Proceedings of Student/Faculty Research Day, CSIS, Pace University, May 4th, 2007

Story-Wall: Lightweight Requirements Management for Agile Software Development

Lorena Delgadillo
Senior Year Honors Student, BS in Computer Science
Seidenberg School of CSIS, Pace University, New York
lorena.m.delgadillo@gmail.com

Abstract

The majority of commercial requirements management tools tend to be costly, document-driven and used by large organizations undertaking traditional forms of software development. While they are not immediately in the spirit of the agile philosophy, which advocates live dialogue over documentation and encourages small teams of developers to do the simplest thing possible to satisfy a requirement, there are some fundamental practices supported by these tools that play a role in more agile forms of software development. This paper examines the core requirements management needs that are common to software development of all flavors and describes a tool concept designed to bring lightweight requirements management to the agile (predominantly XP) context. This work is based on experiences in using agile development practices within ibm.com, and on the transition from manually handling paper-based story cards to the use of first generation story management tools. The paper discusses early feedback on the concept from practitioners.

1. Introduction

Requirements are needed in order to develop a system. Requirements are defined as the needs of the stakeholders for a system. Requirements engineering is a part of systems engineering whose goal is to better understand what a system should do and who for [2]. Requirements engineering is applied throughout the lifecycle of a system's development. From the beginning of a project, requirements must be established, detailing the functionality and constraints of a system. Requirements engineering encompasses the creation and development of requirements, as well as the management of requirements over time.

In software engineering there exist various techniques or approaches to developing a software system. There are traditional approaches which include: Waterfall, Spiral, Iterative or Incremental processes, and other types of processes created by and tailored to specific organizations. On the other hand, there exist alternative lightweight approaches referred to as 'agile'. The key differentiating factors are their approach to communication, team structure, the build and testing approach, and the ability to respond to change [5]. There are many forms of agile development process, such as Scrum, DSDM, Crystal and others. The most extreme form of agile development is eXtreme Programming, also known as XP, which takes a set of combined agile practices to the extreme.

Whether a software system is being built with a traditional approach or an agile approach, fundamental requirements engineering activities are basically the same. Stakeholders need to be identified, candidate requirements need to be determined, analysis has to be made on such requirements, and these requirements need to be checked or validated with customers. Requirements management is all about providing some mechanism to deal with inevitable requirements changes. The major difference is possibly the explicit emphasis and support given to each activity in requirements engineering and, in some cases, how much of one activity is undertaken prior to another can be started.

2. Research Method

A combination of different approaches aided in the development of this work and in the prototyping of a lightweight requirements management tool. Figure 1 shows the research methods.

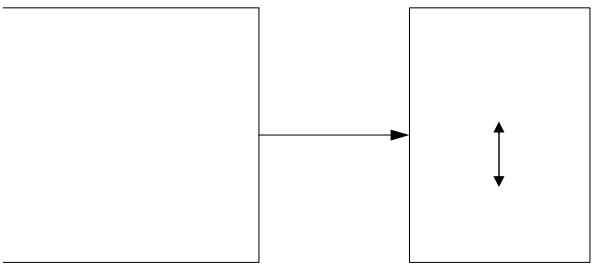

Figure 1: Research methods used.

2.1 Literature

Literature on the topic of requirements management, agile methodologies, and requirements engineering were studied and reviewed to gather knowledge about the different topics and issues addressed in this paper. This helped in understanding the essential requirements management requirements.

2.2 Questionnaires

Questionnaires were created in order to gather information from practitioners to corroborate the above and find out more. The questionnaire for practitioners was intended to gather their use of agile development practices in their work force, as well as any requirements management tools used, to get an understanding of the state of the practice. Questionnaires were also given to other IBM practitioners that work with the Rational Unified Process and Rational product suite.

2.3 Practitioner Feedback

Feedback of the early concept and prototype of the tool was obtained from practitioners using agile methodologies within *ibm.com*. The ideas were discussed with other agile project management leaders at a meeting held at Pace University in April 2007, including XP practitioners from IBM and Google.

2.4 Observation and Participation

Observation of and participation in agile professional presentations aided in developing knowledge of agile methodologies and their most basic requirements management needs.

2.5 Tool Critiques and Use

Critiques were gathered from Pace University graduate students regarding different tools that they had been exposed to in their Systems Requirements Engineering class (CS 775). These critiques, plus personal exposure to over twenty various tools via demo versions, helped in comparing the different features between current requirements management tools to understand the state of the art.

