very familiar with the software development process, and the supplier lacks the customer's domain-specific expertise. As a result, both sides need to work together to produce a well written specification (IEEE Std 830-1998).

The software developer must acquire domain knowledge, often from a subject matter expert. This helps the developer infer tacit knowledge that stakeholders fail to articulate; it gives valuable context when assessing user needs and when making design decisions (SWEBOK 2004).

The role of requirements analyst bridges this divide. “This process is fundamentally interdisciplinary, and the requirements specialist needs to mediate between the domain of the stakeholder and that of software engineering. There are often many people involved besides the requirements specialist, each of whom has a stake in the software. The stakeholders will vary across projects, but always include users/operators and customers” (SWEBOK 2004)

On the customer's side, it is important when gathering requirements to equally represent all stakeholder interests, otherwise the interests of one group will dominate while another group is under-served (SWEBOK 2004).

Even within the supplier's organisation, a study by Cadiz, Gupta & Grudin (2000) found that many different roles from designers, to testers, to product managers, as well as people not directly on the product team, were equally involved in the specification process.

### **2.1.4 Elicitation**

Software requirements are a form of tacit knowledge held by the stakeholders. As such they must be elicited rather than collected (Coulin, Sahraouhi & Zowghi 2005). Failing to elicit adequately results in poorly documented requirements (Grunbacher & Briggs 2001B)

One effective technique for eliciting requirements is to hold a series of brainstorming workshops. “The purpose of [brainstorming] is to try to achieve a summative effect whereby a group of people can bring more insight into their software requirements than by working individually. They can brainstorm and refine ideas which may be difficult to bring to the surface using interviews.” (SWEBOK 2004)

Such a forum allows many stakeholders, not just the domain expert, to make potentially important contributions (Arias et al. 2000). Conflicts and assumptions can be exposed, which in turn improves the coherency and integrity of requirements produced (SWEBOK 2004). It has the added benefit of improving stakeholder buy-in and project ownership (Arias et al. 2000, Coulin, Sahraouhi & Zowghi 2005).

During such meetings developers should remember that users may be unable or unwilling to articulate what their work, and thus their requirements, entails (SWEBOK 2004).

Likewise, it is an opportunity for the analyst to make the domain expert aware that the others lack the knowledge that the expert takes for granted.

As people discuss, they gain insight into the motives for each other's stance. Key constraints and assumptions can hinge on the various interpretations of a word. Jargon gets clarified and people arrive at a shared understanding of the meaning of terms, while distracting synonyms can be identified and discarded (Grunbacher & Briggs 2001B).

During elicitation, it is helpful to use a glossary to clearly define terms that may have more than one meaning (IEEE Std 830-1998).

With each fragment of information contributed, the difference between objective fact and subjective opinion emerges, and participants can appreciate others' contributions with a deeper insight (Aria et al 2000, Grunbacher & Briggs 2001B).

The SWEBOK suggests that such meetings be chaired by a facilitator to avoid certain problems: a vocal minority or senior stakeholder might monopolise at the expense of others for example, or discussion may drift from the agenda (SWEBOK 2004).

Grunsbacher and Briggs (2005b) suggest that this forum happen in an anonymous electronic medium so that inter-personal issues can move to the background, and logical discussion move to the fore.

However, it may be optimistic to think that users would actually remain anonymous, especially among an intimate group such as those studied by Shirky (2004a,b,c). Users will recognise each other's style of speech and their opinions from interactions off-line.

On the contrary, Paulus *et al.* (1996) favour an electronic forum where identities are known; they claim that people are “less likely to loaf because the computer task emphasizes individual accountability”.

Regardless of how a meeting is conducted, the result is a collectively created “visible artefact of tacit knowledge, approximating unarticulated hunches, hopes and concerns” (Grunsbacher and Briggs 2005b)

## 2.1.5 Specification

In engineering disciplines a system specification is derived from requirements so that they can be “systematically reviewed, evaluated, and approved.” (SWEBOK 2004). In the software development process, a software requirements specification (SRS) serves that purpose. An SRS does not describe a design, but only documents the constraints that the final design should abide by (IEEE Std 830-1998).

According to IEEE Std 830-1998 a well formed SRS should consist of a set of requirements each of which is:

- a) Correct
- b) Unambiguous
- c) Complete
- d) Consistent
- e) Ranked for importance and/or stability
- f) Verifiable
- g) Modifiable
- h) Traceable

Requirements should be clearly understandable, and free from subjective interpretation (SWEBOK 2004). A specification where items are still “to be determined” is considered incomplete (IEEE Std 830-1998).

It should also be possible to empirically test that a software product satisfies a given requirement (IEEE Std 830-1998, Coulin, Sahraouhi & Zowghi 2005). The specification should give empirical measures, known as acceptance criteria, by which this can be achieved.

It is this authors experience that teams most neglect to make requirements verifiable and unambiguous and that this neglect is a source of considerable difficulty much later in a project.

IEEE Std 830-1998 highlights a paradox whereby making a specification more clear for a technical audience may simultaneously make it less clear for a non-technical audience. To overcome this, a project may consider using a series of related documents that gradually become more detailed and technical in language. "The style must be precise, full and accurately detailed [...] This tends to make manuals dull reading, but precision is more important than liveliness" (Brooks 1995, p62).

### **2.1.6 Validation**

Requirements specifications are subject to “bugs” just as software code is. This “knowledge artefact” should be scrutinised for bugs just as software is (IEEE Std 830-1998). Elicitation and specification should have flushed out omissions, conflicts, and ambiguities. Validating requirements is a way of testing for such problems early while they are still easy to correct and before resources have been committed to development work. As shown in Illustration 1, the earlier a bug is found the better.

Peer review sessions are one tool that can be used to validate requirements. Genuchten, Cornelissen & Van Dijk (1997) claim that “mature software groups spend 10 to 20 percent of their resources on inspections”. It is recommended that a diverse group of project stakeholders schedule one or more review sessions to check the correctness of the requirements before building begins (SWEBOK 2004).

Peer review has several benefits. DeMarco & Lister (1987 p.119) describe The Hawthorne Effect whereby the very act of observing employee performance tends to improve performance. Similarly, preparing for a group review session raises the quality of work submitted: Genuchten, Cornelissen & Van Dijk (1997) found that engineers apparently spent more time checking their work when they knew it would be reviewed.

By making an extra effort before review, engineers can flush out the most obvious defects from their work (Mandiwalla & Olfman 1994). Removing these trivial mistakes makes a review doubly valuable; not only are bugs removed, but senior inspectors are free to focus their time on more worthwhile issues.

Moreover “the defects reported by other inspectors function as triggers to detect more defects” (Genuchten, Cornelissen & Van Dijk 1997). One discovery leads to another so that a review environment tends to expose clusters of related bugs, rather than finding them in isolation.

“Weinberg observed that in [software companies] where developers are not territorial about their code, and encourage other people to look for bugs and potential improvements in it, improvement happens dramatically faster than elsewhere.” (Raymond 1998)

So, just as reviewers can look for syntactic bugs in code, they can also look for bugs of omission, completeness or ambiguity in specifications.

Finally, on a practical level, the SWEBOK (2004) advises that “it may help to provide guidance on what to look for in the form of checklists” when performing a review.

## **2.2 Groupware / Tools**

The platform used to collectively author and review can become a barrier to user participation if they find the medium awkward to use or find that it does not fit their style of working (Decker et al 2007, Fischer 1998).

Groupware can refer to a range of software products used to support the work of a group. This can range from standard desktop software used in a group context to custom-built systems used to support collaboration and mediation. Briggs, De Vreede & Nunamaker (2001) define a groupware support system as “a suite of collaborative software tools that can be used to focus and structure a team’s deliberation while reducing cognitive costs of communication and information access and minimizing distraction among teams working collaboratively toward a goal” (Briggs, De Vreede & Nunamaker 2001).

Such systems can be classed as either meeting-centric, where users interact at the same time, or as asynchronous, where there is a time difference between content being written and subsequently responded to (Mandiwalla & Olfman 1994). Fischer (1998) describes a groupware system that incorporates both an “action space” and a “reflection space” which cater to meeting-centric and asynchronous work-flows respectively. A system that works exclusively in one mode or the other goes against time-and-motion studies showing that group work is a “mixture of interaction and solitary work” (Mandiwalla & Olfman 1994)

### **2.2.1 Wiki**

A wiki satisfies some of Briggs, De Vreede & Nunamaker's definition of groupware (2001), in that it can be used to focus a team’s deliberation and reduces the costs of communication and information access. As such it represents an appropriate tool for collective authoring. Small to medium enterprises rely largely on familiar office productivity suites and email to support their requirements gathering efforts (Decker et al 2007). This thesis seeks to investigate if a wiki presents a viable alternative.

Wikis do not impose the rigid meeting-centric or asynchronous modes that Mandiwalla & Olfman (1994) warn against. Although they are typically used in an asynchronous fashion (Decker *et al.* 2006), the ubiquity of laptops and wireless network access now makes it practical for everyone in a room to be online simultaneously (Shirky 2004b).

In their 2006 paper, Decker *et al* highlight the advantages a wiki offers over a standard office suite. Advantages include:

- one place publishing,
- simple and safe versioning and locking of documents,
- easy linking between documents.

This allows a single source of truth in a known location, rather than differing versions of out-of-date documents spread across file and email systems.

When co-authoring a document, users need to be notified of who changed what and when. It is especially important for long documents that differences are easy to identify (Cadiz, Gupta & Grudin 2000). The versioning features of a wiki cater to this.

In writing about a wiki-like authoring system, Cadiz, Gupta & Grudin (2000) report that program managers liked the system because they could “receive feedback on specs when it was written, rather than waiting a few days for all comments to be compiled into a spreadsheet”. The tasks of content generation and collation can now happen in a single step. This relieves the workload of a document manager by delegating the work across all the original authors, and feedback is more immediate as a result.

Decker *et al.* (2006) also highlight some practical difficulties with using a wiki. To link to a destination page the user must remember the page title, which can be a burden. The fragmented nature of the document makes it very difficult to print a number of related pages as a single document, nor is it possible to work offline with the current generation of wiki software.

Finally, Decker *et al.* (2007) fail to give sufficient consideration to the single biggest issue with wikis in this author’s opinion. To format a page in a wiki, the user must learn a mark-up syntax that introduces a learning curve for technical and non-technical users alike. This acts as a strong disincentive to using the system. There is no standard wiki syntax, several products use similar but different syntaxes.

In these regards, traditional office suites are more user-friendly than a wiki for basic editing. Wikis better support multiple simultaneous authors, but they stop short of offering the structured integrated environment of the sort of purpose-built groupware systems described by Fischer (1998), Arias *et al.* (2000), and Briggs, De Vreede & Nunamaker (2001).

## **2.3 Social Space**

Genuchten, Cornelissen & Van Dijk (1997) suggest that at a basic level software, such as groupware, is merely a tool, but at higher level of abstraction, the social interactions that overlay this transform it into a social space, in this case for the purpose of negotiation.

A social space is a set of social interactions inspired by circumstances hinged around a focal point like a wiki.

When Grunsbacher and Briggs (2005b) talk about negotiating software requirements, they talk in terms of politics, inter-personal opinions, “hurt feelings”, and “social sanctions”. They describe the negotiation as happening in “social space” where each contribution alters the negotiating space.

The metaphor of software as a social space is one that Clay Shirky of New York University strongly subscribes to. “The practice of software design is shot through with computer-as-box assumptions, while our actual behavior is closer to computer-as-door, treating the device as an entrance to a social space” (Shirky 2004c)

Shirky's work tries to inform of the rich social contexts in which software is routinely used and to inform on how that might be exploited.

“There are [...] behaviors that can only occur in groups, from consensus building to social climbing. And yet, despite these obvious differences between personal and social behaviors, we have very little design practice that treats the group as an entity to be designed for” (Shirky 2004c)

Mandiwalla & Olfman (1994) criticise groupware systems that fail to take account of social issues such as status, power, and interest differences. Indeed the technical issues are often less complex than the social issues (Shirky 2004c).

The advantages and disadvantages of working as a group come about from working in a social space, and the technical and social issues cannot be completely separated (Shirky 2004a).

