2014

# A Hybrid Approach Using RUP and Scrum as a Software Development Strategy

Dalila Castilla
*University of North Florida*, n00820115@ospreys.unf.edu

# A HYBRID APPROACH USING RUP AND SCRUM AS A SOFTWARE DEVELOPMENT STRATEGY

by

Dalila Castilla

A thesis submitted to the
School of Computing
In partial fulfillment of the requirements for the degree of

Master of Science in Computer and Information Sciences

UNIVERSITY OF NORTH FLORIDA
SCHOOL OF COMPUTING

August, 2014

The thesis "A Hybrid Approach Using RUP and Scrum as a Software Development Strategy" submitted by Dalila Castilla in partial fulfillment of the requirements for the degree of Master of Science in Computer and Information Sciences has been

Approved by the thesis committee: Date

_____ _____
Dr. Robert F. Roggio
Thesis Advisor and Committee Chairperson

_____ _____
Dr. Ching-Hua Chuan

_____ _____
Dr. Sherif A. Elfayoumy

Accepted for the School of Computing:

_____ _____
Dr. Asai Asaithambi
Director

Accepted for the College of Computing, Engineering, and Construction:

_____ _____
Dr. Mark A. Tumeo
Dean

Accepted for the University:

_____ _____
Dr. John Kantner
Dean of the Graduate School

ACKNOWLEDGMENT

I would like to express my special appreciation to my thesis advisor, Dr. Robert F. Roggio for his guidance, support, and advice throughout the realization of this thesis. His expertise in the area of software engineering methodologies was an important factor in generating an innovative research idea that eventually became a motivation for carrying out this study. Also, I would like to thank my committee members, Dr. Ching-Hua Chuan and Dr. Sherif A. Elfayoumy as well as Dr. Asai Asaithambi and Dr. Swapnoneel Roy for their valuable comments and suggestions in my thesis defense. Also, I want to thank Dr. Asai Asaithambi and Dr. Roger Eggen for their academic support and advice during my graduate studies. I also would like to thank Dr. Behrooz Seyed-Abbassi, whose advice and words encouraged me to keep advancing in my thesis. In addition, I want to thank Mr. Jim Littleton for his comments and suggestions to make this document fit the required format and Ms. Shawn Broderick for helping me with the administrative procedures in the thesis process.

Finally, I especially want to thank my husband, and my son and daughter for their patience, support, and understanding during the past three years I dedicated to my graduate studies to achieve this milestone in my life and career.

CONTENTS

FIGURES

TABLES

# ABSTRACT

According to some researchers, a hybrid approach can help optimize the software development lifecycle by combining two or more methodologies. RUP and Scrum are two methodologies that successfully complement each other to improve the software development process. However, the literature has shown only few case studies on exactly how organizations are successfully applying this hybrid methodology and the benefits and issues found during the process. To help fill this literature gap, the main purpose of this thesis is to describe the development of the Lobbyist Registration and Tracking System for the City of Jacksonville case study where a hybrid approach, that integrates RUP and Scrum, was implemented to develop a major application to provide additional empirical evidence and enrich the knowledge in this under-investigated field.

The objective of this research was fulfilled since the case study was described in detail with the specific processes implemented using RUP and Scrum within the context of the IBM Rational Collaborative Lifecycle Management Solution. The results may help researchers and practitioners who are looking for evidence about conducting a hybrid approach. However, more case studies that successfully combine RUP and Scrum need to be developed in order to have enough empirical evidence.

# Chapter 1

# INTRODUCTION

To stay competitive in the current market and gain market share, software development companies need to deliver high-quality products that meet or exceed customers' expectations within budget and schedule. For this purpose, companies are constantly looking for ways to improve their software development process. Currently, there are many software methodologies from which to choose. However, all have strengths and weaknesses because they focus on different aspects of the software development life cycle. As a consequence some academics and practitioners have proposed different hybrid approaches that combine two or more software methodologies with the purpose of leveraging respective strengths thereby improving the entire development process.

Some frameworks that combine different methodologies have been proposed and some of them have been implemented in organizations to prove their effectiveness.

For instance, Batra *et al.* mention that by combining Scrum and the Project Management Body of Knowledge (PMBOK) concepts a cruise company was able to deliver a large distributed project whose previously development attempt had failed [Batra10]. Another company was able to improve its productivity by combining Scrum practices with its RUP process [Carvalho11]. Del Nuevo *et al.* proposed a framework that combines Scrum and RUP for distributed software development. This framework was successfully

implemented in an organization, and, as anticipated, some adjustments were needed to fit the organization's practices [del Nuevo11].

These researchers' results validate the statement "one size does not fit all," which means that every project may require a different approach to allow its goals to be achieved. From these research results the need to describe the implementation of different hybrid approaches to add knowledge in this effort and to reveal other aspects that influence the successful delivery of a product that meets the customer expectations are clearly warranted.

The aforementioned cases motivated research to provide a detailed description of the hybrid approach implemented at the University of North Florida for the graduate Software Engineering I and II courses during the academic year 2012-2013. This implementation was special because it was used to deliver a major real-world application (Lobbyist Registration and Tracking System) for the City of Jacksonville, Florida. Also, another important aspect was that the UNF team had direct interaction with the customer during the course of the project which is not common for most of the school projects, where the students are sometimes isolated from real meaningful customer interaction and information. In many of these cases, critical information is passed through the professor, who frequently may not possess the level of detail that users of an application have. The objective of the above hybrid approach is to integrate within a collaborative lifecycle tool, namely IBM's Collaborative Lifecycle Management Solution, two widely used

methodologies, the RUP and Scrum. This integration supports the software development activities from the inception to implementation.

## 1.1 Background

Until the mid-1990s traditional methodologies such as the waterfall and the spiral model among others dominated the software development landscape [Williams12]. They were characterized by up-front requirements and design which followed a plan and document-driven process [Cho09, Boehm04, Nerur05,Williams12]. Also, these methodologies allowed a disciplined and structured process that permitted complying with the many policies and rules placed inside an organization. However, the arrival of the agile methodologies in the late 1990s, introduced a different approach based on the agile manifesto and the principles behind it [Williams12]. These agile methodologies besides utilizing an iterative and incremental approach, promoted the continuous delivery of software, constant interaction with the customer, and elaborating only the documentation needed.

According to Nerur *et al.*, the introduction of agile methodologies divided the software community in two main groups—traditionalists and agilists. Each group claimed that its approach was better than the other. However, there were other researchers and practitioners who believed that both approaches had their own strengths and could be used accordingly in an organization to fit a particular project [Nerur05].

In their book, Boehm and Turner suggested the need for balancing agility and discipline, especially for this evolving Internet economy that demands software that is more complex due to the size of the applications, the rate of changing requirements, and the quality and usability expected of the product [Boehm04].

Recently, some practitioners and academics have developed or proposed different process frameworks that combine two or more methodologies [Batra10, Carvalho11, Cho09, Bashir12, Nisa12, del Nuevo11, Nortier11]. The objective of these hybrid approaches was to create a process that maximized the strengths of the involved methodologies while at the same time reducing their weaknesses to improve the software development lifecycle and produce high-quality products [Cho09, Bashir12].

Two methodologies sometimes used in the above mentioned hybrid approaches are the Rational Unified Process (RUP) and Scrum, both of which embrace an iterative and incremental approach and have been widely used with long successful track records of their own [Ionel08, del Nuevo11]. RUP, which is commonly misunderstood as a traditional methodology because of its many roles and artifacts, can be adapted to a particular project or organization to provide discipline, structure, and guidance throughout the entire software development lifecycle [Collaris10, Krutchen03]. Scrum, on the other hand, is an agile management process that helps steer a project with an iterative and incremental approach during the software development lifecycle [Krebs05].

In the literature, there is a lack of empirical research that addresses different approaches on how to combine two or more methodologies and the issues found during this process. For example, Batra *et al.* claim that more case studies are needed to address the different issues when combining methodologies and to obtain a better understanding of how to combine their processes [Batra10]. In addition, Del Nuevo *et al.* state that the research about hybrid approaches only mention techniques and practices applied successfully in real cases without giving a more detail of the activities and tasks performed [del Nuevo11]. Further, in the literature there are very few case studies developed by practitioners or academics that combine RUP and Scrum [Carvalho11, Bashir12, Nisa12, del Nuevo11, Nortier11].

## 1.2 Problem Statement

According to the hybrid approach school of thought, software development practitioners can optimize their software development lifecycle by combining traditional and agile methodologies. Particularly, it is claimed that RUP and Scrum are compatible and successfully complement each other to improve the software development process [Collaris10]. However, the literature has shown that there is a scarcity of empirical research on exactly how organizations are applying a hybrid methodology and the benefits and issues found during the process. Further, there are only few case studies that show how a hybrid methodology implementing RUP and Scrum can be successfully combined.

Another question arises as to whether there is just one way or there are more ways of combining several methodologies. Recognizing that no one size fits all, there is an industry-wide need for developing more case studies to add knowledge in this area and to provide a database of hybrid strategies.

Given the above information, the main purpose of this thesis is to develop a case study where a hybrid approach, which integrates RUP and Scrum, is implemented to develop a major development application to provide additional empirical evidence and enrich the knowledge in this under-investigated field. Thus, this thesis will supplement existing case studies developed and published in the literature to provide additional insight on the different contexts where a hybrid approach can be successfully applied to address the needs coming from diverse environments and projects.

## 1.3 Overview of the Research Methodology

From the literature review of this study, multiple gaps in studying hybrid approaches were found; however, two gaps were identified that are relevant to the field of hybrid technologies and specifically the one combining RUP and Scrum. For the purpose of this research one area clearly in need of additional research was chosen, which is related to the process of combining both methodologies successfully. Very little empirical evidence was found in the literature.

In the research design of this study, the research purpose and objective, two inquiry questions, and the research methodology were stated. To address the inquiry questions, initially two guiding propositions were defined. However, this research was focused on the first guiding proposition that is related to the processes used to combine the best features of RUP and Scrum.

Then, to support this guiding proposition a case study combining RUP and Scrum to validate and/or add new information to previous cases studies was undertaken. To address this purpose, a research objective was established as: "develop a case study to describe the process of combining both RUP and Scrum methodologies." To fulfill this research objective, the Lobbyist Registration and Tracking System for the City of Jacksonville (COJ) was selected for this study. Once the research objective was established, a descriptive and single case study research methodology that uses a qualitative approach was selected as the research methodology.

The case study is described in detail with the specific processes implemented using RUP and Scrum within the context of the IBM Rational Collaborative Lifecycle Management solution (IBM-CLM). This description was carried out using data collection protocols and multiple data sources from a wide variety of stakeholders with varied interests. Using the combination of these multiple sources ensured the validity of the data collected. This increased the level of confidence of the findings obtained and reduced possible bias from a single source or protocol.

The data analysis followed the phases stated in the research design of this study. For this research, the first step in data analysis was to focus on “within-case analysis” as a standalone entity and to distinguish the unique patterns of this lobbyist case study. In these terms the lobbyist case study was described with the specific RUP and Scrum processes within the IBM-CLM environment and how they were conducted.

The follow-on “cross-case examination” indicated that when comparing the above within-case results with findings from previous published studies, results may vary for different projects and organizations. Finally, the results of this data analysis were summarized, conclusions were derived, and further research is suggested.

# Chapter 2

# LITERATURE REVIEW

## 2.1 The Need for Hybrid Approaches

According to Dyck and Majchrzak a possible case for project failures may be the improper choice of a software methodology. These authors state that every methodology has its own characteristics that make it suitable for a type of project or organization. However, Dyck and Majchrzak add that an organization using a software methodology may follow ideas from other methodologies or may even combine conventional methodologies with agile approaches. This is done with the purpose of tailoring a development strategy for the organization and the project [Dyck12].

Thus, for the past decade researchers and practitioners recognize the need for combining software development methodologies to improve the software development process. Also, they realize that every methodology has its own advantages and disadvantages and that by combining the strengths and mitigating the weaknesses of two or more approaches a better software development process can be achieved [Cho09, Batra10, Nisa12].

## 2.2 Hybrid Methodologies

This section describes several proposed hybrid methodologies and case studies that reveal how these methodologies were implemented inside an organization. Also, a critical review is added at the end of each case study.

### 2.2.1 A Proposal for a Hybrid Methodology Combining RUP and Scrum

Cho proposed a hybrid software development method that is suitable particularly for large-scale projects in the business software industry. This method integrates the Rational Unified Process (RUP) and Scrum to maximize their strengths while reducing the weaknesses of both methodologies. RUP is used to give structure to the hybrid method by following its four phases Inception, Elaboration, Construction, and Transition; however, only seven (Business Modeling, Analysis, Design, Implementation, Testing, Deployment, and Configuration) of the nine disciplines are carried out to make the process faster and more efficient— the Project Management and the Environment disciplines are not considered in this framework [Cho09].

On the other hand, Scrum's roles, artifacts, and events are integrated into the RUP phases. For example, Cho explained that each RUP phase can consist of one or several sprints depending on the size of the project. The daily Scrum meeting, the daily Scrum of Scrums, the sprint planning meeting, and the sprint review meeting can be carried out iteratively in each RUP phase, where the daily Scrum meeting will serve to monitor each

of the seven RUP disciplines. The roles for the Scrum master and the product backlog owner can be played as defined in the Scrum process. The product backlog can be created during the business modeling discipline and later, the tasks from the product backlog and the sprint backlog can be carried out during the sprints and monitored through the daily Scrum meeting [Cho09].

Critical Review: Cho does not mention any organizations implementing this approach, so challenges or issues on implementing this approach are not discussed.

### 2.2.2 A Hybrid Methodology for a Small Project Using RUP, Scrum, and XP

Nisa and Qureshi stated that some studies have indicated that a number of agile models have been carried out by combining them with conventional software development models with the goal of optimizing the strengths and weaknesses of both agile and conventional models. However, they added that there are many combinations that have not been carried out. Therefore, they claimed that integrating Scrum, XP and RUP models is a good combination to accommodate the features of both conventional and agile models. For this objective they proposed a hybrid model named SPRUP that combines the strengths of Scrum, XP, and RUP and eliminates their weaknesses to produce high quality software with a low defect rate [Nisa12].

This model is validated through a controlled case study on a small scale project during a five week period. During the case study, four releases called SPRUP versions were

produced where the first version lasted two weeks and the subsequent versions lasted only one week. The SPRUP model adopts the four phases of RUP which are executed in sequence during each sprint cycle. This model adds the anticipation, crafting, execution, and assessment activities to execute the work that is carried out in the planning, designing, coding, and testing activities of the XP model. These activities are performed according to the RUP phase. For example, the anticipation activity is performed during the inception phase; the crafting activity is carried out during the Elaboration phase; the execution activity is completed during the construction phase, and the assessment activity is executed during the transition phase. Also, the Scrum practices such as the product backlog, sprint backlog, sprint review meetings, daily Scrum meetings, and Scrum roles were incorporated into the SPRUP. The Scrum management practices and XP engineering practices were integrated into the SPRUP model to reduce rework, cost, and effort [Nisa12].

During the case study some data were collected from the four sprint versions. Among the most important attributes measured during the software development were work effort, customer involvement, customer satisfaction, team productivity, and number of interfaces built [Nisa12].

Nisa and Quereshi [Nisa12] concluded that further research can be applied to the SPRUP model on a large scale project and in an outsourcing development environment. Also, they mentioned that there is a need for more case studies with different combinations of RUP and agile methods [Nisa12].

Critical Review: This research is important as a first effort for this combination of RUP and agile methods. However, more case studies are needed where the SPRUP model is implemented in different environments that may not be controlled, and for projects with longer duration as well as for large-scale projects.

### 2.2.3 A Hybrid Approach for Up to Medium Scale Projects Using RUP, Scrum, and XP

Bashir and Qureshi proposed an integrated framework to combine the strengths of RUP, XP, and Scrum in order to achieve high quality software and enhance the team productivity. A case study for the Online CIIT Library Management System was developed by a team composed of six members to validate such proposed integrated framework. The team members were trained before the starting of the project to have a better understanding of the proposed hybrid model and to be acquainted with the management practices of Scrum, the engineering practices of XP, and the RUP process. This case study required four iterations to be completed [Bashir12].

In this framework, the four phases of RUP were reduced to three major phases and the nine major disciplines were reduced to six to streamline the process. The three phases are Domain Analysis and Design, Production and Validation, and Evolution and all of the six activities can be performed in each of the phases. Thus, Business Investigation and Design are the major activities in the Domain Analysis and Design phase. The Implementation and Testing activities play the main role during the Production and Validation phase, and the Deployment and Configuration activities are the main players

in the Evolution phase. The authors mentioned that the Scrum practices such as roles (Scrum Master, Developers and Product Owner), artifacts (Product Backlog, Sprint Backlog, and Burn down Chart), and ceremonies ( Daily Scrum Meeting, Sprint Planning Meeting, and Sprint Review Meeting) can be part of the proposed model without difficulty and the ceremonies can be organized iteratively in every phase. Also, the XP engineering practices can be performed during the logical activities of each phase without a problem [Bashir12].

Bashir and Qureshi concluded that this proposed framework combines the strengths of Scrum, RUP and XP while narrowing their weaknesses. They claimed that Scrum provides best managerial practices throughout the software development, and RUP delivers a structured and disciplined approach throughout the software development and reinforces the XP practices through its philosophy. Also, the authors stated that this framework emphasizes some very useful practices of RUP, which can be applied according to the project [Bashir12].

Critical Review: This is a case study developed for a very small project where the proposed framework was carried out for a specific objective. However, there is a need for more case studies to validate such a framework.  Also, the authors provided a very general description of the framework and they did not give details about how or what activities are executed during the development process.

### 2.2.4 A Hybrid Approach Using RUP and Scrum to Address Modern Challenges

Nortier *et al.* proposed a framework, which combines RUP and Scrum. The main objective of this hybrid approach was to deliver a development process that handled uncertain requirements and adapted to changing requirements late in the development cycle while delivering high quality products within budget and schedule. This framework was applied in an organization that is a subsidiary of a global organization. This organization was facing problems such as trouble meeting datelines and budget, uncertain requirements at the beginning of the project, organization not able to adapt and change to fast trends, difficulty in organizing and coordinating distributed development teams, needed to adhere its development strategy to a higher management process, not very experienced development team, and lack of time for testing activities [Nortier11].

Their framework is aligned to the four RUP phases and their milestones, which provide the skeleton for this framework. Scrum provides project management practices and tracking mechanisms through its ceremonies, roles, and artifacts. The RUP disciplines are applied in the different iterations inside a phase. The Daily Scrum meetings, the Sprint Planning meetings, and the Sprint Review meetings are used to monitor that the activities performed during the process adhere to the six main RUP disciplines. The product backlog is used as a high-level software requirement specification which takes information from use case to create the user stories, and also includes non-functional, architectural, and technical requirements. Also, it is used for iteration planning. The

sprint backlog is used as a detailed software requirement specification and the rest of the artifacts that are produced come from executing the RUP disciplines [Nortier11].

The software development framework has five phases that are aligned to the four phases of RUP [Nortier11]. These five phases are:

- The Feasibility Phase. This phase is aligned to the RUP's inception. During this phase the business case and the customer vision are developed. The business case includes high-level requirements and a project plan. Also, a risk analysis is performed during this time and the initial product backlog and the high-level architecture is produced.
- The High-Level Planning and Design Phase. In this phase the use case model and the business model are created. To follow the Scrum methodology the product backlog requirements are defined 60% of the total and will be further elaborated in the next iterations when they are implemented. Also, at this time the high-level architecture design is produced. During this phase a project plan is produced which contains the phases with their milestones and iterations.
- The Development Phase. This phase is based on iterations, which include the elaboration of the product backlog's requirements and the high-level architecture, development, testing and a Sprint Review. During this phase the validation and verification activities are concerned only with the software product and the system.
- The Customer Validation and Stabilization Phase. This phase concentrates on the validation and verification of the software product against customer's

requirements. All software and supporting documentation are ready to be deployed when they reach an acceptable level and customer's approval is attained.

- The Commissioning Phase. During this phase the software is deployed to the user community and users and maintenance people are trained.

According to the authors, this software development framework can be tailored to suit any project within an organization.

Critical Review: In this paper the authors provided an overview of the proposed methodology and how this was incorporated into the process of the organization. This is an approach that can be implemented in an organization that deals mostly with distributed teams and large projects. However, there is a need to develop case studies where this proposed methodology may be tailored for different projects in different types of organizations.

### 2.2.5 The Impact on Process Productivity Using a Hybrid Approach that Combines RUP and Scrum

According to Carvalho *et al.*, this research was carried out in a Brazilian CMMI Maturity Level 2 medium size company. This company has a reliable maturity process and its software development is based on a RUP-customized and CMMI-adherent process in a traditional fashion, with upfront cost and schedule formal agreement. Since 2008 this company began developing projects using a hybrid approach that incorporates Scrum practices through the RUP process [Carvalho11].

Their hybrid approach is organized in different sub-processes. The life cycle model starts with the Initial Concept sub-process in which the focus is to elicit software requirements and customer expectations. In this model the Architecture is elaborated early in the process. According to the authors, the Architecture, the Initial Concept, and the Requirements Management must be performed in strict accordance with traditional principles and should embrace principles of process capability, verification, and validation. On the other hand, the Design, Code, Test & QA, Transition, and Backlog Management sub-processes take advantages of the dynamics of the agile principles. Also, the product evolutions are divided into sprints to deliver product increments [Carvalho11].

The authors recommended that the Architecture should be performed early in the process to enable the analysis and detailed design tasks as opposed in the agile methodologies where this is done in increments. Further, they recommended that the specifications and validations which occur throughout the life cycle should focus on the early iterations of the project to prioritize the requirements that impact the architecture development. Also, they mentioned that Backlog management is the interface between the agile development tasks and the Architecture and Requirement sub-processes which are performed according to the traditional model principles. The Backlog Management is performed through the whole life cycle and gets input from the information generated by the Requirements and Architecture sub processes [Carvalho11].

In order to accomplish this research, they used a sample of a set of projects performed in accordance with their hybrid approach. The duration for these projects was 18 months during the period of July/2008 to December/2009. For the control group, they used the same data from another set of projects developed from January/2008 to June/2009—18 months durations as well—with an overlap of one year. Thus, the experiment lasted for two years [Carvalho11].

This experiment was based on a set of development metrics that the company gathers by automation software. Also, there is a quality department responsible for managing and monitoring metrics for process and quality. This company uses function points as a way to measure their software project's functionality. Thus, the effort of a particular requirement is calculated based on its functional size, technology, risk, and historical productivity knowledge. Therefore, the project functionality effort estimation is the sum of all its requirements effort [Carvalho11].

In their paper the authors described the measures and metrics adopted by the company and how they are calculated. At the end of their study, it was concluded that there was a 16% increase in productivity in projects developed using this hybrid approach. They believed that this increase was due to the advantages and benefits that the agile principles incorporated in this hybrid approach. Also, they recognized the importance of conducting rigorous requirements management and architecture according to the traditional process. Further, they said that the structure and artifacts used in the RUP customization were

similar to those proposed in the hybrid approach, suggesting that the integration of the Scrum practices was critical to increase the productivity of the process [Carvalho11].