3. Requirements Management

Requirements management is the enabling process in which requirements, both technical and non-technical, are developed and maintained by all of the stakeholders throughout the project life cycle. It provide the ability to see which requirements have been implemented and where, and to assess the impact of potential changes, as well as to help see these changes through [4].

3.1 Requirements

The definition of requirements (i.e. what exactly are these things and where do they come from) has been a topic of much discussion. Most practitioners interpret in their own way what requirements are and mean. Most define requirements as some expression of the needs of the stakeholders for a new system.

3.2 Types of Requirements Management Tools

Current requirement management tools can be defined as either lightweight or heavyweight in their process support. Heavyweight tools tend to be costly and usually are database based or more complex systems. Complexity in these tools can be due to a multi-tiered structure, multiple components that need to be installed, as well as demands on the procedural configuration of the system (including enforcement of prescribed processes). Such complex systems can include server-client interaction, where more than one software tool has to be installed in order for the requirements management tool to function. There can be a high start-up cost and barrier to use. Some requirements management tools are dedicated only to requirements management. Others, such as DOORS, can be used to manage requirements throughout the project life cycle by integrating with other tools. Alternatively, there exist lightweight tools that are less expensive and easier to install and use (discussed later).

3.3 Stakeholders

Developers, requirement engineers, quality assurance (QA) testers and, to some extent, the business clients/customers that are linked to the project at hand are all stakeholders in the requirements management process. Each has their own goal and task in the requirements management process. The task of the developer is to build a functional product that meets expressed and expected requirements, so it fit for purpose. A requirements engineer focuses on writing comprehensible, representative and agreed requirements that can be developed by the application

developers. QA testers need to validate processes are followed and conduct proper testing of the requirements for a project. The goal of the business clients/customers is to have a well developed and quality product that meets their requirements, without exceeding allocated time and cost. All of these parties need to look at requirements and be able to prioritize and approve them, as well as make changes to them as all stakeholders learn more about the problems and opportunities at hand.

3.3.1 Business Client/Customers

The role of the business client is to supply to the technical team a project which can be to improve a current product or create a new product to take advantage of a business opportunity. The business client seeks to have the project released by a given date. Most business clients seek minimal cost and high productivity and quality. A requirements management tool must provide to the business client the ability to organize and prioritize requirements to their needs and deadlines.

3.3.2 Developers

The role of the developers is to transform the requirements into a design or a prototype. Their main goal is to develop a successful product with minimal amount of errors. This can be achieved with the help of having a robust requirements tool. By having requirements traced to design and code, developers can determine what requirements have been completed and which are left to be prototyped. Tracing to test cases for acceptance testing is also important.

3.3.3 Requirement Engineers

The role of requirement engineers is to write requirements that can be agreed by both the technical team and the business clients. Their task is to be able to track the status of a requirement and to track the changes to the requirements. The use of a requirements management tool for a requirement engineer is to facilitate the way in which requirement engineers investigate and develop requirements, as well as handle changes. A requirements management tool must have this feature in order for the requirement engineers to reach commitment to project tasks, provide high quality requirements and be able to track and trace them into high quality code.

3.3.4 QA Testers

QA is a key concept in software development. Without having any QA testers, how can developers (also their project managers and paying clients) know that the system they developed is of acceptable and anticipated quality? QA testers ensure that the software that is being developed is being done according to agreed processes and standards. Such members of the quality assurance team will also be using requirements management tools to verify that the requirement was implemented as specified and satisfies test cases.

3.4 Issues

Requirements management is that part of requirements engineering that deals with the problems in the traceability of project requirements, especially when checking requirements satisfaction and handling changes. The traceability problem is compounded by the types of requirements management tools that exist. Many are expensive and are not user friendly. Another aspect of the traceability problem is the human aspect. Customers and developers do not agree on requirements and, with many changes in the requirements, up to date requirements are not stored, captured, or managed by the developers. At the end of the day, the potential of having a certain portion of the project not being traced back to its original requirement has a high risk. This poses a potential risk to both the cost and efficiency of the product since it brings delays to the project plan.