## **2.3.1 Benefits of Group Work**

### **Lateral Thinking**

One of the main benefits of collaboration in intellectual work is the lateral problem solving provided by different minds.

By re-framing the definition of a problem, by stating it differently or considering it from a different perspective, a brainstorming group can arrive at a novel solution to a problem (De Bono 1990, p58) where a lone problem solver may get stuck in a mental rut. “Often, the most striking and innovative solutions come from realizing that your concept of the problem was wrong” (Raymond (1998).

Once someone in a group identifies a problem, even if that person is unable to resolve it, there is a possibility that another person can (Raymond 1998, Grunbacher & Briggs 2001B).

### **Linus' Law**

The social act of peer review, of both requirements and code, has been a major factor in the viability of the Linux movement.

Raymond (1998) attributes Linux creator Linus Torvalds' success to his ability to leverage a large community of developers to debug on a large scale. He observes that reviewing or debugging can be done in parallel – as you add more people, you can continue to gain benefits without blocking others from working. In fact, the practice scales remarkably well (Raymond 1998).

Not only that, but adding more reviewers will exercise the system more quickly, in more diverse ways, and give better test coverage than a single developer/analyst could alone (Raymond 1998). When a problem is found, it is more likely in a large group to find someone who can readily address the issue. Raymond (1998) quotes this as Linus' Law; “Given enough eyeballs, all bugs are shallow”.

Even at development time, peer review can help reduce problems. Agile pair programming, where developers work in pairs to code and solve problems, has the same development costs as traditional practices but reduces the error rate by a factor of 10 (Brooks 2007, 00:45hr).

In software testing terms you could say that the larger and more diverse a test community is, the more fringe cases and test paths are explored, thus the more test coverage a system gets (SWEBOK 2004).

This early and frequent proactive detection of specification bugs leads to economy of effort (Raymond 1998) by avoiding the expensive reactive bug fixing of Illustration 1.

## **Division of Labour**

People are social by nature, and working in teams is a natural way to organise and divide work. In the context of business, we work in teams due to increased complexity and to win the first entrant advantage in the market (Brooks 2007, 00:10hr)

Traditional media models, such as newspaper or television, divide people into producers or consumers. “One of the major roles of new media is to provide the opportunity and resources for social debate and discussion, rather than to merely provide access to pre-digested information” (Fischer 1998).

With an interactive medium, the user can be sometimes a consumer and other times a producer. Users can act in either role depending on the context of their interaction. Indeed, informed participation requires that individuals actively contribute knowledge as well passively consume it (Fischer 1998).

That participation also starts to change the traditional division of labour. We move from the large efforts of a small group of people to the small efforts of a large group of people (Fischer 1998). The unit of work becomes much smaller. The many small efforts of the community of Linux testers and debuggers (Raymond 1998) are a good example of this.

## **2.3.2 Problems with Group Work**

### **Focus On Tools**

During collaborative work, and especially during tele-collaboration, there can be an over-emphasis on tools and an under-appreciation of the social issues. Tele-collaboration is (incorrectly) being driven by tools rather than by social need. Many of the tools are irrelevant; instead there is a need for more analysis of when collaboration itself is useful (Brooks 2007, 00:51hr).

It has been found that people prefer working by telephone and corresponding by documents and drawings rather than by using teleconferencing (Brooks 2007, 01:02hr). When describing tele-collaboration Brooks suggests that each office has an "ambassador" in the foreign office because there is no substitute for face-to-face contact (Brooks 2007, 00:46hr).

## **Conceptual Integrity**

When designing software, it is essential to retain conceptual integrity (Brooks 2007, 00:10hr). It is important to get wide participation when specifying the requirements. There is also considerable value in a design being critiqued by multi-disciplinary groups. Nonetheless, the design itself should originate from a single designer or a small group of designers. "If a system is to have conceptual integrity, someone must control the concepts. That is an aristocracy that needs no apology" (Brooks 1995, p46)

## **Organisational Resistance**

Hasan and Pfaff (2006) document a case study where senior executives rejected using a wiki partly for fear that knowledge exchange would undermine their power and subvert the organisational hierarchy. Such resistance is counter-productive. One of the hallmarks of an excellent company is that it gives employees autonomy, supports innovative thinking, and treats people with trust (Peters & Waterman 2003 pp.227, 238,310)

## **Credit for Authorship**

Authorship or level of contribution may be a point of contention between wiki users, since anonymous users will not be motivated by recognition (Hasan & Pfaff 2006). However they offer the counterpoint that users' dependence on each other can overcome disagreements about authorship (Hasan & Pfaff 2006). In a professional context, it is also likely that users will recognise authorship of each page - especially where specialist knowledge is shown, or where previously voiced opinions are committed to writing.

## **Flaming**

"Flaming" is a form of anti-social aggressive behaviour seen on the public Internet, especially where users' real identities are unknown to each other. Shirky (2004c) describes flaming as a sort of public performance rather than just a personal attack on other users. He categorises this as a problem of the tragedy of the commons. Briefly stated, the tragedy of the commons occurs "when a group holds a resource, but each of the individual members has an incentive to overuse it" (Shirky 2004c) – in this case the resource is communal attention.

On a public wiki flamers don't have exclusive ownership of their contributions. "If someone acts out on a wiki, the offending material can be subsequently edited or removed" (Shirky 2004c).

Abusive online behaviour would clearly have consequences in a small group of people who know each other in real life. As such, flaming is an unlikely problem in a professional environment where identities are known.

## **Vandalism**

Somewhat related to flaming are vandalism and edit wars (Wikipedia 2007a), where users deliberately contribute junk content, or try to enforce their opinion by continually editing and reverting content.

Two things mitigate against these problems in a work environment: users typically know each other so are bound by social constraints (Shirky 2004a), and the topics covered in a professional context are not typically very controversial or emotive (Hasan & Pfaff 2006)

## **Social Blocking**

Paulus *et al.* (1996) describe the phenomenon of social blocking in brainstorming sessions. This happens when people lose their train of thought while waiting for others to finish speaking, when discussion idles or digresses off-topic, or when people withhold opinions for fear of being judged critically. Paulus *et al.* (1996) claim that “In the typical oral idea generation paradigm, group interaction reduces performance by about 50%”.

He suggests that brainstorming electronically can optimise contributions by avoiding social blocking. People “do not have to take turns to express their ideas” (Paulus et al. 1996)

“One advantage of computer-based interaction is that it limits the amount of time individuals engage in off-task social conversation. Individuals in this paradigm are less likely to repeat verbally others' ideas or to give elaborate explanations or stories associated with their ideas” (Paulus et al. 1996)

Paulus *et al.* (1996) refute the conventional wisdom that brainstorming in a groups inherently stimulates more ideas than working solo (De Bono 1990). They claim that group participants only outperform lone brainstormers when the group is very large. Brooks echoes Paulus' recognition of social blocking by stating that solutions to problems often come when you are thinking alone (Brooks 2007, 00:45hr)

In terms of using time efficiently, their suggestions make sense, however we should not dismiss verbal repetition and storytelling lightly. They may serve an important psychological or social function in reinforcing or confirming ideas, or in contesting authority for example.

### **2.3.3 Emergence of Moderators**

As a group commits to its existence as a group it reaches a point where it needs structure to maintain the orderly working of the group in the face of its members' conflicting interests (Shirky 2004b).

The medium must have an owner and there must be social values in place that are effective at resolving differences (Brooks 2007, 00:51hr).

It is no longer feasible to give all members full and equal rights as this leads to what Shirky calls “the tyranny of the majority” (Shirky 2004b). A core set of users who have a stronger than average interest or passion in the successful working of the group will emerge as moderators who steer the tone and etiquette of the group (Shirky 2004b).

Shirky (2004a) describes how reputation becomes a moderating force in social software, especially in small groups. Non-technical sanctions like communal judgement are sufficient to prevent unwanted behaviour (Shirky 2004c). Social software need not include a technical solution to these issues, simple persistent user-names are sufficient for a reputation “system” to exist in the users' minds.

In a professional work setting, it would be expected that someone with managerial authority would adopt the role of moderator for a work group.

Decker *et al.* (2006, 2007) also suggest that it is within the moderators' remit to set out the structure of content on the wiki, given its initial free-form nature. This will mitigate against content sprawl and ensure that the site remains easily navigable.

## **2.4 Motivation to Participate**

Social systems start to enjoy a network effect if they reach a critical mass, and risk failing otherwise (Briggs, De Vreede & Nunamaker 2001). A system's usefulness attracts users, which in turn makes it more useful to others, attracting them in turn in a positive feedback loop. In order to attract a group of requirements stakeholders and reach that mass it is beneficial to understand the motivations of participants.

Hertel, Nieder & Hennmann (2003) summarise two theoretical models of motivation from the field of social psychology: the Extended Klandermans Model (EKM) and the VIST model (Valence, Instrumentality, Self-Efficacy, Trust)

The EKM relates to wider social movements, whereas the VIST model relates to processes in small work teams and builds on expectancy/value models of behaviour (Hertel, Nieder & Hennmann 2003).

Together, the two models describe six components of motivation:

- Expectancy/value
- Social norms
- Rewards/valence
- Instrumentality
- Self-efficacy
- Trust

### **2.4.1 Expectancy/Value**

People consider the feasibility and potential value of a project before committing to it. “Such expectancy × value constructs have a long tradition in social psychology” (Hertel, Nieder & Hennmann 2003)

Raymond (1998) remarks that open source efforts need a base of substance from which to attract contributors. “it’s almost always easier to start from a good partial solution than from nothing at all [...] When you start community-building, what you need to be able to present is a plausible promise” (Raymond 1998). That is, people must see the project as feasible.

The initial effort need not even be particularly robust. Shortcomings may provoke contributors to fix those shortcomings (Fischer 1998). The aim when seeding an effort is primarily to establish the expectancy that a goal can be achieved.

## 2.4.2 Social Norms

We are also motivated by others' perceptions of us, especially 'important' others, such as friends, family members, and people of high status.

“Motivation to contribute to a movement should be higher the more positive the expected reactions of significant others are, weighted by the perceived importance of these significant others” (Hertel, Nieder & Hennmann 2003)

In brainstorming sessions Paulus *et al.* (1996) find a form of the Hawthorne effect (DeMarco & Lister 1988 pp.119-120) whereby “information about the performance of others can increase performance of groups” (Paulus et al. 1996)

The Linux project provides an example of norm-based motivation where a *gift for prestige* culture exists (Brooks 2007). Torvalds appealed to developers' sense of reputation and professional pride in front of their peers to elicit assistance (Raymond 1998). Reciprocity can also be a strong motivator (Dacher 2008 00:37hr), especially in a gift culture like open source software.

Norms can equally motivate or discourage: “The open-source community’s internal market in reputation exerts subtle pressure on people not to launch development efforts they’re not competent to follow through on” (Raymond 1998).

## 2.4.3 Rewards / Valence

People also do a mental cost/benefit analysis before undertaking a project in anticipation of personal rewards for their efforts.

Costs can include the investment of time, money, mental effort, and risks to health, among others (Hertel, Nieder & Hennmann 2003). In the case of a software design effort, rewards can include a feeling of control, the enjoyment of problem solving, of mastering a tool in depth, a sense of community citizenship, or a satisfied ego, among others (Fischer 1998).

Fun is also an important motivator (Raymond 1998, DeMarco & Lister 1987 p.157) and merits further consideration. These sources do not elaborate on fun can be exploited for encouraging behaviour. There appears to be opportunity for further research in this area.

Even mundane work can be rewarding if colleagues enjoy good interpersonal relations. People can be more interested in who they work with than what they work on (DeMarco & Lister 1983, p.148).

The VIST model categorises these motives using the term “valence”. The Oxford English Dictionary (2008) defines valence as the “Emotional force or significance, spec. the feeling of attraction or repulsion with which an individual invests an object or event”.

Fischer finds that the personal value of contributing to “computational memory” (discussed below) must outweigh the effort. If value is marginal, or if the effort distracts from real work, then there is little incentive to contribute (Fischer 1998).

#### **2.4.4 Instrumentality**

“Instrumentality is defined as the perceived importance or indispensability of one’s own contributions for the group outcome” (Hertel, Nieder & Hennmann 2003). The more important one feels to the successful outcome, the more motivated to take part.

