The Thesis Committee for William Charles Dalton
Certifies that this is the approved version of the following thesis:

Improving Software Development Project Execution at a Financial Services Company

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor:	Steven P. Nichols
Co-Supervisor:	Robert B. McCann
	Ken M. Fitzpatrick

Improving Software Development Project Execution at a Financial Services Company

by

William Charles Dalton B.S.

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

December 2011

Abstract

Improving Software Development Project Execution at a Financial Services Company

William Charles Dalton, MSE

The University of Texas at Austin, 2011

Supervisor: Steven P. Nichols

Co-Supervisor: Robert B. McCann

Information Technology (IT) is inextricably tied to financial services; the business can no longer view IT as simply a part of discretionary spending. More particular to the financial services industry, technology is a way to gain competitive advantage through innovation. Financial services companies discover ways to utilize technology in order to generate product and process innovation, for example, consider the innovation to use scanners and, more recently, smart phones to deposit checks from home or elsewhere instead of with an ATM or a teller at a brick and mortar bank. As the market becomes more competitive, financial services companies must rely even more on product and process innovation. A key enabler of this innovation is the ability to fully understand the current state of how business value is delivered through the use of IT. This understanding can in turn help financial services companies to plan more effectively both strategically and tactically as the environmental factors change constantly. As companies spend vast amounts of money on projects, it is imperative to understand how ideas flow through a lifecycle and are ultimately realized by some process or product offering that in turn deliver some value to the business. The goal of IT for the business is often stated as an improvement in triumvirate: better, faster, cheaper. The difficulty with this goal is that these facets are often mutually exclusive. How then, can IT deliver? This thesis will examine how one financial services company may improve its software project delivery process by examining its historical and current operating state and then discussing some recommendations to achieve improvement.

Table of Contents

List of Figures

Chapter 1: Introduction

Over the past few decades the IRS has attempted to upgrade their computer systems with limited progress and exorbitant costs. "But after more than 20 years and over $5 billion, there's still no end in sight."[1] How could such an endeavor that was apparently of paramount importance consistently fail and essentially waste money in the process? In the 90s, congressional oversight determined the issues to be the lack of project sponsors, clear contract deliverables, and accountability.[2] To remedy the situation, the IRS hired Computer Sciences Corporation (CSC). Even with the supposed right tools, methodologies, processes and practitioners, the IRS continued to struggle with their Business Systems Modernization Program. In a progress report published in 2000, the IRS acknowledged their struggles and limited success over the years as a result of the enormity of the task and the inherent difficulties and risks.[3] There is no argument that what the IRS attempted to accomplish is a monumental task. But, why after all this time has the organization been unable to fulfill a goal they identified years ago? It is reasonable to assume that the success of a project is more reliant on some factors other than the tools, methodologies and processes. As the IRS acknowledged, a lack of sponsors and ultimately the guidance needed for clear goals to be implemented is a major factor. The issues they identified in the 90's point to a more important factor, people. The IRS has had a large internal IT staff. Even with CSC at the helm of the initiative, internal

[1] Anne Broache, "IRS trudges on with aging computers," *CNET News*, August 14, 2010, http://news.cnet.com/IRS-trudges-on-with-aging-computers/2100-1028_3-6175657.

[2] Elana Varon, "No easy IT fix for the IRS," *CIO*, August 14, 2010, http://www.cio.com/article/32197/No_Easy_IT_Fix_for_the_IRS.

[3] United States, "Internal Revenue Service Business Systems Modernization Program Progress Report," *Internal Revenue Service*, September 1, 2000, http://www.irs.gov/pub/irs-utl/bsm-prog.pdf, 3.

conflict arose as midlevel management felt threatened; the culture and politics of the situation caused gaps in understanding, requirements and design.[4]

The case of the IRS and their costly journey to upgrade an IT system is a great illustration to emphasize three specific points that directly impact the success of a project. First and foremost, a project must have a clear direction and goal. Second and no less important, the people involved with a project must understand the purpose and goals and accept and commit to the goals, tools, methodologies and processes associated with the project. Third, software projects are for building software. In general, these software systems or applications exist long beyond the life of a single project and so successful projects and good software systems are dependent upon mature software development lifecycle management.

ANALYSIS

In order to execute successfully on projects we must have good analysis. Analysis covers topics such as risk identification, gap analysis, impact analysis, requirements elicitation, requirements management, business process modeling, and analysis modeling. This list should suffice to illustrate the discipline even though it may be lacking. Analysis is all about capturing and articulating business needs in terms that both business and information technology (IT) people can understand and from which proceed to do work. "Developing a mutual understanding requires that individuals pass information about how they understand and interpret the world around them, as well as processing to make sense of the passed information itself."[5] This, perhaps, is one of the most critical and difficult areas of the project lifecycle to master. On the one hand a business need can be

[4] Varon, "No Easy IT Fix".
[5] Roger Chiang, Bill C. Hardgrave, Keng Siau, *Systems Analysis and Design* (Armonk, New York: M.E. Sharpe, Inc., 2009), 33.

written down as a simple sentence. On the other hand, a single sentence can result in thousands of lines of complex code and several integrated hardware and other software systems to deliver a seemingly simple statement of need.

The business need is usually referred to as requirements. "Requirements communicate needs from stakeholders to developers on a development project."[6] Requirements can be considered as a contract between a business customer and an information technology (IT) provider. This contract is extremely important because it is the basis for all future work on a project; it is the foundation for successful communication between the business user and the technology provider. Contracts are typically written documents that specify the bounds of an agreement signed by the parties involved. "You create a requirements document to explain what you need to someone capable of meeting the need."[7] How exactly requirements are documented varies greatly amongst organizations and even areas within organizations. Requirements can be written in plain language narratives. Rinzler suggests that requirements should be documented as naturally as a story, "you can make a narrative that is engaging and easy to follow for the people who have an interest in solving the problem at hand."[8] Rinzler's fundamental argument is that we need to write requirements so that all parties can understand. "The challenge when trying to write a story about what you want software to do is to make it understandable to two groups that communicate in very different ways."[9]

[6] Ian Alexander, Ljerka Beus-Dukic, *Discovering Requirements: How to Specify Products and Services* (England: Wiley, 2009), 5.
[7] Ben Rinzler, *Telling Stories: A Short Path to Writing Better Software Requirements*, (England: Wiley. 2009), 2.
[8] Ibid., 3.
[9] Ibid.,

DESIGN

Design produces the system architecture specification such that technology solution developers should be able to create and implement the system according to the specification. Much work has been done to bridge the gap between analysis artifacts and design artifacts so that intent and traceability can be attained. "Achieving validity in a software artifact depends critically on the ability of requirements consumers to correctly interpret the intentions of requirements producers."[10] The difficulty with translating business language into a system specification is that a computer system relies on a specific language domain and syntax. Ali Arsanjani stated "the gap between business and IT is primarily due to the different terminology, levels of granularity, varied models, approaches, tools and methods that each employ."[11] It is essential, therefore, to have skilled analysts/designers and developers working closely to ensure that the design meets the requirements. Ideally, with an adequate design the matter of coding the system should be relatively straightforward even with all the incumbent complexities.

CONSTRUCTION AND IMPLEMENTATION

With a good design, constructing the system is not a difficult task assuming the language and system component infrastructure, whether software or hardware, is available and sufficient. Code may be considered the most important deliverable, because code is the automated business value. However, there are many aspects of a system all of which are required to achieve business value. For example, functional code may not be secure or may be extremely slow. A lack of an adequate disaster recovery plan or availability infrastructure can easily result in a lack of realized business value. With these

[10] Kimberly S. Wasson, "A Case Study in Systematic Improvement of Language for Requirements," *Requirements Engineering: 14th IEEE International Conference* (2006): 10.
[11] Ali Arsanjani, "Empowering the business analyst for on demand computing," *IBM Systems Journal* 44:1 (2005): 76.

considerations in mind, code is still extremely important and to the technology solution providers, it will likely always be considered king. Code quality, therefore, is very important. One of the difficulties in the software development lifecycle is the ability to ensure that a requirement has been met by the software. This is typically achieved through acceptance tests on the working software but good developers should want to go through acceptance testing without any defects and testers should hunger for ways to break the code. With an adequate design to guide them and sufficient testing, the code can move to a production system and start providing business value. How much time we spend on coding and testing is what a managed software development lifecycle should try to address. Obviously we want to spend the right amount of time and do the right amount of documentation and ancillary activities to support development but we need to ensure that the right amount does not only take the current project into consideration but rather the life of the system we are building or enhancing.