The authors stated that this study was carried out in small and medium projects. Thus, further studies have to be made for large projects. However, they indicated that the hybrid approach proved to be useful in the environment where it was applied and can increase productivity [Carvalho11].

Critical Review: This case study revealed how the process productivity increased by introducing Scrum practices, which allowed for daily micro planning, controlling work, and enhancing team communication. As the authors indicated, more case studies are needed where this methodology is implemented in large projects.

### 2.2.6 A Hybrid Approach for Distributed Software Development: scRumUP

In this case study Del Nuevo *et al.* proposed a methodology named scRumUP. This methodology integrates the Scrum management practices into the software development process of RUP and is designed to be applied in a distributed environment. Also, during the development the agile practices of test-driven development (TDD) and continuous integration (CI) are applied as they allow for continuous testing and integration which is critical in a distributed environment [del Nuevo11].

One of the goals of their methodology is to describe which activities and tasks are carried out during its application. As a part of this methodology, some guidelines are given for the formation of the distributed teams and propose different ways of communication among the teams. Their methodology consists of two parts—the Release Planning phase and the Sprint. In these two phases the Scrum roles, artifacts, and ceremonies are incorporated to perform and coordinate the activities in the project and enhance communications among the teams. The RUP is mainly used for its established software development process and to provide the necessary documentation to facilitate the distributed development environment [del Nuevo11].

The Release Planning phase involves three activities: Requirements Elicitation, General release Planning Meeting, and the Local Release Planning Meeting. During this phase the requirements are gathered, the product vision is defined, and the Product Backlog is created and prioritized by the Product Owner. Also, at this time the Product Backlog is divided into sections which are assigned to the different teams. Each team estimates the features of its assigned section [del Nuevo11].

The Sprint is an iteration that is considered a very important element of the process because it results in a potentially shippable product increment. The sprint involves four phases: Sprint Planning, Development Work, Sprint Review, and a Sprint Retrospective [del Nuevo11].

- The main objective of Sprint Planning is to identify which part of the Product Backlog will become part of the working increment.

- The Development Work phase involves several activities that have as an objective the delivery of the shippable product increment which consists of executable code and documentation. The activities carried out during this phase are: Analysis and Design, Implementation and Testing, Deployment, Daily Scrum, and Weekly Meetings.
- The target of the Sprint Review is to ensure the teams have met the goal and demonstrate the results to the stakeholders. Three activities are performed in this phase: Team Demo, Product Owner Demo, and Stakeholders Demo.
- The main objective of the Sprint Retrospective is to analyze the sprint and learn what was done correctly and what needs to be changed for the next sprint.

The authors also defined a list of artifacts that are produced and maintained during the process.

This proposed methodology was adapted to fit the software development methodology of a software development company in Spain and was applied in a pilot project. At the time of the paper development, the authors were waiting for results on the performance and utility of the application [del Nuevo11].

Critical Review: The authors provided very detailed information of the process including the activities followed and artifacts produced in each part of the process. However, at the time of the realization of this paper, there was insufficient information about the benefits or challenges found during the application of scRumUP.

### 2.2.7 Implementation of a Hybrid Methodology for a Large Distributed Project Using the Project Management Body of Knowledge (PMBOK) and Scrum

Batra *et al.* presented a case study from a cruise line company that demonstrates that balancing agility and discipline in a project can be a critical factor for the success of a project. This project consisted of creating a web site for this cruise line. This was an important large distributed project with limited risks that became strategically important. The project faced several challenges such as extended scope and changes on requirements, executive sponsors, and project structure. In addition, it was clear that the initial cost and budget estimation of the project was far from reality. However, over budget could be accepted but over schedule needed to be minimized and the project needed to be successful in terms of meeting requirements and delivering results such as increasing web traffic and revenues [Batra10].

The leaders of the project thought of integrating Scrum into the PMBOK framework. Their rationale was that the PMBOK's management practices will provide the necessary structure to deal with a large project and Scrum's agile practices will provide the adaptability and flexibility to deal with the evolving scope and changing requirements [Batra10].

First, Batra *et al.* described the characteristics of the project and their challenges based on thirteen dimensions from which nine were taken from the nine management areas of the PMBOK [Batra10, PMBOK04] such as Scope, Cost, Schedule, Quality, coordination/integration management, communications management, risk management,

procurement management, and human resources management. The other four dimensions —objective, size, changes in top management, and outsourcing— were added to consider the dynamic environment of the project. Second, the authors indicated which of the twelve agile principles and key characteristics of the structured approach were supported to deal with the challenges faced by the project. Third, Batra and his colleagues described how each challenge faced by the projects was handled by the agile and structured approach. Further, the authors defined that if a project is strategically important, is distributed, involves some outsourcing, has some new and changing requirements, faces organizational and technology changes, and requires incremental delivery of the product then using a hybrid approach will be critical for the success of a project. As a consequence, requirement specifications will be met and budget and schedule can be controlled [Batra10].

This project was considered successful even though the cost of the project was five times more of the initial estimation and the length of the project doubled. This decision was based on the facts that the requirements were met and the increased revenues and benefits of the project outweighed the cost of the project. Some paradoxical findings in this case study were that the agile method brought discipline to the project and can be used on large projects [Batra10].

Critical Review: Although these case study results are relevant for large distributed projects using PMBOK and Scrum, more case studies are needed to understand how hybrid approaches behave in different distributed software development environments.

### 2.2.8 Combining Extreme Programming (XP) Agile Methodology with a Stage-Gate Project Management Model

Karlström and Runeson studied the feasibility of applying agile methods in the development of large software that use stage-gate project management models. This study was undertaken as a response to questions raised at an industrial seminar on Extreme Programming (XP) and agile development methods sponsored by the Software Engineering and Research Group at Lund University held in 2003. The seminar included representatives from more than 20 different software companies. The questions raised in this seminar were if agile development might coexist with their existing project management models and how [Karlström05].

According to the authors, a state-gate project management model describes a work process from the product's inception to its delivery. The stage-gate model is connected to a software development methodology such as waterfall or an iterative software development process. Two stages of the software methodology are connected through a gate, where top management decides whether to continue investing in the project and move to the next stage. The milestones of the software methodology are connected to a gate [Karlström05]. Figure 1 shows the Cooper stage-gate model.

In this research Karlström and Runeson developed three case studies on three large software development projects at three different large companies. They described the benefits and pitfalls of integrating agile methodologies such as XP with state-gate project management models. Their results showed not only the feasibility but also the benefits of

this approach as well as issues that must be addressed to ensure its success. The development teams in all three companies were enthusiastic about introducing agile methodologies to the process and the company management in all three companies saw the need for making process changes. However, the question was what to change and how to introduce change in an environment of higher-priority delivery deadlines [Karlström05].

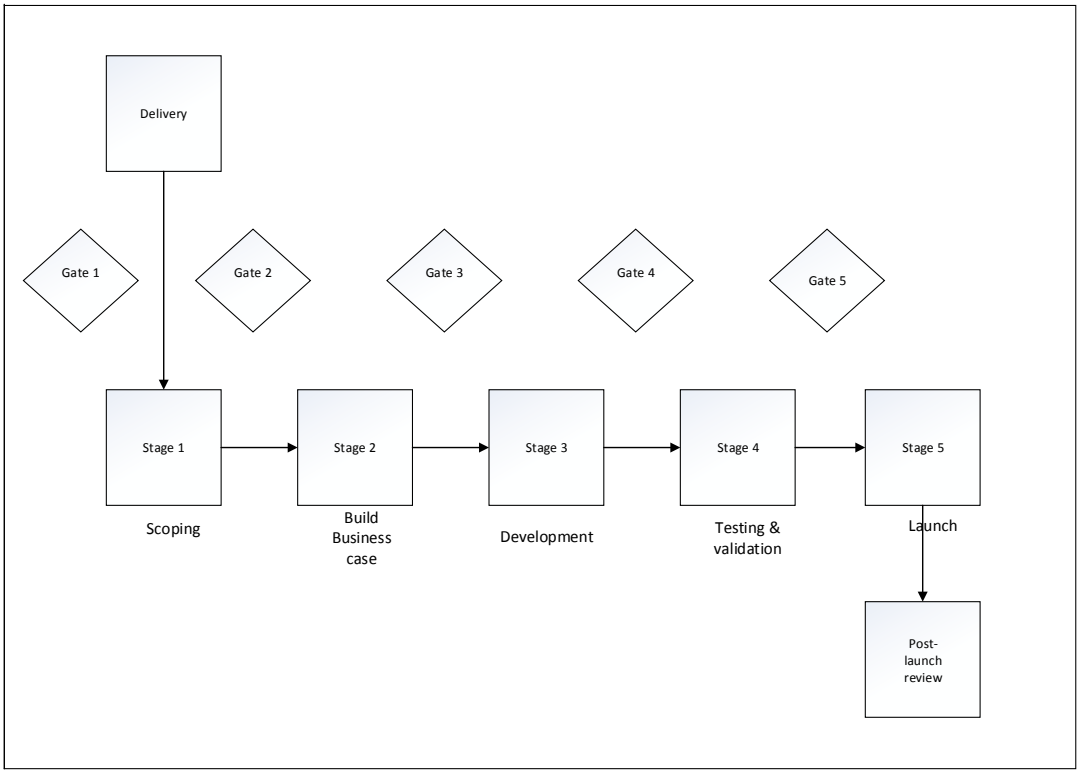



Figure 1: Cooper Stage-Gate Project Management Model [Cooper01, Karlström05]

As a first step in their study, the authors found that there was a lack of information on the research literature on how to integrate agile methodologies into stage-gate models. The three companies studied belong to a global, product-driven market environment and they

used stage-gate models to manage development projects of which their objective is functionality. By contrast, the authors found that agile methods are derived from small-scale, contract-driven software development projects, and their objective is time [Karlström05].

Thus, the objective of their research was "focused on investigating the integration of agile teams into stage-gate software product development, adhering to a strict qualitative research process [Karlström05]." This qualitative approach was based on semi-structured interviews and analyses of official documentation in the three companies. For this study, 18 semi-structured interviews were conducted at the three companies among different employees, including engineers, managers, product managers, and project managers. The interviews were recorded, transcribed, and analyzed and several techniques were used to validate the study [Karlström05].

The findings of this analysis were in four areas: planning and prioritization, communication and follow-up, process model and roles, and project management [Karlström05].

- Planning and prioritization. Agile teams were working and completing most important features first. Therefore, there was earlier feedback in the most important features and this kept the project deadline from affecting their scope. Also, it is easy to make trade-offs of features, but there is little support for long-term plans.

- Communication and follow-up. There was improved communication among team members and person-to-person communication proved to be more effective than documentation-based communication. Also, partial working software has more effect on customers, developers, and management than documentation. The agile practice of breaking down tasks made it easier for the teams to focus, gave them a sense of control, and a better understanding of the technical working of the system. As a consequence the quality of the product increased dramatically.
- Process model and roles. Agile methodologies require identifying a customer representative and this requirement helped increase the feedback on the performed work and having a quick feedback earlier in the stage of the software development. Also, documentation tasks were scheduled as any other tasks since they consume the same resources as code development. This helped prioritize the tasks more effectively.
- Project management. Management in the three companies acknowledged the need to change the traditional way of developing software, but they did not know what changes to make. After the study, the three companies are considering seriously agile methodologies to influence their processes. Also, agile methodologies helped the team members focus on past and current releases while managers focused on current and future releases.

The authors concluded that some adjustments need to be made to engineering and project management to ease the integration as the agile team must be prepared to interface with the gate model. Also, they added that while the technical staff generally is ready for a

change and welcomes agile methods, management sometimes fears them. Thus, they suggested that it is important for everybody to accept this change, which can be accomplished in small steps over a longer time [Karlström05].

Critical review: The research is very informative for using XP and stage-gate models. Other interesting research effort would include different agile methodologies other than XP. Also, the authors used multiple case studies to validate their findings and conclusions.

### 2.2.9 The Challenge of Implementing Scrum in a Traditional Development Environment

In this study Nishijima and Dos Santos compared agile methodologies against traditional methodologies. Also, the authors provided an overview of the advantages and disadvantages of implementing agile methods in a traditional development environment [Nishijima13].

The authors summarized from the literature traditional methods such as waterfall, spiral, iterative and incremental, RUP, and PMBOK. They also described briefly the background of agile methodologies, and gave an overview of the Scrum methodology. Then, the authors made a comparison of the traditional models versus agile methodologies. Later, the authors compared Scrum versus RUP and finally, they drew some conclusions [Nishijima13].

What methodology is more suitable, traditional or agile? The authors explained that it depends on many factors. They stated that according to Soares, traditional methodologies should be used where requirements are stable and future requirements can be predicted. On the other hand, they mentioned that agile methodologies are more flexible and allow to evaluate and to respond to frequent changing requirements [Nishijima13, Soares09].

When comparing RUP and Scrum, the authors stated that both methodologies are based on an incremental and iterative approach. However, they explained that RUP is frequently used for large projects even though it can be used for small projects. Also, they commented that RUP is a heavy-weight methodology with emphasis in documentation and well-defined roles, but slow to changing requirements [Nishijima13].

On the other hand, the authors stated that the Scrum methodology is based on the Agile Manifesto where the focus is on delivering increments of executable software rather than delivering comprehensive documentation, and responding faster to customer's requirements. Also, they explained that Scrum is more flexible in the size of the iterations, is based on small teams (5-9 team members) and has only three roles—Scrum Master, Product Owner, and Development Team. Also, Scrum is adaptable to different situations and can complement traditional methodologies such as RUP among others [Nishijima13].

However, the authors indicated that when implementing Scrum in a traditional environment three factors need to be considered: culture, people, and communication.

Also, they added that once the challenges are overcome, the advantages of Scrum are: 1) it is a good communication tool, 2) it strengthens the relationship with the customer, and 3) it provides a great motivation among the development team [Nishijima13].

Critical review: This article is an introduction for agile and traditional methods. However, it lacks substantial evidence to support their point of challenges that organizations face in the implementation of both methodologies. Therefore, this article needs more sources, cases, and examples to strengthen the original goal of this research.

## 2.3 Hybrid Methodologies Summary

Table 1 contains a summary of the hybrid methodologies described in the literature review.

| Authors | Description | Critical Review |
|---|---|---|
| Cho | A hybrid software development method that is suitable particularly for large-scale projects in the business software industry. This method integrates the Rational Unified Process (RUP) and Scrum to maximize their strengths while reducing the weaknesses of both methodologies. | Does not mention any organizations implementing this approach, so challenges or issues on implementing this approach were not discussed. |
| Nisa and Qureshi | A hybrid model named SPRUP that combines the strengths of RUP, Scrum, and XP and eliminates their weaknesses to produce high quality software with a low defect rate. | More case studies are needed where the SPRUP model is implemented in different environments that may not be controlled, and for projects with longer duration as well as for large-scale projects. |
| Bashir and Qureshi | An integrated framework that combines the strengths of RUP, Scrum, and XP in order to achieve high quality software and enhance the team productivity. | A very general description of the framework is provided and details about how or what activities are executed during the development process are not mentioned. |

| | | |
|---|---|---|
| Nortier *et al.* | A framework that combines RUP and Scrum to deliver a development process that handles uncertain requirements and adapts to changing requirements late in the development cycle while delivering high quality products within budget and schedule. | This is an approach that can be implemented in an organization that deals mostly with distributed teams and large projects. However, there is a need to develop case studies where this proposed methodology may be tailored for different projects in different types of organizations. |
| Carvalho *et al.* | A hybrid approach that incorporates Scrum practices through the RUP process. This hybrid approach was carried out in small and medium projects and it proved to be useful in the environment where it was applied and increased productivity. | This case study reveals how the process productivity increased by introducing Scrum practices, which allowed for daily micro planning, controlling work, and enhancing team communication. As the authors indicated, more case studies are needed to be developed for large projects. |
| Del Nuevo *et al.* | scRumUP integrates the Scrum management practices into the software development process of RUP and is designed to be applied in a distributed environment. Also, during the development the agile practices of test-driven development (TDD) and continuous integration (CI) are applied. | The authors provided very detailed information of the process including the activities followed and artifacts produced in each part of the process. However, at the time of the realization of this paper, there was insufficient information about the benefits or challenges found during the application of scRumUP. |
| Batra *et al.* | A hybrid methodology that integrates the Project Management Body of Knowledge (PMBOK) and Scrum practices. The PMBOK's management practices provide the necessary structure to deal with a large project and Scrum's agile practices provide the adaptability and flexibility to deal with the evolving scope and changing requirements. | Although these case study results are relevant for large distributed projects using PMBOK and Scrum, more case studies are needed to understand how hybrid approaches behave in different distributed software development environments. |
| Karlström and Runeson | The benefits and pitfalls of integrating XP with a state-gate project management model. | The research is very informative for using XP and stage-gate models. Other interesting research effort would include different agile methodologies other than XP. Also, the authors used several case studies to validate their findings and conclusions. |
| Nishijima and Dos Santos | A comparison between agile methodologies and traditional methodologies. Also, an overview of the advantages and disadvantages of implementing Scrum in a traditional development environment such as RUP. | This article is an introduction for agile and traditional methods. However, it lacks substantial evidence to support their point of challenges that organizations face in the implementation of both methodologies. Therefore, this article needs more sources, cases, and examples to strengthen the original goal of this research. |

Table 1: Summary of the Hybrid Methodologies

## 2.4 Literature Gaps

The literature review presented in the previous section permit us to conclude that there are multiple literature gaps in several areas on hybrid methodologies. For instance, the need for hybrid methodologies is increasing due to their potential benefits in software development in different organizations such as in the academic area as well as public and private. However, there is a lack of empirical research and only few case studies have been published in the literature on hybrid methodologies. As a consequence, there is insufficient evidence related to their elements, issues, and findings to make them valid and reliable for users. Also, in the previous research there is no general methodology that integrates the interacting elements of hybrid methodologies. Neither there is a hybrid approach that presents a general process for combining two or more methodologies that has been proven successful. This suggests that different projects require different management practices [Shenhar01].

In particular, there is very limited research on combining specific hybrid methodologies as RUP and Scrum and even fewer case studies developed and published. This narrows even further potential empirical evidence for users.

Specifically, two gaps relevant to this field have been identified in the literature review:

Gap 1: At the present time there is limited empirical research reported in the literature on the process of successfully combining RUP and Scrum.

Rationale 1: The process of combining RUP and Scrum successfully is recognized as the activity undertaken to ensure proper combination of their respective phases on a software development project. There is a lack of evidence in the literature as to exactly how this linking process has to be carried out.

Gap 2: At the present time there is limited empirical research reported in the literature on the nature of combining RUP and Scrum.

Rationale 2: The nature of combining RUP and Scrum can be defined as the interrelationship of managerial elements in a software development project using RUP and Scrum such as strategy, organization, culture, metrics, issues, and people. Further, according to the existing literature review, different projects may need different management practices to manage such elements in order to be successfully delivered—one size does not fit all projects [Shenhar01]. However, in the literature there is little evidence as to exactly what managerial elements and activities are interrelated, and how these interrelationships are used to obtain good results.

# Chapter 3

# RESEARCH DESIGN

The purpose of this chapter is to present the research design process, which includes the research questions and objective, the research methodology to be implemented, and the data collection process.

## 3.1 Research Questions and Objective

The literature review presented in chapter 2 revealed some relevant research gaps that were identified in relation to hybrid methodologies in general and specifically for RUP and Scrum. The current empirical evidence is insufficient to be reliable and valid. Further, only few case studies have been documented, which creates the need to carry out and document more case studies to strengthen limited findings.

Thus, to help fill these gaps, the main purpose of this research is to develop a new case study that combines RUP and Scrum to validate and/or add new information from previous cases studied. Therefore, a research objective is established as: “develop a case study to describe the process of combining both RUP and Scrum methodologies.” This case study will embody a lobbyist software development application that was developed for the City of Jacksonville, Jacksonville, FL.

To fulfill this research objective, two research questions associated with the two gaps identified in the literature review for this study are defined as follows. Related to the first gap**:** "The process of combining RUP and Scrum successfully does not have sufficient empirical research," the first research question is stated as:

Research Question 1: How should the process of combining RUP and Scrum be carried out for achieving a successful implementation?

Then for the second gap**:** "The nature of combining RUP and Scrum has insufficient empirical research," the next question is stated as:

Research Question 2: Which are the important managerial elements such as strategy, organization, culture, metrics, issues, and people that should be considered when combining RUP and Scrum for delivering a successful project?

## 3.2 Research Methodology

Research methods are generally classified into two major categories: Quantitative and Qualitative methods [Yin94]. Both are inquiry processes that investigate a human or social problem. The former is built on the observation of the natural phenomena and a whole view is formed with words, reporting detailed of informants, and is carried on a natural setting [Creswell94, Denzin94]. The latter is based on testing a theory composed

of variables, measured with numbers, and analyzed with statistical procedures to validate if the proposed theory is true [Creswell94].

There are major differences between qualitative and quantitative methodologies. To define which method is more appropriate for a research objective there are different aspects to take into consideration [Yin94]:

- Aspect 1. Reliance on existing literature and prior evidence: qualitative methods have less reliance.
- Aspect 2. The nature of the research objective and question: qualitative methods ask how and why rather than what and how many.
- Aspect 3. Extent of control over actual behavior extents: qualitative methods have less or no control.
- Aspect 4. Rigor and Validity: same level for both qualitative and quantitative methods.
- Aspect 5. Amount of data: not determined until saturation is reached.

Moreover, there are other methodologies to carry out research such as a case study research, which may also use quantitative and/or qualitative approaches. The criteria to select a case study research using qualitative and/or quantitative approaches, are similar to the above-mentioned aspects [Yin94].

Generally there are several types of approaches needed to carry out case study research [Yin94, Baxter08]. Yin categorizes case studies as explanatory, exploratory, and

descriptive. Also, Yin differentiates them as single, holistic, and multiple case studies [Yin94]. Thus, according to the above information, for this research study the best fit was to select a descriptive and single case study research methodology that uses a qualitative approach. This research may also use building theories from case study research [Eisenhardt89] as the core methodology to draw valid and reliable results and contributions from carrying out the lobbyist case study.

A typical qualitative case study research consists of an overview or introduction of the research, a literature review, a research design, data collection, results, and conclusions. In this type of research, guiding propositions are suggested to assist data collection and data analysis by identifying the important prior concepts or constructs with references to existing literature. Thus, guiding propositions will help direct attention to something that will be examined within the scope of the research. Without these propositions a researcher may be tempted to collect everything [Yin94, Eisenhardt89]. Similarly, Yin suggests these propositions as an option to organize the research.