It is this author's experience that stakeholders, especially front-line staff, underestimate their importance to a project. They can assume that analysts, managers or team leaders will be able to somehow specify the requirements themselves. However front-line staff typically hold knowledge about work-flows and business data that is key to the design of a new software system.

The sense of instrumentality can equally demotivate if the person feels their contribution will not be noticed, or will be insignificant

#### **2.4.5 Self-Efficacy**

Team members must believe they are capable of achieving their goals. They should not feel overwhelmed or intimidated. If they believe they can succeed in a task, then members may gain confidence from each other and be more likely to get involved (Fischer 1998; Hertel, Nieder & Hennmann 2003).

Brooks remarks that open source developers typically do not have to convey their requirements to another, they are skilled enough to build for themselves and know their own requirements intuitively (Brooks 2007, 00:34hr).

#### **2.4.6 Trust**

Finally team member must trust in each other that their work will be reciprocated and not exploited by others (Hertel, Nieder & Hennmann 2003). Team members should trust that there will not be a tragedy of the commons.

“Trust is particularly relevant in virtual teams in which misunderstandings and fear of exploitation can escalate more quickly due to fewer face-to-face interactions” (Hertel, Nieder & Hennmann 2003). It must also be relevant in a competitive corporate environment where individuals would be reluctant to cede advantage to a competing colleague.

The issue of credit for authorship, described above, would presumably be less problematic in an environment where people trust each other.

### ***2.5 Documentation as Knowledge Management***

Requirements analysis must be a multi-disciplinary exercise, and should happen in a social space. The benefits and drawbacks of group work as well as peoples' motivations dictate the level of participation.

Once there is participation, the fundamental problem in software development is one of communication (Brooks 1995). The essential complexity, according to Brooks, lies in

resolving communications difficulties to arrive at a shared understanding of a problem and solution.

Complexity arises from the need merge different perspectives of large amounts of information into a shared understanding (Arias et al. 2000). “The predominant activity in designing complex systems is that participants teach and instruct each other” (Arias et al. 2000)

Documentation is a powerful means to address this (Parnas 2007, 00:24hr). As projects grow ever larger, designers need to rely ever more on documentation to communicate all details amongst each other.

## 2.5.1 Externalisations

Documentation is one way to overcome the human memory's limited capacity for detail, a document is an externalisation of knowledge held in the mind.

“When distributed cognition is at work between the individual human mind and artefacts, such as memory systems, it often functions well because the knowledge an individual needs is distributed between his or her head and the world (e.g., an address book, a system of email message folders, or a file system). But in the case of distributed cognition in operation among groups of minds, a group has no head, no place for the information about this distribution of knowledge to be available to all members implicitly— therefore externalizations are critically more important for collaborative design” (Arias et al 2000)

When collaboration is compounded by differences in geography or time zone, shared documents, then voice, then video are most useful in that order (Brooks 2007, 00:48hr). Brooks does not expand on the reason for this, but it may be due to the relative permanence of the written word over the spoken, or due to the maturity of supporting tools.

Assumptions about the user, indeed assumptions in general, should be documented. The team leader should create a single shared mental image of the users and of the application (Brooks 2007, 00:26hr). In other words, tacit knowledge should be externalised. If details are not externalised then it leads to people making incorrect assumptions (Brooks 2007, 00:30hr).

Brooks (2007 00:31hr) considers vague information to be worse than incorrect information. Some organisations may argue for a centrally controlled document management system with formal edit and review stages for reasons of quality control (Hasan & Pfaff 2006). However, such a centralised and structured environment “will make it difficult to adopt a ‘community approach’ towards knowledge acquisition.” (Hasan & Pfaff 2006).

Decker *et al.* (2006) recommend capturing all scraps of information however vague, and importantly, recommend flagging them as vague. They believe in making all information public so it can be contested and stripped of ambiguity. Once details are surfaced they can be expanded on and refined into useful knowledge (Hasan & Pfaff 2006), and omissions and conflicts can be resolved through discussion.

A knowledge base can be seeded by experts and subsequently fleshed out by the rest of the group as happened with a transportation design system discussed by Arias *et al* (2001). Likewise, linking to blank wiki pages, say Decker *et al.* (2006), will encourage users to

populate those pages. Using the built-in linking features of a wiki, a team can maintain a special “overview” page of open issues to act as a starting point for conflict resolution (Decker *et al.* 2007).

## **2.5.2 Group Memory**

Externalising is not only useful in the present. “Group memory” is important over extended periods of time. Not only must the essential complexity be resolved, but a history of the decisions and rationales that led to that resolution should also be kept (Arias *et al.* 2000).

It is particularly important during staff turnover. If a system architect leaves there must be someone who understands the history of the system and is concerned for the integrity of its design (Brooks 2007, 01:13hr). Shared documentation is concerned with knowledge retention by a group rather than by an individual (Hasan & Pfaff 2006)

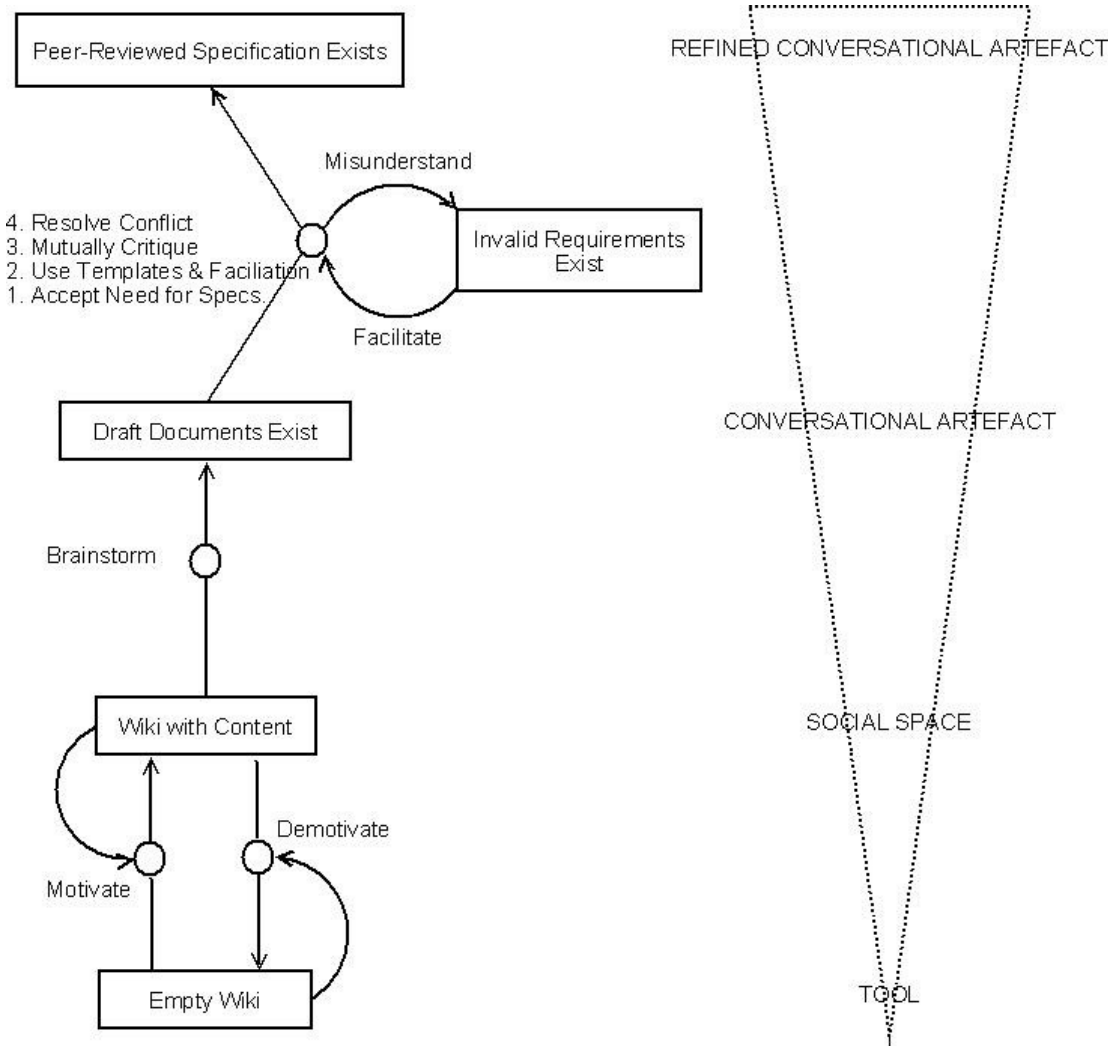
Requirements and design are negotiated and change over time. Their rationale is not always obvious unless there is a known course of decisions that led to them.

A wiki is very well suited as a group memory tool where information creation and storage is conducted through collaborative writing (Hasan & Pfaff 2006). Shirky (2004b) describes Wikipedia articles as conversational artefacts where the product is a record of ongoing discussion around a subject.

A conversational artefact, then, is the sum of the statements, perspectives, rationales, compromises, and agreements of a dialogue over time.

Eliciting requirements is an iterative process that discovers and exposes requirements through dialogue, with stakeholders negotiating back and forth. This thesis suggests that the software requirements specification should emerge as a conversational artefact from that process.

## 2.6 Conceptual Model of Co-authorship



*Illustration 2: Stages of Adoption; Source: Author*

The preceding paragraphs lead to the model above, showing stages of adoption on the left, and the ascending abstractions of those stages on the right.

A project begins with a need to specify what software should be built. A software requirements document is used to meet this need. The document should include input from all stakeholders.

An empty wiki represents a basic tool to facilitate collaborative authoring. Features like centralised storage and versioning make it a preferable tool over standard office suites.

The wiki can become the focal point for the co-authoring effort. Fostering an environment of cooperation attracts a group of collaborators. Motivations can encourage or discourage people to participate. Once people are working together, the wiki takes on the characteristics of a social space, wherein people both generate and consume content.

In that social space, people can present their different perspectives and aggregate their fragments of information. A dialogue of negotiation produces rough-draft documents as conversational artefacts.

These rough drafts must be evolved through discussion and peer review into valid requirements to derive a quality software specification. Conflicts and ambiguities come to light and can be resolved. A software specification produced in this way benefits from the combined abilities and knowledge of the whole group.

The group will have explored and tested for assumptions, omissions and ambiguities in greater depth and breadth than the traditional analyst could alone. This will produce a more robust, better quality specification. With that, it is hoped that the specification would have fewer bugs, thus reducing the overall cost and risk of the project.

This concludes this dissertation's statement of hypothesis. What follows describes a pilot study established to test this conceptual model. The findings of the pilot study are also presented.

## **3 Research Method**

Secondary research consisted of an initial literature review using sources from IEEE, ACM, and DIT and Trinity libraries as well as books by recognised subject matter experts. Topics researched include: wiki usage, requirements analysis, collaboration, co-authoring, motivation and project management.

Two business analysts and a group of wiki users were subsequently interviewed. A pilot study was then carried out with a group who were starting to develop educational software. At all times in this dissertation, the word “pilot” refers to the period of observation, not the project itself.

### **3.1 Curam Wiki Users**

This author interviewed a team of 4 developers in Curam Software who use their company wiki regularly as a mandatory part of their work practices. Curam is a Dublin based company that develops a software framework for integrated Social Services programs. Curam maintains a wiki which is read by approximately 300 employees. Roughly 80 these are also regular contributors. It was unknown among the interviewees how long the wiki has been in use, but this author assumes greater than two years, based on the interviewees' work histories.

The wiki is used to store release notes, initial development proposals, design documents, and test plans. Documents on how to set up and use particular tools are also stored. The developers have, on occasion, used it to document development requirements which, in turn, prompted further discussion. A tips and tricks section is also maintained. Use of the wiki forms a mandatory part of their work practices.

### **3.2 Business Analyst Interviews**

A senior project manager (Meteor PM) and a senior business analyst (Meteor BA), both from Meteor, were interviewed. The interviews included questions on stakeholder discovery and inclusion, and eliciting and verifying requirements.

The interviewees were then asked about day to day problems they meet in documenting and managing requirements. They were also asked to compare the different issues encountered on small versus large projects. Finally the interviewees were asked for their opinions on the suitability of a wiki to address the points above.