PEOPLE

A company or organization can determine exactly what they want to accomplish but if they don't have the right people for the project then all the tools, methodologies and processes likely won't amount to much success. The IRS case is living proof of this fact. People introduce a dynamic to any project because they are unpredictable. A project has to deal with the difficult task of ensuring they have quality requirements and building a system according to those requirements, all the while dealing with interpersonal relationships that are necessary to carry out the work of the project. Emotions and other psychological factors come into play and can either wreak havoc or bring about much success on a project in every facet of the project life cycle.

People must be engaged in discovering and documenting requirements and ultimately all other associated artifacts necessary to produce a valid and quality system. In discussing people, we need to understand there are a number of roles that are essential including but not limited to a business analyst and/or a systems analyst, the project sponsor, the project manager, the developers, the architects, the business subject matter experts (SMEs), and testers. "All aspects of requirements definition ultimately succeed or fail based on how well people can work together."[12] The ability to communicate effectively and precisely is a key factor for requirements and also for project success in general. Projects will succeed or fail based on communication and availability of people.

Observation has shown that when given a particular framework that seems to put bounds on a project team "people sometimes ignored the provided structure."[13] Wasson furthermore found some interesting results regarding human nature. Sometimes people would not volunteer information unless asked and give conflicting or different answers to the same questions.[14] As humans we are highly contextually oriented, particularly when it comes to communication and memory. This presents a difficult variable when thinking about executing projects. Do we have the right people in the right positions, with the right skills and the right tools to do the job that needs to be done? This is really a management problem but plays a significant role in whether or not a project will be successful.

STATEMENT OF INTENT

Using the IRS case as a microcosm, the importance of proper management of the software development project lifecycle may be seen. This thesis shall elaborate more on these concepts in addition to discussing specific tools, methodologies and processes as

[12] Hugh R. Beyer, Karen Hotzblatt, "Requirements gathering: the human factor," *Communications of the ACM* 38:5 (1995): 30.
[13] Wasson, 16.
[14] Ibid.,

they relate to software projects. The author’s company, Financial Services Corporation (FSC), shall be the basis for discussion. The goal of this thesis is to analyze the way FSC currently executes software projects, identify areas of improvement based on past and present literature as well as personal and anecdotal experience from co-workers and discuss key opportunities that the company could realistically take to raise the level of performance in quality, cost and time to market.

Chapter 2: Financial Services Corporation

In order to set the stage properly, it is essential to have a good understanding for how Financial Services Corporation (FSC) is structured and executes software projects. For the sake of avoiding any corporate communication policy infringement, Financial Services Corporation is a pseudonym. The details and discussion will reference factual information about the real company. FSC is a privately owned financial services company that provides insurance, investment and banking services to its customers. FSC has a very strong company culture and prides itself on its mission statement and its adherence to it. The company is subdivided into organizational business units with an emphasis on a certain class of products or services. For example, there is one organization for banking and another organization for investments. Each of these organizations focuses on such things as sales, customer service, marketing, and product management. Separately, FSC has organizations dedicated to providing services that facilitate the growth, execution and maintenance of the business such as business intelligence, portfolio management, process engineering and information technology (IT) among others. This thesis will focus primarily on where the business and IT meet through project execution.

BUSINESS ORGANIZATIONAL STRUCTURE

As was previously mentioned, the business organizations have distinct focus areas. Two areas or disciplines of the business most concerned with IT projects are change and product management. Within a business organization there are several product lines or lines of business. For example, in banking there will be an organization dedicated to consumer loans and a separate organization dedicated to bank deposits and another organization dedicated to mortgage and so forth. Within these lines of business there will be an organization dedicated to the product management and another organization dedicated to the change management.

Product management focuses on building the project portfolio in order to acquire internal investment dollars. They work a lot with marketing and vendors to generate more opportunities for the company. Product management is more strategic and is primarily concerned with watching and planning for the environmental factors that affect the business such as competition, product and technology innovation and regulatory changes.

Change management is more concerned with the daily operations as well as the tactical execution or implementation of the strategic plan for the corresponding line of business. This organization plays a much more critical role with project execution as they own the investment and the corresponding IT product. Change management consists of an executive sponsor and teams dedicated to either new project work or operations and maintenance work. Within each of the teams there are operational or practitioner subject matter experts and business analysts or business managers who manage the implementation of changes into the business operations.

IT ORGANIZATIONAL STRUCTURE

IT is aligned with the business. Just as the business is broken down into product and change management teams by line of business and product, IT teams are aligned by components or domains. For each product line, such as mortgage or auto insurance, there is a corresponding project execution team responsible for delivering new project work for that product and change management area of the business. There are also operations and maintenance teams that align more with the lines of business such as banking or investments. These maintenance teams will manage the systems developed through project execution for the life of the system. Project and maintenance teams consist of a manager, at least one project manager and an army of technical professionals including developers and analysts, although primarily developers. FSC has IT architecture and testing aggregated into separate IT organizations although each has resources aligned to specific components or domains. FSC also has infrastructure and operations teams who manage the technologies behind all of the technology solutions, such as but not limited to databases, servers, telephony, desktop and networking.

Project execution or delivery and maintenance teams work hand in hand with the business change management team. Project delivery teams provide the project management and technical expertise to manage the detail of large efforts, usually over 250 man-hours. Through experience, FSC has determined the need for different funding models to facilitate greater agility with IT in delivering technology solutions. Maintenance teams are responsible for monitoring production and taking on the maintenance of the work packages delivered from the project team. In general, a project or maintenance team consists of a project manager, a technical lead, a functional or systems analyst, a UI designer, a test lead, a project architect and many developers. These teams play a crucial role in all facets of the project lifecycle including but not limited to requirements engineering and management, analysis and design, construction, testing, implementation and maintenance of the IT product.

IT architecture and testing are unique only in that they provide resources that work in a matrix capacity for a given domain; that is to say, there are architects and testers for business areas and specific IT components who do not report to the same management as the rest of the core project or maintenance team. A project or maintenance team generally works with the same set of architects and testers. Architects are split into project architects and domain architects. Domain architects set the strategic goals and direction for the IT systems, while project architects work more intimately with project detail design and compliance.

Testers, while belonging to a separate organization, are integrated with project teams to ensure that the business requirements are implemented and to ensure the system meets the expected level of quality. The testing organization is not a technical organization at FSC, and its workers are not highly technical. A technical degree or certification is not required to be a tester.

FUNDING

Each year the leadership across FSC meet for a strategic planning conference. They discuss the major initiatives that will be needed over the next several years and what kinds of investment will be made in projects and technology. The leadership team prioritizes initiatives by current market, economic and regulatory conditions among others. An overall project and process investment budget is determined and then divided amongst the business units at the highest level. The leadership teams of the various business units compete for funding to execute their respective portfolios of work. Once the funding is allocated to each business unit they, in turn, allocate funding to each of their respective lines of business. For example, the banking business unit has a line of business for consumer loans, credits cards, mortgage, deposits etc. The executives of each product line work to allocate funds to projects. This usually involves improvements such as new product offerings, defect fixes and enhancements to existing systems, or compliance with changing regulatory requirements. A change management executive sponsors each project that is substantiated by a cost benefit analysis. Once the project is authorized, high level estimates are used to create a general project budget although this estimate can lead to unfortunate consequences. Due to the nature of the estimates, there is some flexibility in the budget although typically the change management executive will not accept too much variation in these numbers; plus or minus ten percent is determined as an acceptable project at implementation. Many projects are forced to execute despite a fatal flaw in the initial estimates provided. While this is not always the case, it has happened. The anticipated maintenance cost of the changes is handled separately from the project budget. Naturally, there are IT operations and maintenance expenses for which the business change management area must pay, but as far as project budgets are concerned, those costs don't exist.

PROJECT METHODOLOGY

Projects at FSC are executed with a custom waterfall process or a customized agile methodology. One of the difficulties with adopting methodologies is getting acceptance and ubiquity; this is why no industry model or practice will be found in its purest form, although key aspects of the industry practices will be identifiable. For example, agile development is fairly new at the organization but will manage around 20 – 25% of the project portfolio by the end of 2011. FSC refers to its agile method as Lean-Agile, which takes practices from both Lean and Agile methodologies, therefore it is not Lean and it is not Agile but a unique flavor of process, attempting to take the best of both. Even the waterfall process is not used precisely as prescribed. Some elements of the Unified Software Development Process (USDP) are adopted into the waterfall incarnation. Figure 1 below on the next page shows the lifecycle of the waterfall process.