Based on the existing literature, this study develops two initial propositions: Guiding proposition 1 supports Research Question 1, whereas Guiding Proposition 2 supports Research Question 2. Table 2 contains these two guiding propositions with their rationale.

| Guiding Propositions | Rationale |
|---|---|
| Guiding Proposition 1. Specific processes are used to combine the implementation of RUP and Scrum for different projects in different organizations. | This guiding proposition suggests that processes carried out with RUP and Scrum during the software development life cycle may vary for different projects in different organizations. This implies that different mechanisms or tools are needed at different phases of the project life cycle to ensure synchronization of both processes for delivering a project successfully. |
| Guiding Proposition 2. Combining Hybrid methodologies like RUP and Scrum influences different managerial elements that need to be addressed to successfully deliver a software development application. | This guiding proposition implies that emphasis on strategy, organization, culture, people, tools, and metrics, among others may differ from one type of project to other, thus requiring a different approach or style to implement this particular hybrid methodology implementation. This is consistent with Shenhar, who suggests that different projects may need different management practices to be successful [Shenhar01]. |

Table 2: Guiding Propositions and Rationale

It is important to emphasize that these guiding propositions are only used as guiding propositions for empirical research, which may develop more detailed propositions that may help better answer the proposed research questions [Yin94, Eisenhardt89].

## 3.3 Data Collection

To carry out the data collection, the following phases are identified: case selection criteria, data collection process, data sources, and data analysis.

### 3.3.1 Case Study Selection Criteria

The selection criteria for this case study included:

- The length of the project to be accomplished with no more than two academic terms.
- Some team members with experience in participating and managing software development projects.
- Commitment to the project where participants are willing to share in-depth information.
- Geographical location in Jacksonville area to facilitate customer and students interaction.
- An organization interested in developing an application using a hybrid methodology integrating RUP and Scrum.

### 3.3.2 Data Collection Process

A data collection process needs to ensure that the information obtained during this process is reliable and valid for further research and practical purposes [Yin94]. For this purpose, in this research the data was collected from: development team's meetings, meetings with customers and feedback, and other data produced during the development of the project and other relevant activities.

### 3.3.3 Data Sources

The data sources for this research originated from the stakeholders in the lobbyist software development project such as participants with different roles as business analysts, software developers, customers, and managers. Another data source came from the quality assurance activities and the documentation produced during the software development life cycle phases such as requirements, analysis and design, implementation, testing, and deployment. Also, other major sources of evidence for this research include documentation generated during the software development project such as memos, emails, meeting minutes, conference calls among members and customers, and templates for required documentation coming from customer's organization standards, among others.

Thus, using the combination of these multiple sources ensures the validity of the data collected, because it increases the level of confidence of the findings obtained and reduces possible bias [Yin94].

### 3.3.4 Data Analysis

During the data analysis phase several steps may be carried out like: within-case analysis, cross-case analysis, and conclusions [Eisenhardt89]. For this research, the first step in data analysis was to focus on within-case analysis as a standalone entity and to distinguish the unique patterns of our lobbyist case study. Next, the research compared

within-case results with other case studies' findings already developed and published for cross-case examination purposes. Finally, the results of this data analysis process and therefore, the conclusions of the Lobbyist case study analysis were delivered.

# Chapter 4

# OVERVIEW OF RUP, SCRUM, AND THE IBM-CLM TOOL

## 4.1 Introduction

As mentioned in the research design, the main purpose of this research is to show the results of implementing a hybrid approach in the development of the Lobbyist Registration and Tracking System for the City of Jacksonville. This hybrid approach integrated the RUP and Scrum methodologies within the IBM Rational solution for Collaborative Lifecycle Management (CLM). Therefore, to address this objective, this section includes a general introduction to this collaborative tool, an overview of the RUP, and the Scrum processes.

## 4.2 The IBM Rational Solution for Collaborative Lifecycle Management

The IBM Rational solution for Collaborative Lifecycle Management integrates several tools that work together as one in a transparent way and can be used efficiently for a traditional or agile development team. This solution is built in the Jazz platform and its main components are the Rational Requirements Composer, the Rational Team Concert, and the Rational Quality Manager. These three products are linked together to provide a collaborative environment for the different roles in a development team (Figure 2).

Graphic redacted, paper copy available upon request to home institution.

Figure 2: The IBM Rational Solution for a Collaborative Lifecycle Management [IBMDeveloperWorks11]

The Rational Requirements Composer (RRC): This tool utilizes the Requirements Management (RM) application, which provides tools to capture, organize, collaborate, and manage requirements for any size development team. These tools help teams to capture sketches and storyboards, use cases, glossaries, and process models. Because of this, "project teams can manage their requirements, write good use cases, improve traceability, increases stakeholders collaboration, reduce project risk, and increase quality [Chawla13]."

The Rational Team Concert (RTC): This product facilitates team collaboration through the software development lifecycle. This collaborative development system provides a

repository where teams can manage and organize their work allowing for a clearer view of the project progress. RTC integrates work item management and iteration planning, source control management, build management, and team health [Göthe08]. Rational Team Concert provides agile projects with tools to create the product and sprint backlogs, and allows teams to track their progress during an iteration.

The Rational Quality Manager (RQM): This product allows quality assurance teams to track the activities of the quality assurance effort to ensure the application will meet the business objectives. The main artifact of this tool is a customizable test plan that contains business objectives, requirements, test environment, tests cases, test schedules, builds, reports, strategy, and the quality process [Göthe08]. Further, these activities can be linked to the requirements and development work items to allow for full traceability.

## 4.3 The Rational Unified Process

The Rational Unified Process (RUP) is a process framework based on software engineering principles with the goal of producing high-quality software that meets the customer needs under a predictable schedule and budget. RUP takes an iterative, user-case driven, architecture centric approach and incorporates many of best practices in software development [Kruchten03].

### 4.3.1 RUP Six Best Practices

RUP's process framework can be easily adapted or extended to fit the needs of a project and provides guidelines, tool mentors, templates, and concepts so that the development team can take advantage of its six best practices [Rational01]. These six best practices are explained according to [Kruchten03].

Iterative Development. By following this approach, the development process is divided in iterations. Each iteration follows the four phases of the sequential approach in which teams can address some requirements and risks, design, implementation, and integration and testing. As a result, at the end of each iteration the team will have a working product that is evolved in the next iterations. This allows teams to take into account changing requirements and address risks early in the process.

Manage Requirements. Since having up-front requirements is almost impossible for most of the projects, it is important to expect new or changing requirements. Thus, as the product evolves and changes or new requirements are integrated into the product, it is necessary to manage requirements. This means that it is important to have a well-defined process to elicit, organize, and document the original requirements; evaluate changes to these requirements and assess their effects on the budget and schedule; and track and document trade-offs and decisions.

Use Component-based Architectures. This approach allows building a resilient software architecture that is flexible, accommodates changes, and enables reuse [Rational01]. This concept together with an iterative approach permits to identify components gradually and helps decide which components can be developed, reused, or customized from available commercial products during the whole development process.

Visually Model Software. A model helps have a better understanding of the system. Therefore, using modeling helps visualize how the different elements of the software development process fit together; helps identify, create, and document the structure and behavior of the software's architecture; ensures consistency between the design and implementation; and avoid misunderstandings [Rational01].

Continuously Verify Software Quality. This approach is based on continuously verifying the product with respect to its functional and non-functional requirements. Thus, the increment should be tested in each iteration so that defects can be identified earlier in the project.

Control Changes to Software. In an iterative process work items are updated frequently. Because of this it is important to keep track of changes that occur during requirements, design, and implementation. Managing changes ensures that everybody is in synch and has the latest version of the work item. Therefore, embracing, controlling, tracking, and monitoring changes is vital in an iterative process.

### 4.3.2 RUP Process Description

Roles, artifacts, activities, workflows and disciplines are the five primary elements used to represent the process [Kruchten03].

- Roles: A role describes the behavior (activities) and responsibilities (artifacts) of an individual or group of individuals working in a team.
- Activities: Activities define the work a role performs.
- Artifacts: Artifacts are the concrete products of a process. They are the input and the output of an activity performed by roles.
- Workflows. A workflow is a sequence of activities that produces a valuable result.
- Disciplines: Disciplines are used to organize and group the roles, activities, artifacts, and workflows of the process.

The RUP process covers the entire software development lifecycle and is represented by two dimensions as shown in Figure 3. The horizontal axis represents time and shows how the process moves forward through the different phases, iterations, and milestones. The vertical axis represents the static aspect of the process and contains the main nine disciplines. Thus, the four phases are performed sequentially while the disciplines are executed iteratively throughout the four phases [Kruchten03, Rational01].

Graphic redacted, paper copy available upon request to home institution.

Figure 3: RUP Process [Rational01]

### 4.3.3 RUP Disciplines

The RUP nine disciplines are represented on the vertical axis and they are divided into six technical disciplines and three supporting disciplines which are explained as follow based on [Kruchten03].

The six technical disciplines are:

- Business Modeling. The goals of this discipline are to understand the organization, identify stakeholders, learn about the currents process, roles, and deliverables, and understand what the currents problems are and how they can be improved. This information helps identify how the application will fit in the organization and how the users will ultimately benefit from this.

- Requirements. One of the main goals of this discipline is to gather the user's requirements with the objective of defining user's needs and the high-level features for the system which will be contained in the vision of the product. These features are later translated into more detailed requirements, which are captured in use cases. These activities help establish a better understanding of what the systems should do and thus be able to define the scope of the system.
- Analysis and Design. During this discipline the requirements are converted into specifications that describe how to implement the system. First, the analysis, which is driven by the use cases, transforms the requirements into a set of classes and subsystems and concentrates on the functional requirements. Later, the design adapts the output of the analysis to the non-functional requirements constraints and the output of the design is the solution to be implemented.
- Implementation. The goal of this discipline is essentially the realization of the design which is performed in iterations. One of the objectives in each iteration is to have a plan that describes which subsystem will be implemented and the order of the classes to be implemented. The activities performed during this discipline include coding, fixing code defects, and performing unit tests and code reviews to ensure its quality. Then, every implemented component is integrated and tested into a new version of the subsystem, and later the subsystem is integrated and tested into the system.
- Test. The purpose of this discipline is to assess the quality of the product by validating that the requirements are implemented appropriately, detecting defects

or problems, and validating that the product works as designed. The main focus of this discipline is to have the right approach to find defects in the product.

- Deployment. The goal of this discipline is to deliver the finished software to the user community. Some of the activities performed during the software deployment include beta testing which involves testing the software in the user's site, preparing the software for final delivery, installing the software, and training end users.

The three supporting disciplines are:

- Configuration and Change Management. The purpose of this discipline is to maintain control of the different artifacts created during the software development process. Since during an iterative process artifacts are updated, it is important to maintain control of the different version of the artifacts, keep information on why these artifacts were modified and who is responsible for each artifact. Another important activity in this discipline is to capture and maintain control of the change requests and what artifacts were affected by these.
- Project Management. The goals of this discipline are to provide a structure that allows managing software projects, specifically projects involving an iterative software development process; to offer guidelines to execute and monitor the progress of the project; and to provide a framework to manage risks.
- Environment. The main activities during this discipline involve helping the organization select and acquire the tools to support the development software

process, configure the RUP process according to the organization's needs and the project's needs, and describe this chosen configuration in the Development Case.

### 4.3.4 The Four Phases of RUP

According to Kruchten [Kruchten03], the RUP process is divided in four phases which are represented on the horizontal axis and are described as follows:

- The Inception Phase. This phase concentrates on reaching an agreement among all the stakeholders on the initial project concerns such as defining the project's scope, estimating the schedule and cost of the entire project, and evaluating the project's feasibility. Also, this phase aims to identify the critical use cases that will drive the system's functionality and will help establish at least one candidate architecture.
- The Elaboration Phase. The objective of the Elaboration phase is to establish, validate, and baseline the architecture, address the major risks, and evolve the project plan. Therefore, during this phase it is necessary to have a good understanding of the critical requirements to ensure that have been covered in the architecture. Also, designing, implementing and testing the initial architecture helps mitigate major risks early in the project. These activities allow having detailed information about what needs to be done so that the project plan can be updated to reflect better estimates.
- The Construction Phase. Some of the goals for this phase are to complete the requirements, elaborate or expand the design, develop the rest of the components

and features, test them and integrate them to the rest of the application, and asses the product against the defined acceptance criteria. Also, during this phase it is important to achieve an acceptable level of quality and develop useful versions of the product as rapidly as practical.

- The Transition Phase. The purpose of this phase is to deploy the software to the user community when the software has reached an acceptable level of quality and user documentation has been prepared. During this transition some activities are carried out which include performing some beta testing to validate that the product meets users' expectations; making corrections and enhancements to ensure good performance and usability; training users and maintenance people; and evaluating the deployed version against the product's vision and acceptance criteria. Sometimes during this phase issues will emerge that will require future evolutions of the product.

The four phases represent a development cycle which results in a product. Sometimes this is referred to as one evolution of the product. The duration of each phase varies according to the project therefore they can consist of one or several iterations.

## 4.4 Scrum Framework Overview

Scrum is an agile management process that helps manage and organize the software development process. This process uses an incremental and iterative approach that provides the practices to respond to changing requirements and the feedback to build the

product that is needed. Because of this, risks are controlled and mitigated early in the process. Scrum provides a structure through its roles, events, and artifacts and a collaborative environment so that team can solve complex problems little by little.

### 4.4.1 Scrum Process Description

Scrum organizes the development process into sprints, which have a duration of up to four weeks. At the beginning of each sprint there is a Sprint Planning Meeting in which the length of the sprint is decided and the Development Team selects a set of features from the Product Backlog. The Product Backlog is a prioritized list of the features established by the clients needed to develop the product from the business standpoint. The set of features is what the Development Team commits to or forecasts to finish during the sprint. This set of features is broken down into tasks to form the Sprint Backlog. No one is allowed to make changes to the Sprint Backlog or add items except for the Development Team. When a sprint begins, a Daily Scrum Meeting is held at the start of the day. The length of this meeting is no more than fifteen minutes. During this meeting each member of the Development Team has to answer three questions: What did you do yesterday? What are you doing today? What impediments do you have? The Scrum Master is in charge of removing impediments to facilitate the progress of the project. At the end of the sprint, the Development Team has a potentially shippable product which is demonstrated to the stakeholder in a Review Meeting with the purpose of obtaining feedback. After this meeting, the Development Team holds a Retrospective

Meeting, where the process is inspected. The goal of this meeting is to discuss what is working and what needs to be improved if that is the case.

Each sprint typically follows this pattern where an evolving product is created by successively adding increments of workable software. Figure 4 summarizes the Scrum process.

### 4.4.2 Scrum Roles, Events, and Artifacts

According to Schwaber, this process consists of Scrum teams and their associated roles, events, artifacts, and rules that bind them together. The following sections describe these roles, events, and artifacts based mainly on [Schwaber11, Schwaber09].


Figure 4: Scrum Process [Deemer10]

#### 4.4.2.1 Scrum Roles

Scrum teams consist of three roles: the Scrum Master, the Product Owner, and the Development Team.

The Product Owner is responsible for obtaining the highest value of the product and the work of the Development Team. The Product Owner is in charge of the Product Backlog and is in charge of prioritizing or ordering the list according to which has the highest business value or highest business risk .

The Development Team is self-organizing and cross-functional. As such, the size of the Development Team needs to be optimized to remain agile and to deliver work. Also, the Development Team decides what to commit to and how to achieve this commitment [Deemer10].

The role of a Scrum Master is different from the role of a project manager because the Scrum Master does not manage the project and the team. The Scrum Master ensures that Scrum's theory, practices, and rules are used and understood as intended. Thus, the Scrum Master supports the Product Owner to obtain the highest business value and coaches the Development Team to create a high-value product and removes any impediments that prevent reaching these objectives [Deemer10].

#### 4.4.2.2 Scrum Events

The sprint is a time-boxed event that lasts four weeks or fewer and starts immediately after the previous sprint has concluded. A sprint includes the followings events: Sprint Planning Meeting, Daily Scrum, the Sprint Review, and the Sprint Retrospective.

The goal of the Release Planning Meeting is to create a release plan that defines the goal of the release, general features and functionality for this release, the major risks, and the estimated delivery date and cost. This release plan will change according to the progress made during each Sprint. The product is released when enough increments have been created for the product to have value.

The Sprint Planning Meeting takes place at the beginning of each sprint and it is time-boxed. It will typically last no more than eight hours for a four-week sprint. This meeting includes the Product Owner, the Scrum Master, and the Development Team. This Sprint Planning Meeting consists of two parts: the Sprint Planning Part One and the Sprint Planning Part Two [Deemer10].

The purpose of the Sprint Planning One is to decide what will be achieved during the sprint. The Development Team understands and selects the items that can be accomplished during the sprint and discusses the goal for this sprint. Also at this time, they review the Definition of Done.

The Definition of Done describes what the Scrum team understands for the work to be complete. For example, what are the code standards the items must meet, the testing activities that should be performed on the items, and which documentation is required [Deemer10]. This ensures that when an item is "done," it complies with the expected quality.

During the Sprint Planning Two, the Development Team focuses on how to implement the features selected from the Product Backlog to convert them into a working increment. The Daily Scrum is a meeting that will take no more than fifteen minutes. Its purpose is to know what the Development team has achieved until this moment, what is stopping its progress, and what work will be achieved until the next Daily Scrum. Therefore, the Daily Scrum helps assess how the team is progressing to achieve the sprint goal.

The Sprint Review is a meeting held at the end of the sprint where the Scrum Team gets feedback from the stakeholders. In this meeting the Development Team demonstrates the product increment and discusses what has been accomplished and what has not been accomplished. Also, from this meeting come new ideas of how to improve the product and those ideas go into the product backlog.

The Sprint Retrospective is a meeting that takes place after the Sprint Review and prior to the next Sprint Planning Meeting. In this meeting the Development Team does a review about the process. Also, they agree on what is working and what is not and what needs to

be changed to be more effective so that new improvements can be incorporated into the new sprint.

According to Deemer [Deemer10] the Sprint Review represents an opportunity to inspect and adapt the product while the Sprint Retrospective provides the opportunity to inspect and adapt the process.

#### 4.4.2.3 Scrum Artifacts

Scrum artifacts include the Product Backlog, Sprint Backlog, Sprint Burndown, and Release Burndown.

The Product Backlog is the business perspective of the list of requirements needed to create the product and is the Product Owner's responsibility. Therefore, the Product Owner is in charge of the Product Backlog's contents and also of ordering this by business value, risk, priority, and necessity. The Development Team is in charge of implementing the items in this list and because of this, of providing an estimate of the effort needed to develop each item. The Product Backlog is a dynamic list that evolves during the process to reflect new features or changing requirements.

The Release Burndown chart shows the total estimated effort remaining from the Product Backlog using the sprints as units of time.

The Sprint Backlog is a subset of the Product Backlog items. These are the items with the highest priority or order that the Development Team forecasts or commits during the Sprint Planning. These items are broken down into tasks and become the Sprint Backlog**.** This artifact reflects all the necessary work to complete the product increment and achieve the sprint goal. The Sprint Backlog belongs and can only be updated by the Development Team [Schwaber11, Seikola10].

According to Deemer, the purpose of the Sprint Burndown chart is to show the progress toward the goal instead of how much time was spent on the tasks. Thus, this chart is updated every day to reflect the estimated remaining work in hours for the entire team until all the tasks are finished. Therefore, with this chart is easy to see total hours of work remaining for the whole team [Demmer10].

# Chapter 5

# LOBBYIST CASE STUDY

## 5.1 Background

In the UNF School of Computing curriculum for the Masters in Computer and Information Sciences degree, the graduate course of Engineering of Software is offered in two semesters. One of the objectives of this course is to provide the students the opportunity to work on real-world projects, while learning software engineering principles. For this purpose, at the beginning of the course during the Fall 2012 a project from the City of Jacksonville (COJ) was presented to the software engineer students. This project was presented by the director of the Office of Ethics, Compliance and Oversight, who will be referred as the project sponsor throughout this paper.

During this time the project sponsor expressed the needs of her office, which had an outdated, error-prone system to register, track, monitor, and validate the lobbyist activities. These activities were carried out manually which was a cumbersome task for the COJ employees and delivered an inefficient service to lobbyists and citizens.

This process will be described according to the three different users involved in the lobbyist activities (Lobbyists, COJ employees, and Citizens):

Lobbyists. They are required to register every year to perform lobbyist activities. Also, they need to provide information about their clients and the issues represented by them. To initiate this process, a lobbyist follows the next two steps:

1. Fill out a registration form. This form can be downloaded from the COJ Ethics Office website following a link to the COJ City Council website. This form contains four sections—A, B, C, and D.
    - Section A (Figure 5). The purpose of this section is to gather the lobbyist's personal information, lobbyist status (active or inactive), if the registration is a renewal, and date of initial registration. This section needs to be renewed once a year.

| Section A. REGISTRATION OF LOBBYIST | |
|---|---|
| Lobbyist Name: | |
| Street Address: | |
| City: | |
| State | |
| Zip Code | |
| E-mail Address | |
| Phone | ( ) |
| Status: (check one) | Active: ☐ Inactive: ☐ |
| Renewal: | ☐ (Check box if renewing registration) |
| Date of Initial Registration: | |

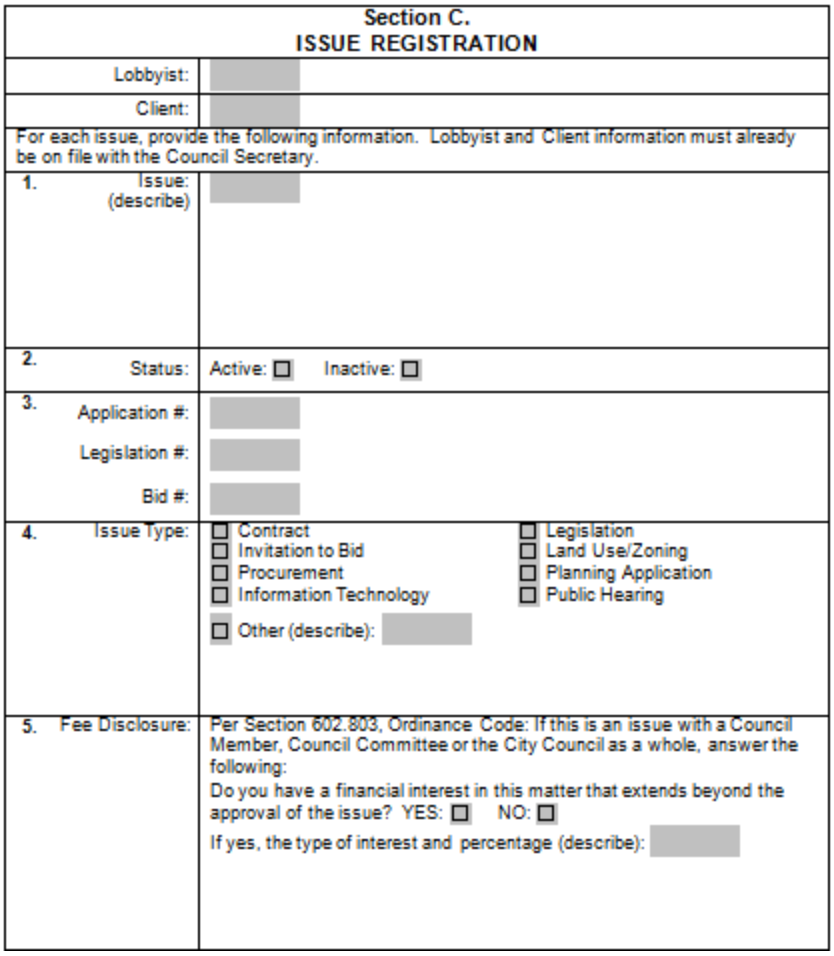
Figure 5: Lobbyist Registration Form, Section A

- Section B (Figure 6). This section contains the client information such as client type, name, address, and phone number.

| Section B.<br>CLIENT INFORMATION | |
|---|---|
| Client Type:<br>(check one) | Individual: ☐ Business: ☐ |
| Client Name:<br>(and/or Business Name) | |
| Business Type: | |
| Street Address: | |
| City: | |
| State | |
| Zip Code | |
| Phone | ( ) |

Figure 6: Lobbyist Registration Form, Section B

- Section C (Figure 7). The purpose of this section is to register an issue. If lobbyists are registering only the issue, their personal information and their client information must already be on file with the Council Secretary. This section contains information such as description of the issue, status (active or inactive), type, fee disclosure, and other information concerned with the issue.