### **3.3 The MLPSI Group**

A pilot study was also conducted to test the validity of the conceptual model proposed.

The Modern Languages in Primary Schools of Ireland (MLPSI ) initiative is a government sponsored initiative, active since 1998, to introduce European languages to primary schools in Ireland.

In 2008 the MLPSI programme worked in conjunction with the coordinators of Dublin Institute of Technology's Serious Games programme to develop educational computer games

for school children between the ages of 8 to 12 years old. In this project the National Coordinator represented the customer for the product.

The DIT / MLPSI group comprised of

- the MLPSI National Coordinator
- a project chair person
- a project manager/analyst
- a technical lead
- five language lecturers
- a professional graphic artist
- a graduate developer
- this author as observer.

The group was contacted in March 2008 about acting as a pilot group for this thesis. At that time, they had done some preliminary brainstorming, but the project had not been officially launched yet. This offered a good opportunity to study the requirements gathering at the early stages of the project. A timeline of the pilot study can be found in the appendices below.

### **3.4 Structuring the Pilot**

The goal of this pilot was to allow comparison between a baseline set of requirements and a set of requirements authored collaboratively by the group to test the conceptual model above. To reach the point where this was possible, the pilot first had to gradually build up the usage of a wiki through the levels described in the conceptual model above.

To that end, preparatory work was done to incentivise collaboration, and mitigate against the disincentives

#### **3.4.1 Seeding with Supporting Content**

According to the conceptual model, a wiki needs something to attract users to begin using it initially.

First, background project management material was added. The team's existing project mission statement was added to the front page. Minutes of initial meetings and brainstorming sessions that had circulated as email attachments were collated into designated pages.

A glossary of terms was also added as suggested by the SWEBOK (2004). A page describing "People, Roles & Responsibilities" was also added. A sample profile was provided for others to follow.

An initial layout for content was also set up as suggested by Decker *et al.* (2006).

#### **3.4.2 Training the Users**

Steps were taken to address ease of use as suggested by the background reading. Initially this author created pages and posted content for users while they became acquainted with the wiki. A page of frequently asked questions (FAQ) was compiled and added to each time a user raised a question about using the wiki.

One training session was also arranged which gave a conceptual overview of collaborative authoring as well as showing basic editing features and explaining the syntax needed to format content. The session was attended by three of eight team members due to conflicting commitments.

### **3.4.3 Addressing Motivations**

While seeding the wiki and training team members, a mail was also sent to everyone specifically addressing some of the motivators mentioned above. The project chairperson was asked to draft a mail in his own words to the effect that:

- the project was worthwhile and achievable (expectancy/value)
- that each team member was capable of achieving their goals (self-efficacy)
- that each member was a necessary part of the team (instrumentality)

The chairperson was asked to send the message since he was the authority figure on the project, and by social norms, a message from him would have more impact than a message from an external observer. It was hoped that this would predispose or bias people towards participation.

### **3.5 Taking a Baseline**

Having seeded the wiki with content and encouraged team members to use it, the next objective was to take a set of baseline requirements and assess if the group could refine and expand them to a further level of detail through discussion and review.

No requirements for the project had been defined before the pilot. However, the technical lead and the project manager together had produced a “game design document” for each mini-game to be developed. The game design document described the learning objectives of the game, the players' roles, and a brief description of game-play in a single A4 page of prose.

It was the group’s intention that the developer would develop the games based on these design documents, therefore they were a close equivalent to specifications for the project. It was decided to take the game design document for one game to act as the baseline against which to compare any co-authored and co-reviewed work that might come from the pilot.

### **3.6 Briefing Stakeholders**

The game design document was structured into a set of requirements which followed a template (also added to the wiki). These constituted the baseline requirements.

To help with reviews, a check-list was adapted from, and informed by, several sources (Tervonen 2008, IEEE/EIA 12207.1-1997). It was kept brief, with the specific team and the project subject in mind.

The checklist and the requirement structures were explained and discussed at a face to face meeting. The team was then asked to review the requirements and append to them online as

they saw necessary for one month before the developer was due to begin coding the game in question.

The intention was then to compare the baseline to the refined version according to attributes such as readability, ambiguity, weak phrasing, options, continuances, and orthogonality (SWEBOK 2004). That is to say, the requirements refined by the group would be checked to see if they were actually more clear, singular, comprehensive, and measurable than the initial set.

## **4 Consideration of Findings**

### ***4.1 Reflection on the Group***

The MLPSI group was very suitable for this study in a number of ways. The team was small in size such that each person knew all other team members, and was free from any disruptive inter-personal issues. Varying levels of experience and diverse disciplines were represented, and discussions were always open and fluid.

The team was also working within budgetary and expertise constraints typical of small scale development efforts. This type of team is more likely to adopt low-cost tools and ad-hoc development processes, such as those offered by a wiki, compared to a team with access to specialist skills and tools.

Given the constraints on available expertise, the role of business analyst was, de facto, shared among the project manager, project chair, and technical lead. As such, it can be said that the group's skill-set lacked the expertise of a professional business analyst. This proved to have significant impact on the pilot.

Since the group was not co-located, it would convene once a month to meet and discuss progress. The majority of team members were more comfortable with face-to-face discussion than online discussion, so the distributed nature of the team limited the opportunities to discuss how the wiki worked, and to discuss requirements themselves.

It is worth noting that there were different levels of comfort with online tools. The graphic artist and the developer would have preferred more online collaboration, whereas the less technically inclined users were still only becoming comfortable navigating the wiki when this pilot concluded.

It is also noteworthy that the MLPSI Coordinator was willing to trust the team with a lot of responsibility for the games. This thesis accepts that she was reliant on their expertise, and acknowledges that the team benefited from lots of creative scope. However there is a risk that the customer is under-represented and this could lead to mismatched expectations when the final product is delivered.

Finally, the project schedule aligned well with the timeline of this thesis, the main brainstorming and analysis phase of the project corresponded closely to the research phase of this thesis.

### ***4.2 Reflection on Conceptual Model***

This thesis depicts a conceptual model for participation that is described in four layers of ascending abstraction. It is this author's assertion that the model is essentially coherent.

However, the pilot study informs of some key aspects of the model that should be identified and emphasised. It would appear from the pilot study for example, that, in the environment studied, the levels of 'wiki as tool' and 'refined conversational artefact' are more significant than social space and conversational artefact.

## 4.2.1 Wiki as Tool

Curam uses its wiki primarily as a groupware tool to disseminate up-to-date technical details. The main advantages they cited included concurrent centralised authoring and built-in versioning of documents.

When documenting requirements, Meteor use an indexed cross-referenced documentation practice based on Microsoft Office since “email trails would be far too unstructured” (Meteor BA). Data entry is laborious, however, and needs a dedicated documentation manager to coordinate work across teams. Nonetheless, it allows the analysts to easily make consistent changes to documentation.

The wiki was initially accepted as a tool by the MLPSI group because it was championed by the project technical lead, and because there was a precedent for using the wiki for similar projects in the DIT.

The “people, roles & responsibilities” page proved an effective way of serving two needs at once. Firstly, such information made it clear who everybody on the team was, who did what, and what everyone could expect of each other. More importantly though, it gave users a chance to learn how to contribute content and edit pages by themselves. After a short time all team members had filled in their own profile information.

Regarding a wiki as purely a text editing tool, the significance of WYSIWYG editing (“what you see is what you get”) cannot be over-stated. Business users accustomed to modern office suites found it unintuitive switching between read and write modes and using a mark-up language to format their content.

Even technical users from both Curam and the pilot study found mark-up inconvenient (the more advanced user interface of Google Docs proved superior for the pilot team in one exercise). Although everyone was able to edit content, overall people found it unduly tedious and difficult.

On the MLPSI pilot one user was frustrated by difficulty tracking changes to the wiki. Others informed him that the “recent changes” page could be used – a tip that they were able to share from the training session. This detail could also have been added to the FAQ. Ongoing email and verbal support were a necessary precursor to using the wiki. A single training session was inadequate to address the whole group’s questions. It is questionable whether more training would have overcome the ease-of-use barrier.

At an early team meeting it emerged that non-technical team members were unfamiliar with the terms “avatar” and “mini-game” being used by the technical members. These were subsequently defined in the glossary. Once the terms had been explained face to face the glossary was not referenced often. The main value of the glossary then was that it simply drew attention to the need to identify and define terms.

Decker *et al.* (2006) raise some practical difficulties with using a wiki. They warn that the sometimes arcane page titles make it difficult to link pages. This was not a big problem for the pilot users – once the initial content structure was laid out, most pages were rather self contained.

They also state that the fragmented nature of wiki articles makes it very difficult to print a number of related pages as a single document, The initial content structure also mitigated against this for the same reason.

Off-line access was another potential issue raised by Decker *et al.* (2006). Pilot participants had broadband or wifi access at all times so access never became an issue.

However, site navigation did prove awkward until a set of navigational links was provided on every page.

In collaborative authoring, the tool should be incidental, and focus should be on the collaborative effort. If the tool itself is cumbersome it distracts from the business task at hand. Functionally the wiki proved to be adequate as a tool, but a WYSWYG interface would be of great benefit.

Seeding the wiki had two purposes: to make it the focal point of the project and to encourage centralised documentation. In both regards, the seeding effort was successful – the pilot group readily arrived at this conceptual level of usage.

## **4.2.2 Social Space and Moderation**

### **Stakeholder Inclusion**

When asked about the day to day difficulties in gathering requirements, the Meteor business analyst emphasised the social nuances of the work. Inclusion is an important precursor to user acceptance. If stakeholders feel they were overlooked during the project they may resist accepting the system on delivery.

Stakeholders help the analyst identify further stakeholders and owners of information. Without this help, the analyst may question the wrong people, miss important information or omit stakeholders entirely (Meteor BA).

Inclusion is too important for the analyst to rely solely on referrals from stakeholder to stakeholder. Instead Meteor tries to methodically discover all interested parties from an integrated model of their business processes (Meteor PM). The interviewees did not elaborate on this aspect of their methodology. Stakeholder discovery in a group as small as the MLPSI pilot was not a problem, all stakeholders were known to the project.

High-ranking or long-serving staff can also pose a problem if they mistakenly believe that they should be consulted (Meteor PM). This is political ground. Such influential users, if snubbed or ignored, can become a considerable obstacle to achieving project goals. One needs to consider their sense of instrumentality. Nonetheless, the analyst can circumvent them by “having a chat with their manager through informal channels” (Meteor BA). The MLPSI team was too small for this to be a problem.

### **Conflict Resolution**

One technique used in Meteor is to meet people individually to identify contradiction, followed by group meetings where discrepancies can be resolved.

The Meteor business analyst was of the opinion that small teams work more efficiently due to informal structure, saying that "informal contact is oil in every organisation". There is closer common interest, more sharing of responsibility and less "finger pointing". Departmentalised organisations are more bureaucratic and have more pockets of self-interest (Meteor BA).

In Curam, content ownership was based clearly on areas of expertise. Content is factual rather than controversial; it is seldom necessary to 'moderate the social space'. Nonetheless, an implicit moderation system exists based on company hierarchy. Where contradictions arise they are resolved informally in face to face meetings. The 'moderators' and content authors arrange content along the company's lines of business.

Flaming and vandalism were non-issues in a professional environment where users all know each other.

For the Meteor analyst it was important to keep open channels of communication with frontline staff and to remain on good relations with them; they are the people who can provide empirical fine detail to the analyst.

The MPLSI group did not see the emergence of moderators during the pilot study. It would seem from Shirky (2004a) that moderators only come from a group's need to police itself. Moderators could equally be seen as stewards and champions for a group. Not only do they indicate that a group has reached a large size, but they also indicate that a group is led by project champions. Feedback from the pilot group suggested that the group wanted more guidance from a facilitator during the specification review of requirements.

## **Motivations**

In describing their motivation to use the system, the Curam users describe the typical cost-benefit proposition of work relevance versus ease of use. One user remarked that he would contribute content that was of interest or relevance to his work but was discouraged by poor ease of use. Ease of use also strongly discouraged pilot participants to use their wiki.

A mail was sent by the MLPSI project chairperson to address instrumentality, self-efficacy and expectancy/value motivations. The impact of this mail was not measurable. However all members did participate actively, and the tone at meetings was one of optimism in meeting goals.