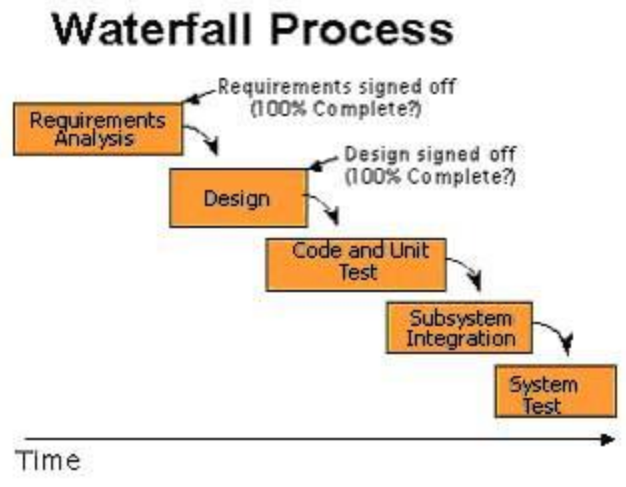

Figure 1: The Commonly Used Waterfall Approach: Requirements and Design Are Finalized Before Development Begins[15]

This is the standard and most ubiquitous process at FSC. While it is true that requirements are "frozen" at the transition into the design phase, there is still a mechanism in place for change requests. Requirements can change throughout the lifecycle, so design, code, unit test and subsystem integration can occur multiple times in much shorter durations. In fact, a common practice is for development teams to begin coding in design for those things that are fairly certain. The trend of development teams is to follow a more iterative approach, which is the basis for the Unified Software Development Process (USDP) and more particularly the Rational Unified Process (RUP). Once again, it is necessary to emphasize that FSC does not follow any industry model or process but rather creates its own processes based on best practices found in several industry standard processes.

Figure 2, shown below, illustrates the RUP lifecycle. Phillipe Kruchten states that this graphic shows the architecture of the process and that the vertical axis represents the

[15] Glenn Tattersall, "Supporting Iterative Development Through Requirements Management," *IBM developerWorks,*Oct. 15, 2002, http://www.ibm.com/developerworks/rational/library/2830.html.

specific disciplines and the horizontal axis represents time and the intensity of the various disciplines over the lifecycle of a project.[16] FSC's project methodology looks much more like the RUP lifecycle in practice. More and more, on the development teams, early coding or prototyping is done to validate requirements and design so that late lifecycle change requests can be minimized or avoided.

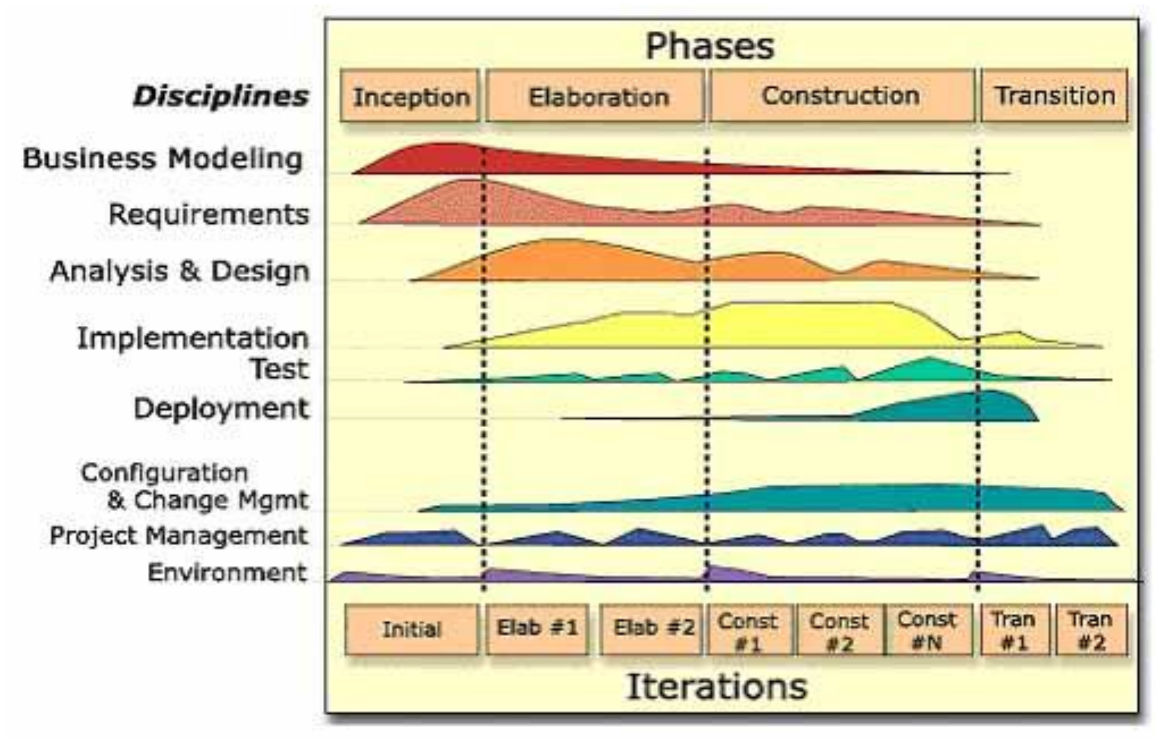

Figure 2: RUP Phases, Iterations, and Disciplines[17]

An IBM consultant stated in an article on iterative development in 2002 "often, practitioners continue to pursue a waterfall approach and attempt to 'sign off' on

[16] Phillippe Kruchten, *The Rational Unified Process An Introduction*, 2. ed. (Reading, MA: Addison Wesley Longman Inc., 2000). 22.
[17] Tattersall, "Supporting Iterative Development".

requirements and design while paying lip service to iterative development."[18] The problem is that many at FSC have grown accustomed to this approach, a mostly waterfall, somewhat iterative process. However, the reality is that FSC generally adheres to structured waterfall phases and therefore requirements, analysis, design and coding phases are generally expected to be 100% complete at the given phase transitions. "The reality is that requirements will change, so it is better to develop an initial set of requirements and get started, rather than agonize over trying to get them 'perfect'."[19] This is why FSC has more recently attempted to adopt an agile methodology.

Around 2007, FSC attempted to try Agile but the attempt failed with serious backlash from several of the business and IT units. The reason Agile failed is because of the coupling a lot of the company has to the waterfall approach. For example, the business likes to use the analogy of building a car when referring to their software products. What good is a chassis with four wheels and a steering wheel if it doesn't have the motor and the body? Basically, the business infers that the product can't have value unless it is show floor ready. Another reason both business and IT personnel rejected the method was that they felt the project would never end and that they couldn't put a concrete end date in place and the budget had to be flexible. While these facts can be realized this isn't always the case; the method merely brings the cost of changing requirements to the forefront. In 2010, FSC was ready to once again try the Agile methodology. Learning from mistakes of the past, the methodology was adopted on a pilot basis and closely monitored to ensure adherence to the practices and principles espoused. With this re-introduction, FSC called the methodology Lean-Agile and worked much more slowly to ensure it would take root. It is interesting to note that the name of

[18] Ibid.,
[19] Ibid.,

the methodology is integral for its success at FSC. Names and labels carry with them perceptions and connotations. Lean-Agile is basically the same, conceptually, as what was tried in 2007; however, there has been a level of marketing and sensitivity involved with the re-introduction. So far, the approach has worked. There is reason to question, however, whether the perceived efficacy of the methodology and the way it is being implemented will really help FSC achieve sustained improvement. One should not infer that there is anything wrong with what FSC is doing but there should be reason for concern. Perception does matter and so do real results.

PROJECT PROCESSES AND TOOLS

At the inception of a project the business must understand two things very well: where they are and where they want to be or go. At FSC, the business relies heavily on the knowledge of SMEs and business managers to explain the current operating state and in turn identify the gaps and opportunities for improvement. Obviously product management plays a role in identifying opportunities but this is primarily in technological or product offerings. A key tool for the business to communicate the current state and future state is through the use of business process models. The extent to which the business uses modeling techniques and tools is limited. The business primarily focuses on the use of Microsoft Visio for the creation of process flow diagrams. The business does not currently document process flows with narratives nor do they have direct traceability to business requirements. Historically, there has been difficulty in maintaining business process documents at FSC although there is a repository and a business process organization to ensure coverage and accuracy of the business processes but it appears to require specific individuals on project teams to ensure the process document is updated. Through personal experience, the author often saw process documents not re-used from project to project; the practice would be to recreate as-is and to-be artifacts on a regular basis for those parts of the process needed for the project. The real point to note here is that a lot of the value or knowledge is with people.