**Section C.**
**ISSUE REGISTRATION**

Lobbyist:

Client:

For each issue, provide the following information. Lobbyist and Client information must already be on file with the Council Secretary.

1. Issue: (describe)

2. Status: Active: ☐ Inactive: ☐

3. Application #:

Legislation #:

Bid #:

4. Issue Type:

☐ Contract
☐ Invitation to Bid
☐ Procurement
☐ Information Technology
☐ Legislation
☐ Land Use/Zoning
☐ Planning Application
☐ Public Hearing
☐ Other (describe):

5. Fee Disclosure: Per Section 602.803, Ordinance Code: If this is an issue with a Council Member, Council Committee or the City Council as a whole, answer the following:
Do you have a financial interest in this matter that extends beyond the approval of the issue? YES: ☐ NO: ☐
If yes, the type of interest and percentage (describe):

Figure 7: Lobbyist Registration Form, Section C

- Section D (Figure 8). In this section the lobbyist signs to certify that he/she reviewed and understands Chapter 602, Part 8, Jacksonville Ethics Code, which is annexed to the form or can be seen in the same web page where the registration form was found.

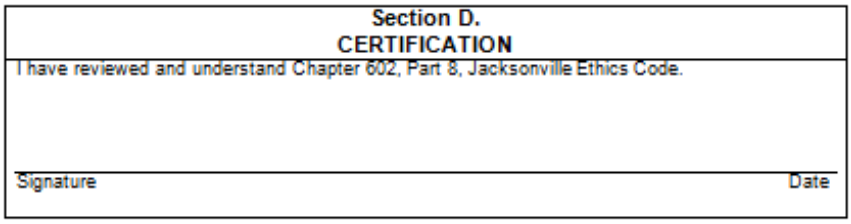



Figure 8: Lobbyist Registration Form, Section D

2. Print and submit the form via mail or personally to the Office of the City Council Legislative Services Division.

Once the registration form is received, the COJ employees follow the next steps:

1. The form is revised and approved or rejected by the Council Secretary.
2. If the form is rejected, the legislative assistant sends the form back by mail to the lobbyist for correction.
3. If the registration form is approved, the legislative assistant enters the lobbyist name; client name; a brief issue description; and expiration date into an Excel document. This Excel document contains a list of all the active lobbyists and is updated each day, if necessary, with new information.
4. Every time this document is updated, it is converted into a PDF document and uploaded to the lobbyist registration area in the City Council web page. This site contains all the past versions of this document.
5. The registration forms are filed on binders and kept on-site for any inquires.

Citizens interested in lobbyist activities can either:

1. Download the PDF files posted on the City Council web page, which contain general information as described above; or,
2. For more detailed information, citizens can visit the COJ Legislative Services Division office in downtown and browse the binders that contain the lobbyist registration forms.

Thus, this office was in need of automating its lobbyist registration system to provide a more efficient service. Exacerbating this need was the City of Jacksonville IT department's inability to develop this system due to the lack of financial resources. However, if this project were to be carried out by the UNF software engineering students, they would need to coordinate with the IT department and make certain that the system would be developed using technologies in use by this department in the City of Jacksonville.

A team of UNF software engineering students took the challenge of designing, developing, validating, and delivering a simpler, more efficient, and reliable system to keep track of the lobbyist activities. This project was carried out under the direction of the software engineering professor who served as a mentor and project manager for the execution of the project and also acted as a liaison between the team and the sponsor of the project who was the Director of the Office of Ethics, Compliance, and Oversight.

## 5.2 Hybrid Approach Conception

One of the objectives for this course was to follow a structured approach during the software development lifecycle. For this reason, the execution of the project was divided into two stages. The first stage would involve the Requirements Analysis and Architectural Design phases and would be executed during the first semester. The second stage of the project would include the Construction and Testing and Transition phases and would be carried out during the second semester as shown in Figure 9. Another objective was to provide the students with the experience of working in an incremental and iterative environment through an agile-hybrid approach within the context of a collaborative tool.

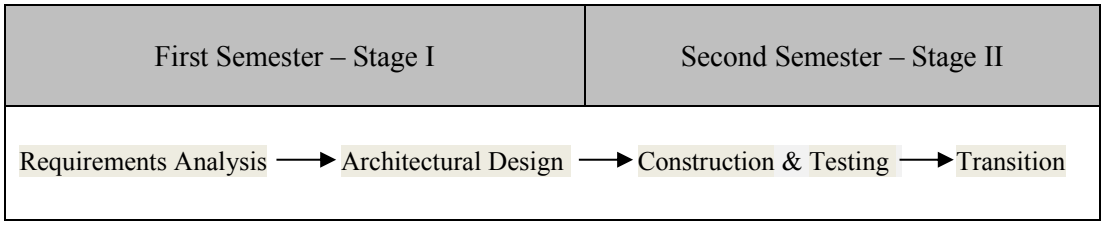



Figure 9: The Four Phases of the Software Development Cycle

To accomplish the first objective, the software development lifecycle would follow a RUP cycle. This would mean that the first stage would include the Inception and Elaboration phases and the second stage would include the Construction and Transition phases.

For the second objective, RUP and Scrum elements would be integrated under the umbrella of the IBM Rational CLM solution. RUP's activities, roles, and artifacts would guide the project through the first stage, while Scrum's agile practices would drive the second stage to enhance and manage the software development. The collaborative tool, then, would serve to enhance team collaboration, provide visibility of the project activities and project progress, provide a repository for the project deliveries during the first stage using the Rational Requirements Composer, and support the agile process during the second stage using the Rational Team Concert. However, using the full capabilities of the CLM tool would entirely depend on the team's familiarity with this tool and the available time to learn it.

Following the above, the first stage of the project was divided into six iterations and the second stage was divided into eight iterations called sprints. Each iteration or sprint was defined to have a duration of two weeks and accompanied by a set of deliverables. Figure 10 shows the conceptualization of the hybrid approach.

<table>
<tr><td colspan="13">IBM Rational CLM Solution</td></tr>
<tr><td colspan="7">Stage I - RUP</td><td colspan="6">Stage II - Scrum</td></tr>
<tr><td colspan="4">Inception</td><td colspan="2">Elaboration</td><td></td><td colspan="4">Construction</td><td colspan="3">Transition</td></tr>
<tr><td>Iteration #0</td><td>Iteration #1</td><td>Iteration #2</td><td>Iteration #3</td><td>Iteration #4</td><td>Iteration #5</td><td>Sprint #1</td><td>Sprint #2</td><td>Sprint #3</td><td>Sprint #4</td><td>Sprint #5</td><td>Sprint #6</td><td>Sprint #7</td><td>Sprint #8</td></tr>
</table>

Figure 10: Hybrid Approach Conceptualization

## 5.3 Stage I Using RUP

The goal for the first stage was to learn about the organization and the environment where the application will operate, understand the user's needs and expectations, gather requirements, define the project scope, have an initial risk assessment, and establish an initial baseline for the software architecture.

### 5.3.1 Inception Phase

The main goal of the Inception phase was to have an agreement among stakeholders about what were the objectives of the project such as what would be or would not be included in the product. For this reason, it was important to understand the structure and dynamics of the organization as well as to learn how the application will be used and what the user's expectations were toward this. To meet this goal, the activities performed during this phase were to determine the scope of the product and share a vision with the most important requirements and constraints for the end product and to identify how the software solution will interact with the end user. Also, a preliminary software development and an initial list of risks were created at the beginning of this phase. These activities were performed through three iterations, which will be described with more detail in the following sections.

The output produced during these activities were the following artifacts largely subscribing to RUP artifacts: the business vision, the business use-case model, the

business rules, the domain model, a glossary, the product vision, an initial use case model, a software development plan, and the initial risks list. Figure 11 shows the activities and artifacts that guided the process during this phase.

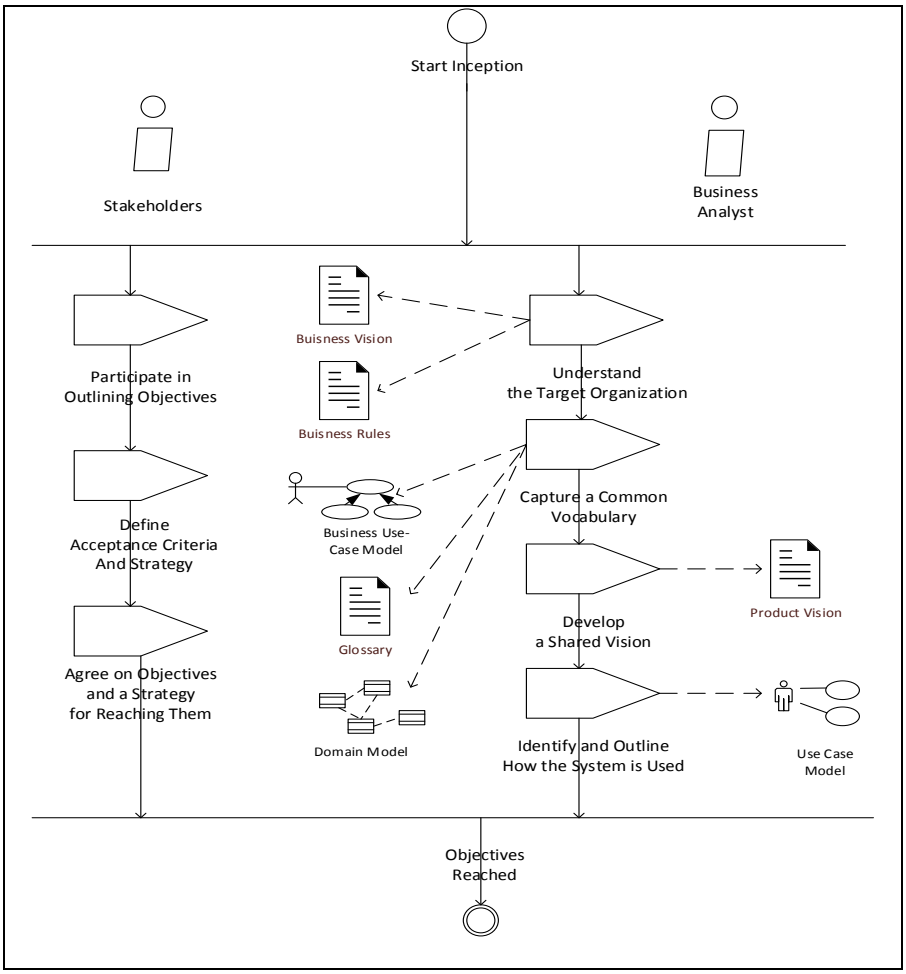



Figure 11: Inception Phase Workflow (Adapted from [ScrumUP14])

Iteration #0. The main purpose of this iteration was to craft an initial assessment of the project scope, risks, and organize the team essentially according to the RUP roles. Thus, after the initial presentation given by the director of the City of Jacksonville's Office of

Ethics, Compliance, and Oversight the high-level requirements were identified and the following activities were performed:

Identify Roles. As mentioned before, the software engineering professor served as the Project Manager for the development team. Therefore, the purpose of this activity was to identify the roles that the different team members, which include students only, would play while performing the required activities. These roles would be modified as team members would take on new roles through the project. Also, the team was required to identify a Team Leader and Backup and a Quality Assurance (QA) Manager. The Team Leader was responsible for bringing the team together, making sure that the activities for the iteration were completed, and submitting an executive summary for each iteration. This executive summary included the list of activities undertaken during the iteration, changes or revisions performed to any artifact, and comments about what worked and what did not during the iteration. The team QA Manager was responsible for ensuring that all the artifacts were properly reported, formatted, and uploaded to the CLM tool. Figure 12 shows a sample of the roles and responsibilities played during this project. The output for this activity was a team formation document in which it is described the roles and responsibilities for each member.

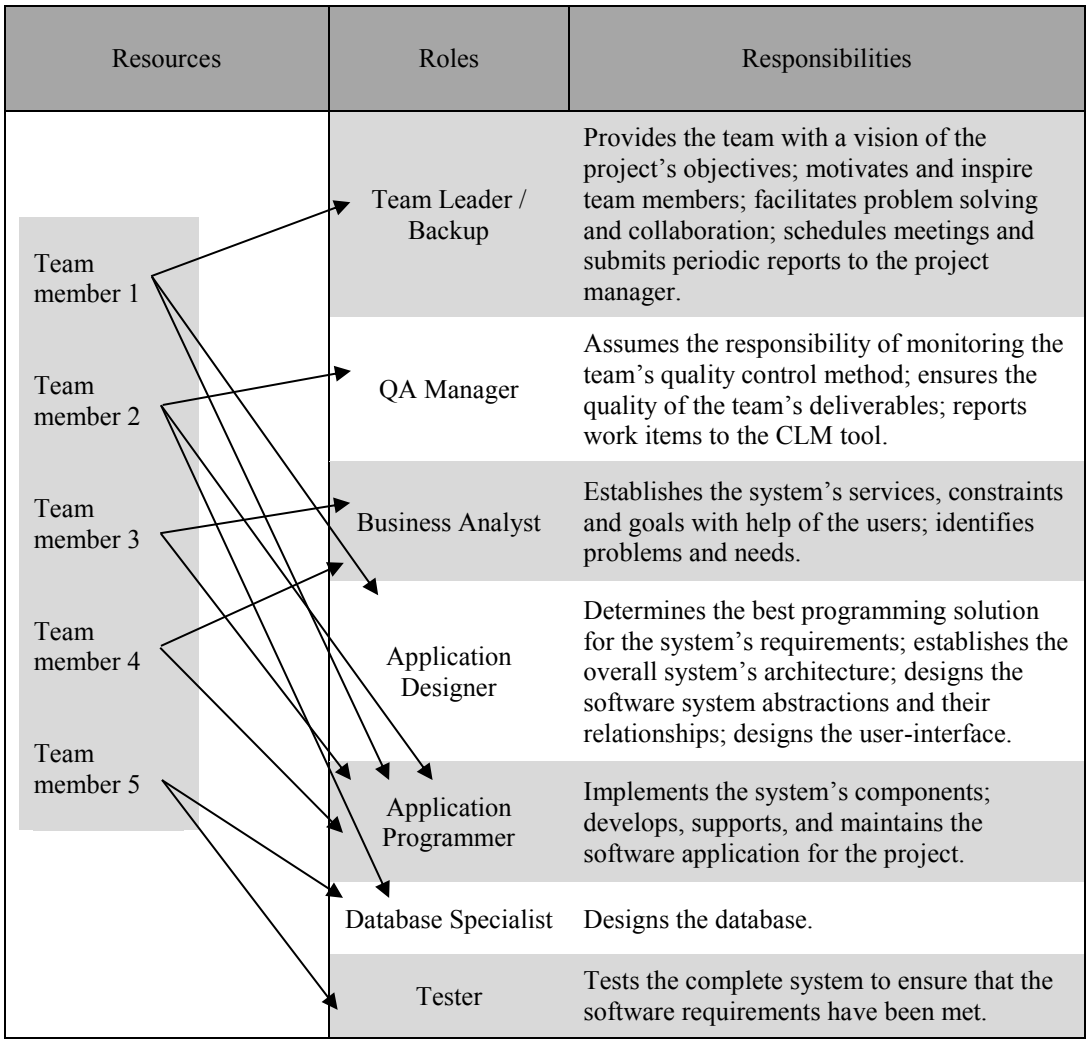



Figure 12: Team's Roles and Responsibilities (Adapted from [Kruchten03])

Develop a Software Development Plan. The main objective of this activity was to create an initial software development plan to identify the scope of the project, number of iterations, date lines for deliverables, and environment constraints for the new system.

Identify Risks. During the initial risk assessment of the project some risks were identified, quantified, and prioritized. This activity followed a model for judging risk assessment where the overall range for any item of risk equals between 0 and 5. This

overall range was obtained using the following formula, where the probability of this item of risk is multiplied by its potential impact. The probability of an item of risk to occur ranges from 0.0 to 1.0 and its potential impact ranges from 1 to 5.

$$Overall\ Range = Probability\ of\ Ocurrence\ \times\ Impact$$

Some of the major risks identified during this time were the technical risks with respect to the COJ IT Division limitations. This risk was identified with a magnitude of 4.5, which was very significant and needed to be addressed early in the process. One of the issues was that the development team was planning to use Microsoft technologies for the development of the application. Therefore, it was necessary to identify the consequences of this decision early in the process. This issue needed to be addressed early in the first meetings with the client and IT stakeholders. Also, another issue was that if the application were to "live" on the COJ website, it was essential to find what technical tasks were needed to be accomplished to have the finished product published to the website. To address these issues the development team needed to develop a strong relationship with the COJ IT staff to be able to deliver a product on time.

Other two items of risks were identified and documented in the list of risks artifact, which cover some important aspects for each item of risk such as the risk magnitude, impacts, mitigation strategy, and a contingency plan.

CLM Tool. During this iteration, the area for this project was setup in the CLM tool. For the first stage of the project every artifact produced, also called deliverable or work item, would reside in the Requirements Management (RM) area, which is part of the Rational Requirements Composer (RRC). This would allow that every artifact produced during each iteration be accessed with the latest version for further reference or update. Ideally, the client should be able to receive this information through the tool, but it was considered that this tool would be used primarily for the development team. However, this tool would allow the team to deliver the latest version of the artifacts to the clients when necessary. Once the artifacts for this iteration were produced, the team QA Manager reviewed them and uploaded them to the project's RM area, where they would be inspected by the team Project Manager.

By the end of this iteration the team had identified the major roles the team members would be playing during the full life cycle of the project, had made an initial assessment of the risks, had created an initial scope of the project which would grow as the team learned more about the requirements for the project, and had developed an initial software development plan.

Iteration #1. The main objective of this iteration was to understand the structure and dynamics of the organization within which the application would operate. Therefore, the activities performed during this iteration were centered on the RUP's Business Modeling discipline. Thus, as a first step to understand the organization and its current needs a meeting was setup. The purpose of this meeting was to identify all primary stakeholders

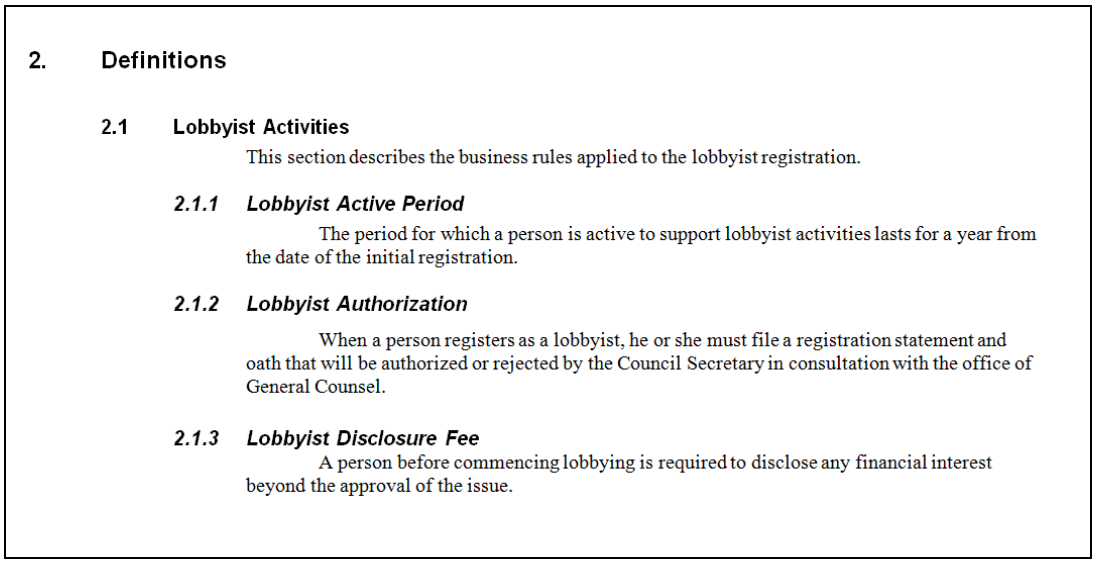
involved in the process as well as others stakeholders who will be affected by the new system.

Before the meeting, a questionnaire was created and sent to all the stakeholders involved in the project so they could have an idea about what was going to be covered during the meeting. The purpose of this questionnaire was to find detailed information about the current process, the users involved in the process, the environment where the new system will operate, the IT Division's requirements for the new system, and who will be the point of contact between the development team, the IT Division, and the Office of Ethics, Compliance, and Oversight.

During this meeting, the development team met with the staff that supported the lobbyist activities; the Director/Council Secretary who would ensure that the system met the needs for the registration and tracking of lobbyist activities and conformed to the Council Office requirements; and a representative from the COJ IT department. As a result of this meeting, the team was able to understand more about the Ethics Office and its relationship with other offices, the current process involved to track the lobbyist activities and its goals, what were the current needs, and how they expected the application to facilitate the process. Also, the development team was able to identify some of the technical requirements and standards that the application needed to meet to be able to live in the COJ web site. After this meeting, the development team was ready to analyze the information and the following artifacts were produced:

The Business Vision Document. On this document the development team was able to define the goals of the Office of Ethics and its environment and relationships with other offices, as well as to identify the stakeholders and level of interest in the project. Also, a description of the current process concerned to the lobbyist activities was elaborated and an initial description of how this process could be improved was provided.

The Business Rules Document. One of the objectives of this document was to describe the business rules for the entire organization. However, for this project the main purpose of this document was to identify the business rules that affected the lobbyist activities. The process of the lobbyist activities was divided into three groups: the lobbyist registration process, handling of the documentation produced during this process, and the disclosure of this information to the citizens. Thus, the business rule document was divided into these three sections that addressed the procedures and rules followed in this process. Figure 13 shows a sample of the business rules for the lobbyist registration process.



2. **Definitions**

**2.1 Lobbyist Activities**

This section describes the business rules applied to the lobbyist registration.

***2.1.1 Lobbyist Active Period***

The period for which a person is active to support lobbyist activities lasts for a year from the date of the initial registration.

***2.1.2 Lobbyist Authorization***

When a person registers as a lobbyist, he or she must file a registration statement and oath that will be authorized or rejected by the Council Secretary in consultation with the office of General Counsel.

***2.1.3 Lobbyist Disclosure Fee***

A person before commencing lobbying is required to disclose any financial interest beyond the approval of the issue.