Interestingly, the "people, roles and responsibilities" page gave an insight into the graphic artist's and the language lecturers' interests in the project. The graphic artist was able to link to his portfolio, and the language teachers saw the wiki as something that could potentially be re-used in their teaching environment, while this author was motivated by academic & professional interest. All these can be considered variations of the reward motivation. The project manager felt encouraged to contribute knowing that others depended on their work – a form of social motivation.

During the brainstorming meetings where games were being prototyped there was a palpable sense of fun in the pilot study group, people clearly enjoyed the creativity in conceiving the games.

Curam were obliged to use the wiki as part of their work practices. The MLPSI were not obliged to use the wiki, but the technical lead strongly favoured its use. Usage can be encouraged by authority figures in this way.

Anonymity was not considered by the pilot group. Indeed they were keen to know the origin of ideas so they could direct responses accordingly.

People did not find social blocking to be a problem at face to face meetings, although some meetings did fork into several simultaneous conversations.

### **4.2.3 Conversational Artefacts**

#### **Discussion Protocol**

When the pilot group were asked to discuss requirements online, there was uncertainty on how to do this. Should they raise questions in the body of the article itself? Should they use the associated discussion page for each article, or have a single designated discussion section?

This issue was anticipated in advance. Two alternative options were offered to the users. This confused people and compounded the ease-of-use problem. In retrospect, choosing a single option in advance would have given clearer direction. This issue could then have been addressed as part of user training.

Finally issues were discussed face to face in meetings rather than online, and outcomes were recorded afterwards. Conducting discussion online would have allowed all comments to be captured, whereas some comments were lost by recording after the meeting.

#### **Knowledge Mismatches**

The Meteor project manager emphasised that the single biggest problem is to achieve shared understanding across different business areas. Everyone has a different understanding of how things currently work and how they ought to work (Meteor PM).

All information should be double checked for correctness. For example, a team leader or supervisor will typically give a broad overview of their area of operation. However, their frontline staff may be able to give a more accurate and detailed account (Meteor BA). Others will provide incorrect information in good faith based on their false assumptions (Meteor BA).

When stakeholders mistakenly take their knowledge for granted, granularity of expression becomes problematic. Lacking common ground, they will state their needs in terms that are either too general or too specific to be meaningful to the analyst (Meteor BA).

So, while there is dialogue, there is room for misunderstanding. If this knowledge mismatch is identified upfront then it can be addressed sooner. In Meteor, the business analyst is expected to identify and resolve these conflicts. Although he has the advantage of being professional and methodical, having the stakeholders discuss *en groupe* could also help reveal these points of knowledge mismatch.

For one game in the pilot study, the developer needed to design labels to display translated text on limited screen space. A constraint emerged during discussion when the German language expert explained that German sentence structure led to a much longer translation than Italian or French. Initially the language team were unaware of the display size restrictions, and the developer was unaware of verbosity of that particular German phrase.

Using acceptance criteria for a requirement, having a testable measure of success, and recording the rationale for a requirement can help bridge such knowledge mismatches and pre-empt discussion

In another instance, the developer was initially unaware of a requirement that was raised before his involvement in the project. Once aware of it, he was still unaware of the rationale behind it: the National Coordinator had mandated that it was required to satisfy the teaching curriculum. This put his design effort at a potential disadvantage until both the requirement and the rationale were brought to his attention.

### **Perceived Feasibility Mismatches**

When users ask for requirements, they often have little cognisance of the wider impacts beyond their area of concern. “It's not deliberate, it's just not their priority, they don't even think about it” (Meteor BA). For example, one department may ask for a feature that seems trivial to them, but which would require considerable data entry or a change of work practices in another department.

The language team and the technical team had a different level of awareness of the technical difficulty in implementing each request. The language team would (understandably) treat all requests as equally technically feasible, while the technical team continued to give feedback on what was easy and what was difficult to implement. The whole team soon realised that there were mismatched perceptions of difficulty and began to manage that mismatch through discussion.

### **Ambiguities & Assumptions**

During the course of the discussion, the pilot exposed and clarified several assumptions. All the games were being designed to support teaching French, German, Italian, and Spanish. At one point the graphical artist, who was Irish, was asked to depict a kitchen scene. He based his depiction on his idea of a typical Irish kitchen. During a review of his prototype the Italian and French language teachers corrected him on a few points: the typical French and Italian kitchen does not feature a bread-bin, a kettle or a loaf of sliced bread, and the electrical sockets are a different shape. The kitchen was subsequently re-drawn to a culture-neutral depiction.

The assumptions were rooted in the cultural backgrounds of the participants. This can equally happen among people with different professional or organisational backgrounds and can only be surfaced by making explicit the assumptions and testing their validity, by review or otherwise.

## **Document Ownership**

When there was a clear delineation of expertise, such as language expertise, people readily co-authored a document. The various language experts all contributed to a table of translated terms without hesitation. However when adding design ideas to the game design documents, where expertise was less clearly delineated, there was a reluctance to intrude on someone else's document. People still sensed that the original author was the sole owner of a document.

Perhaps people are comfortable with consuming content but are still unpractised as producers. The concept of collective ownership of information must be stressed. The analyst needs to actively encourage people to amend and append to content as well as just reading it so the team can move from the large efforts of a small group of people to the small efforts of a large group of people.

## **Prototyping**

Visual prototypes helped to build shared understanding at least as much as the wiki – the prototypes became the focal point of discussion and debate, while the wiki became the point where decisions were captured. Prototyping served to elicit information.

The debate, the alternatives, the rationales and the rejected ideas were not captured. As such, the wiki content could not truly be considered a conversational artefact.

## **Group Memory**

It can be valuable for people to understand the history of decisions that led to the current point (Meteor PM, Arias *et al.* 2000). The developer tended not to refer back to the wiki for a record of decisions made, but the project manager did.

The MLPSI wiki pilot ran for five months whereas Curam has used its wiki for over two years. Although Curam use their wiki primarily at the level of documentation tool, their “tips and tricks” section acts more like a true externalised group memory.

## **Communications Program**

Use of a wiki might have more success if it is done within the framework of a company communications program. The Meteor project manager and business analyst establish the requirements gathering process at the outset of a project. They undertake an exercise of educating other team members on what will be needed of them and how the process will be conducted. A company communication program, described by policy or standard procedure, can be of great value in resolving such issues. However a business can easily underestimate the time and resources needed for such a task (Meteor PM).

A brief presentation was made to the pilot group regarding use of the wiki and how the pilot study would be conducted. On reflection, more time should be invested in informing stakeholders about what the requirements process entails and in how to express and discuss their ideas.

#### 4.2.4 Requirements as Refined Artefacts

Once elicited, the conceptual model suggests that to be useful, co-authored requirements need to be refined into a valid Software Requirements Specification. Specification and validation need to happen at this point.

Meteor achieves this refinement by regularly sending every document out to stakeholders with a structured comments sheet. Traceability and accountability of feedback is important, and any unaddressed issues remain open. This feedback promotes discussion. The analysts may also give a final presentation if the business process under discussion is important enough.

When verifying requirements for completeness, Meteor do not use any metrics per se. Rather they rely on a method of accounting for all information inputs and outputs in the business processes being modelled. The interviewees did not elaborate on the specifics of this method. As with conflict resolution, responsibility for specification and validation lie with the analyst as expert.

In the pilot study group, people understood roughly what requirements were, and could understand why they would be useful, but did not appreciate why they needed to be so detailed.

They were unconvinced of their necessity and were deterred by the upfront effort. They were satisfied that the game design documents would serve their purpose and saw more detailed requirements as unnecessary and time consuming work. They saw it as a facilitator's responsibility to drive the specification process as it was not their primary duty on the project and cited lack of time for the task.

There was a requirement that all games “should incorporate a cultural aspect” as well as the basic language teaching element. This is a statement that is open to wide interpretation that was not fully addressed.

The problem revealed itself when the group was unable to decide what supporting audio-visual material to source.

The group was then asked what the requirement meant, and how it could be satisfied. In the ensuing conversation each person had a different interpretation of what constituted cultural detail and how they thought the requirement could be satisfied. This was addressed by prototyping different suggestions.

Another requirement stated that “the games must satisfy the syllabus” – but it was unclear how it would be decided that the games fulfilled this requirement. This was not yet addressed at the conclusion of the pilot for this dissertation. There is a risk that this could lead to mismatched expectations when the final product is delivered.

The Meteor interviewees were asked if they considered a wiki an appropriate forum to gather and verify requirements. The project manager was not in favour of it. He thought it would be too difficult to consistently manage change. He was thinking of having to manage change himself, instead of allowing the group to self-regulate under his guidance.

Using a wiki assumed a level of technical writing that is beyond the lay-person; he maintained that business people would be unable to distil user needs into system requirements (Meteor PM). An analyst and a process would be needed to facilitate people. The pilot study proved him correct on this point.

The business analyst thought it would be useful for including a wide range of people, and for collecting valuable low level detail you would otherwise miss. From observations on the pilot group, this would rely on people overcoming the sense of document ownership to add detail.

However he suspected that people would be “too lazy” to contribute. This is addressed by the reward/valence aspect of the EKM and VIST models (see above). If the benefit of contributing outweighs the effort to contribute then people will participate. The pilot study showed that a wiki's ease-of-use added notably to the effort to contribute.

He shared the project managers concern about the process becoming chaotic, but thought that user education could overcome this. He would rather use the wiki as a forum for elicitation and verification, but would keep the authoritative specification outside the wiki.

## 4.2.5 Summary of Findings

There should be an ongoing period of training and support for users who are new to wikis. The “people, roles & responsibilities” exercise was a very useful first step for getting participation. Even with training, ease of use should be a priority in the user interface and WYSIWYG editing should be strongly favoured over modal editing and syntactic mark-up.

Unlike in open source or voluntary projects, private organisations can compel employees to participate, and other motivators would be largely moot. Where that is possible widespread scheduled peer reviews could be made a mandatory part of the development process.

In a very small team, members' understanding of terminology soon converged - the glossary was useful mostly in highlighting the need to define terms at the outset. It may be more useful on a larger team.

Fun was a notable aspect of this project – people enjoyed the creativity of designing games and were very engaged by it. It would be useful if a project could avail of this to motivate participation.

The role of analyst as facilitator is a very significant one. Educating project stakeholders on the importance of the requirements process may be a crucial early step. The role is then needed to champion the requirements process and take responsibility for it. A facilitator needs to encourage stakeholders to contribute and discuss on an ongoing basis. This thesis was suggesting that review could happen on an ad-hoc basis. This may be possible, but it would also be advisable to schedule regular review sessions as well to guarantee a minimum effort.

People think in terms of documents and are still conscious of ownership. This needs to be replaced by a sense of collective ownership. Even if this is stated at the outset, it needs to be re-affirmed until stakeholders change their perception of document ownership.

Laying out templates, document structures and formats in advance avoids non-essential discussion and allows all to focus on content. A discussion protocol can also be established within a program to streamline communications. This should be presented as part of training at the outset of a project.

Some people strongly preferred face to face prototyping to elicit requirements over online discussion. Doing so, people will arrive at draft requirements without much difficulty, but it is a considerable conceptual step up to progress to a refined specification.

Assumptions are a function of “cultural” background. Those of different backgrounds invariably hold different assumptions. Recognising this can allow people to make their assumptions explicit and validate them.

Throughout this thesis there is an implied assumption that a medium- to large-sized team is working on a complex project. If this were so then documentation is certainly relevant. However, if a project is under a certain size or complexity then a pure prototyping approach can be more pragmatic. Likewise, if the software product will have a short useful life then the cost of retaining knowledge of the design might not be justified.



## **4.4 Critique of Methodology**

Concerning the methodology used in this dissertation, there are some lessons learned that may inform subsequent similar experiments.

This dissertation was based on stating a hypothesis, devising an experiment to test it, and then comparing the results with expectations. The hypothesis was based on secondary research, and was subsequently tested by primary research. Results are compared with expectations in the section on findings, above.

Interest in this topic was first prompted by this author's personal experiences working on software projects as an analyst-developer. However first-hand action research has been omitted since this author is not working as a designated business analyst at this time of writing. If the researcher was actively working as business analyst on a project it would greatly ease access to the pilot study group.