During the elaboration phase, the business works daily with IT personnel to identify the business and system requirements. While UI specification and use-cases are part of the requirements artifacts, currently, Microsoft Excel is the predominant tool used to track requirements in what is called the Requirements Specification Matrix. An unfortunate trend is that both business and IT personnel often refer to requirements as simply the list of all of the "system shall" statements. In the past several years project teams have had the option to use the RSM, IBM DOORS, IBM RequisitePro and most recently Microsoft SharePoint to facilitate the requirements management discipline. Some areas do use DOORS or RequisitePro but these teams comprise a minority across the enterprise. Most recently the organization is looking into the adoption of yet another IBM requirements management tool that will be the prescribed tool to facilitate requirements management activities. Currently, however, requirements artifacts are often not re-used from project to project although they are retained to a degree with the maintenance teams. These artifacts are usually archived with project SharePoint sites and are not easily retrieved and so, like business process models, requirements artifacts are recreated from project to project.

The design phases at FSC typically produce artifacts such as system context diagrams, class and sequence diagrams, component diagrams, and other technical specifications. There is a lot of flexibility in these specifications. While class and sequence diagrams will typically follow the UML standard, many of the most valuable and re-usable design artifacts such as deployment and component diagrams are made up and do not follow a standard. A lot of artifacts are produced based on some standard templates that typically reside in a Microsoft Word, Excel or PowerPoint document. Developers use a Microsoft Visio or IBM Rational Software Architect or some other graphic utility to produce models although the skills to do so in a standard consistent way are lacking across the enterprise. Narrative based specifications are documented in Microsoft Word documents, which are often large and difficult to read through. Like requirements artifacts, design artifacts are rarely re-used or retained except for the purposes of maintenance team transition. Maintenance teams are primarily concerned with knowing the system context dependencies, performance implications and problem resolution. While design artifacts are retained more often than requirements and business process diagrams they are not typically re-used by the project teams.

Construction or coding phases produce the working software. FSC is primarily a JAVA shop with regard to online applications although a lot of processing happens on mainframes, which of course run languages such as C++, PL/1 and COBOL among others. Developers primarily use integrated development environments (IDE) to do their coding and unit testing. In the last year, even mainframe development will be able to occur in an Eclipse based IDE. For most development Genuitec MyEclipse Blue, IBM Rational Application Developer (RAD) and Rational Software Architect (RSA) and Microsoft Visual Studio are the primary IDEs while IBM Rational Developer for Z/OS (RDZ) is currently being introduced to mainframe developers. Individual workstations use Microsoft Windows whereas most code is executed on Unix based and mainframe systems for test and production systems. Code is managed through version control tools such as Borland StarTeam, IBM Rational Team Concert and Microsoft Team Foundation Server. Code is deployed through custom tooling or existing application programming interfaces or infrastructure on IBM WebSphere Application Server, JBOSS Application

Server running on AIX, Linux or Solaris servers, Microsoft Internet Information Services for Windows Server and the mainframes systems primarily IBM CICS on z/OS.

Testing at FSC is done by the centralized testing organization as well as with business users and the IT developers. Developers carry out all levels of testing, but they primarily focus on unit and integration testing. Unit testing is writing code to test code and is usually done during coding activities by individual developers on their code. Integration testing is more functional and is done by developers executing business like test cases; this is usually done on a shared server where code from many developers is brought together to ensure everything works. Once development is basically complete the code moves to formal testing by testers and business users. During this period, defects are tracked against the project and developers work to fix them in a timely manner to prepare for implementation. Business users also determine the need for change requests during this phase since they have working software to use, however change requests during this phase are much more expensive. Testers, as opposed to business users, try to minimize the number of manual test cases executed. During development, testers are typically scripting or recording scripts to execute against the software. Business users will execute exception scenarios primarily while leaving the bulk of testing to testers. At FSC, testers are generally non-technical in their background and use tools such as HP's Quick Test Professional (QTP) to record and execute test cases. Performance testers also have tools such as Quest Software's JProbe and HP's Load Runner to simulate and monitor production-like requests on the application to determine memory, elapse-time, and CPU processing resource issues. Testers also request security scans against web applications to identify any new or existing vulnerability.

The implementation of code to production passes through several stages. As previously mentioned, code is first developed on a workstation and then integrated into a server environment called a Component Integration Test (CIT) environment. From there it moves to a system test environment where testing is done by business and testers. Once the code has been validated and accepted by business and IT testers it moves to a pre-production staging environment. FSC has a beta-like environment where certain customers will have access to the latest code before the rest of the customer base. This subset group primarily consists of employees and it is used to find any glaring issues to prevent a larger customer impact. Once again, testers and business users run a subset of test cases and provide a written confirmation. Developers also watch logs and look for exception messages throughout the pre-production time. Once the code is deemed ready it moves to production systems and goes live through configuration management activities. Immediately upon availability of the software in production, both business users and testers once again perform their validations. Implementation to production occurs close to midnight and personnel work through the night to make sure the system is available and applications are ready before customer service representatives come in for work the next day. Releases typically occur on the weekend and all scheduled changes happen late at night. If there are ever issues that need to be fixed immediately, the project team will go through the entire release process very quickly, assuming the root cause is identified and a fix is made.

Once major defects or issues have been resolved in the implementation, a warranty period ensues for a given period of time, depending on the size and complexity of the changes. During this time the maintenance team takes ownership of key project artifacts such as requirements and design artifacts. They would have already had transition meetings with the project team to become acquainted with at least the nature of the changes as well as the cost impact to maintenance. During the warranty period, the project team must provide primary support; they will handle defect resolution on all defects related to their project. After the warranty period expires, the maintenance team takes full ownership of the project implementation. Most project work at FSC consists of enhancing or modifying existing systems so the maintenance teams are more like appliance or component owners. Any defects or specific business changes less than 250 hours or operational necessities will be conducted through the maintenance team.

Chapter 3: Microcosms

When discussing FSC, it is interesting to note that the company runs projects quite well. For a company that executes roughly half of a billion dollars in project work annually it has impressive production quality by internal and external measures. As good as FSC is, there are some serious opportunities to improve and achieve an even better operating state. The goals of the business are for IT to provide solutions at a lower cost, higher quality and in less time. The emphasis on any one of these facets shifts over time but there are always efforts to improve all three. In the last year, FSC has newly adopted Lean-Agile which is already proving capable of delivering business value at or better than the current goal of 150 days, which is the time from project inception to implementation. There are internal IT efforts to identify tools and processes that can be improved or adopted so that the defect curve can be moved earlier in the development cycle and, thus, test quality should improve. In the past two years there was an effort to truly improve the area around identifying, organizing and planning the business need such that by the time a project started it would be able to focus on detail design, coding, testing and implementation. An improvement organization was created to improve the tools, training and processes for the IT community. All of these initiatives are good and have or are making things better. What is missing or what needs more attention?

First Tasks – Experience Is The Best Teacher

I have worked at Financial Services Company for nearly five years. In that time I have held roles as a developer, a systems/business analyst, a project manager, and technical lead. While I do not claim to be an expert and I fully realize that most individuals that write scholarly articles and books have decades of experience, I know what my short experience has taught me and I have taken great effort to validate my experience and ideas against literature and those in the company who can be considered experts, at least at FSC.

My first work assignments were as a developer and I vividly remember the coding assignments like they were a few months ago. My first task was to implement a batch process that would get data from an automatic clearing house (ACH) return file and post messages to a system for customer service representatives to see those financial transactions that failed. After that task I was able to work to implement an interface with a vendor web application so that customers on our web site would go to a vendor site when they clicked on a hyperlink and we would send certain customer information and financial information securely to make for an integrated experience while on the vendor site. The details of these assi gnments may not seem germane to the topic at hand but during their execution I learned my first lessons about the importance of good requirements.

In my first task, as with all technical tasks, I had to design and implement a solution. After I started making connections and understanding what needed to be done I found that I did not know certain details about my task. What was the message to the customer service representative to say for a given entry in the automatic clearing house (ACH) file? In working with my technical lead I phrased what seemed like a good message and asked if it would meet the requirements. My code was implemented and I never heard if the message was good or bad. It could have been my being new and not knowing where to look, but had it been documented in a requirements artifact a tester would have written up a defect if I had not implemented properly. I don’t think I ever understood what the benefit or impact that change would have on the business but I can only imagine had it been important enough for me to code, it was important enough to the business and ultimately to a customer. Why wasn't this innocuous detail documented? Was it even validated when it went to production?