The definition of requirements traceability has also been the cause of the traceability problem. Many practitioners and experts have their own understanding of the various project and software development tasks that requirements traceability should help simplify and support [2]. Most requirements management tools do not, however, help the user to formulate requirements. Rather, they leave a free form text area where the user can input whatever they please. The user may not know that instead of collecting and managing requirements, they are managing nonsense. Therefore, overly bureaucratic requirements management processes and tools can actually sometimes end up managing out of date garbage! The tools always rely on the people to do a good job, and some tools make this easier than others to encourage and realize.

4. Requirements for Requirements Management Tools

4.1 Essential Requirements

A requirements management tool must have, at a minimum, the following features in order to manage requirements:

- Requirements Storage
- Requirements Prioritization
- Change Control
- Requirements Progress
- Requirements Traceability

These requirements for requirements management do not have to only be implemented through a software product; they can also be implemented through some form of manual documentation, filing system or some form of human process. The stakeholders wish to have a system, whether automatic or not, that can be able to allow them to keep track of the requirements and all changes to them.

4.1.1 Requirements Storage

A requirements management tool should be able to store requirements. Requirements can be stored in the simplest form via a filing system or spreadsheet or, more usually, as a database schema. Requirements need to be stored in order for them to be managed and tracked for any changes. The requirements management tool should be able to store requirements with as much detail about them (i.e. metadata) as necessary. The more detail, the easier it is for a developer to understand his or her tasks and the impact of the change to be assessed.

4.1.2 Requirements Prioritization

Not all requirements are considered equal. Trade-offs will always need to be negotiated as to what can and what should be done on a project. A requirements management tool must have a requirements prioritization feature. Business managers as well as developers must be able to view and organize requirements according to their need and release dates. A requirements management tool should be able to allow the user to change or add attributes to a requirement, such as cost, effort, risk, priority, etc. (as above). Such attributes help managers to prioritize requirements.

4.1.3 Change Control

Change control is an essential part of requirements management. Every requirement can change and it can change more than once. Changes in requirements should be tracked in a requirements management tool in order to know the cause for the change and the impacts due to the requirement change, on quality, cost, schedule, etc. An ideal level of granularity should be on an atomic level. The more detail maintained the better it is to understand what has changed and to roll-back any problematic changes. Consequential changes to requirements should be authorized and agreed upon by project managers and business managers.

4.1.4 Requirements Progress

Requirements are subject to changes and keeping track of the status or state of a requirement is vital to the requirements management process. Requirements can be preliminary, to be negotiated and discussed, or can be in a development stage, tested, implemented, approved, etc. If a requirement cannot be linked to any of these stages then it is possible that the requirement is not needed.

4.1.5 Requirements Traceability

Requirements traceability is the relationship between a high-level requirement and a low-level requirement, and all other project artifacts derived from and contributing to them. Traceability analysis has three different types: impact, derivation, and coverage [3].

A requirements management tool must be able to link requirements between design and code and back to requirements. This is referred as forward-backward traceability. Further, vertical traceability is the linkage between a requirement, its design and its code. Horizontal traceability is the linkage between versions of requirements. These distinctions are illustrated in figure 2. Traceability between requirements and the source of requirements is known as pre-requirements traceability. Traceability between requirements and the target feature is known as post-requirements traceability [2]. The existence of requirements traceability in a project helps manage which requirements have changed and cross-impact the changes throughout the project.

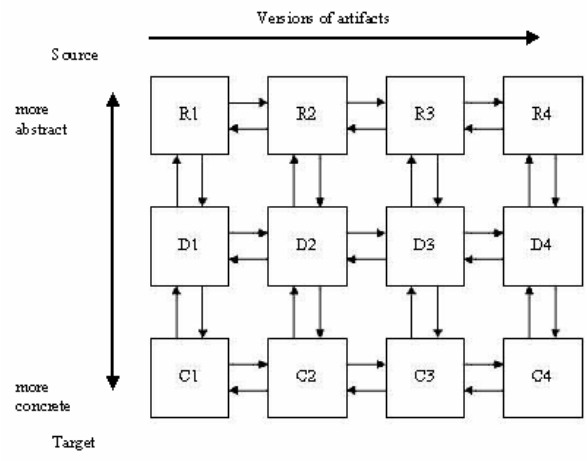

Figure 2: Traceability diagram showing forwards and backwards, horizontal and vertical traceability.

5. Current Requirements Management Tools

There is a vast diversity of requirements management tools currently in use by many companies. Such tools can support the full software development life cycle and all its project artifacts, while others only manage requirements. A study was conducted on a sample of approximately twenty requirements management tools on the market.