Figure 13: Sample of the Business Rules Document

The Business Use-Case Model. The intended function of this model was to identify the business use-cases, roles, and deliverables with respect to the lobbyist activities. Thus, each business use case described a sequence of activities performed for this office that delivered a valuable result to the business actor. In this activity five business use-cases were identified: the lobbyist registration form, approval of the issues, update of the list, upload of the list, and the browsing of the records. For each business use-case a diagram, a description, goals, and a workflow of the process were provided. Figure 14 shows a sample of the business use-case for the lobbyist registration form.

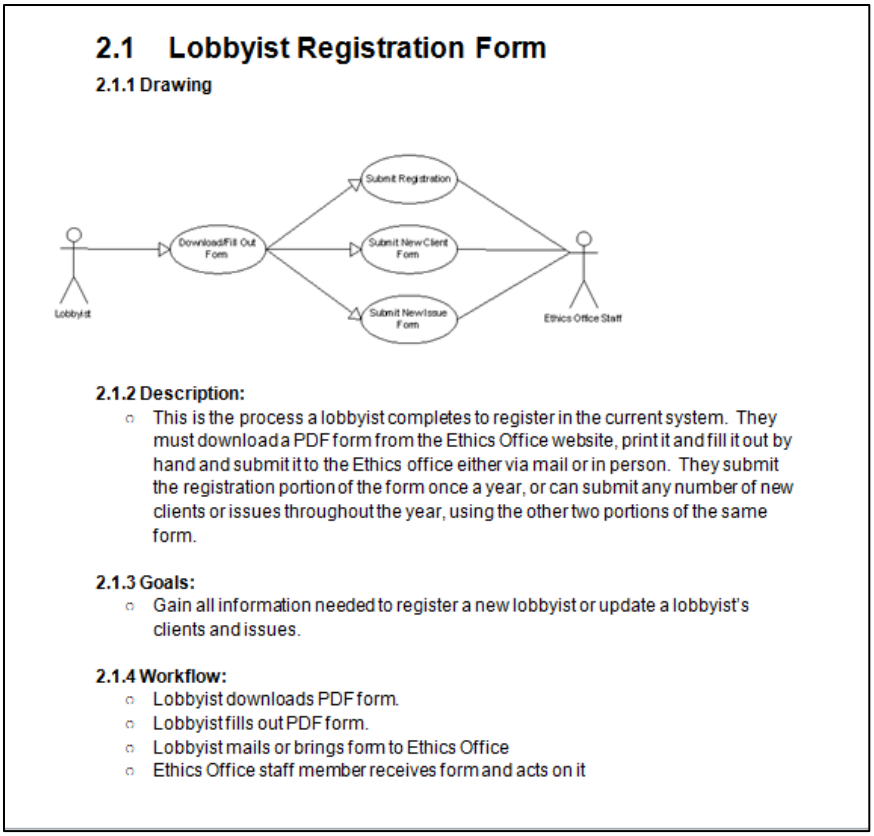



Figure 14: Sample of the Business Use-Case for the Lobbyist Registration Form

The Business Glossary. The business glossary contained the definitions of important terms related to the lobbyist activities. Some of these terms corresponded to the entities of the process that were graphically represented in the Domain Model during the next iteration. Figure 15 shows a sample of the business glossary.

| Term | Definition |
|---|---|
| Citizen | Any person accessing the publicly available portion of the application. |
| COJ Employee | An employee of the government accessing the system with administration capabilities. |
| Registration | A record filled out by a lobbyist and approved by an employee that ties a lobbyist to their client and to an active issue. |
| Lobbyist | An individual required to register with the government because they are lobbying government officials on behalf of another person or entity. |
| Client | The person or entity a lobbyist is representing. |
| Issue | The specific reason the lobbyist is working for their client. Each Issue requires a separate registration and is Active from registration until marked Inactive. |

Figure 15: Sample of the Business Glossary for the Lobbyist Activities

The Updated Risks List. After the meeting, the development team had a clearer understanding of the process involved in the registration and tracking of the lobbyist activities and also of requirements and constraints for the application. Thus, the development team was able to identify new potential risks, which were added to the risks list.

CLM Tool. Once the artifacts were produced, they were reviewed by the team QA Manager and uploaded to the Requirement Management area for further review by the team Project Manager.

At the end of this iteration the development team had a better understanding of the organization and its lobbyist processes, and the importance of this project for the Office of Ethics. This project was strategically important because it would help meet one of the goals of this office, which is to provide a more efficient and transparent service to its employees, lobbyists, and citizens.

Iteration #2. One of the purposes of this iteration was to continue with the business modeling to have a better understanding of the elements and their interactions with each other during the lobbyist process. Also, one of the goals for this iteration was to define the needs and requirements for the new system; thus, the requirements discipline played an important role during this iteration. The artifacts produced during these activities were the Domain Model and the Product Vision. In addition the Business Glossary was updated to add terms missed in the last iteration. A description of these artifacts is provided next:

The Domain Model. This model (Figure 16) was created to identify all the key entities involved in the lobbyist process. Also, this model helped illustrate the entities' attributes, the relationships among them, and their associations.

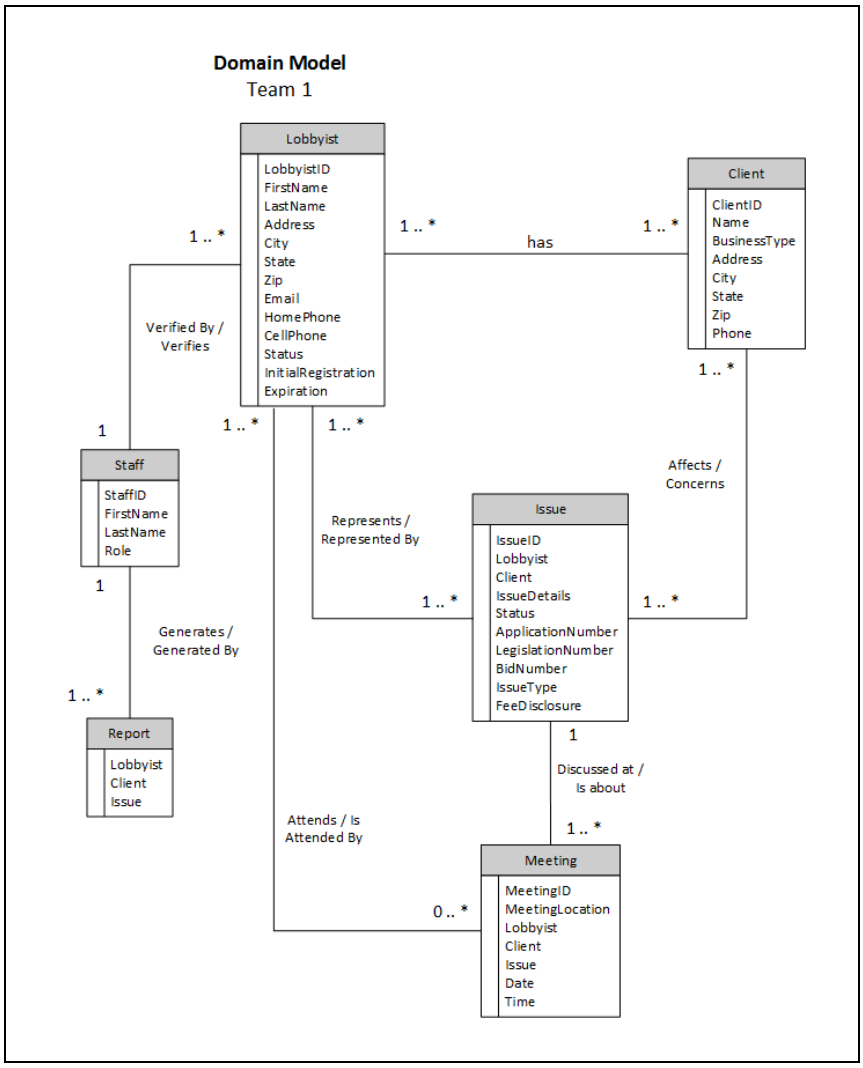



Figure 16: Domain Model for the Lobbyist Process

The Product Vision. Another essential artifact produced during this iteration was the Product Vision. The main objective of this artifact was to define the scope and to provide a complete vision of the system. The product vision document provided the problem statement and the product position statement. Also, it included a summary of the stakeholders involved in the project, a summary of the users of the system, a description of the current process, and the product perspective (Figure 17). In addition, this document listed at a higher level of abstraction the user needs of which a sample is shown in Figure

18 and the features required for the system. The list contained a description of the feature, the need addressed, its benefit, and its development risk (A sample of the feature list is shown in Figure 19). The list of features was especially important because it drove the development of the use cases for the next iteration.

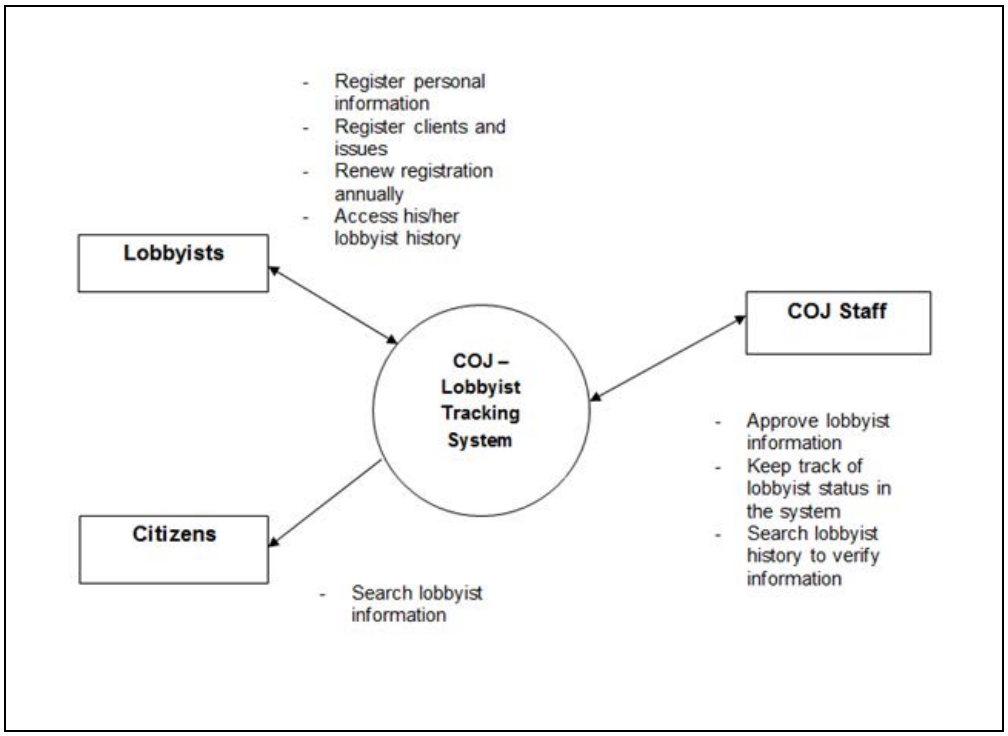



Figure 17: Product Perspective

| Ref# | Need | Priority | Current Solution | Proposed Solutions |
|---|---|---|---|---|
| N01 | A more efficient process to store lobbyist registration information. | High | Registration forms are kept in binders | A database system that keeps the lobbyist information. |
| N02 | More efficient process to update records of lobbyist activities | High | An Excel Spreadsheet contains the lobbyist name, the client name, the issues, and expiration date of the lobbyist. This information is updated manually when a new lobbyist registers and when a new client or issue is added. | A web application where lobbyists are able to register their information, clients and issues online. |
| N03 | The Ethics Office needs an easier way to upload lobbyist record information to the website | High | Through the City of Jacksonville website, under the City Council department there are PDF lists within the last five years and the Lobbyist registration forms. | A web application that dynamically updates lobbyist records on a publicly available website |

Figure 18: Sample of the User Needs

| Ref# | Feature | Need Addressed | Benefit | Development Risk |
|---|---|---|---|---|
| F01 | Lobbyist shall be able to create an account with a username and password | N01 | High | Medium |
| F02 | Lobbyist is required to enter their Florida ID / Driver's License number when registering. | N01 | High | Low |
| F03 | Lobbyists shall be able to fill out their personal information. | N01 | High | Low |
| F04 | Lobbyists shall be able to register their clients. | N02, N03 | High | Medium |
| F05 | Lobbyists shall be able to register issues for their clients | N02, N03 | High | High |
| F06 | An issue can be marked as inactive | N02, N03 | High | Low |
| F07 | When a new Lobbyist issue is created it must be approved by Ethics Office staff. | N05 | High | Medium |

Figure 19: Sample of the List of Features

The Business Glossary. This artifact was updated to include some of the definitions of the attributes included in the Domain Model.

CLM Tool. Once the artifacts were created, they were reviewed and uploaded to the RM area by the team QA Manager for further inspection by the team Project Manager.

At the end of this iteration the development team had a better understanding of the elements involved in the lobbyist process, the user needs, and a list of the system's features needed to accommodate the user needs.

Iteration #3. One of the fundamental characteristics of RUP is that it is a use-case-driven approach. The use cases were created to describe the functionality of the system based on the features list captured in the Product Vision. Thus, the main activity for this iteration was to develop a use-case model that contained the main interactions of the system with the actors. Another activity that took place during this iteration was the creation of the traceability matrices. The outputs of this iteration were the following artifacts:

The Use-Case Model. The first step of this activity was to create a set of façade use cases. This set of façade use cases was created as a placeholder of the main interactions of the system with the actors. Basically, the façade use cases contained a use-case number, a use-case name, the actors, a brief description of the use case, and assumptions for this event. Figure 20 shows a sample of a façade use case. During this activity fifteen use cases were identified.

| | | |
|---|---|---|
| **Use Case Number:** | UC13 | |
| **Use Case Name:** | Generate Lobbyist Activity Report | |
| **Actor (s):** | Citizen | |
| **Maturity: (Façade/Focused/…** | Facade | |
| **Summary:** | A citizen generates a report of lobbyist activity | |
| **Basic Course of Events:** | **Actor Action** | **System Response** |
| **Triggers:** | | |
| **Assumptions:** | The lobbyist the citizen is searching for has registered and is active in the system. | |
| **Preconditions:** | | |
| **Post Conditions:** | | |
| **Activity Diagram:** | | |
| **Author(s):** | Team member's name | |
| **Date:** | 11/02/2012 | |

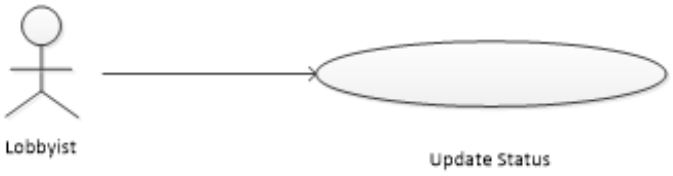
Figure 20: A Façade Use Case to Generate a Report on the Lobbyist Activities

The second step was to create a use-case index to keep track of the developed use cases. This index listed the basic information found in the façade use cases and the level (façade, filled or focused) and the date of the last update as shown in Figure 21.

| Use Case Number | Use Case Name | Description | Actors | Level | Date of Update |
|---|---|---|---|---|---|
| UC01 | Create new account | Lobbyist will be able to create a new account | Lobbyist | Façade | 10/29/2012 |
| UC02 | Renew account | Lobbyist will be able to renew their account | Lobbyist | Façade | 10/29/2012 |
| UC03 | Log in account | Lobbyist will be able to access their account by ID number | Lobbyist | Façade | 11/1/2012 |
| UC04 | Account expiration notification | Lobbyist will be notified when their account is about to expire | Lobbyist | Filled | 11/1/2012 |
| UC05 | Access account history | Lobbyist will be able to access their account history(Issues) | Staff/ Lobbyist | Filled | 10/29/2012 |
| UC06 | Add Issue | Lobbyist will be able to add a new issue | Lobbyist | Façade | 10/29/2012 |

Figure 21: A Sample of the Use-Case Index

The third step was to create the filled use cases; however, before starting this activity, the set of façade use cases was reviewed. After reviewing the use cases, the development team found that the interaction of the user with the system was duplicated in two use cases so that those two cases were removed and the list was reduced to thirteen use cases. Following this revision, the filled use cases were created and for each use case some items were added such as a use case diagram, the basic course of events, the triggers, and the pre and post-conditions as shown in Figure 22. Once the filled use cases were created a peer review was performed and some use cases were modified according to the feedback obtained.




| | | |
|---|---|---|
| **Use Case Number:** | 12 | |
| **Use Case Name:** | Update Status | |
| **Actor (s):** | Lobbyist | |
| **Maturity: (Façade/Focused/...** | Filled | |
| **Summary:** | This use case is started when a lobbyist wants to update his/her status (active/inactive). | |
| **Basic Course of Events:** | **Actor Action**<br>1. This use case is started when a lobbyist selects to update his/her status from his/her dashboard.<br>3. The lobbyist updates his/her status.<br>5. The lobbyist confirms that the change of status is correct.<br>7. This use case concludes when the lobbyist receives visual confirmation of the updates. | **System Response**<br>2. The system displays the lobbyist's current status and prompts the lobbyist if he/she wants to change it.<br>4. The system prompts for confirmation of the change of status.<br>6. The system saves the current status and confirms the change of status was successful. |
| **Alternative Paths:** | | |
| **Exception Paths:** | | |
| **Extension Points:** | | |
| **Triggers:** | The lobbyist would like to update his/her current status. | |
| **Assumptions:** | Lobbyist is aware of the steps for updating his/her status. | |
| **Preconditions:** | The system is functioning properly.<br>The lobbyist has created an account and has a current status (active/inactive). | |
| **Post Conditions:** | The lobbyist was able to update his/her status. | |
| **Reference: Business Rules:** | | |
| **Reference: Risks** | | |
| **Author(s):** | Team member's name | |
| **Date:** | November 3, 2012 | |

Figure 22: Filled Use Case to Update the Lobbyist Status

Traceability Matrices. During this activity two traceability matrices were created, one was created to map the user needs to the features (Figure 23) and the second one to map the features to the use cases (Figure 24).

| | Needs | | | | |
|---|---|---|---|---|---|
| **Features** | N01 | N02 | N03 | N04 | N05 |
| F01 | X | | | | |
| F02 | X | | | | |
| F03 | X | | | | |
| F04 | | X | X | | |
| F05 | | X | X | | |
| F06 | | X | X | | |
| F07 | | | | | X |
| F08 | | | | | X |
| F09 | | | | | X |
| F10 | X | | | | |
| F11 | X | | | | |
| F12 | X | | | | |
| F13 | | X | X | | |
| F14 | | X | X | | |
| F15 | | X | X | | |
| F16 | | X | X | | |
| F17 | | X | X | | |
| F18 | | | | X | |
| F19 | | | | X | |
| F20 | | | | | X |
| F21 | | | | | X |
| F22 | X | | | | |

Figure 23: Traceability Matrix to Map Features to Needs

| | Use Cases | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Features** | UC01 | UC02 | UC03 | UC04 | UC05 | UC06 | UC07 | UC08 | UC09 | UC10 | UC11 | UC12 | UC13 | UC14 | UC15 |
| F01 | X | | | | | | | | | | | | | | |
| F02 | X | | | | | | | | | | | | | | |
| F03 | X | | | | | | | | | | | | | | |
| F04 | | | | | | | | | | X | | | | | |
| F05 | | | | | | X | | | | | | | | | |
| F06 | | | | | | | | | | | | X | | | |
| F07 | | | | | | | X | | | | | | | | |
| F08 | | | | | | | X | | | | | | | | |
| F09 | | | | | | | X | | | | | | | | |
| F10 | | X | | | | | | | | | | | | | |
| F11 | | | | X | | | | | | | | | | | |
| F12 | | X | | | | | | | | | | | | | |
| F13 | | X | | | | | | | | | | | | | |
| F14 | | | | | | | | | | | X | | | | |
| F15 | | | | | | | | | | | | | | X | |
| F16 | | | | | X | | | | | | X | X | | | |
| F17 | | | | | | | | | | | | X | | | |
| F18 | | | | | | | | | | | | | X | | |
| F19 | | | | | | | | Removed as a duplicate of UC07 | Removed as a duplicate of UC01 | | | | X | | |
| F20 | | | | | | | X | | | | | | | | |
| F21 | | | | | | | | | | | | | | X | |
| F22 | | | | | | | | | | | | | | | X |

Figure 24: Traceability Matrix to Map Features to Use Cases

CLM Tool. Once the artifacts for this iteration were finished, the team QA Manager reviewed them and uploaded them to the RM area to be reviewed and obtain feedback from the team Project Manager.

By the end of this iteration the development team had created a set of use cases that described the main interactions of the system with actors according to the requirements. Also, the team had ensured that the use cases mapped to the features and the features mapped to meet the user needs.

Until this point the team had fulfilled most of the Inception phases objectives, which aim to define the project scope, understand and agree on the user's needs and features for the product, and have an initial assessment of the risks.

### 5.3.2 Elaboration Phase

One of the first activities during this phase was to finish up the elaboration of the use cases and to have a better understanding of the most critical use cases that would drive the software architecture. This was important so that the team could elaborate an initial baseline of the architecture. For this purpose, one of the first steps was to create an initial design of the database and the construction of a prototype that contained the main interaction of the system with the users.

These activities were performed through two iterations, which will be described with more detail in the following sections.

Iteration #4. The analysis and design discipline played a part in this iteration as the development team was completing the use cases. Therefore, the first step for this iteration was to complete the use cases. After this activity was completed, the next step was to translate the requirements into a design that described how to implement the system. One of the first steps into the analysis and design was to create an initial database design based on the requirements and constraints for the system and a prototype that contained the main functionality of the system as described in the use cases. However, in this iteration the development team focused on completing the filled use cases, designing the initial database model, and reviewing and preparing the artifacts into a package to be submitted to the stakeholders for their revision and feedback. The package was planned to be submitted along with the user-interface prototype, which was planned to be ready for the next iteration. The outputs for this iteration were:

Complete Filled Use Cases. The goal of this activity was to add the alternate paths and exceptions to each use case. These two events refer to the interaction of the system and the actors when the basic course of events is not followed as explained in [Kulak03]. Therefore, the exceptions were used to describe the steps to execute when an error has occurred and the alternatives to describe a path that is not the most likely to occur. After the exception and alternatives paths were added to each use case, there was a peer review to validate the completeness of the use cases and to find any vague points that would

need clarification with the stakeholders. After the review the development team found that the interaction described in one of the used cases would be part of another use case, therefore, this use case was removed and the list was reduced to 12 use cases. Figure 25 shows an example of a filled use case with exceptions and alternatives paths.


| | | |
|---|---|---|
| **Use Case Number:** | UC6 | |
| **Use Case Name:** | Add Issue | |
| **Actor (s):** | Lobbyist | |
| **Maturity: (Façade/Focused/…** | Filled | |
| **Summary:** | *Lobbyist* will be able to add a new *issue* to the system. | |
| **Basic Course of Events:** | **Actor Action** | **System Response** |
| | The actor (*Lobbyist*) selects to add *information* about an *Issue* from his/her personal dashboard. | |
| | | Form to add *issue* is displayed. |
| | The actor (*Lobbyist*) enters issue *information* and links issue to a *client*. | |
| | The actor (*Lobbyist*) selects to save. **A1** | |
| | | The system validates and logs the *information* to the database and *issue* is set to "Pending" status. **E1** |
| | | The system emails the *COJ Employee* to inform that a new *issue* has been submitted. |
| | | The system redirects to the personal lobbyist dashboard. |
| **Alternative Paths:** | A1. The actor (*Lobbyist*) decides to cancel the changes instead of saving them. In this case, the system will not save the changes, and display a message stating the changes were not saved. The flow of events ends here. | |
| **Exception Paths:** | E1. The actor (*Lobbyist*) submits the issue *information* without completing all the required fields. The system will alert the actor of the missing data and redisplay the page so the actor (*Lobbyist*) can correct the *information*. | |
| **Extension Points:** | | |
| **Triggers:** | The actor (*Lobbyist*) selects to add a new issue. | |
| **Assumptions:** | The actor (*Lobbyist*) is aware of the steps for adding an *Issue* and has successfully logged in into his account. | |
| **Preconditions:** | The actor (*Lobbyist*) has created an account and account is active (not expired). | |
| **Post Conditions:** | A new *issue* is added to the system. | |
| **Activity Diagram: See diagram** | | |
| **Author(s):** | Team member | |
| **Date:** | 11/12/2012 | |

Figure 25: Filled Use Case with Alternative and Exception Paths

Database Design. An initial database (Figure 26) was created based on the elements that the development team had identified from the domain model, features, and use cases.