My second task was a little more complex. I was to program an interface that would redirect users to a vendor site. Technically, this was more interesting because I prefer working with the web. The complex part was to interface with another application to gather some finance details unique to the customer. Building the interface was rather trivial, but gathering the finance details was interesting. As I looked at the code I noticed there were a lot of details. In this case, I worked directly with a functional or systems analyst who, in turn, worked with a business manager. Ultimately the business wanted to pass a finance rate to the vendor, but that rate could represent an actual rate, a potential rate or some default rate. I did not know what it should be so the business had to synthesize a priority matrix before I could implement the code. Naturally, I have to have requirements before I can do work but why wasn't this already defined? How could I be in the construction phase without this requirement put down and how did the business not know they needed to do this particular requirement? How was a tester going to validate the code?

1ST PROJECT – AGILE BATTLE SCARS

After doing some development tasks I apparently showed potential or promise and was made a technical lead around the time FSC was trying to adopt Agile for the first time. The promise of Agile was that we would focus on delivering business value faster and incrementally, if possible. For my first project, I was given a relatively small scope. The task at hand was to improve the workflow of an existing business process in an existing web application. The improvement had high business value. So my team and I set out to deliver. We met with the business and began requirements elicitation and documentation. We asked what aspects of this project would have the highest value and then we focused on the requirements around that item. Then we would tackle the next item and so on. We planned to deliver those first items and then plan another release or two when the rest of the requirements would be finished. Although I felt we had a successful project our business sponsor didn't see it that way and so my management also didn't see it as successful. We delivered functionality in three phases and the business got the majority of their value two months earlier than had we waited until everything was in place. Why did many see it as a failure? First, our IT change management processes were not equipped to handle multiple releases for a project and so there were some process impediments. The release process also ate up project budget several times instead of once. The business teams had not been adequately trained on the methodology. As a result there was a breakdown in communication between the project team and the business sponsor. In our efforts to go fast the sponsor was not fully aware of what was happening. Because the sponsor felt his decision authority had been bypassed he did not find the methodology acceptable. Both the business and IT had tuned their processes to the Waterfall methodology and so it was difficult trying to run Agile as a microcosm for just requirements, design and construction phases in the project lifecycle. We implemented in production and had a major defect. We had requirements that conflicted with the current business process and no one knew it until we were in production and the back office personnel discovered it. We quickly remedied the issue and went on to have the remainder of the releases rather uneventfully, which is the ideal. How did we bypass the sponsor? How did we miss a requirement conflict?

2ND PROJECT – THE COST OF LACK OF KNOWLEDGE

In my next project assignment, which was quite a bit larger, our team was tasked to implement functionality that would assess late fees for past due payments on certain products and would provide a way for customer service representatives to waive the fees as needed. Most of the work would be done by other technical teams. This was a strange project because my immediate team was very small but I needed to manage the design and coordination of construction for several other IT teams, which made the task very difficult. Most of the technical work would be done by others on the mainframe and related technologies, of which I had almost no experience. While this was a learning opportunity for me, I quickly came to a realization that neither my project architect nor I understood how the existing system worked. In fact, it took weeks of meetings with individuals from several teams to collaborate in a room and go through the design with each area detailing the specifics because no one person had all the knowledge needed to specify the whole system. A major risk to the project timeline was due to the authoring and review and approval process for documents. It is absolutely essential for documents that a financial services company publishes to be scrutinized and scrutinized again to make sure we are not exposing any financial or litigious risk to the company. However, I learned in this project that it would take at least twelve weeks from the time the document was drafted to when IT could reference it as reliable content for doing their document work. This was compounded by another issue we faced, which was a huge oversight in the number of documents that needed to be modified. There was a clear lack of understanding of how many documents were involved as defined by the scope of the project. Both business and IT did not have the information; we based our project plan on a large assumption that was not known to be an assumption until later in the project. Also, due to the specific nature of the changes needed, there were key resources that had limited or no availability due to how project work was scheduled for them. Because of missed requirements and resource constraints the project was actually put on hold right before entering the testing phase. At FSC, either a project is cancelled or personnel work longer hours to make the project complete and implement to production as planned. The project later resumed four months later, had a different technical lead because I was on another project by then, and implemented at a much greater cost and longer schedule than originally planned. There are several questions that arise from this effort. First, why did it take a seemingly herculean effort to figure out what the current system did? Second, why does it take so long for documents to go through the authoring and approval processes and if we knew it took that long, why did we have the project timeline we had? Third, how did the business not know all the documents that truly needed to be involved in the project?

3RD PROJECT – A LESSON IN RISK MANAGEMENT

In my third project, which was three times larger than my second project, we were to undertake the retirement of a legacy system which has been in service in conjunction with our newer system for several years. The legacy system still provided functionality that the newer system did not. So, we were to move that functionality over to the new system with enhancement of course and actually turn off the legacy system. This project inherently had a lot of risk. First, while the business understood how to use the legacy system, they did not fully know all of the functionality it provided them. On the IT side, there were two individuals in the company that understood how the legacy system worked. Second, the amount of change was massive. While we had an existing product, and therefore an existing design, a lot of code would need to be written to handle everything in scope. It wasn't so much that the change was unknown but rather that we knew that it was a lot of change. Third, due to organizational politics and another much larger project receiving a lot of negative attention due to planning and budget issues, the technical lead on this project was moved right before construction and I inherited the project, having to finish a detailed design and proceed to construction within a few weeks. I had not been involved in any design decisions and any requirements discussion, however I was familiar with the newer system and had worked on projects that enhanced or modified it before. At the beginning of my involvement I was worried about the success of the project. The construction timeline was aggressive and I was given a fairly inexperienced team. Besides me and a few others, all of my developers were new to the product or lacked capabilities. This project was the perfect storm. Due to all these factors the construction, testing and implementation phases of the project consumed extra hours from personnel. My lead developer even became physically ill during testing because we had an inordinate number of defects. We released on time but had a few significant defects and so we worked to fix them and had a subsequent release the next month to fix more things and implement change requests. We also had to cut scope at one point because the amount of change was certainly too much. But, how did we get into that situation to begin with? Why did we have a lack of technical expertise on the legacy system? More importantly, why wasn't there some sort of system specification for it? Why didn't the business know more about their own product?

4TH PROJECT – NEW PROCESS, NEW TECHNOLOGY, NEW ISSUES

By my fourth project, I had moved areas and was supporting software project initiatives for areas like the one I had left. Specifically, my team worked to make tools and processes so that development teams working directly with the business units would be faster and more effective in their project work. My project was to build a tool to aggregate process document questionnaires into a single cohesive location, making them easily accessible and available for all those involved in a project to find information about the project. The ultimate objective of the tool was to make it easier for development teams to take care of the process overhead and for operations areas to be able to find information to ensure compliance and perform adequate impact analysis and planning for change. To build this tool we decided we would be an early adopter of the Lean-Agile methodology that was rolling out throughout 2010. We also decided to adopt and implement a new technology or application programming interface for building web applications, called Wicket. This technology was to be the new standard for developing the web or view components of a web application at FSC but we were to learn it and build an application with it before any internal training was available. My project team was a small team, about four developers. As the technical lead on the project as well as the analyst and business manager I worked with the sponsor and my development team to execute the project. As the methodology promises, we quickly attacked user stories and implemented an application iteratively. The sponsor, who was also in IT, was extremely pleased to see new or modified functionality every Friday. At each review we would walk through the new items or changes that were requested and took feedback. By the following Friday we would repeat the process. After some weeks, the sponsor began inquiring as to when we would finish. We would go back to the stories and explain what was left. As we would have subsequent reviews, the sponsor would ask for more features or changes to existing features. At these times we would have the discussion that we would need more time to do it but that we could do it. Managing the sponsor's expectations was key and fairly easy because we met often with working software. The project actually lasted a lot longer than the sponsor or me wanted, however, we were constantly adding new features. We had to have a paradigm shift in our thinking about the project because traditional projects at FSC had a defined start and end. We had an open backlog of features and were always working to implement it, so the project lasted as long as it needed to, which was longer than expected. However, the sponsor was getting exactly what was wanted. On the technical side however, we spent more time in construction having to learn the new technology and how to apply it correctly or following best practices. I had received training and, in turn, had to train my developers, which was not easy. Luckily, our product was not overly large or complex so code changes were fairly straight forward. Surprisingly, managing the code changes and ensuring they were adequate beyond functioning was a challenge even with such a small team. I mention these facts because I have become aware, recently, of larger projects in the full scale development areas working with the business units that have run into problems due to using the new technology and not fully understanding it. Of course, it is a risk to introduce new frameworks and with the early adopters there will be a learning curve, but how can we mitigate the risk better? As for the Lean-Agile process, I found it to be fairly successful. It was difficult to get my few developers to estimate their work packages accurately taking into account all the needed activities like code review and unit tests and integration tests. Developers look for the most efficient way to get things working and implemented. The problem I faced was having clear expectations with regard to all the artifacts required. As such, I feel like our final product was not as polished as it could have been had the whole team been on the same page.