Secondary research consisted of background reading from sources in the libraries of Trinity, DIT, the ACM, IEEE and from books by recognised subject experts. This dissertation covers only a very small section of the overlap between information elicitation and the social sciences. Further research can avail of a wealth of material that was outside the scope of this paper. Bjorn Decker of Fraunhofer Institute for Experimental Software Engineering has conducted very relevant work in this area. The area of Industrial Psychology would also be directly applicable.

A conceptual model derived from the research gives an excellent base from which to conduct an experiment. It provided a very useful framework for thought throughout the lifetime of this dissertation. Further reading on Systems Thinking models (Soderquist 2006) may lead to a more useful model and suggest how to improve feedback to refine the model further.

Primary research took the form of interviews and a pilot study. The interviewees from Curam and Meteor represented mainly technical staff. It would be beneficial to get equal representation from non-technical stakeholders. The Meteor interviewees in particular were very experienced in their roles and gave clear and useful insights.

The effort to control the environment of the pilot by seeding the wiki and briefing the stakeholders was relatively successful as evidenced by the group's smooth progression through the first three stages of the model.

The MLPSI group was not co-located but rather convened monthly to discuss progress. In terms of logistics this introduced additional issues into observing the pilot.

A co-located study group would give an observer more contact-time to build up a rapport with participants, which would make it easier to encourage participation. More time could also be spent overcoming training issues.

Regarding the desired size of the pilot group, a study may be constrained by availability of such pilot groups. It would be preferable though, where possible, to explore a study group with multiple developers because the need for clear requirements would be greater.

The need for a facilitator was identified in advance by the model but this author failed to ensure that either he or one of the project members acted formally in that capacity during the requirements specification and review stages. This was of significant relevance to the outcome of the study.

Taking a meaningful baseline poses a problem. It is arguable whether the game design documents represent a fair baseline since they were not explicitly structured as requirements. However, they were chosen as such because they represented de facto requirements for the project.

Alternatively, a project that has already developed a valid software requirements specification could be asked to repeat the exercise using the process suggested by this thesis (if a willing group could be found). The first set of requirements would form the baseline, and the second set could be compared. However, the results would be biased by the group's knowledge gained in preparing their first specification.

An alternative may be to have a facilitated team “compete” against an analyst using conventional elicitation, specification and validation techniques and compare the outcomes. The conventional analyst's specification would form the baseline against which the team-produced results could be compared.

The metrics of ambiguity, orthogonality and measurability are subjective and simplistic. Nonetheless, had requirements been produced, one would expect a notable difference between baseline and refined requirements so that a difference would be observable even by simplistic measurements. This thesis suggests that the larger the group, the more notable would be the difference.

If more objective and detailed comparison were needed, the field of lexical analysis may offer a guide on how to compare for such attributes. Detailed lexical analysis is outside the scope of this dissertation.

## 5 Future Directions

There is an upfront cost in carrying out requirements analysis. If done badly, then the negative effects may not be noticed until much later in the project. If done well, the positive effects might not be noticed at all. This time delay makes it difficult to impress the advantage of solid requirements at an early stage. There would be considerable merit in studying effective techniques to bridge this perception gap to improve stakeholder participation from the outset.

Unfortunately this dissertation has not succeeded at establishing whether or not there is a link between diversity of contributors and quality produced. It would be beneficial if empirical evidence could be collected relating Linus' law to specification quality.

This dissertation considered only that the business analyst as facilitator was a key role. It would be interesting to look further at the minimum or optimum skill-mix needed within a group to conduct collective requirements analysis. Likewise the group involved in this study was relatively small. Group size as a success factor should be considered further.

Another experiment that could prove useful is to allow a community of stakeholders to build a “folksonomy” of tags around requirements. A folksonomy is an informal ontology of words or phrases used by a social group to add metadata to a resource of some sort. The linguistic and statistical study of the tags could be highly informative of how different stakeholders and the group as a whole perceive each requirement.

It may be that prototyping is a naturally more suitable elicitation technique for groups such as the MLPSI pilot group and their visually-oriented games project. Online discussion and prototyping are complementary techniques for eliciting requirements. Prototyping offers the benefit of visual immediacy while online discussion has the advantage of permanency for group memory. There would be benefit in studying how these two approaches could best compliment each other for a given project or team type.

At the time of writing, the Dublin Institute of Technology runs a course in Serious Games. The course focuses on (graphical computer) games as training and education tools. The pilot group showed how strong a motivator fun can be during work. It would be very interesting to infuse a groupware system with elements of play to encourage positive behaviour. A social space can be a space for play as well as for work. The MLPSI pilot consisted of a team of language teachers designing games; this author considered planting word games within the wiki content to encourage participation, but was constrained by time.

### 5.1 Addendum: Facilitators & Thinklets

The absence of an analyst as facilitator proved to be pivotal in the pilot study. Late research during the study discovered the subject of thinklets. These also may prove a useful area of further research if working with a team where a facilitator is needed but is unavailable.

Briggs, De Vreede & Nunamaker (2001) have found a repeating pattern whereby groupware systems are used briefly in organisations before they are discarded again. Such groupware systems are initially managed by a professional facilitator who subsequently leaves or is promoted away from the project (Kolfshoten 2004). When systems are handed over to users to operate themselves, they find it difficult “to understand what the system was supposed to do for them and for their group” (Briggs, De Vreede & Nunamaker 2001).

A set of techniques that can potentially mitigate against the loss of a facilitator are “thinklets”. A thinklet is to facilitation as a design pattern is to software design. It is a recognised approach to a problem: “A thinklet is a named, tightly scripted, process for creating a single repeatable predictable pattern of collaboration among people working together towards a goal [...] In other words, a thinklet is a codified facilitation routine” (Kolfshoten et al. 2004).

A team member who is not necessarily a trained analyst might organise a set of thinklets into a script to guide a group through a brainstorming session. Results of such scripted sessions or “modules” promise greatly improved productivity over unscripted sessions (Briggs, De Vreede & Nunamaker 2001).

Classes of thinklet include: diverge, converge, organise, evaluate, and build consensus (Kolfshoten et al. 2004). A session can be scripted to address requirements gathering whereby a team focuses first on conceiving of a variety of possible requirements, organising them by importance, considering the merits of each, and then coming to a consensus on which are valid needs.

## 6 Appendices

## **6.1 Log Of Activity**

This log shows a time-line of key events in pilot study. It spans the duration from this author's first contact with the group until conclusion of the pilot study.

- |            |  |
|------------|--|
| 12 Mar 08  | First contact with the MLPSI group. The entire team is in attendance. Preliminary brainstorming on game ideas has already begun.   |
| 19 Mar 08  | This author seeds the wiki with initial brainstorming content to date, adds supporting project material, introduces the wiki to users.   |
| 12 Apr 08  | Project manager starts to layout the game ideas according to the Game Design Document template   |
| 17 Apr 08  | Wiki training session held. This explained the conceptual purpose of a wiki and also explained how to contribute content. In attendance were Ute, Maria-Jose, and Isabelle of the language team. |
| 7 May 08   | Users actively adding content: the majority of the team members (both technical and non-technical) have uploaded files or added text to the wiki at this point.                                  |
| 4 June 08  | Whole team meet. The team seems to accept the need for requirements. This author explains the template to the team. We choose two games to examine.  |
| 2 July 08  | Whole team reconvenes. This author reiterates the need for requirements. Review has yet to begin..   |
| 30 July 08 | Pilot study concludes  |

## 6.2 Transcript: Curam Wiki Users Interview

Transcript of interview held with users of the Curam wiki, 27<sup>th</sup> February 2008

[dev1] : Senior Software Engineer and Technical Team Lead

[dev2] : Software Engineer

[dev3] : Senior Software Engineer

[author] *What types of people (job titles/roles) use your wiki? What do they use it for?*

[dev3] All roles from associate software engineer to principal engineer use it. In application development they use it to see any release notes for a particular release. A new person may use it to set up a new machine. In general it's used for almost everything relating to internal development.

It is used especially in Technical Infrastructure, for how long I don't know. Roughly about 80 people maintain their project documentation on wiki, about 300 people use it for information relating to the product. We maintain all internal technical documentation on wiki. Release notes, initial development proposals, test plans and how to use a particular tool.

[author] *Do many people actively contribute, peer review, or amend content?*

[dev3] Tech. Inf. [Technical Infrastructure] use it extensively in development and documenting releases. App. Dev. [Application Development] use it as a reference guide in their development to see what has been released. The material does not get peer reviewed as such, however it is freely editable. Information is updated frequently and informally in my experience.

[author] *For different job titles, what encourages or discourages contribution?*

[dev1] There are very few discouragements. Our team uses the wiki as an easy to modify repository of most of our development documentation. This of course excludes end user documentation.

The reason our wiki is favoured by developers is the ease of modification and version history provided. The ease of modification is particularly important as we try to keep our documentation matching the code base. For example our design documents are on the wiki when the code is changed for whatever reason the corresponding changes to the design documents can be made quickly.

Our internal processes documents, tips and tricks and other information is also very well suited to a wiki. The developers in the department are actively involved in the definition of our processes and so an easy to maintain medium accessible to all is very useful.

[dev2] Content that interests or is of relevance to me would encourage me to contribute.  
Difficulty of formatting and editing would discourage me from contributing.

[author] *Is the wiki used as a group discussion medium or is it more for posting up reference info, meant mainly to be read?*

[dev3] It is mainly posted to be read, however I have used it to document requirements for future development and this in turn has lead to discussion.

[dev1] It's mostly for just for reference

[author] *How are problems like inconsistencies, omissions, contradictions, and ambiguity in information resolved?*

[dev2] Through discussion, in person or via email.

[dev1] There is no ongoing formal process for addressing these kinds of problems other than the fact that when you create a page you are the page owner. In future if people find problems with it they can fix them themselves. Only in the case were there might be some debate as the changes is the page owner consulted. On occasion full reviews of certain sections of the wiki are done if many problems are being encountered or some major change has occurred that warrants that. This is done as needed.

[author] *What do people feel are the pros & cons of the wiki compared to alternative approaches they've used?*

[dev2] It's simple to get simple content onto the page (once you've learned the syntax) but you need to learn syntax and it's not easy to format content. It's not easy to compare different versions.

[dev1] The benefits of the wiki are subtle we produce basically the same documents as before. The main pro is the ease of modification and the visibility to the whole department. The seamless version control is also another big plus.

[author] *Who manages the wiki or acts as moderator? (e.g. project manager, front-line staff, technical staff, department secretary, no one in particular?)*

[dev2] No one in particular.

[dev1] The wiki has no overall moderator, page owners are the moderators of the content they create. This makes sense as most of the information on our wiki is of a technical nature. Team Leads and Managers are of course responsible for the output of their teams but this part of their normal managerial duties and not specific to wiki management. The IT department support the wiki at a technical level (upgrades backup etc.), but the content is managed by the teams themselves.

### **6.3 Transcript: Meteor Project Manager Interview**

Transcript (abridged\*) of interview with Senior Meteor Project Manager, 04th March 2008

[author] *What methods/methodologies do you currently use to gather requirements?*

[PM] We use interview, observation and existing business process documentation, then we redesign with the stakeholders and they sign it off.

[author] *How do you currently collate and disseminate requirements? Are there any issues around how that's done?*

[PM] We use an indexed system, all processes are numbered. Not all stakeholders are aware of historical changes and it can be difficult to reach shared understanding. The management of the documentation itself isn't a big problem. We have one person managing it at all times.

[author] *What are the most significant day to day problems you face in practice?*

[PM] Getting shared understanding across the business is the biggest problem. It's not clear who the upstream/downstream stakeholders are, we try to methodically discover stakeholders from an integrated model of the processes. Stakeholders refer the analyst to each other, but we don't rely on it. Another problem is when you get conflicting requirements, then you have to go with the business decision. Another problem is people who think they are stakeholders but who are not really stakeholders.

[author] *In your experience how does requirements gathering differ on different sized projects/teams/organisational cultures?*

[PM] Small projects tend to have a low level of best practice or governance. An organisation might not make allowance for a communications program, it will be a considerable draw on human resources.

[author] *What metric(s), if any, do you use to gauge the quality of a requirement?*

[PM] We don't use metrics per se. We know what inputs and outputs we should have. If the process can convert all the inputs to the required outputs then we know we're covered.