CLM Tool. Once these artifacts were completed and reviewed by the team and the team QA Manager, they were uploaded to the RM area to be inspected by the team Project Manager.

After the team Project Manager's inspection and feedback, a package was prepared to be submitted to the COJ stakeholders with the purpose of obtaining some feedback.

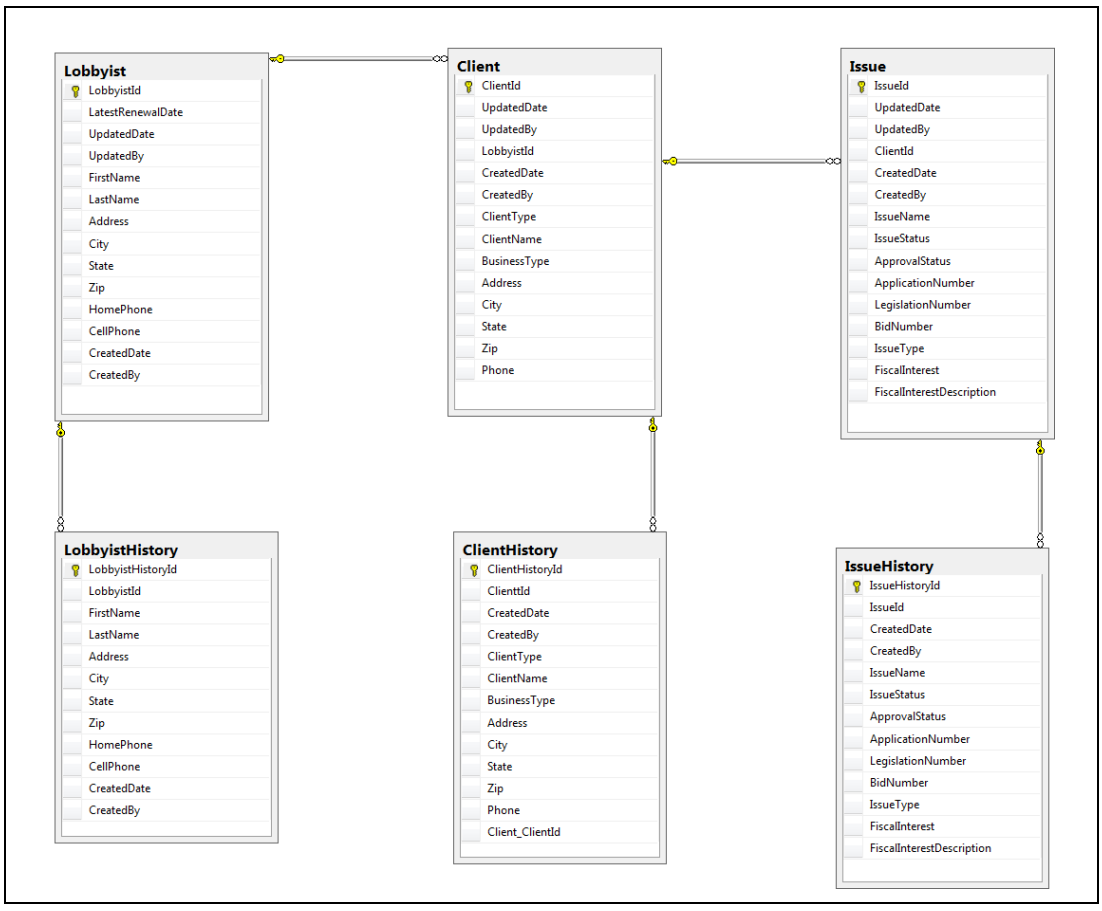



Figure 26: Initial Database Design for the Lobbyist System

Iteration #5. This iteration had a short duration and its goal was to design and produce a user-interface prototype that had the main interactions of the system with the actors based on the functionality described in the use cases and the list of features.

At the end of this iteration the user-interface prototype was presented to the team Project Manager and after his approval, the user-interface prototype along with the package prepared in the last iteration was submitted to the COJ stakeholders.

After this iteration the development team was preparing to enter the second phase of the project. For this purpose a meeting was scheduled to meet with the COJ stakeholders to demonstrate the user-interface prototype, clarify certain points, and obtain feedback. In addition, this meeting would be important to prioritize the customer's needs in the product backlog, which would be the starting point for the next phase.

### 5.3.3 Stage I Artifacts

This section summarizes the RUP artifacts produced or updated during the Inception and Elaboration phases during this stage of the project as shown in Table 3.

These artifacts were created by following RUP templates which were adjusted to the project. Also, they provided the means to document the process without producing long tedious documentation.

| Artifacts | RUP Phases and Iterations |
|---|---|
| Software Development Plan | Inception, iteration #0 |
| Initial Risks List | Inception, iteration #0 |
| Business Vision | Inception, iteration #1 |
| Business Rules | Inception, Iteration #1 |
| Business Use-Case Model | Inception, iteration #1 |
| Business Glossary | Inception, iteration #1 |
| Domain Model | Inception, iteration #2 |
| Product Vision | Inception, iteration #2 |
| Use-Case Index | Inception, iteration #3;<br>Elaboration, iteration #4 |
| Use-Case Model | Inception, iteration #3;<br>Elaboration, iteration #4 |
| Traceability Matrices | Inception, iteration #2 and #3;<br>Elaboration, iteration #4 |
| Database Model | Elaboration, Iteration #4 |
| User-Interface Prototype | Elaboration, Iteration #5 |

Table 3: Stage I Artifacts

By the end of this stage, the development team had been able to complete the selected activities and artifacts and had a clearer idea of the user's requirements, the vision of the product, and had established the baseline architecture to support this vision.

## 5.4 Stage II Using Scrum

One of the goals during this stage was to ensure that the development team had a clear and detailed understanding of the user's requirements, so that the development team could validate and baseline the architecture of the system and could refine the development plan to reflect more precisely what needed to be done until the delivery of the project. Therefore, it was important to have an executable architecture where some of the risks had been addressed so that the team could concentrate on implementing the rest of the functionality and performing the necessary tasks to deliver the project.

To accomplish these tasks, the Scrum practices were integrated to the process during this stage and the Rational Team Concert was used as a collaborative tool so that the team would plan, organize, and collaborate to reach a tangible goal.

### 5.4.1 Elaboration Phase

At the end of the first stage the development team had accomplished some specific goals and created some artifacts that helped the development team have a better understanding of the organization, user needs, expectations for the new system, and also to establish an initial baseline of the software architecture. Also, these artifacts were sent to the COJ stakeholders for their revision.

The goals for this phase during the second stage of the project were to validate and baseline the architecture and to create a more fine-grained development plan. Also, to prepare the development environment, process, and tools.

To meet these goals, it was essential to get feedback from the COJ stakeholders to ensure the success of the project. Also, as the Scrum process indicates, it was necessary to create the product backlog with the list of features and prioritize them according to the Office of Ethics, Compliance and Oversight needs. To accomplish this objective a third meeting was set up to initiate the process.

The attendees for this meeting were the project sponsor, the project sponsor's IT consultant, a lobbyist, and the development team. In this meeting the development team obtained some feedback with respect to the artifacts and also had the opportunity to demonstrate the user-interface prototype to the COJ stakeholders. The demonstration of the user-interface prototype elicited feedback that included the addition of some nice-to-have features, some small changes so that the information requested from the lobbyists could comply with the city council lobbyist law, and other additional changes to improve the lobbyist registration process.

During this meeting the project sponsor who would play the role of the product owner helped identify the features that had more priority for her office. This helped the development team prioritize the list of features that would form the product backlog. Scrum rules indicate that the product owner should be responsible for the product

backlog however, in this case due to time conflicts from both parties, the development team took responsibility for this.

Therefore, after the meeting, the development team created the product backlog using the list of features, which was updated first to reflect the COJ stakeholders' feedback. Later, these features were prioritized in two ways, first, they were ordered as they were going to be developed and second they were assigned a priority of high, medium, or low. High priority was assigned to those features that delivered the main functionality, medium priority was assigned to those features that give detail to the high priority features, and low priority was assigned to the nice-to-have features.

After the product backlog was prioritized, the development team began to estimate the effort needed to implement each feature in the product backlog. This effort estimate was given in story points even though this term may be confusing since the development team was working with features instead of user stories. This effort estimate was based on the Fibonacci series; thus, the development team assigned 3, 5, 8, or 13 story points to each feature depending on the effort. For example, 3 meant the least effort and 13 the greatest effort needed. Also, the development team ensured that each feature was traced to a use case. The development team created two versions of the product backlog. The first version showed the priority level for each feature/activity within a sprint and the other version showed the estimate effort in story points (see Figure 27 and Figure 28).

| Feature # | Features Description | Use Case# | Priority | Sprint # |
|---|---|---|---|---|
| F01 | Lobbyist shall be able to create an account with a username and password | UC1-Create New Account UC3-Login to Account | High | 1 |
| F02 | Lobbyists are required to enter their Florida ID / Driver's License number when registering | UC1-Create New Account | High | 1 |
| F03 | Lobbyists shall be able to fill out their personal information | UC1-Create New Account | High | 1 |
| F16 | The system must allow lobbyists to update their registration information at any time | UC11-Update Profile | Medium | 1 |
| F17 | The system shall allow lobbyists to update their active status to inactive status if they stop lobbyist activities | UC11-Update Profile | Medium | 1 |
| F31 | Each Lobbyist should check a box to certify that they either do or do not have any financial interest while representing an issue | UC1- Create New Account & UC2-Renew Account | Medium | 1 |
| F32 | Tooltip and description for each field during the registration(lobbyist, client, and issues) | UC1-Create Account, UC6-Add Issue, UC10-Add Client, UC11-Update Profile | Low | 1 |
| F04 | Lobbyists shall be able to register their clients. | UC10-Add Client | High | 2 |
| F05 | Lobbyists shall be able to register issues for their clients | UC6- Add Issue | High | 2 |
| F06 | An issue can be marked as inactive | UC11- Update Profile | Medium | 2 |
| F13 | During re-registration the system should ask lobbyists which clients are expired and which are still active | UC2-Renew Account & UC11-Update Profile | Medium | 2 |

Figure 27: Sample of the Product Backlog with Priorities Levels

| Feature # | Features Description | Use Case# | Story points | Sprint # |
|---|---|---|---|---|
| F01 | Lobbyist shall be able to create an account with a username and password | UC1-Create New Account UC3-Login to Account | 8 | 1 |
| F02 | Lobbyists are required to enter their Florida ID / Driver's License number when registering | UC1-Create New Account | 3 | 1 |
| F03 | Lobbyists shall be able to fill out their personal information | UC1-Create New Account | 8 | 1 |
| F16 | The system must allow lobbyists to update their registration information at any time | UC11-Update Profile | 8 | 1 |
| F17 | The system shall allow lobbyists to update their active status to inactive status if they stop lobbyist activities | UC11-Update Profile | 3 | 1 |
| F31 | Each Lobbyist should check a box to certify that they either do or do not have any financial interest while representing an issue | UC1- Create New Account & UC2-Renew Account | 5 | 1 |
| F32 | Tooltip and description for each field during the registration(lobbyist, client, and issues) | UC1-Create Account, UC6-Add Issue, UC10-Add Client, UC11-Update Profile | 8 | 1 |
| F04 | Lobbyists shall be able to register their clients. | UC10-Add Client | 5 | 2 |
| F05 | Lobbyists shall be able to register issues for their clients | UC6-Add Issue | 8 | 2 |
| F06 | An issue can be marked as inactive | UC11-Update Profile | 8 | 2 |
| F13 | During re-registration the system should ask lobbyists which clients are expired and which are still active | UC2-Renew Account & UC11-Update Profile | 8 | 2 |

Figure 28: Sample of the Product Backlog with Story Points

After this activity, the development team elaborated a development plan based on the product backlog and other activities needed to accomplish the project. The development plan included the number of sprints undertaken during this stage of the project with a

detailed plan for the first sprint and a decreasing level of granularity for the subsequent sprints. This document also described the development environment for the application.

According to the development plan, the second stage was organized to have five sprints and two releases. The first release was planned at the end of the second sprint and the second release was planned at the end of the fifth sprint.

Since the project was executed as a part of the Software Engineering course, the Scrum ceremonies such as the Sprint Planning Meeting, Daily Scrum, the Sprint Review, and the Sprint Retrospective were adapted to the course. Therefore, a time was assigned during the class for team meetings and working time. The Daily Scrum meetings for the development team were at the beginning or before the class started. During this time the development team talked about their progress and/or difficulties found, and what was left to do for the rest of the sprint. Later, this was communicated to the professor, who during this stage played the role of the Scrum Master. At the end of each sprint the team leader wrote a retrospective summary about what the team achieved, any difficulties found and how they were going to be solved.

Sprint #1. At the beginning of the sprint the development team revisited the software development plan and the product backlog to select the features that were going to form the sprint backlog. Also, some of the work items or tasks included in the sprint backlog included the user interface tests and test scenarios from the use cases. Also, the development team needed to set up the development environment for the application, and

to create a project area in the Rational Team Concert (RTC) under Change Management. The work items undertaken during this sprint were:

Sprint Backlog. At the beginning of the sprint this artifact was formed with some of the highest priority features in the Product Backlog and other tasks that the development team needed to complete such as creating the project area in the RTC, setting up the development environment, setting up the application visual style, creating the manual test scripts, defining the validation criteria for the implemented features, and updating the documentation.

RTC. During this activity the development team set up the project under the Change Management area and created the product backlog, the sprint backlog as well as the project timelines as shown in Figures 29, 30, and 31. Also, this area served as a repository for any documents created during this stage of the project. Further, the development team would use the dashboard to have visibility of the team progress and productivity through some charts such as the team velocity, current sprint burndown, and the release burndown.

One more activity under RTC was to set up the development environment. However, it was not possible to upgrade RTC to make it compatible with Visual Studio 2012.

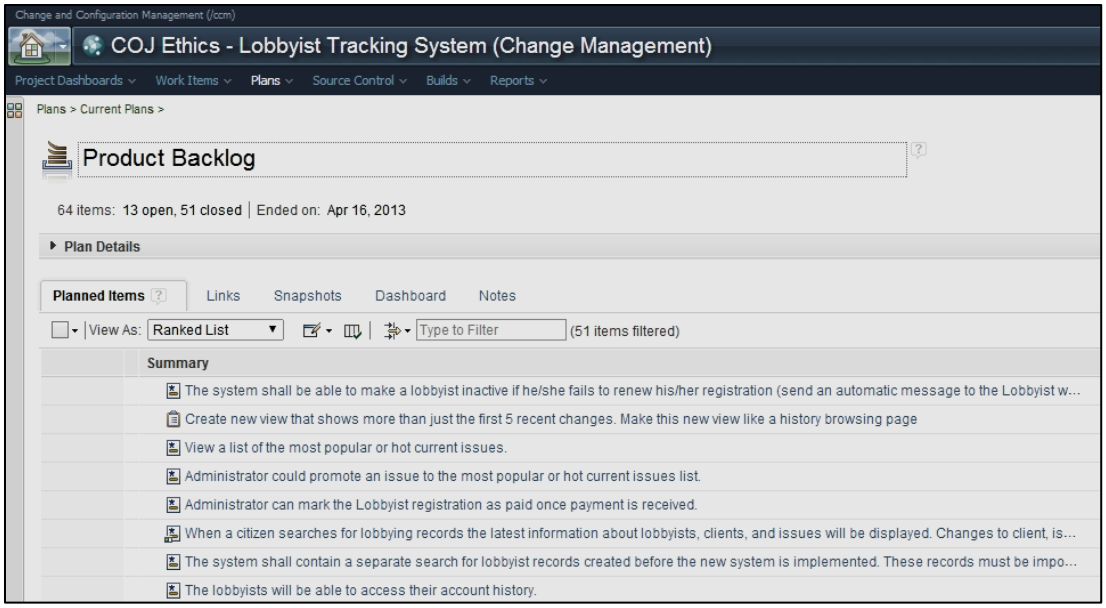



Figure 29: The Product Backlog under RTC

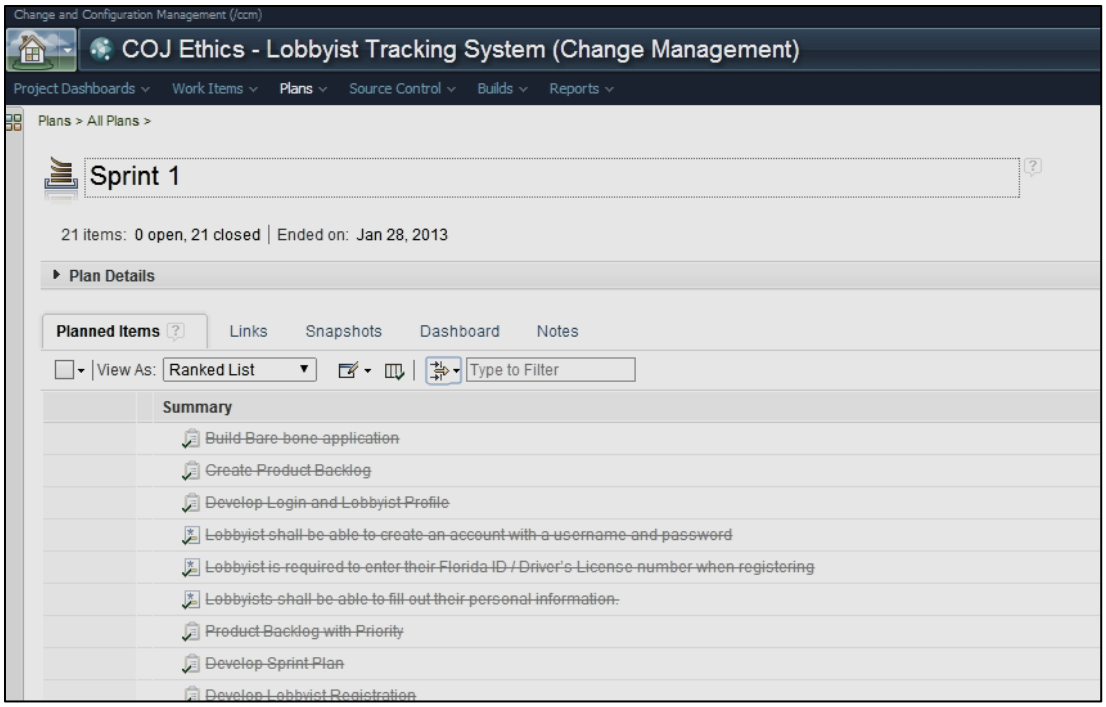



Figure 30: The Sprint Backlog under RTC

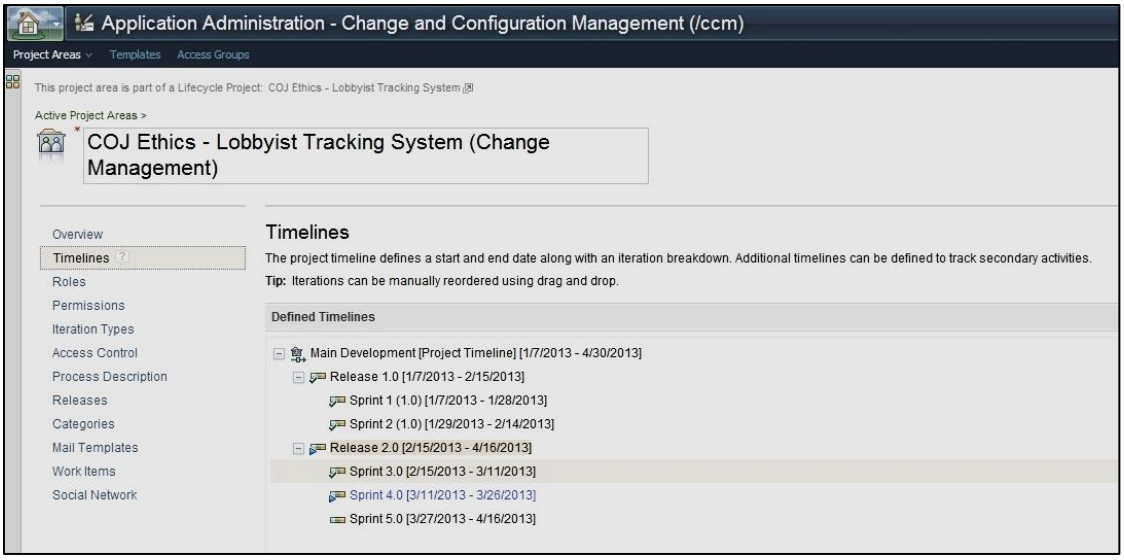



Figure 31: Project Timelines under RTC

Development Environment Set up. The project environment was set up using Team Foundation Server and Visual Studio 2012. The solution was uploaded to Team Foundation Server, which served as the version control system and code repository.

Delivered Increment. At the end of this sprint the development team had created an initial visual style for the application based on the COJ website and had implemented some basic functionality for the features undertaken in this sprint. These features were not finalized because the development environment set up took longer than planned. Therefore, the completion of these features was moved to the next sprint.