Recent Work – The Need for Mature Process

Most recently I have been working in the operations organization at my company doing development and process improvement work. In operations, the mentality is to get things done so that we can maintain system availability and stability. As part of that culture or mentality when tools and processes are developed time to market is paramount. One of the first things I noticed was the lack of formal processes to get things done. If a developer has an idea and manager support a lot can be automated. Larger efforts, for cross team automation or tools, of course, require more documentation and more rigor, in general, but the solutions are often delivered very fast.

One team has most recently been created to handle development of automation tools across the entire organization spanning disciplines such as store, database, server, networking, desktop, etc. This team realized what the lack of formal process can do to efforts, particularly when the tools have to be maintained by another person or team. Progress has been made with the automation team but this team is a perfect microcosm for some of the other improvement effort going on in the development area at FSC. How much rigor is needed for a particular project? How much documentation should be required? These are the questions many development teams ask because the process overhead is often what is complained about as the cause of slow-downs and bottlenecks in the overall process of executing and delivering IT solutions for business customers. Little or no process leads to lack of adequate documentation for those left to support and later enhance systems. Little or no standards lead to inconsistent documents, such as requirements or design templates. All of these facts were realized by this operations automation team. Luckily, over time, the team has matured and is finding a suitable process to run effectively and consistently.

In all the examples mentioned so far, I've drawn from my own experience and observations. I have consistently asked myself if this is really how it is all over the company or just where I've happened to be. Perhaps it is just coincidence, due to other circumstances, that I have witnessed the realized risks of inexperience or insufficient process or adherence to process. Either way, there are areas of FSC that still need improvement.

Chapter 4: Processes

Over the past decade several process methodologies have been tried by FSC. The Capability Maturity Model developed at Carnegie Mellon, aspects of the Unified Software Development Process, and Agile have all been attempted but never successfully and never ubiquitously. The Waterfall methodology has been the one constant over all the attempts at change. Sometimes a methodology didn't succeed because of a lack of tools, training or other processes. FSC does not readily adopt a process just because it is the latest and greatest or the industry standard for financial services companies with substantial internal IT shops. Changing process requires a paradigm change or a culture change and currently there is an effort at FSC to do just that, learning from the lessons of the past and attempting to take the best practices and techniques from other process methodologies. As mentioned previously, FSC likes to customize how they do things. It is not simply Waterfall or Agile methodology; it is the FSC Waterfall or FSC Agile methodology. Those in leadership, desire FSC to embrace Agile. Currently, FSC does about 90% of projects using the waterfall approach. All supporting processes are patterned after this model, which makes it more difficult to adopt iterative models.

SOFTWARE DEVELOPMENT LIFECYCLE MANAGEMENT

In FSC, the technical architects responsible for the engineering disciplines from requirements to configuration management have come together to pitch the idea of a suite of tools and processes to facilitate a target state of end to end lifecycle management. The fundamental idea behind this approach is that it is not governed by any one specific model or process but rather facilitates effective management of the best practices involved in any project methodology. Figure 3 below depicts generic phases or stages in a project lifecycle. This figure implies a flow from business need to implementation which is accurate in any execution methodology. The focus of the figure however is to note some of the key artifacts typically produced in these phases.

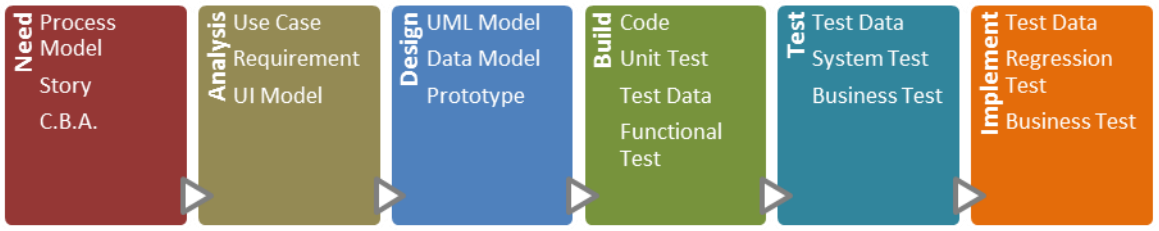

Figure 3: Generic Software Development Lifecycle Artifacts

The lifecycle of a project begins with conceptualizing a business need. While the artifacts listed are common they are obviously not always used. At FSC, user stories are not typically used except in the percentage of projects currently running under the FSC Agile methodology. In the Need phase, the business must be able to articulate a need and justify its cost, hence the inclusion of a cost benefit analysis in this phase.

Once the business need is articulated through some formal artifacts, the Analysis phase can expound the business need to create more detailed specifications. These specifications or artifacts include use-cases and requirements narratives as well as user interface (UI) models. What is wanted and why are determined at a high level during the needs phase. What is wanted is elaborated with enough detail during the analysis phase so that IT can translate business need into system specifications.

The design phase is precisely for IT to articulate the business need into a representation of a system such that it can be constructed, tested and implemented. In a sense, design provides the architecture schematics. Of course, each project does this at inconsistent levels at FSC. This is a significant issue for long term strategic management of systems. Although it is not currently the standard at FSC, the Unified Modeling Language (UML) models are an ideal candidate for output out of the design phase. Data models are used but not to their full efficacy especially in terms of integration with the rest of the software development environment. Prototypes are also used when necessary. Prototypes are working code samples that are usually developed to demonstrate how something will be implemented. It can be used to show the business something functional and is often used to help the development team understand or even complete a complex part of the system early to eliminate risk.

In the build phase of a project, the developers will be diligently writing code, writing unit tests and performing integration tests. Unit tests are also code that exercises functional code to minimize defects during integration. Integration tests may be code in the form of test harnesses for testing working software actually running on test servers with other parts of the system being developed. Since most teams at FSC work with medium to large teams of developers, development is done in parallel to reduce the time in the phase. Whenever a web service, for example, needs to be consumed or new modules must interact with each other, there are often defects that should be identified and fixed before the testers and business users test the system.

Testers at FSC work to script the acceptance tests, usually during the design and build phases in order to execute them during the test phase. A motto at FSC is "no manual testing". Many tools are used to automate the execution of actual business scenarios to ensure the working software meets the business requirements. The work of the testing organization is the way FSC accomplishes a measure of traceability from requirements to working software. The goal of the testing phase is to find and eliminate all defects. As part of that, business subject matter experts (SMEs) will often perform manual testing on highly peculiar or exception scenarios to test the extreme business use-cases.

Upon implementation of the software to a production or production-like system, testers and business users will perform a subset of validations to ensure the code was released as expected and all software is working as expected. Typically the testers will have a suite of regression tests to ensure existing functionality works in addition to new ones developed for the new functionality. The business may also have special data available in order to execute test scenarios in production systems. This is limited by the nature of financial services and ability to execute actual financial transactions.

The common thread among all these phases is that some artifact is produced whether it is a business process model in Microsoft Visio or a test case in HP's Quick Test Pro. Artifacts are stored, read and used to facilitate the execution of a project. Projects at FSC generate a lot of documents or artifacts. These artifacts are put into project shares or Microsoft SharePoint sites. Once the project implements some documents are transitioned to be used for maintenance purposes but the rest are archived and not easily retrieved in the future. And therein lays the crux of the matter. Except for the immediate project or recently implemented projects, there appears to be a loss of the documented context about the systems developed through erosion. Detail is lost about what was to be delivered, why, how and even the specific implementation details for what was delivered. Traceability and reuse are weak. There is a lot of opportunity to mature the processes at FSC. What can be done?