In the study, a comparison was conducted between requirements management tools used both for traditional and agile approaches. Each tool was compared to the essential requirements defined in section 4.1 of this document. Requirements management tools must be able to store requirements, prioritize requirements, keep track of changes, progress to requirements, and be able to trace requirements.

Each requirements management tool surveyed had a means of storing requirements. Tools such as TopTeam Analyst and TRUEreq use servers in order to store the requirements and other data. Other tools are standalone tools which can be installed on a shared drive for multiple users to use.

Not all requirements management tools prioritize requirements. RMTrak for example, does not provide features to prioritize requirements, but there are other tools that help users prioritize requirements. RequisitePro helps users customize the priority level of each requirement. Not all tools give the user the ability to customize the software development process they can use to undertake their project.

Changes to requirements need to be tracked for each requirement. Most tools on the market have some form of change control. It can be as simple as generating a change report that details the changes at a summary level or as complex as having notifications emailed to managers or ensuring change requests be approved by managers.

Most of the tools surveyed showed the progress of requirements. Some showed which phase the requirement was in, such as in analysis or in development. Others such as TRUEreq can show if a requirement is overdue or completed. RaQuest shows if a requirement was proposed or actually approved.

Traceability is an essential part of requirements management. Tracing the links between requirements determines the relationship of requirements in a project as well as the relationship between design and code of the requirement. Since not all requirements management tools support the full development life cycle, most only provide traceability between requirements and not between design and code. Tools such as OptimalTrace by Compuware and RequisitePro by IBM have the feature to develop requirements through the full development life cycle. Others such as TRUEreq do not address the traceability issue, so do not have a traceability matrix or report.

A good requirements management tool will have all the essential requirements. They should all be able to store requirements, prioritize requirements, keep track of changes and the progress of requirements, and trace links between requirements. Many strongly support these features, but generally require that rigorous processes are followed, and some do not support them at all.

Unfortunately, not much of the requirements creation and development problem has been addressed by many of the requirements management tools. In the study only one tool helped users to actually write ‘good’ requirements. This tool, Leap SE, has requirements template depending on the type of requirement. Users can choose to create functional, structural, or technical requirements from a list of various templates. Although this tool helps the user to compose requirements, it does not have any other features, so is limited.

Current requirements management tools can improve only if they can first help users create better requirements and support this creative and exploratory interchange process that surrounds their development. Well written requirements ease the process of managing requirements; stakeholders will at least know that the requirements that they are managing are not nonsense.

6. Agile Requirements Management

In contrast to the traditional requirements management techniques, agile software development practices do not focus on detailed requirements documentation. In fact, XP [1] uses physical paper index cards, known as 'story cards', to record requirements. Story cards are used as a way to prompt discussion about requirements between the developers and the clients. Communication between the clients and development team is one of the main practices of agile.

Most XP teams do not use a tool to manage their story cards. This poses an advantage, but also a potential problem since story cards can be lost or misplaced in the wrong pile which can affect the project at hand. It is even a higher risk of not having a tool to manage stories cards or requirements when agile becomes distributed in implementation.

Throughout the software development life cycle, stories are created, estimated, prioritized, and placed in iterations for development. Iterations are short cycles in which features of the software are built, tested and released. Clients choose those stories they want developed during each iteration, as well as decide which stories are no longer needed. Stories are usually placed on a whiteboard or a wall that has different stages in which a story can be place in. Usually the stages are "to do", "in progress", "completed", as well as others. The terms are usually defined by the team.

During the creation of stories (aka very loose requirements), the developer and the client have various forms of communication which more than likely are not recorded or stored. This poses a problem when developers try to remember what the story was about and the person who they communicated with is no longer present. Managing stories becomes more problematic when dealing with a globally distributed project, where the different team members are placed in various places around the globe, because development teams will not be able to view the physical wall of stories. Tracking changes of stories becomes more difficult because each team might have different versions of the story. Agile methodologies don't hold requirements traceability as essential to the development and change process as do traditional methodologies. This is somewhat due to the perception that traceability is, by definition, always costly and heavyweight. However, agile projects can be subject to requirements change issues too.