---

\* Audio recording available on request

[author] *What do you think a wiki could contribute from the different stakeholders' perspectives?*

[PM] Personally I would be against it. Our documentation is available to all but people tend to want documentation explained to them. Even with versioning, with a large number of users changing various pieces of it, changes have to be reflected throughout the document and it's very difficult to maintain. We had a consultative process to establish a documentation process. Then we designed a comments sheet that we send out so they can review documents. That will prompt discussion and then sometimes we will do a final presentation. There is an element of peer reviewing.

[author] *What would deter an extended team from adopting a wiki to “write the spec themselves” (under the guidance of the analyst)*

[PM] It would be too chaotic because you're assuming a level of knowledge of requirements writing, a lot of business people aren't in a position to convert their business requirement into an IT system requirement, so that has to be carefully managed.

[author] *Can you imagine a prose version for business people and technical version that would be derived from it.*

[PM] I don't think you would get the business people to do it, that's not their job, they need to be facilitated, that's an analyst's job, they're trained for different things.

## 6.4 Transcript: Meteor Business Analyst Interview

Transcript (abridged\*) of interview with Senior Meteor Business Analyst, 05th March 2008

[author] *Just to get started, could you give a quick overview of the methods you use to identify stakeholders and to gather requirements? Interview, observation, reviewing document flows?*

[BA] The first thing is to identify the stakeholders, if the customer gives you the wrong stakeholders you'll ask either the wrong questions or will miss information. That can backfire on you, they won't accept the system later. The method depends on the project. You have to agree the method with the stakeholders.

Everything you hear you need to double check, I have to go to the people who cover the area in question and ask if you're the one I need to ask questions or is there someone else I should be asking. Are you the one signing this off?

Everything you hear, you have to confirm that it's correct. You have to do it in a nice way, you don't tell people you're checking it.

[author] *What are the most significant day to day problems you face as analysts? E.g. managing documentation itself is not a real problem. Finding the stakeholders, people issues, conflicting requirements?*

[BA] When you start the project you have to confirm the level of requirements gathering. Some people give you too little detail and others give you too much detail. Sometimes the context is missing. People give requirements but don't say why they're asking for it.

[author] *Peter said shared understanding across business units is a big problem. Shared understanding of what exactly (processes/requirements)? How can you foster shared understanding?*

[BA] For example Sales might have an operational requirement and they ask for a few extra fields on a screen, but they've no idea how that might affect Customer Care – they don't understand that someone will have to run reports over that information, and the data warehouse guys will have to store that. It's like a lack of context. It's not deliberate, they don't even think about it.

[author] *What sort of techniques can you use to resolve problems like inconsistencies, omissions, ambiguities, prioritisation*

[BA] What happens a lot is, when there's a change, you need to meet people individually to identify conflicts, and then afterwards get everyone in a room to resolve that. Often

---

\* Audio recording available on request

people will mistakenly just think things work a particular way so you need to check everything, every statement they make.

[author] *How do you mitigate against the influence of senior/long-serving staff who incorrectly think they are stakeholders?*

[BA] If someone is a senior and they act as a stakeholder you need to be very careful. Senior staff have a lot of influence, you can never ignore them. As soon as they suspect you're ignoring them they can work against you. There are always the informal channels. You can always have a chat with their manager through informal channels and ask who signs-off on things.

[author] *I understand Meteor use a review process whereby documents are sent out with comment sheets and that promotes discussion. Could you identify the strengths and weaknesses of those reviews.*

[BA] Every comment is a one liner, who raised this and when, and has an official status. Any unaddressed issues remain open, it provides traceability and accountability. If we have a spreadsheet I can read it and say this issue is closed because person X did this and this and this. Email would be too unstructured. Data entry is laborious though and needs a dedicated documentation manager to coordinate across teams – you have to type up every single comment, and there are three teams involved, so you need three spreadsheets and very careful document management.

[author] *Leading on from the comment sheet, What kind of communication channels are important or effective between the analyst and the business, and amongst stakeholders themselves?*

[BA] In Meteor we call the stakeholder the business manager who will sign-off requirements, but he won't have a low level knowledge of the day to day operations. At a lower level the team leads and front-line staff have a much better idea of what happens. It's important to be in touch with those and be on good terms with them.

[author] *What metrics, if any, do you use to gauge the quality of a requirement?*

[BA] We have to start with a good understanding of the as-is processes. Then on a case by case basis we decide if we have enough detail, we have to go step by step, every single bit of the process must be mapped in detail, we need to know where every bit of information comes from. That's the only way to guarantee some kind of quality.

[author] *In your experience how does requirements gathering differ on different sized projects/teams/organisational cultures? E.g. Small teams have these dynamics, large teams have these other dynamics... the big things for Peter were governance & best practice.*

[BA] There's a big difference. Small teams work more efficiently due to informal structure. Informal contact is oil in every organisation. There's closer common interest and less finger pointing and more sharing of responsibility. In a bigger organisation you have bigger structures. Departmentalised organisations are more bureaucratic, there are more pockets of self-interest. They tend to work in little islands.

[author] *What do you think a wiki could contribute from the different stakeholders' perspective?*

[BA] A wiki would be an interesting concept. I wouldn't allow anyone to delete anything. You could address a wide view of people. People would feel included, nobody is left out, you would get valuable detail you would otherwise miss, but people would be too lazy to do it.

[author] *Do you think it would turn chaotic?*

[BA] Well, you'd have to manage it, You can avoid chaos by communicating how you are going to use it. I would rather just use this for verification, but I wouldn't have the definitive document up there.

## 6.5 Transcript: MLPSI Post-Pilot Discussion

Transcript of discussion held after the MLPSI pilot, 04<sup>th</sup> August 2008

In attendance:

[PM] : Project Manager

[GA] : Graphical Artist

[Dev] : Developer

[author] *Did the wiki help you to understand what the project was trying to achieve?*

[GA] Yes

[Dev] Yes

[PM] It did. It also made me think more about the structure and details of what we were trying to accomplish much earlier than I would have without the wiki.

[author] *Were there ways in which the wiki was particularly helpful for teamwork?*

[Dev] not really.

[PM] It was good as a point of reference to see where things were at. But it also encouraged and almost forced people to articulate or write their points of view, make contributions and play the roles they had been assigned. You could see at a glance what part of the project needed some work done.

[GA] Yes. It focus ideas discussed in meeting into a game document for myself and mark to follow.

[author] *Did the wiki have any particular shortcomings in supporting teamwork?*

[PM] A few members of the team struggled with the techie side of things but once they got the hang of it, they were fine.

[GA] Yes. It was never really embraced as a forum for discussion of know game issues. I believe this was due to people's ability to use the technology. I don't think anyone wants to admit they didn't understand and kept queries and information until meetings. I think a series of smaller tutorials relevant to editing topics may have been helpful.

[Dev] It's not very interactive. People don't know when other people have added to it.

[author] *Are there any ways it could be made more useful for any future teamwork?*

[GA] Yes. I think so, I still update it with art work when I get the chance.

[Dev] Not really. It's a wiki, not adaptable enough for design and development projects.

[author] *What encouraged/discouraged you to contribute content to the wiki?*

[PM] I was encouraged by knowing that others were waiting to use my input to do their own work. As all the areas were linked, the wiki was a representation or reminder that it was a team project.

[Dev] I was put off just by the time it took, a blog was easier and better for me to get any news across to team members and email was the best way to contact people.

[GA] I thought it was a great way to share content. However Google docs proved better suited to sharing text files. With Google people could edit documents online without fear of two people editing at the same time and one copying over the other work as they re-uploaded it. That said no complaints about contributing artwork.

[author] *How confident are you that that people shared the same understanding of what the games should be like?*

[PM] Fairly confident, The game design documents were discussed and clarified so I 'd reckon everyone was on the same page. The fact that we had the sample graphic there as well was a huge help

[GA] Not very confident. I don't think we were on the same page until we saw artwork. Recent problems with language content being supplied for the kitchen game made it clear that a lot of people didn't understand how it would work. We asked for cultural recipes to have 5 ingredients and for each of them to come from the vocabulary list. Some people added more than 5 ingredients and items that weren't on the list.

[Dev] I don't think they grasp the amount of time and work that goes into creating a game in the first place.

[author] *What do you think affected the effort to define requirements?*

[PM] it was hard to find time for it on top of normal workload, it needed to be driven or coordinated by a facilitator, the game design documents were the right level of detail, more detail was overkill

While the requirements are a good idea, for this project, I felt they might be a bit too restrictive for the programmer. It was important given the time scale that we accomplish what we set out to do, and to do this, it has become necessary to give more flexibility to the programmer in how they accomplish our objectives. For example for the "Cook-It" game, many of the initial requirement have changed, these do not alter the objective of the game which is to reinforce the language learning but from the programmers end, it just couldn't work the way we envisaged in the proposed time frame.

[GA] the game design documents were the right level of detail, more detail was overkill.  
Because of programming and time restraints we haven't been able to fulfil everything set-out in the documents.

[Dev] We didn't really have time on top of our regular work. The wiki was awkward to use and the game design docs were enough anyway.

*[author] Do you feel people understood the relevance of requirements, what they were for?*

[PM] Some did understand but others didn't or were not necessarily too interested in that aspect

[GA] Yes. But again, because of programming and time restraints we haven't been able to fulfil everything set-out in the documents.

[Dev] They were not that relevant, maybe if there was a large team of developers who didn't attend any meetings instead just coded, the requirements might help.

*[author] Did you find that other peoples ideas distracted you or gave you further ideas during brainstorming?*

[PM] I found that other peoples ideas stimulated my own

[GA] Likewise.

[Dev] Likewise.

*[author] would you profile online or face to face brainstorming*

[PM] online - there are fewer distractions, to be honest I am okay with either!

[GA] there are fewer distractions online.

[Dev] either.

*[author] Did you ever refer to the wiki to keep track of what was being agreed on over time?*

[PM] Yes I did.

[GA] Yes, still do.

[Dev] Not really.

## 6.6 Requirement Template

- **Requirement Id:**
- **From Game Design:** *which game design does this requirement originate from?*
- **Description:** *give a brief description of the requirement*
- **Rationale:** *give a justification of the requirement*
- **Originator:** *the person who raised the requirement*
- **Status:** *potential / under discussion / accepted / dropped / delivered*
- **Likelihood of change:** *how likely is this requirement to change in the future: low / medium / high*
- **How to measure success:** *an empirical measure to test if the solution matches the original requirement*
- **Conflicts:** *other requirements that cannot be implemented if this one is implemented*
- **Importance to Customer:** *how important is this requirement for the customer: low / medium / high*
- **See Also:** *link to related material...*
- **Possible implementation / Further notes:** *ideas on how this requirement might be achieved*

## 6.7 Requirements Review Checklist

### Purpose of this page

Each requirement should be checked for clarity, correctness, completeness, and relevance. This page acts as a checklist of questions to ask when reviewing requirements.

*It might be useful to keep a printout of this page to hand while reviewing requirements.*

Questions to ask about each requirement:

- Is each requirement written in **clear, concise, unambiguous** language?
- Is each requirement **free from content and grammatical errors**?
- Is any necessary **information missing** from a requirement? If so, is it identified for discussion?
- Have **internationalisation issues** been adequately addressed? (e.g. have different languages' money/number/date formats been considered)
- Is each requirement **in scope** for the project?
- Are all requirements **actually requirements, not design or implementation solutions**?
- Is each requirement **verifiable by testing**, demonstration, review, or analysis?
- Can all of the requirements be implemented **within known constraints**? (time/budget/technology)
- Do any requirements **conflict with or duplicate** other requirements?
- Are there any **legal issues** relating to each requirement?