CAPABILITY MATURITY MODEL INTEGRATION

Pankaj Jalote wrote that the "use of proper processes is extremely important for an organization that seeks to deliver high-quality software and increase its own productivity."[20] Process is important but it is a common human behavior to not want to

[20] Jalote, Pankaj, introduction to *CMM in practice : process for executing software projects at Infosys* (Reading, Massachusetts: Addison Wesley Longman Inc. 2000), xi.

follow the rules, especially if there is a perception that those rules seem to get in the way of accomplishing tasks. Based on experience and observation, it seems there is a tendency to deal with problems or tasks in a way most accustomed. Process is the way people deal with problems and tasks. So, it reasonable that people want to follow the process of which they are accustomed. Generally, people aren't all that interested in trying to solve a problem in a new way because it often seems like an unnecessary risk. Since part of project execution is about eliminating and mitigating risk, it is intuitive to deal with well-known risks associated with current processes. In 2000, Jalote wrote about his company Infosys as a practical example for how the CMMI can be, and has been, implemented at a successful company. "Infosys Limited (NASDAQ: INFY) was started in 1981 by seven people with US$ 250. Today, we are a global leader in the 'next generation' of IT and consulting with revenues of US$ 6.35 billion (LTM Q1-FY12)."[21] A key item to note is that Infosys' core business is IT. FSC is a financial services company but IT is integral to the success of the business, therefore as far as evaluating the CMMI, Infosys stands as a reasonable example. Jalote presented the argument that "the range of results that can be expected in a project when it is executed using the software process of an organization is the software process capability" and if an organization wants to see a better range of results they need to improve the process capability.[22] The following are the maturity levels outlined by the Capability Maturity Model Integration (CMMI):

1. Ad-hoc, no processes in place.
2. Managed at a project level, basic project management.
3. Defined, at an organizational level, standardized process across organization.

[21] Infosys, Jun. 20, 2011. http://www.infosys.com/about/what-we-do/pages/index.aspx.
[22] Jalote, 6.

4. Quantitatively managed, measured and controlled.

5. Continuous process improvement[23]

Jalote provides a way to put these five levels into perspective:

> In level 1, an organization executes a project in a manner that the team and project manager see fit…The repeatable level (level 2) applies to an organization in which project management practices are well established, although organization-wide processes may not exist. At the defined level (level 3) the software processes for the organization have been precisely defined and regularly followed….At the managed level (level 4), quantitative understanding of the process capability makes it possible to quantitatively predict and control the process performance on a project…At the optimizing level (level 5), the process improves continuously, with level 4 providing the mechanisms to quantitatively evaluate the effectiveness of process enhancement initiatives.[24]

Each of these levels encapsulates process areas. Except for level 1, each level has "key process areas (KPAs) which specify the areas on which the organization should focus to elevate its process to that maturity level."[25] In reviewing the goals for KPAs listed for Level 2 and 3 at Infosys, FSC appears to also implement many of them.[26] The question is whether or not FSC is achieving the goals and is interested in the KPAs and goals at level 4 and 5.[27] Basically, the model provides a way to look at what is being done and what is needed to improve process maturity for a given area. In the webinar "CMMI on the web" it was pointed out that the maturity level is not a goal in and of itself but rather a way to help organizations improve.[28] At Infosys there are five groups of key practices:

- Commitment to perform – actions the organization must take to support KPA

[23] Deen Blash, Shane Mcgraw, "CMMI on the web," Webinar, *SEI*, Sept. 25, 2008. http://www.sei.cmu.edu/library/abstracts/presentations/20080925webinar.cfm.
[24] Jalote, 6.
[25] Ibid., 7.
[26] Ibid., 10-11.
[27] Ibid., 12.
[28] Blash, Mcgraw, "CMM on the web".

- Ability to perform – training, resource requirements and control structures for personnel
- Activities performed – actual process activities that are recommended
- Measurements and analysis – what measurements should be done
- Verifying implementation – process is verified independently and by senior management[29]

These practices may seem like common sense. At FSC, there is usually a verbal commitment to perform, a perceived ability to perform, a lack of activities performed or activities performed but not at their optimal level, some measurement and analysis and some verification of that measurement and analysis. Process has to be perceived positively and at FSC, process is too often considered an annoyance when it is not well understood or simply seen as a roadblock to success because of the prescribed rules, rigor and documentation required. Process is a good thing, but FSC must find the right amount of process.

> Having a standard process reduces variations in performance of different projects. Without a standard process, projects may follow different process, resulting in different outcomes along various dimensions like quality and productivity. In such a situation, process becomes less predictable in that past performance has only a weak correlation with performance of future projects. Consequently, past data and experience cannot be used for estimation, and a new project cannot effectively learn from past projects (and learning from the past is essential for improving).[30]

Even Jalote recognizes that no one process fits every project and so process tailoring is used.[31] At FSC, some efforts for tailoring are used for the standard Waterfall process. FSC uses risk based-rigor, which means that the more risk on a given project, the more

[29] Jalote, 8.
[30] Jalote, 73.
[31] Ibid.,

rigor is needed. Rigor, in this sense, means more documentation and more quality assurance controls. One of the key reasons CMMI did not flourish at FSC as it did at Infosys was that it was considered too formal and requiring too much time and rigor. A cursory glance on the Software Engineering Institute's website publications shows that the organization considers CMMI to work with anything from Waterfall to Agile and that CMMI is not a methodology for execution in itself but a way to improve the chosen methodology. At the time FSC tried CMMI the focus was on the waterfall methodology, which is formal and requires a lot of rigor. At FSC, almost every attempt to really improve the process seems to erode without a champion at the right level of management and a groundswell of support from the practitioners.

UNIFIED SOFTWARE DEVELOPMENT PROCESS

Around 1999 Ivar Jacobsen, with others, had formalized the Unified Software Development Process (USDP). In discussing process Jacobsen stated, "Whereas process is critical, in practice it both shapes and is shaped by the tools used to implement and automate the process."[32] If FSC desires to improve project execution, there must be an examination not only on process but also on tools. There must also be a re-evaluation of certain job roles and skillsets to become more effective. "Adopting new technology, particularly in the context of re-engineering the software development process, requires more than high-quality products (process and tools) can deliver alone."[33] Jacobsen was refering to using object-oriented principles when he stated this but it seems reasonable that he would also recognize that it takes people to execute processes and use tools, and that people are, perhaps, the most significant factor in the success of any project. A key

[32] Ivar Jacobsen, *The road to the unified software development process* (Cambridge, United Kingdom: Cambridge University Press, 2000). 6.
[33] Jacobsen, 6.

theme of the USDP Jacobsen wanted to bring, was that each discipline in the project lifecycle does not have to be time boxed. Any discipline or process can be executed throughout the lifecycle of a project but the intensity of each varies throughout the timeline of the project. This was illustrated by Figure 2 in chapter 2 of this thesis as a software engineering process. “A Software Engineering Process is the total set of activities needed to transform a user’s requirements into software.”[34]

Jacobsen has played a key role in the development of the Unified Modeling Language (UML). He argued that “what the field badly needs is a set of models that will (a) help architects and developers themselves think and (b) let others see what they are thinking.”[35] Thus, the UML was developed although it is uncertain as to whether its usage truly accomplishes what Jacobsen and the Object Management Group fully intended. “All those concerned with software development worldwide, e.g., architects, developers, testers, managers, users, customers, and stakeholders, can comprehend the common models.”[36] If the definition of comprehend means that as long as there is a developer or architect in the room walking through the model and explaining its legend or syntax to testers, managers, users, customers and stakeholders then Jacobsen’s is more agreeable. Jacobsen is mostly correct about the UML. Currently, it is the most efficient way to document design ideas and communicate them to others. The issue with the UML at FSC is that not all developers and architects are familiar with it or familiar enough with all the key model types. The reality with people is that unless the artifact is essential or required for them in their job role they will not care about it nor be inclined to use or understand it. The UML is for IT alone at FSC. Like code, the design diagrams create an

[34] Quoted in Ibid., 122.
[35] Ibid., 104.
[36] Ibid., 105.

interesting response from the business users. It is as if users take leave of their cognitive abilities when technical specification documents are placed before them. This is because not all people associated with software development at FSC have a background in IT.