Testing Activities. For this sprint some basic manual test scripts were created to test the functionality of the product increment. These manual test scripts outlined the necessary steps to validate the output correctness of the increment based on an input (Figure 32).

| SL.NO | TEST SCENARIO | DESIGN STEPS | DESCRIPTION | EXPECTED RESULT |
|---|---|---|---|---|
| 1 | Last/ First Name | Step 1 | Lobbyist enters the First name and Last name in "firstname" / "Last name" | Information should be filled successfully in the First name/Last name field. |
| 2 | Username | Step 1 | Lobbyist enters the Username in the "Username" field. | Information should be filled successfully for the Username field. |
| | | Step 2 | Lobbyist enters the Username in the "Username" field which already exists | Dsiplays error message" Usename already exists,please enter again." |
| 3 | Password | Step 1 | Lobbyist enters Old Password in the "Old password" field | Information should be filled successfully for the password field. |
| | | Step 2 | If Lobbyists enters Incorrect Old passsword. | Prompt "Wrong Password. Please enter again" |
| | | Step 3 | Lobbyist enters new Password in the "New password" field | Information should be filled successfully for the password field. |
| | | Step 4 | If Lobbyist enters password which doesn't match requirements. | Dsiplays error message" Password requirements does not match , please try |
| | | Step 5 | Lobbyist retypes the password in "Confirm password" field | Information should be filled successfully for the Confirm password field. |
| | | Step 6 | Lobbyist enters wrong password in the "Confirm password" field. | Displays error message" Password does not match , please try again" |

Figure 32: Manual Test Scripts

Documentation Update. Also, during this sprint the list of features in the product vision was updated to reflect the new requested features originated from the stakeholders' feedback. In addition, the traceability matrices were updated to ensure that the new features mapped to the user needs and the use cases.

Retrospective. At the end of this sprint the development team had accomplished all of the tasks except for completing all the functionality of the committed features. However, the development team was confident that the functionality of these features would be finalized during the next sprint.

### 5.4.2 Construction Phase

During the past sprint the development team had met with the COJ stakeholders and had demonstrated the user-interface prototype which had elicited good response from them. As a result of this meeting there were some small requests for changes and additions of

nice-to-have features and the development team was able to add more detail to the development plan for the project. Thus, most of the activities for the next sprint would be focused on implementing the main functionality of the system and any new requests.

Sprint #2. The goal for this sprint was to deliver the first release. Thus, one of the first activities carried out during this sprint was to add more detail to the development plan about the tasks needed to be carried out in this sprint. Also, the plan for the next sprint was updated during this time. The additional activities performed in this sprint were:

Sprint Backlog. This artifact was created under RTC and contained all the tasks for the new features and the ones needed to complete the features from last sprint. Also, other tasks that were included were the correction of bugs found during the testing activities and the initial elaboration of the user guide.

RTC. This tool helped maintain up-to-date record of the completed features in the product backlog. Also, it helped visualize the progress of the work items in the sprint backlog because it was easier to track which activity had started, which one was completed, who has in charge of it, and the time it took to complete it. In addition, any document produced from this activity could be linked to it. Moreover, it helped the Scrum Master evaluate the performance of the development team. However, this tool was generating neither the burndown chart nor the velocity chart. Therefore, these artifacts were created manually in an Excel document until the problem was solved. The

development team was able to solve the issue for the burndown chart in the next sprint, but the issue related to the velocity chart was not solved.

Testing Activities. The development team continued elaborating the manual test scripts for the new features implemented during this sprint. Once the new functionality was finished, the regression testing was performed to ensure the correctness of the outputs. The bugs found at this time were corrected before the first release was deployed.

User Guide. By the end of this sprint the team had created a user guide that contained screen-shots of the user interface and step-by-step instructions of the current functionality of the application. This document was ready to be sent to the stakeholders along with the first release.

First Release. The purpose of the first release was to show to the COJ stakeholders the advances the development team had achieved so far. Also, this first release would provide an opportunity for the COJ stakeholders to try and test the application and provide new feedback. With this purpose, the first release was deployed to an area of the development server so that it could be easily accessed by the COJ stakeholders.

Retrospective. At the end of this sprint most of the features had been completed and some had been left with some basic functionality, which needed to be polished in the following sprint. Nevertheless, the development team had achieved the first milestone, to deliver

the first release. In addition, new test scripts for the new functionality had been created and tested and a user guide had been created to include the completed features.

Sprint # 3. The development team continued to implement the following features from the product backlog while awaiting the COJ stakeholders' feedback. The following work items were carried out as part of this sprint:

Sprint Backlog. The development team committed to the next features from the product backlog and added some work items related to the testing and documentation activities.

RTC. The development team had found that in order to generate the burndown chart, it needed to create tasks as child elements to the features. Thus, the sprint backlog was created with these tasks and other work items needed during this sprint. However, the development team was not able to solve the problem of generating the velocity chart.

Testing Activities. Some test scenarios and a traceability matrix were elaborated to map the test scripts and test the current functionality of the application. The bugs found were added as tasks in the RTC to be corrected in the next sprint.

Delivered Increment. At the end of this sprint the development team had added new functionality that was added to the first release and deployed to the development server for the COJ stakeholders' review and feedback.

User Guide. The elaboration of the user guide continued to cover the new completed features.

Retrospective. During this sprint only one task needed to be pushed to the next sprint since it took longer than the estimated effort. Also, the new functionality had been added to the deployment server so that it could be tested by the COJ stakeholders.

Sprint #4. At the beginning of this sprint the development team together with the Scrum Master took a decision based on the features left in the product backlog and the velocity of the team. The decision was that the development team was going to be able to deliver a functional system that included all the core requirements as part of the second release. However, more sprints needed to be added so that all those nice-to-have features left in the product backlog could be implemented. Since there was still time left in the academic semester, the extra time for the additional sprints was not an issue. Also, there were other activities that the development team needed to schedule in order to deploy the application to the operational environment. Therefore, this decision needed to be communicated to the COJ stakeholders in the first opportunity since the additional sprints would impact the final delivery date.

As the dateline for the second release was approaching, the project sponsor contacted the development team for any new updates and also to inquire if it was possible to do a product demonstration in front of the COJ Ethics Commission. The development team

and the Scrum Master agreed to perform this demo, which was scheduled at end of this sprint.

Also, during this period the COJ Information Technology Division (COJ ITD) had recently assigned a project manager to oversee the last phases of the project and coordinate the transition of the software to the COJ operation environment. The COJ ITD Project Manager had requested a phone conference at the end of the sprint to learn about the status and datelines of the project and to coordinate a meeting with the COJ ITD staff to discuss the technical aspects of the handover.

Most of the activities performed in this sprint were in consideration with the demonstration. Thus, the following work items were carried out during this period:

Sprint Backlog. The tasks for two more features were included. Also, some tasks were included to improve the styling of the application in preparation of the demonstration and to correct bugs found in the last and current sprint. Other work items were included to continue the elaboration of the manual scripts, testing activities, and documentation.

RTC. The sprint backlog was added to the RTC which contained all the scheduled tasks which needed to be carried out and also included the links to any deliverable produced in the sprint. Even though this tool was not used to its full capability, it helped the development team track the progress and duration of the tasks.

Testing Activities. Besides the continuous elaboration of the manual scripts, the development team performed exhaustive testing and corrected all the bugs found.

User Guide. The elaboration of this document continued as new functionality was added to the software.

Additional Delivered Increment. The team had been able to add two more features to the product and improved the application styling. This increment had been integrated and tested with the rest of the application and was ready for the demonstration.

Retrospective. During the conference call with the COJ ITD Project Manager, the development team informed her about the project status and the delivery date for the second release. Also, they discussed the extension of the project and the delivery date for the final product. In addition, the development team requested a teleconference with the COJ ITD personnel to discuss technical issues before the second release. Further, a meeting was scheduled with the COJ ITD personnel after the delivery of the second release to discuss the deployment aspects such as the operational environment for the application, the handover process and documentation, and the ongoing handling of remedial and adaptive maintenance.

Also, the development team had performed the product demonstration in front of the COJ Ethics Commission which was impressed with the excellent work the team had performed until this point. This product demonstration was especially important for the

Project Sponsor because it had shown how the COJ Ethics Office had partnered with the University of North Florida to develop a product that would bring benefits to the City of Jacksonville. The benefits of this partnership were that the product would provide a more efficient process for the lobbyists, the COJ staff, and the citizens and would save the City of Jacksonville around $100,000 since this software was to be donated. Overall, this demonstration had elicited an excellent response and evoked great motivation for all the stakeholders involved in the project.

At the end of this sprint the development team was confident that the second release was on schedule since most of the core features were in place and the rest could be implemented in the follow-on sprint.

Sprint #5. For this sprint the development team dedicated all their efforts to complete the core requirements and deliver a version of product to the COJ ITD staff for their revision. As a result the following activities were carried out during this period:

Sprint Backlog. The tasks to completed last set of features were added to this artifact as well as the ongoing testing and documentation activities. Also, other tasks were included to cover the correction of bugs, and the implementation of input validation.

Testing Activities. The elaboration of the manual test scripts continued to cover the last set of features. After these features were tested, some bugs were corrected. However, any

bugs found that did not affect the functionality of the system were scheduled to be corrected in the following sprints.

User Guide. The elaboration of this document continued to cover the last implemented functionality and it was delivered along with the latest version of the application.

Delivered Increment. The last set of features that were part of the core requirements were implemented, tested, and integrated to the current application. Then, some styling bugs were corrected to improve the overall appearance of the application. Also, the input validation was implemented to ensure data integrity through the whole application. Lastly, some regression tests were performed to validate the correct functionality of the whole application.

Retrospective. The development team along with the Scrum Master held a conference call with the COJ stakeholders, which included some personnel from the IT Division and the City Council and the project sponsor. Some of the items addressed during this time were:

- How the security for the application and database could be handled.
- The delivery date for the second release and the artifacts produced through the software development process.
- The pickup date for the copy of the application and documentation by the COJ ITD Project Manager, and

- The date for the next meeting with the COJ ITD staff to set up a schedule for the implementation of the last features and the transition and final acceptance of the system. However, before the meeting, the COJ ITD staff was going to conduct a technical review of the application.

At the end of this iteration the application had all the basic functionality and had been tested and released to the COJ ITD staff for their revision. The development team had created a document that contained the future development plans and activities for the final delivery of the project.

### 5.4.3 Transition Phase

After the second release, the project was in the last stage of the construction phase and entering the transition phase. In the past sprint the development team had released an operational product that fulfilled the core requirements. The main functionality was in place, there were some bugs that needed to be corrected, and the nice-to-have features were scheduled to be implemented over the next sprints as well as the refinement of the whole application.

The second release, which included the application code package, the database schema, application installation setup, Visual Studio files, and available documentation, was picked by the COJ ITD project manager for a preliminary technical review and the initial beta testing of the product. At this time the application was delivered in the .NET

framework version that the development team had been using. Later, the COJ ITD staff had requested the development team to downgrade the application to the .NET framework currently used in the COJ IT Division and also to format the application interface style to fit the standards of the COJ.net web site.

A few days later, the development team had the face-to-face meeting with some of the COJ ITD stakeholders. In this meeting the development team performed a walkthrough of the entire application and obtained the following feedback:

- Many technical questions were answered and also new features were added. These new features mostly involved changes in how the information was displayed.
- In addition, there were some new requirements on how the COJ employee should be authorized to access the application.
- The COJ ITD staff requested a technical design document and a user guide that conformed to their standards. To facilitate the elaboration of these documents, they were going to provide templates and guidelines to the development team.
- During this time the Scrum Master from the City of Jacksonville was added to the ITD Project team to facilitate the development and communications during the remainder of the project.
- A preliminary product walkthrough that included the project sponsor and the COJ end users was scheduled for the following week.

In retrospect, this was an effective meeting for the development team because it was clear that the ultimate handover of the product would be compatible to the required operational environment.

One week later, the development team met with the project sponsor, COJ end users, and the COJ ITD representatives that included the project manager, development, and testing teams to conduct the preliminary product walkthrough. The highlights of this meeting were the following:

- Some future enhancements were proposed so that the lobbyists could have links to important information governing the lobbyist activities.
- Other feature enhancements were added such as how to handle lobbyist's confidential information, facilitate user inputs, and display of information.
- Other issues were clarified such as no need for the application to have programmed features for printing web site pages or exporting information.
- Feature enhancements from the last meeting with the COJ ITD staff were revisited again and some were confirmed and others were deleted.
- The product backlog was reviewed and features were added, changed, deleted, and prioritized by all the COJ representatives.
- It was agreed that future enhancements will be handled by the COJ ITD as new Request for Information Technology Services (RFIS). For this purpose, the development team needed to provide detailed technical documentation with instructions on how to implement changes to the application and database and how to upload data from an excel file to the database.

- The development team agreed to send a baseline of the product and updated documentation that included all the items agreed in this meeting.
- The COJ stakeholders agreed to send the information necessary to perform the new changes and additions to the application.
- The development team indicated that the rest of the product backlog and documentation would be finished in three more sprints as long as there was a consistent active participation and feedback from the COJ stakeholders. They also committed to deliver a release at the end of each sprint and send it the COJ stakeholders for beta testing.
- The COJ stakeholders agreed that the product acceptance was going to happen at the end of the last sprint after the completion of the new requirements, the user testing, and the user guide and technical design documents. Also, they added that the acceptance might not coincide with the full implementation of the system due to several action items that included adjustments in the existing legislation and public notification.

Sprint #6. After the meeting with the COJ stakeholders, the development team met to start planning the rest of the features and activities needed to complete the product. As a result, three sprints were planned with three releases so that the COJ stakeholders could start the beta testing of the product. The first release was the baseline of the product and was going to be delivered during this sprint with all the updated documentation such as the product vision, product backlog, manual test scripts, and user guide. The second and

the third releases were scheduled at the end of the second and third sprint. The work items for this sprint were:

Product Vision. The list of features in this artifact was updated to include the new features requested by the COJ stakeholders.

Product Backlog. This artifact was updated to include all the features and activities that the development team needed to implement and perform to deliver the final product. The items in this artifact were prioritized based on the COJ stakeholders' feedback and organized by sprints. Also, it showed the complexity and status for each feature/activity and mapped each feature to a use case. This artifact was updated at the end of the sprint to reflect the status of "Done" for each feature/activity that was completed. This definition of "Done" meant the feature has been implemented and tested or that the activity has been completed and reviewed it. Figures 33, 34, and 35 show how the features/activities in the product backlog were organized by sprints.

| Product Backlog | | | | | |
|---|---|---|---|---|---|
| **Related Feature** | **Task** | **Related Use Case** | **Complexity** | **Sprint #** | **Status** |
| **Sprint 1** | | | | | |
| N/A | Search results should show for all lobbyists and clients who have ever had an approved issue | | | | Done |
| F33 | Every web page should have contact information email address) and hotline for complaints | UC11-Update Profile | Low | 1 | Done |
| F20 | Complete Change History Interface | UC14- change history | High | 1 | Done |
| F34 | Administrator can add comments in an approved issue or why this was denied – need to limit the text by just two sentences | UC11-Update Profile | Low | 1 | Done |
| F39 | When adding a new client the system should allow lobbyists to mark address and name as private | UC10-Add Client | Medium | 1 | Done |
| F36 | The system will be able to send an email to all the Lobbyists | UC7-Approved or Reject Issue | Medium | 1 | Done |
| F03 | Update Lobbyist BAR number | UC1-Create New Account | Medium | 1 | Done |
| F03, F04, F05 | Red* for all required field on all forms | UC1-Create New Account | Low | 1 | Done |
| F41 | Active Directory Integration | N/A | High | 1 | Done |

Figure 33: Product Backlog (Sprint 1)

| Sprint 2 | | | | | |
|---|---|---|---|---|---|
| F37 | The user should be able to select sort options on all search result lists | UC14-View Change History | Low | 2 | |
| F14, F16 | Limit Data and input sizes for the database | UC1-Create New Account | Low | 2 | |
| F30 | Each page in the web application must be printable | UC11-Update Profile | High | 2 | |
| F18 | Display client lobbyist and issue active status and dates. | N/A | Medium | 2 | |

Figure 34: Product Backlog (Sprint 2)

| Sprint 3 | | | | | |
|---|---|---|---|---|---|
| F40 | Update the text on the homepage | N/A | Medium | 3 | |
| | Manage administrator email addresses | N/A | Medium | 3 | |
| F42 | Develop detailed exception handling and logging for the application | N/A | High | 3 | |
| F22 | The system should have a separate search for lobbyist records created before the new system is in place. There should be an import tool to update these records | N/A | Medium | 3 | |
| F42 | Develop Custom Error Page | N/A | High | 3 | |
| F32 | Add tooltips to fields | N/A | Low | 3 | |
| **Recurring** | | | | | |
| N/A | Work with COJ (ITD) to deploy and test the application | N/A | High | 1,2,3 | |
| N/A | Security test application | N/A | High | 2,3 | |
| N/A | Update the application documentation and send it to COJ | N/A | Medium | 1,2,3 | |

Figure 35: Product Backlog (Sprint 3)

Sprint Backlog. This artifact was not created physically during the current and following sprints since the development team had organized the features/activities in the product backlog by the sprint in which they would be completed as shown in Figures 32, 33, and 34.

Testing Activities. The manual testing scripts were updated to test all the functionality in the product baseline. Also, a bug list was created as a result of the testing activities to keep control of what was or needed to be corrected (Figure 36).

| BUG # | SCENARIO | BUG DESCRIPTION | URL | Status |
|---|---|---|---|---|
| 1 | Contact information-Missing email address | Every web page should have email address and hotline for complaints | | Fixed – Added Modal to bottom of every page in the "Ethics Contact" link |
| 2 | Email ID-REGISTRATION PAGE | Not accepting email address if given in Uppercase | http://localhost:8080/home/index/Account/Register | |
| 3 | Phone number-REGISTRATION PAGE | Accepting Invalid format of phone number | http://localhost:8080/home/index/Account/Register | |
| 4 | Remember me-LOGIN PAGE | | http://localhost:8080/home/index/Account/Login?ReturnUrl=%2fhome | Fixed – Removed |
| 5 | Phone number-CREATE CLIENT PAGE | Accepting Invalid format of phone number | http://localhost:8080/home/index/Lobbyists/Client/Create | |
| 6 | SEARCH | Search records must be imported from the current list registered lobbyist- upload most recent Excel file | http://localhost:8080/home/index/Administration/Home/Index | |
| 7 | CREATE CLIENT,EDIT CLIENT-financial | Private flag – checkbox for financial interest. | http://localhost:8080/home/index/Lobbyists/Client/Create | Fixed -Private boxes now in place. Financial interest only on issues. |

Figure 36: List of Bugs

User Guide. This document was updated to follow the COJ ITD's documentation guidelines and to include all the functionality in the product baseline.

Technical Design Document. The development team started the creation of this document which required a description of the software architecture; a description, class diagram, an activity diagram, and user's screen for each module; and the database diagram.

Product Baseline. In the middle of this sprint the product baseline was released and was ready to be downloaded by the COJ ITD staff with all the updated artifacts described above. Later, the COJ ITD staff reported they had been able to download the baseline and documentation and were ready to test it.

Retrospective. After the last meeting, the project sponsor had been testing the application using the application that was uploaded to the development server and had provided some feedback. This feedback required the development team to make small changes to the user interface to make the user input simpler. Also, at the end of the sprint the COJ ITD staff reported that they hadn't been able to upload the application to their server due to an error, which was due to the different versions of the .NET frameworks used by both parties. Because of this, the team scheduled to downgrade the application to the .NET version currently used in the COJ IT Division for the next sprint.

With respect to the development team's activities, all the features included in the sprint backlog had been implemented and the elaboration of the documentation had continued without problems. Also, at the beginning of this sprint the development team had decided to stop using the CLM tool since it was not used to its full potential and the implementation activities and source control were carried out using the Team Foundation Server.

Sprint #7. The development team continued working on the next set of features/activities scheduled for this sprint. Thus, the following work items were performed:

Product Backlog. Besides working on the features/activities already scheduled in this artifact for this sprint, the development team included the downgrade of the application as well. At the end of the sprint the status of each feature/activity was updated to "Done" (Figure 37).

| Sprint 2 | | | | | |
|---|---|---|---|---|---|
| F37 | The user should be able to select sort options on all search result lists | UC14-View Change History | Low | 2 | Done |
| F14, F16 | Limit Data and input sizes for the database | UC1-Create New Account | Low | 2 | Done |
| F30 | Each page in the web application must be printable | UC11-Update Profile | High | 2 | Done |
| F18 | Display client lobbyist and issue active status and dates. | N/A | Medium | 2 | Done |

Figure 37: Product Backlog at the End of the Current Sprint

Testing Activities. The elaboration of the manual test scripts continued to cover the new implemented features. Subsequently, the testing activities were performed to ensure the necessary functionality was in place and detect any bugs in the software. After this, regression tests were performed to ensure the new added features didn't introduce errors. At the end of these activities any bugs found were added to the bug list to be addressed later.

Documentation. The development team continued working in the user guide to include the new functionality and kept on developing the technical design document ensuring that the COJ ITD's standards were followed.

Second Release. At the end of the sprint, the application had been tested and downgraded and a release package had been zipped and sent to the COJ ITD stakeholders. This package contained the new added functionality, all the updated documentation, and a list of specific instructions on what to change so that the application could run in their

operational environment. In addition, the second release had been uploaded to the development server, where it could be accessed and tested by the COJ stakeholders.

Retrospective. During this iteration the development team had continued to implement the new set of features scheduled for this sprint, corrected all the known bugs, and tested the application. The second released had been sent to the COJ ITD staff for their revision and testing. The COJ ITD project manager had approved the documentation and reported that the COJ ITD staff hadn't been able to upload the application to the operational environment due to an error that needed to be fixed on their side. Therefore, the testing of the application in their operational environment had been postponed one week or until the problem was fixed. However, a meeting needed to be scheduled to perform a demonstration of the final product.

Sprint #8. The development team continued to work in the rest of the tasks needed to deliver the final project. Thus, the work items undertaken during this sprint were:

Product Backlog. According to the product backlog, the development team performed the tasks needed to implement the last features, the tasks to verify the security of the application, the tasks needed to handle the COJ employee authorization to administrate the application, and the necessary tasks to perform the final testing and deliver the documentation. At the end of this sprint, this artifact was updated to reflect the status of "Done" for each feature/activity that was completed (Figure 38).

| Sprint 3 | | | | | |
|---|---|---|---|---|---|
| F40 | Update the text on the homepage | N/A | Medium | 3 | Done |
| | Manage administrator email addresses | N/A | Medium | 3 | Done |
| F42 | Develop detailed exception handling and logging for the application | N/A | High | 3 | Done |
| F22 | The system should have a separate search for lobbyist records created before the new system is in place. There should be an import tool to update these records | N/A | Medium | 3 | Done |
| F42 | Develop Custom Error Page | N/A | High | 3 | Done |
| F32 | Add tooltips to fields | N/A | Low | 3 | Done |
| **Recurring** | | | | | |
| N/A | Work with COJ (ITD) to deploy and test the application | N/A | High | 1,2,3 | Done |
| N/A | Security test application | N/A | High | 2,3 | Done |
| N/A | Update the application documentation and send it to COJ | N/A | Medium | 1,2,3 | Done |

Figure 38: Product Backlog at the End of the Last Sprint

Testing Activities. The elaboration of the last manual test scripts continued and they were performed to test the last added features. After the last features were tested, the regression tests were carried out. All outstanding bugs were corrected and tested and the bug list was updated to reflect the completion status.

Documentation. The elaboration of the user guide and the technical document continued until all the new features were included. These two activities were completed few days before the final product delivery.

Third Release. After the testing activities were completed for the last features and all outstanding bugs were corrected, the third release package was prepared. This package

contained all the source code, database files, the latest version of all the documentation, and a new set of instructions on what specific environment variables needed to be modified for the application to work in the operational environment. This package was uploaded to the development server and ready to be downloaded by the COJ ITD staff.