Vendors have created various tools as a solution to the requirements management problems sometimes experienced in agile software development projects. The study found a handful of agile requirements management tools that, in contrast to agile practices, were either complex, heavyweight or had no customer usability. For example, Rally, a web-based tool, was found too complex in its design to use effortlessly, which defeats the idea of going agile. The interface of this tool had too many seemingly unnecessary features and little explanation of its expected process of use. The process flow was not easy to comprehend and not so in line with usual practice. Other tools such as Project Cards, an Eclipse plug-in, do not include the idea of pair programming and developer allocation to stories into its framework. Although it allows you to customize your project, it is time consuming to deal with it. Although it is a very useful tool for developers, it does not give the easy to use interface that would be preferred by clients to create and prioritize their own stories with ease.

Companies such as IBM and Google use requirements management tools for their agile development teams. An agile development team for *ibm.com* uses a tool called Extreme Planner. Although it is a lightweight tool, it does not have any requirements traceability. A group at Google that focuses on agile development uses a brand new and custom made tool created by a third party vendor to suit their needs for agile story management (details forthcoming).

Many of the current agile requirements management tools overload the tool with features that are not needed very often and, in turn, create a more complex tool than may be necessary to support what is meant to be an agile and lightweight process. Hence a need for an open source and lightweight requirements management tool for agile software development.

7. Story-Wall Concept

In order to address the need for a lightweight requirements management for agile software development, it was found in this research that a new tool should be created. This new tool will have the essential requirements mentioned in section 4.1 as well as provide a solution to manage requirements in a globally distributed project.

7.1 Prototype

In order to provide a solution to the issues currently seen in requirements management and in requirements management tools, a prototype of a lightweight requirements management tool concept was developed. The name of this prototype is Story-Wall.

7.2 Users

Story-Wall will benefit and provide the ability for developers, project managers, and clients to all view the stories for a project. Story-Wall is targeted not only for developers, project managers, and clients, but also for whoever would like to be aware of the stories in a project, such as QA testers. According to roles, these stakeholders can create, elaborate, estimate, prioritize and allocate stories to iterations.

7.3 Features

Story-Wall focuses on providing a high level view of the project. Its main features include:

1. Virtual Wall
2. Story Card Simulation
3. History of Changes
4. Lightweight Requirements Traceability

7.3.1 Virtual Wall

Practitioners of XP rely on story cards being placed on a wall or a whiteboard which allows for them to see what needs to be or has been done on a project, and to easily move a story between stages. Having this exact same idea in a tool will allow for easy transition between a physical wall and a virtual wall which they will see in the tool. This concept was adapted in the Story-Wall prototype, as illustrated in figure 3.

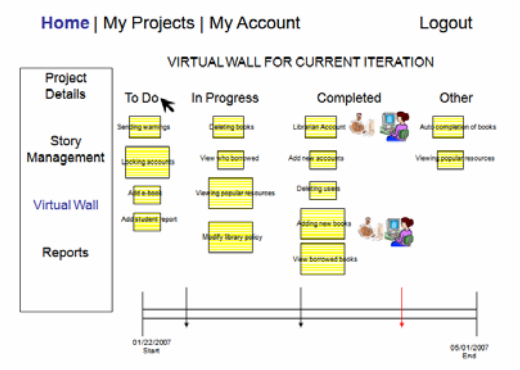

Figure 3: Virtual wall prototype.

Using drag and drop functionality, developers can drag story cards from stage to stage until completion and customer acceptance. The virtual wall allows for users of the tool to see up to date story card information and the progress of the project.

Within the virtual wall users can see where they are at in the timeline and be able to select different iterations and view their own virtual wall. Upon viewing the virtual wall for a project, the current iteration will be displayed in the context of the project timeline.

Users can view the details of a story card by just selecting it. Users can view the complete list of stories that are assigned to a certain phase within the iteration.

7.3.2 Story Card Simulation

In physical reality, story cards have both a front and a back side of the card. Usually, on the front of the card, the details of the story are written, as per figure 4. The back of the story card is often used to show the different tasks the story involves and test cases for the story. This same concept was adapted for Story-Wall. When a user selects a story card, they can view the front and back sides of the story card, and contribute to either as appropriate and needed.

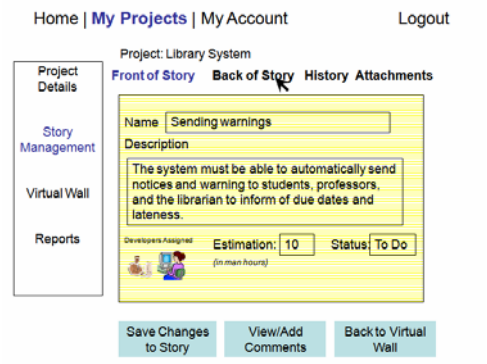

Figure 4: Story card simulation.