## 7 Bibliography

- Arias, E., Eden, H., Fischer, G., Gorman, A., and Scharff, E. (2000) Transcending the individual human mind—creating shared understanding through collaborative design. *ACM Trans. Computer-Human Interaction*, Vol. 7(1), pp.84 - 113
- Briggs, RO., De Vreede, G., Nunamaker, J. (2001) ThinkLets: Achieving Predictable, Repeatable Patterns of Group Interaction with Group Support Systems (GSS), *HICSS '01: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)*, Vol. 1, IEEE Computer Society, Washington, DC, USA
- Briggs, RO., De Vreede, G., Nunamaker, J. (2003) Collaboration Engineering with ThinkLets to Pursue Sustained Success with Group Support Systems, *Journal of Management Information Systems*, Vol. 19(4), pp.31 - 64, M. E. Sharpe, Inc., Armonk, NY, USA
- Brooks, F. (1995) *The Mythical Man-Month*, anniversary ed., Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA
- Brooks, F. (2007) Collaboration and Telecollaboration in Design, Keynote Speech, *OOPSLA*, Montreal, Canada, 21 – 25 Oct 2007, [http://www.oopsla.org/podcasts/Keynote\\_FrederickBrooks.mp3](http://www.oopsla.org/podcasts/Keynote_FrederickBrooks.mp3), Last Accessed 01<sup>st</sup> Nov 2007
- Cadiz, J. J., Gupta, A., and Grudin, J. (2000) Using Web annotations for asynchronous collaboration around documents. *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work (Philadelphia, Pennsylvania, United States)*. CSCW '00. ACM, New York, NY, pp.309 - 318
- Chandler H. (2001) The complexity of online groups: a case study of asynchronous collaboration, *ACM Journal Computing*, Vol. 25(1), pp.17 - 24, ACM, New York, NY, USA
- Coulin C., Sahraoui A., Zowghi D. (2005) Towards a Collaborative and Combinational Approach to Requirements Elicitation within a Systems Engineering Framework, *ICSENG '05: Proceedings of the 18th International Conference on Systems Engineering*, pp 456-461, IEEE Computer Society, Washington, DC, USA
- De Bono, E. (1990) *Lateral Thinking*, Penguin Books Ltd, 80 Strand, London WC2R 0RL, England
- De Marco T., T. Lister (1987) *Peopleware: productive projects and teams*, Dorset House Publishing Co., New York, NY, USA
- Decker, B., Ras, E., Rech, J., Klein, B., Hoecht, C. (2006) Using Wikis to Manage Use Cases: Experiences and Outlook. *Proceedings of the international Workshop on Learning Software Organizations and Requirements Engineering (LSO+RE 2006)*, March 27th - 28th, 2006, Hannover, Germany

- Decker B., Ras E., Rech J., Jaubert P., Rieth M. (2007) Wiki-Based Stakeholder Participation in Requirements Engineering, *IEEE Software* Vol. 24(2), pp.28 - 35, IEEE Computer Society Press, Los Alamitos, CA, USA
- Duggan E.W, Thachenkary C.S. (2003) Higher Quality Requirements: Supporting Joint Application Development with the Nominal Group Technique, *Information Technology and Management*, Vol. 4(4), pp.391 - 408, Kluwer Academic Publishers, Hingham, MA, USA
- Fischer, G. (1998) Beyond "Couch Potatoes": From Consumers to Designers. *Proceedings of the Third Asian Pacific Computer and Human interaction* (July 15 - 17, 1998). APCHI. IEEE Computer Society, Washington, DC
- Fruhling A., Steinhauser L., Hoff G., Dunbar C. (2007) Designing and Evaluating Collaborative Processes for Requirements Elicitation and Validation, *HICSS '07: Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, p. 15, IEEE Computer Society, Washington, DC, USA
- Fuchs-Kittowski F, Kohler A, (2005) Wiki communities in the context of work processes, *WikiSym '05: Proceedings of the 2005 international symposium on Wikis*, pp.33 - 39, San Diego, California, ACM, New York, NY, USA
- Furnas G.W *et al.* (2006) Why do tagging systems work?, *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*, pp.36 - 39, ACM, New York, NY, USA
- Giles, J. (2005) Internet encyclopaedias go head to head, *Nature, Issue 438*, pp.900 - 901
- Grunbacher P, Boehm B, (2001) EasyWinWin: a groupware-supported methodology for requirements negotiation, *ESEC/FSE-9: Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering*, pp.320 - 321, Vienna, Austria, ACM, New York, NY, USA
- Grunbacher, P. and Briggs, R. (2001b) Surfacing Tacit Knowledge in Requirements Negotiation: Experiences Using Easy Win Win. *Proceedings of the 34th Annual Hawaii international Conference on System Sciences (HICSS-34)*, Vol. 1 (January 03 - 06, 2001). HICSS. IEEE Computer Society, Washington, DC, 1062.
- Gruding, J.(1995) Groupware and social dynamics: eight challenges for developers, *Human-computer interaction: toward the year 2000*, pp.762 - 774, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA
- Hasan, H., Pfaff C. (2006) The Wiki: an environment to revolutionise employees' interaction with corporate knowledge, *Proceedings of the 20th conference of the computer-human interaction special interest group (CHISIG)*, pp.377 - 380, Sydney, Australia
- Hertel, G., Nieder S., Herrmann S. (2003) Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel, *Research Policy*, Vol. 32(7), pp.1159 - 1177

- IEEE/EIA 12207.1-1997 (1997) *IEEE/EIA Standard: Industry implementation of intergenerational standard ISO/IEC 12207:1995 Standard for information technology – software lifecycle processes – life cycle data*: Section 6.22 Software Requirements Description
- IEEE Std 830-1998 (1998) *IEEE Recommended Practice for Software Requirements Specification*, IEEE 345 East Street, New York 10017-2394, USA
- Keltner, Dacher (2008) Pysch 160: Lecture 14, Social Influence And Conformity, <http://webcast.berkeley.edu/stream.php?type=download&webcastid=22170> Last Accessed 19<sup>th</sup> Mar 2008
- Kolfschoten G., Appelman J., Briggs R.O., de Vreede G. (2004) Recurring Patterns of Facilitation Interventions, In *GSS Sessions, HICSS*, Vol. 1, IEEE Computer Society, Los Alamitos, CA, USA
- Liu, H., Pattie Maes (2005) InterestMap: Harvesting Social Network Profiles for Recommendations, [http://ambient.media.mit.edu/assets/\\_pubs/BP2005-hugo-interestmap.pdf](http://ambient.media.mit.edu/assets/_pubs/BP2005-hugo-interestmap.pdf) Last Accessed 01st Nov 2007
- Mandiwalla, M., and Olfman, L. (1994) *What Do Groups Need? A Proposed Set of Generic Groupware Requirements*, ACM Transaction on Human-Computer Interaction Vol. 1(3), pp.245 - 268.
- Marlow, C., M. Naaman, D. Boyd, M. Davis (2006) HT06 Tagging Paper Taxonomy Flickr Academic Article ToRead, *Proceedings of the seventeenth conference on Hypertext and hypermedia*, Odense, Denmark
- Martinez A, Arias J, Vilas A. (2005) Merging Requirements Views with Incompleteness and Inconsistency, *ASWEC '05: Proceedings of the 2005 Australian conference on Software Engineering*, pp.58 - 67, IEEE Computer Society, Washington, DC, USA
- McConnell, S. (1998) *Software Project Survival Guide*, Microsoft Press, Redmond, Washington, USA
- Norman, D. (1991) Collaborative computing: collaboration first, computing second, *Communications of ACM*, Vol. 34(12), pp.88 - 90, ACM, New York, NY, USA
- Oxford English Dictionary (2008) <http://0-dictionary.oed.com.ditlib.dit.ie/> Last Accessed 01<sup>st</sup> Jul 2008
- Parnas, D. (2007) Precise Software Documentation: Making Object Orientation Work Better, Keynote speech, *OOPSLA*, Montreal, Canada, 21 – 25 Oct 2007, [http://www.oopsla.org/podcasts/Keynote\\_DavidLorgeParnas.mp3](http://www.oopsla.org/podcasts/Keynote_DavidLorgeParnas.mp3) Last Accessed 01<sup>st</sup> Nov 2007
- Paulus, P., Larey TS., Putman V, Leggett K, Roland EJ. (1996) Social Influence Processes in Computer Brainstorming, *Basic and Applied Social Psychology*, Vol. 18(1), pp.3 - 14
- Peters, T. and R. Waterman (2003) *In Search Of Excellence*, Harper Collins, 10 East 53<sup>rd</sup> St, New York, NY10022

- Potts C, Catledge L (1996) Collaborative conceptual design: a large software project case study, *Computer Supported Cooperative Work*, Vol. 5(4), pp.415 - 445, Kluwer Academic Publishers, Norwell, MA, USA
- Raymond, E.S. (1998) *The Cathedral and the Bazaar*, O'Reilly & Associates, 101 Morris Street, Sebastopol, CA, USA
- Riehle D. (2006) How and why Wikipedia works: an interview with Angela Beesley, Elisabeth Bauer, and Kizu Naoko, *WikiSym '06: Proceedings of the 2006 international symposium on Wikis*, pp.3 - 8, Odense, Denmark, ACM, New York, NY, USA
- Royal Academy of Engineering (2002) The Challenges of Complex IT Projects, <http://www.bcs.org/server.php?show=conWebDoc.1167> Last Accessed 01<sup>st</sup> Nov 2007
- Shirky, C. (2004a) Situated Software, [http://www.shirky.com/writings/situated\\_software.html](http://www.shirky.com/writings/situated_software.html), published 30<sup>th</sup> Mar 2004 Last Accessed 01<sup>st</sup> Nov 2007
- Shirky C. (2004b) A Group Is Its Own Worst Enemy, [http://shirky.com/writings/group\\_enemy.html](http://shirky.com/writings/group_enemy.html), published 01<sup>st</sup> Jul 2004 Last Accessed 01 Nov 2007
- Shirky, C. (2004c) Group as User: Flaming and The Design of Social Software, [http://www.shirky.com/writings/group\\_user.html](http://www.shirky.com/writings/group_user.html), published 05<sup>th</sup> Nov 2004 Last Accessed 01 Nov 2007
- Siddiqi J. et al (2007) A System for Semantically Enhanced, Multifaceted, Collaborative Access: Requirements and Architecture, *ITNG '07: Proceedings of the International Conference on Information Technology*, pp.1051 - 1056, IEEE Computer Society, Washington, DC, USA
- Soderquist C. (2003) Facilitative modelling: using small models to generate big insights. *The Systems Thinker*, Vol. 8(2), Pegasus Communications
- Sterman J. (2006) Learning from evidence in a complex world, *American Journal of Public Health*, Vol. 96(3), pp.505 – 514
- SWEBOK (2004) Abran A., P. Bourque, R. Dupuis, and J. W. Moore, Eds. Chapter 2 Software Requirements In *A guide to the software engineering body of knowledge*, <http://www.swebok.org> Last Accessed 01<sup>st</sup> Jun 2008
- Tamaro SG., Mosier JN., Goodwin NC, Spitz G, (1997) Collaborative Writing Is Hard to Support: A Field Study of Collaborative Writing, *Computer Supported Cooperative Work*, Vol. 6(1), pp.19 - 51, Kluwer Academic Publishers, Norwell, MA, USA
- Tervonin, I Dr. (2008) Checklist for Architecture Reviews, <http://www.tol.oulu.fi/~tervo/WChecklist.pdf> Last Accessed 30<sup>th</sup> Apr 2008
- Van Genuchten, M., Cornelissen, W., and Van Dijk, C. (1997) Supporting inspections with an electronic meeting system, *Journal of Management Information Systems*, Vol. 14(3) pp. 165 - 178.

Wagner C, Majchrzak A. (2007) Enabling Customer-Centricity Using Wikis and the Wiki Way, *Journal of Management Information Systems*, Vol. 23(3), pp.17 - 43, M. E. Sharpe, Inc., Armonk, NY, USA

Wikipedia (2007a) Wikipedia:Why Wikipedia is not so great,  
[http://en.wikipedia.org/wiki/Wikipedia:Why\\_Wikipedia\\_is\\_not\\_so\\_great](http://en.wikipedia.org/wiki/Wikipedia:Why_Wikipedia_is_not_so_great) Last Accessed 01<sup>st</sup> Nov 2007

Wikipedia (2007b) Wikipedia:Replies to common objections,  
[http://en.wikipedia.org/wiki/Wikipedia:Our\\_Replies\\_to\\_Our\\_Critics](http://en.wikipedia.org/wiki/Wikipedia:Our_Replies_to_Our_Critics) Last Accessed 01<sup>st</sup> Nov 2007