Retrospective. In this sprint the development team had completed all the features scheduled for the third release and continued testing the application and correcting any issues found. These activities continued until the development team considered the testing activities had been completed on their side. The COJ ITD project manager had reported that they had been able to download the application to a desktop and it was working fine. However, due to scheduling issues, they hadn't been able to move the application to the operational environment.

After the end of the last sprint, a meeting was scheduled to perform a walkthrough of the final product to the customers. Also, for the formal handover of the application the COJ ITD Project Manager had requested two copies of the application and documentation in CDs. The COJ ITD Project Manager had reported that the application was working well in one of their personal computers; however, they still needed to solve an issue so that the application could be moved to COJ environment and they could start the customer testing.

During the meeting, the development team conducted the final product walkthrough in front of the customers— project sponsor and COJ end users— and COJ ITD

representatives. As a result of this meeting, the final product obtained the approval from the project sponsor and the development team delivered two copies of the final product package to the COJ ITD Project Manager to formalize the handover of the application.

### 5.4.4 Stage II Artifacts

This section summarizes the RUP and Scrum artifacts produced or updated during the second stage of the project as shown in Table 4.

| Artifacts | RUP Phases and Sprints |
|---|---|
| Software Development Plan | Elaboration, sprint# 1 |
| Product Vision | Elaboration, sprint #1; Transition, sprint #6 |
| Product Backlog | Elaboration, sprint #1; Transition, sprint #6, sprint #7, and sprint #8 |
| Sprint Backlog | Elaboration, sprint #1; Construction, sprint #2, sprint #3, sprint #4, and sprint #5 |
| Delivered Increment | Elaboration, sprint #1; Construction, sprint #2, sprint #3, sprint #4, and sprint #5 |
| Manual Test Scripts | Elaboration, sprint #1; Construction, sprint #2, sprint #3, sprint #4, and sprint #5; Transition, sprint #6, sprint #7, and sprint #8 |
| Release | Construction, sprint #2 and sprint #5; Transition, sprint #6, sprint #7, and sprint #8 |
| User Guide | Construction, sprint #2, sprint #3, sprint #4, and sprint #5; Transition, sprint #6, sprint #7, and sprint #8 |
| Technical Design Document | Transition, sprint #6, sprint #7, and sprint #8 |
| List of Bugs | Transition, sprint #6, sprint #7, and sprint #8 |

Table 4: Stage II Artifacts

# Chapter 6

# RESULTS, CONCLUSIONS, AND FURTHER RESEARCH

## 6.1 Results

This chapter includes the results of this research as related to the research methodology, hybrid approach, and the implementation and retrospective of the case study. Also, the conclusions of the study are derived and further research is suggested.

### 6.1.1 Research Methodology

The goal of this research was to address major shortcomings discovered during an extensive literature review in hybrid methodological approaches to software development. Thus, this research was focused on determining the feasibility of undertaking a significant hybrid software development effort that combined traditional RUP and Scrum within an umbrella of Collaborative Lifecycle Management with the added development constraint of first year graduate software engineering students some of whom had considerable development experience while others did not. Documenting the results of this effort to provide a venue for further research was also critically important.

In addition to validating this hybrid approach using two industry-wide popular development approaches another objective was to supplement existing studies addressing similar approaches while pointing out the major differences between them and this effort.

Both the research goal and research objectives for this study were satisfied. The former was achieved by supplementing new information to the database of existing similar, yet oftentimes quite dissimilar cases presented in earlier chapters. The latter was successfully accomplished by describing the developed lobbyist registration and tracking system.

The thesis includes an introduction to the research topic, an extensive literature review, a description of the research design, data collection, case study research results, and conclusions.

The results of the data collection validated the first guiding proposition defined in the research methodology. This guiding proposition states that specific processes are used to combine the implementation of the RUP and Scrum methodologies for different projects in different organizations.

For this research, the first step in data analysis focused on “within-case analysis” as a standalone entity and to distinguish the unique patterns of this lobbyist case study. The lobbyist system was thus described with the specific RUP and Scrum processes involved and how they were conducted.

The “cross-case examination” indicated that when comparing the above within-case results with findings from previous published studies, results may vary for different projects and organizations.

### 6.1.2 Hybrid Process

One of the objectives of combining RUP and Scrum was to provide a hybrid approach that maximizes RUP and Scrum strengths to deliver a high-quality product that meets customer requirements and expectations. RUP strengths provide structure and guidance through the entire software development process, allow easily adaption of the process to the project needs, and promote customer participation. Although RUP involves project management activities, it does not indicate how to organize the daily activities during the software development process [Collaris10]. In contrast, Scrum strengths include the management of the day-to-day activities to organize the software development process especially during the Elaboration and Construction phase [Krebs05] and responding to changing requirements and feedback while some Scrum weaknesses are that this assumes that there is a vision of the product and it is “not architecture-based [Morampudi13].”

Based on these strengths and weaknesses, the software development process followed a full RUP evolution, which gave structure to the software development process and guided the process especially during the first stage, which included the essentials of the Inception and Elaboration phases. Although many implementations of the RUP do not include all classic RUP elements, the software development process followed in this development

application adhered to traditional elements such as roles, activities, artifacts, and disciplines. The RUP thus provided a framework to the development team to understand the business and projects goals, to define the requirements through use cases, and to proactively address high risks by establishing the initial baseline architecture before commencing the software development activities.

The second stage, which subscribed to RUP's Construction and Transition phases, was realized using Scrum, as the development management process. Scrum practices drove all day-to-day activities in which the development team self-organized and self-managed to deliver business value. Also, these practices allowed the development team to have a more continuous interaction and feedback from the customer to validate that the right product was built. Thus, Scrum process started with the elaboration of the product backlog based on the artifacts produced in the first stage of the process such as the product vision and the use-case model and ended with the delivery of the product to the COJ customers.

### 6.1.3 Implementation Results and Lessons Learned – A Retrospective

This hybrid implementation using the RUP and Scrum was successful in several ways. The evolving system was incrementally released by the development team, reviewed by the COJ users, and approved by several other COJ stakeholders after several face to face meetings. In addition, there was a positive hands-on experience for the students that

participated in this project; there is no development experience better than actually implementing a real-world solution to a needed business application.

As stated, one of the main goals for this hybrid approach was to integrate RUP and Scrum within the context of a collaborative tool, in this case the IBM Rational CLM solution. This solution provides a collaborative environment during the requirements and analysis, construction, and testing phases. However, the development team was not fully aware of all the features of this tool and was thus unable to realize the full benefits of the CLM. For example,

- The RRC area provides the tools to capture use cases, glossaries among others and allows teams to manage requirements and maintain full traceability of these requirements during the software development. However, due to the lack of experience the development team had in order to learn the capabilities of the RRC, this area was only used as a repository for all the artifacts produced during the first stage and to obtain feedback from the team Project Manager. For example, use cases were developed offline using templates in Microsoft Word and then imported into the Requirements Management area of the CLM solution. The development team later learned that RRC provides templates and much guidance for use case development and utilization as a driving component of software development. But as time moved on, retrenching was not an option.
- The RTC area provides the team with the tools to track work items, source control, report, and build management. Also, it provides an Eclipse-based client interface, Microsoft Visual Studio client interface, and a Web client interface to

build and deliver artifacts. Again, due to version incompatibilities between the IBM Rational CLM solution and the Microsoft Visual Studio environment used, the development team — more familiar with Microsoft team Foundation Server— opted to work outside this CLM environment and used Team Foundation Server for source control. Although RTC full capabilities for builds and tracking were not used, the RTC area was used to plan the agile process and track the work items. The project timelines, then, for sprints and releases were created in this area as well as the product and sprint backlog artifacts. The development team noted that neither the burndown chart nor the velocity chart were generated in the dashboard area, only to later discover that tasks as child elements to the features needed to be created within the sprint backlog. However, this solved the issue for the burndown chart, but it did not solve the problem related to the velocity chart.

- The RQM area provides teams the tools to track the quality assurance activities. Because the development activities were accommodated using Team Foundation Server, quality assurance features within RQM were not used.

While the CLM solution was used to host many of the development team's activities and artifacts, the dual task of learning software engineering principles and practices while trying to take advantage of the CLM solution proved to be somewhat out of synch and sometimes incompatible at times. Further iterations of this effort in subsequent software engineering classes would certainly benefit with more full use of the many features within the CLM solution.

One of the goals of an agile process is to have frequent interaction with the customer. This goal was at best partially accomplished. Issues included:

- Schedule Problems. First, the graduate courses at the UNF School of Computing are offered in the evenings and the Software Engineering class met twice a week. Because of this, most of the graduate students had full time jobs or were engaged in other activities during the day. This also made it difficult for the customers to schedule meetings during the day.
- Assigned Times. Even though the COJ stakeholders had great interest in the development of this application, they did not have an assigned time to interact with the team. (This should be well-established in the future if a similar undertaking is to reoccur). Thus, during the first stage of the project, the development team only met few times with the COJ stakeholders to gather requirements. For questions or clarifications, phone calls or emails served as a communication between the two parties. However, during the second stage of the project, the COJ stakeholders appointed a project manager from the COJ IT Division to coordinate the transition of the product to the customer environment, and this resulted in significantly more interactions with the COJ stakeholders, as they slowly started to assume ownership of the product.
- CLM Tool. One of the many advantages of the CLM solution is that it can be used by different stakeholders involved in a project and can serve as a repository for all the artifacts produced during the software development. However, for simplicity it was decided that only the development team and the team Project Manager could have access to this tool. This decision limited having more

interaction with the COJ stakeholders and facilitating their feedback. This should be changed for subsequent repetitions of such a software engineering project with the City of Jacksonville (or other corporate) clients.

This research adds knowledge and may well facilitate other hybrid implementations with RUP and Scrum, especially for students carrying out other software projects in software engineering courses in a university setting. Lessons learned via this implementation will help reduce the gap in how a hybrid approach might be performed in an academic setting. Also, the use of the modern CLM tool will ensure students to acquire more skills that may help them be more effective in their careers.

Unfortunately, while the system was delivered successfully, it has not yet been implemented. The application has not yet been uploaded into the operating environment, ostensibly because of some COJ internal policies, which have arisen and are clearly beyond project scope. Regardless, the delivery of the system was successful since it was reviewed, approved, and it was functioning according to the customer's expectations and requirements.

While extensions and enhancements to the system have been identified, the first step would be to upload the system to the operating environment.

## 6.2 Conclusions

The research goal and research main objectives were satisfied and successfully accomplished. This was achieved by respectively describing the developed lobbyist registration and tracking system and adding new information to the database of existing cases presented in earlier chapters.

This research thus helps reduce the first gap derived from the literature review: "The process of combining RUP and Scrum successfully does not have sufficient empirical research."

Further, the results of the data collection— describing the developed lobbyist registration and tracking system processes— validated the first guiding proposition that was defined in the research methodology. This guiding proposition states that specific processes are used to combine the implementation of the RUP and Scrum methodologies for different projects in different organizations.

The data analysis results of within case analysis— the unique patterns of this lobbyist case study that included the specific RUP and Scrum processes involved and how they were conducted— and the cross-case examination indicated that when comparing the above within-case results with findings from previous published studies results may vary for different projects and organizations.

These research results also added specific knowledge as how the hybrid RUP and Scrum lobbyist project in a particular organization was conducted and what specific processes are involved. Thus, these results may help researchers and practitioners that are looking for evidence in how to conduct this hybrid approach.

Also, this agile-hybrid approach offers several strengths. One of them is that by subscribing to RUP's activities, roles, artifacts, and disciplines during the first stage of the project, the development team will have a better understanding of the organization and will share a vision of the core requirements of the product with the rest of the stakeholders to create the product backlog for the second stage of the project. Also, RUP's incremental and iterative approach allows for customer feedback on the different documents and models produced ensuring that the development team has a more accurate idea of the product needed.

Another strength in this hybrid approach is that by incorporating Scrum practices in the second stage, the development is empowered to make decisions to achieve a goal at the end of each sprint. Also, there are more frequent releases of a working product that allow for continuous customer feedback and engagement. This adds confidence in both sides, the customer knows that will get the product that is needed and the development team knows that is building the right product. Thus, risks are mitigated early in the project and delays on the schedule due to changing requirements late in the project are avoided. In fact, reducing these risks helps shorten the duration of the project.

One of the weaknesses of this approach is that early and continuous customer interaction is critical for the successful of the project. Also, the team members experience and knowledge on these methodologies play an important factor to avoid delays in the delivery of the project.

Based on the experience and results obtained during this hybrid implementation some recommendations may be suggested for new projects using this hybrid approach. First, the development team should choose the RUP's artifacts that add more value to their process during the first stage of the project. However, the creation of the product vision document and the use-case model are prescribed since they are essential for the creation of the product backlog and to drive the main functionality of the system. Next, the early involvement of the customer in the project and prompt feedback are essential for the success of the project. Thus, assigning times or setting up a schedule for meetings with the customer from the beginning of the project and identifying a point of contact on the customer side should be done to facilitate customer interaction. Also, selecting the appropriate software development technologies and ensuring that all the team members have some knowledge or acceptable level of experience using these assures a more productive environment and avoids schedule delays on the project. In addition, the use of the full capabilities of the IBM Rational CLM Solution is considered important to organize and manage all the deliverables, promote stakeholder collaboration through the entire project, and keep track of the project progress.

While the results of this research were successful and valid, these findings are not sufficient evidence to fill this gap completely. Therefore, more case studies that combine RUP and Scrum successfully need to be developed to supplement empirical data to fill this gap.

In summary, this research has provided significant evidence that both the RUP and Scrum may be successfully used within a CLM solution in an academic environment. This research and results may be published in journals since it follows a qualitative research methodology to describe a hybrid implementation process and because it helps reduce the gap by adding knowledge. Also, it can be published as a case study in books related to software engineering technologies. For these reasons we can conclude that the research goals of this thesis were successfully achieved.

## 6.3 Further Research

Future research in hybrid methodologies may include developing more case studies that address different ways of successfully combining RUP and Scrum. This eventually may lead to have enough empirical evidence that will help practitioners carry out their new implementations more successfully.

To respond to the second question that addresses the managerial elements that may influence each step of the process such as quality, culture, and people, a new research effort should be developed. In this research these elements may be measured using

information from different data collection processes. These findings will add knowledge to better understand what managerial elements have the greatest significance in a hybrid implementation.

In the software development life cycle, after the product is successfully delivered, the software maintenance phase starts. In this phase some activities take place, among them new requirements are added, some changes in the operational environment may be needed, and errors are corrected. However, this maintenance phase does not receive as much attention as the other SDLC phases [Pigoski01]. Also, there is not enough information in the literature that addresses how software maintenance would be carried out with a hybrid approach. Thus, further research may also look for case studies that address how to carry out continuous engineering during the maintenance phase using a hybrid approach.

Finally, for the purpose of future software engineering courses at UNF, this study may be used as a source to make improvements from this research experience.

# REFERENCES

Print Publications:


[Bashir12]
Bashir, M. S. and M. R. J. Qureshi, "Hybrid Software Development Approach for Small to Medium Scale Projects: RUP, XP & SCRUM," Science International (Lahore) 24, 4 (2012), pp. 381-384.

[Batra10]
Batra, D., X. Weidong, D. VanderMeer, and K. Dutta, "Balancing Agile and Structured Development Approaches to Successfully Manage Large Distributed Software Projects: A case Study from the Cruise Line Industry", Communications of the Association for Information Systems 27, 21 (August, 2010), pp. 379-394.

[Baxter08]
Baxter, P. and S. Jack, "Qualitative Case Study Methodology: Study Design and Implementation for Novice researchers", The Quality Report 13, 4 (December, 2008), pp. 544-559.

[Boehm04]
Boehm, B.W. and R. Turner, Balancing Agility and Discipline: A Guide for the Perplexed, Addison-Wesley, Boston, 2004.

[Carvalho11]
Carvalho, W. C. d. S, P. F. Soares, M. d. Soares, M. A. Teixeira da, and L. C. Buiatte, "A Comparative Analysis of the Agile and Traditional Software Development Process Productivity," 2011 30th International Conference of the Chilean Computer Science Society, 2011.

[Cho09]
Cho, J., "A Hybrid Software Development Method for Large-Scale Projects: Rational Unified Process with Scrum," Issues in Information Systems 10, 2 (2009).

[Chawla13]
Chawla, L. and R. F. Roggio, "Application Lifecycle Management: Marriage of Business Management with Software Engineering," International Journal of Computer and Information Technology 2, 1 (January, 2013), pp. 19-24.

[Collaris10]
Collaris, R. A. and E. Dekker, "Scrum and RUP – A Comparison Doesn't Go on All Fours", Agile Records 1 (January, 2010), pp. 62-65.

[Cooper01]
Cooper, R. G., Winning at New Products, Perseus Publishing, 2001.

[Creswell94]
Creswell, J.W., Research Design: Qualitative and Quantitative Approaches, Sage Publications, London, 1994.

[del Nuevo11]
del Nuevo, E., M. Piattini, and F. J. Pino, "Scrum-based Methodology for Distributed Software Development," 2011 6th IEEE International Conference on Global Software Engineering (ICGSE), (August, 2011), pp. 66-74.

[Denzin94]
Denzin, N.K. and Y.S. Lincoln, Handbook of Qualitative Research, Sage Publications, Thousand Oaks, CA, 1994.

[Dyck12]
Dyck, S. and T.A. Majchrzak, "Identifying Common Characteristics in Fundamental, Integrated, and Agile Software Development Methodologies," System Science (HICSS), 2012 45th Hawaii International Conference , ( 4-7 January, 2012), pp. 5299-5308.

[Eisenhardt89]
Eisenhardt, K. M., "Building Theories from Case Study Research," Academy of Management Review 14, 4 (October, 1989), pp. 532-550.

[Göthe08]
Göthe, M., C. Pampino, P. Monson, K. Nizami, K. Patel, B. M. Smith, and N. Yuce, Collaborative Application Lifecycle Management with IBM Rational Products, IBM Redbook, IBM Corp, 2008.

[Ionel08]
Ionel, N., "Critical Analysis of the Scrum Project Management Methodology," Annals of The University of Oradea, Economic Science Series 17, 4 (2008), pp. 435-441.

[Karlstrom05]
Karlstrom, D. and P. Runeson, "Combining agile methods with Stage-gate Project Management," Software, IEEE 22, 3 (May-June 2005), pp. 43-49.

[Kruchten03]
Kruchten, P., The Rational Unified Process: An Introduction, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 2003.

[Kulak03]
Kulak, D. and E. Guiney, Use Cases: Requirements in Context, Addison-Wesley Professional, Boston, MA, 2003.

[Morampudi13]
Morampudi, N. S. and G. Raj, "Evaluating Strengths and Weaknesses of Agile Scrum Framework using Knowledge Management," International Journal of Computer Applications 65, 23 (March, 2013), pp. 1-6.

[Nerur05]
Nerur, S., R. Mahapatra, and G. Mangalaraj, "Challenges of Migrating to Agile Methodologies," Communications of the ACM 48, 5 (May, 2005), pp. 73-78.

[Nisa12]
Nisa, S. U. and M. R. J. Qureshi, "Empirical Estimation of Hybrid Model: A Controlled Case Study," International Journal of Information Technology and Computer Science (IJITCS) 4, 8 (2012), pp. 43-50.

[Nishijima13]
Nishijima, R. T. and J. G. Dos Santos, "The Challenge of Implementing Scrum Methodology in a Traditional Development Environment," International Journal of Computing & Technology 5, 2 (May-June, 2013), pp. 98-108.

[Nortier11]
Nortier, B., K. Von Leipzig, and C. Schutte, " The Development of a Software Development Framework by Combining Traditional & Agile Methods to Address Modern Challenges," ISEM 2011 Proceedings, September 21-23, Stellenbosch, South Africa, 2011

[Pigoski01]
Pigoski, T. M., "Chapter 6 Software Maintenance," IEEE- Trial Version, Technical Software Services (TECHSOFT), Inc., Pensacola, Florida, May, 2001.

[Rational01]
Rational, The Software Development Company, "Rational Unified Process Best Practices for Software Development Teams," Rational Software White Paper, 2001.

[Seikola10]
Seikola, M. B., " The Scrum Product Backlog as a Tool for Steering the Product Development in a Large-Scale Organization", Master's thesis, Department of Communications and Networking (Comnet), Aalto University, Espoo, Finland, 2010.

[Shenhar01]
Shenhar, A.J., "One Size Does Not Fit All Projects: Exploring Classical Contingency Domains," Journal of the Institute of Operations Research and Management Science 47, 3 (2001).

[Williams12]
Williams, L., "What Agile Teams Think of Agile Principles," Communications of the ACM 55, 4 (April, 2012), pp.71-76.

[Yin94]
Yin R., Case Study Research: Design and Methods, Sage Publications, Thousand Oaks, CA, 1994.

Electronic Sources:

[Deemer10]
Deemer, P., G. Benefield, C. Larman, and B. Vodde, "The SCRUM Primer," 2010, http://www.brianidavidson.com/agile/docs/scrumprimer121.pdf, last accessed July 11, 2014.

[IBMDeveloperWorks11]
IBM DeveloperWorks, "All that you Wanted to Know about CCM@Rational: IBM Rational Solution for Collaborative Lifecycle Management", https://www.ibm.com/developerworks/community/blogs/CCM@Rational/?lang=en, July 22, 2011, last accessed July 11, 2014.

[Krebs05]
Krebs, J., "RUP in the Dialogue with Scrum," http://www.ibm.com/developerworks/rational/library/feb05/krebs/, February 15, 2005, last accessed July 11, 2014.

[Schwaber09]
Schwaber, K., "Scrum Guide," http://www.itemis.de/binary.ashx/~download/26078/scrum-guide.pdf, May, 2009, last accessed July 11, 2014.

[Schwaber11]
Schwaber, K., and J. Sutherland, "The Definitive Guide to Scrum: The Rules of the Game," http://www.scrum.org/portals/0/documents/scrum%20guides/scrum_guide.pdf, October, 2011, last accessed July 11, 2014.

[ScrumUP14]
ScrumUP <Reference>, http://www.scrumup.com/reference/index.html, last accessed July 11, 2014.

[Soares09]
Soares, M. dos S., "Comparison of Traditional and Agile Methodologies for Software Development," http://wiki.dcc.ufba.br/pub/Aside/ProjetoBitecIsaac/Met._Ageis.pdf, 2009, last accessed July 11, 2014.

# VITA

Dalila Castilla has a Bachelor of Sciences degree from the University of Coahuila, Mexico in Computer and Systems Engineering and expects to receive a Master of Science in Computer and Information Sciences from the University of North Florida, August 2014. Dr. Robert Roggio of the University of North Florida is serving as Dalila's thesis advisor.

Dalila has on-going interests in the software development and database fields, and has four years of experience as a programmer analyst developing information systems using software engineering practices in a Unix environment.

Currently she has updated her knowledge with new technologies based on object oriented languages such as Java and C# and new technologies for web development. Also, this knowledge covers new concepts in software methodologies and new developments in the database field. Dalila is fluent in Spanish and English. She is married and has two children, Chris and Hayley.