7.3.3 Prioritizing and Estimating Stories

A direct manipulation drag and drop interface was also created to enable customers to sort their story cards into priority lists according to perceived business value or importance of the story. Likewise, developers directly stretch the virtual story cards to represent their sizing estimates, this representing the anticipated development effort to build a story. Stories are selected per iteration in a further direct manipulation manner not unlike selecting and moving varying sized pieces to fill a limited size container which reflects the development team velocity for the iteration. The priority and sizing information is visually represented in the virtual wall that manages the story cards and displays the big picture context for the agile project.

7.3.4 History of Changes

Yet to be completed stories are always changing and these changes are sometimes never tracked. The Story-Wall tool keeps track of the changes made to a story by adding a simple history to the story. A user can choose to view the history of the story and see from who created the story to the latest change contributor. Annotations and comment facilities are provided. Font types can differentiate contributors, as per a physically annotated card.

7.3.5 Lightweight Requirements Traceability

Requirements traceability, as mentioned previously, is often ignored in an agile context as perceived as burdensome and redundant. However, an increasing number of development situations and contexts are actually seeing the need for some form of traceability to support project longevity. Lightweight requirements traceability can be achieved via the story wall concept as a by-product of everyday use. Additionally, stories can be traced back to the discussions between clients and the development team to recover rationale by recording a meeting, phone conversation, or saving a chat and uploading it to the tool. The uploaded file will accessible for users to either listen or read once they view the story details. Tracking multimedia requirements information of this nature to support understanding is a current research topic at organizations such as Siemens.

7.4 Ongoing and Future Work

Currently the tool is in prototype phase. The implementation of this tool will use the following technologies: AJAX, Web 2.0 and Ruby on Rails. The platform on which this tool will be created will be Wiki-based. Web 2.0, AJAX, and Ruby on Rails were chosen as the ideal technologies to implement drag and drop functionality for the tool. A wiki-based tool will allow for more collaboration between users as well as discussion forums for questions and comments about stories and the overall project. Wikis are very much used in Agile since they allows for an easy and intuitive way for members in teams to communicate. The tool will continue to be validated with targeted practitioners to help refine the concept prior to full implementation.

8. Conclusions

Requirements management tools should be able to ease the management of changing requirements. In order for requirements management tools to work efficiently they must be able to store requirements, prioritize requirements, track changes to requirements, track the progress of requirements, and provide a level of requirements traceability.

Current requirements management tools are known to be heavyweight and hence a turn-off to the growing community of agile software developers. They generally force processes and procedures that are viewed as overly burdensome and contrary to the agile philosophy of 'doing the simplest thing possible' at all stages. Since current agile story/requirements management tools do not have all of the most fundamental of requirements for a requirements management tool, this work has involved exploring a lightweight requirements management tool concept that can go someway towards addressing this gap.

9. References

[1] Beck, K. *Extreme Programming Explained, Embrace Change.* Addison-Wesley. 2001.

[2] Gotel, O.C.Z. and Finkelstein, A.C.W. An Analysis of the Requirements Traceability Problem. *Proc. 1st IEEE International Conference on Requirements Engineering*, IEEE Computer Society Press, Colorado Springs, CO (April 1994), 94-101.

[3] Hull, M. E. C. Jackson K., and Dick J.J., *Requirement Engineering*. Springer-Verlag, London, UK, 2002.

[4] Ludwig Consulting Services, LLC, *http://www.jiludwig.com. July 2006.*

[5] Pressman, R.S., *Software Engineering*, Sixth Edition, McGraw Hill, New York, NY, 2005.

10. Acknowledgments

The author would like to thank Dr. Olly Gotel for her mentoring and support as well as the CS 775 Systems Requirements Engineering class for completing the requirements management tools questionnaires.

The author would also like to thank the IBM practitioners and researchers who provided input and feedback on this work, especially David Leip and Joe Krebs.

This research work is supported by a Pace University Presidential Grant for 2006-2007: a Eugene M. Lang Student Research Fellowship entitled **"**Lightweight Requirements Management for Agile Software Development". The full report for this ongoing work is available from the author on request.