The use-case, which Jacobsen invented, is another tool prescribed by the USDP. To address the short-comings of the UML for non-IT personnel the use-case model can be used to picture the system through how it is used by users.[37] Jacobsen surmised that "if we can identify, early on, all the ways the system will be used and then control development so that the system offers these ways, we will know we are building the right system."[38] According to Jacobsen, use-cases fulfill two key purposes, first "they capture a system's functional requirements" and second "they structure each object model into a manageable view."[39] Use-cases at FSC are typically represented by narratives or use-case descriptions. FSC does not currently practice use-case modeling. Interestingly, use-cases are often developed as analysis afterthoughts at FSC. So much emphasis is on the requirements specification matrix or Excel spreadsheet full of "the system shall" statements that use-case descriptions are often created and reviewed but rarely used by developers. In short, use-cases are an ancillary rather than primary requirements artifact at FSC. While use-cases are requirements artifacts the business can understand, they can be used directly to design much of the system from user interface to the object model.[40] "To perform use-case modeling, it is important to understand the problem domain."[41] Jacobsen presents two ways of understanding the business domain through models although only one shall be emphasized. He suggests that by "mak[ing] a business model

[37] Jacobsen, 168.
[38] Ibid.,
[39] Ibid., 169.
[40] Jacobsen, 202-203.
[41] Ibid., 204.

of the business process you are going to support with information technology (IT)" you can "uncover your true business process."[42] If we know the real business process then we can adequately discuss use-cases and requirements in order to design the system to support the use-cases. There is a distinction between analysis and design, although for many practitioners the line is blurry. If use-cases are adopted and modeled, it follows that a design model can be developed directly from the use-case model. Jacobsen acknowledges that the design likely will be more complex and constrained but that is the purpose of design over analysis; analysis extracts the ideal, whereas design formulates ideal into a manageable solution and then code is used to implement the design.[43] Jacobsen states that by having an analysis model and a design model "maintained throughout the lifetime of a system, different problems are solved using different models."[44] It is important to note that the value concept here is that analysis and design models should be maintained for the life of a system, not for the life of a project.

RATIONAL UNIFIED PROCESS

The Rational Unified Process (RUP) "is a software development approach that is iterative, architecture-centric and use-case-driven."[45] The RUP is an incarnation of the USDP as a defined commercial methodology for executing projects. Two factors that are claimed to make the RUP superior to the waterfall approach are that the waterfall "leave[s] key team members idle for extended periods of time and defer[s] testing until the end of the project."[46] Kroll and Kruchten elaborate the claim of superiority by listing nine advantages:

[42] Ibid.,
[43] Ibid., 208-209.
[44] Ibid.,
[45] Per Kroll, Phillippe Kruchten, *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP* (Boston, Massachussetts: Addson Wesley. 2003), 3.
[46] Ibid., 6.

- It accommodates changing requirements
- Integration is not one ‘big bang’ at the end of a project
- Risks are usually discovered or addressed during early integrations
- Management has a means of making tactical changes to the product
- Reuse is facilitated
- Defects can be found and corrected over several iterations
- It is a better use of project personnel
- Team members learn along the way
- The development process itself is improved and refined along the way[47]

Since the RUP is based on the USDP, emphasis is placed more on how the RUP looks at various stages of the project lifecycle. As depicted in Figure 2 in Chapter 2, the RUP has four phases. Since most projects at FSC are changing existing systems through bug fixes and enhancements, a common example project used throughout Kroll and Kruchten’s work will be referenced.

During the inception phase of a project, existing use-cases that are to be modified are reused [and] extended and new use-cases as well as non-functional requirements are created.[48] The key with this phase on an existing system is that the phase should be relatively short, which is a stark contrast to the comparable inception phase at FSC for many projects. At FSC, the time spent on requirements is usually equivalent or exceeds the rest of the project phases combined. This phase has five objectives outlined by the RUP in an example project:

1. Understand what to build making an inventory of additional capabilities and the known problems in the existing system that should be fixed.

[47] Ibid., 7-9.
[48] Ibid., 96.

2. Identify key system functionality by identifying critical use-cases and documenting the architectural significant use-cases in a architecture document.
3. Determine at least one possible solution making mock-ups and prototypes of some of the critical use-cases and getting customer feedback.
4. Understand the costs, schedule and risks associated with the project documenting the business case, development plan, risks and other project management documents.
5. Decide what process to follow and what tools to use, realizing most of the same team members were involved in development on the system before and may be accustomed to the previous process and tools.[49]

The elaboration phase details requirements and design. For existing systems this phase should be relatively short assuming the number of architecturally significant use-cases is not large. At FSC, this phase would be considered the design phase. Requirements will have been detailed sufficiently so that the technical team can focus effort on architectural impact and design. The example project provides for four objectives:

1. Get a more detailed understanding of the requirements detailing the rest of the use-cases and analyzing in detail the fixes to major defects assessing the impact to the architecture.
2. Design, implement, validate and baseline the architecture ensuring that the critical use-cases do not regress the architecture.
3. Mitigate essential risks and produce accurate schedule and cost estimates updating the business case the development plan risks and other artifacts.

[49] Ibid., 102-110.

4. Refine the development case and put the development environment in place implementing lessons learned from the previous project and inception phase as well as take care of any tooling and training needed.[50]

The construction phase is focused on writing the software. The key component to this phase in the RUP is that the customers have to be involved in some sort of validation of the software components created. At FSC it depends on the system. Since waterfall has long been the methodology of choice, the systems do not necessarily lend themselves to iterative development. The example project lists two objectives:

1. Minimize development costs and achieve some degree of parallelism by dividing the staff into small teams working on subsystems or components that will be implemented and tested.
2. Iteratively develop a complete product that is ready to transition to the user community with some customers doing alpha and beta testing along the way.[51]

The transition phase is a move to maintenance. The software is in production or in the hands of the customer. At FSC many areas have separate maintenance teams that own the components developed beyond the life of a project. However, at FSC there is a concept of warranty in which the development team will still handle any defects that arise soon after project implementation.

AGILE

Agile is one of the newer methodologies in industry and FSC. There are many variations of Agile at FSC, so to establish a common understanding of their respective approaches, it is helpful to review the foundations of agile, as found the Agile Manifesto:

[50] Ibid., 116-136.
[51] Ibid., 151-159.

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the short timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity – the art of maximizing, the amount of work not done – is agility.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.[52]

It is interesting to note that FSC refers to its agile methodology as Lean-Agile. Charles Cobb states, "First and foremost, agile has adopted the fundamental value system

[52] Charles G. Cobb, *Making Sense of Agile Project Management* (Hoboken, New Jersey: John Wiley & Sons, Inc., 2011), 40-43.

of lean: delivering and maximizing net customer value."[53] In his overview and history of agile, Cobb presents how agile is rooted in Lean principles and discusses the agile manifesto in more detail. FSC uses user stories as the fundamental requirement artifact and story points for estimation and planning. User stories are basically narrative statements that follow a format such as: As a user, I need to do such and such so I can get such and such benefit. Story points are relative sizes of effort assigned to stories. Story points are used to try to get user stories broken down to a level where iterations can execute design, development, testing and, if sensible, implementation in the week or month long iteration. The promise of Agile is that all aspects of the project lifecycle are working all the time, much like the RUP with iterative development, except that the levels of engagement are more consistent. Since Agile is focused on lean principles there is a high proclivity for less documentation and process so that working software can be produced.

A typical Lean-Agile project at FSC has daily standups where each team member states what they had accomplished the prior day, what they are going to do today and mention any risks or roadblocks they have and address them offline after the meeting. This team accountability also helps show the progress being made. All work should tie to some form of "card", whether it be an actual post it note or a virtual task that should be completed within 8 man hours or a day. Ideally, staff members would be constantly moving tasks through a workflow where they code, unit test, code review, acceptance test and then implement or wait to implement the software until additional features are developed. Thus, user stories are broken down into much more easily estimated tasks that all filter back up to a story. To summarize, all work is focused on those activities that add

[53] Ibid., 21.

value and the team members develop the tasks that need to be fulfilled to accomplish the story until all stories on the back log are complete.

Something that FSC has been doing in the past year includes a Lab-Agile methodology. As mentioned previously, names or labels are very important at FSC. Officially, the name is Agile Scrum. The lab concept allows for business end users who support end customers to be in the same physical space as the developers who are working on software and testing it with the service representatives and a small portion of membership near-real-time. In this environment there are dedicated configuration management resources, testers, developers and business people. Software has been released to a dedicated production environment in as little time as five days from inception of the business need. The basis of the lab methodology is that the fundamentals of agile are in practice and the entire environment from people to tools and processes are encapsulated so that all aspects of the project lifecycle exist in an isolated laboratory. Typically the execution and release cycles are one or two months in this environment but it illustrates a large contrast with the rest of the portfolio running under the near-waterfall methodology, which execute in much longer periods of time.

COMPARISON AND DISCUSSION

In a recent meeting discussing the Agile methodology and its future expansion at FSC, a senior level architect explained the vision for the future is to make Lean-Agile the de facto process for executing projects. The desire is to expand the Lab concept so that the labs are increasingly focused down to the system or component level rather than just general business area. The goal is to execute 20% of the portfolio value in labs in 2012. With a measured approach it seems as if FSC will move ahead and actually change from waterfall to agile.

So how do these and other processes compare? Kroll and Kruchten provide a helpful analysis by providing two dimensions to compare these processes in Figure 4 below.

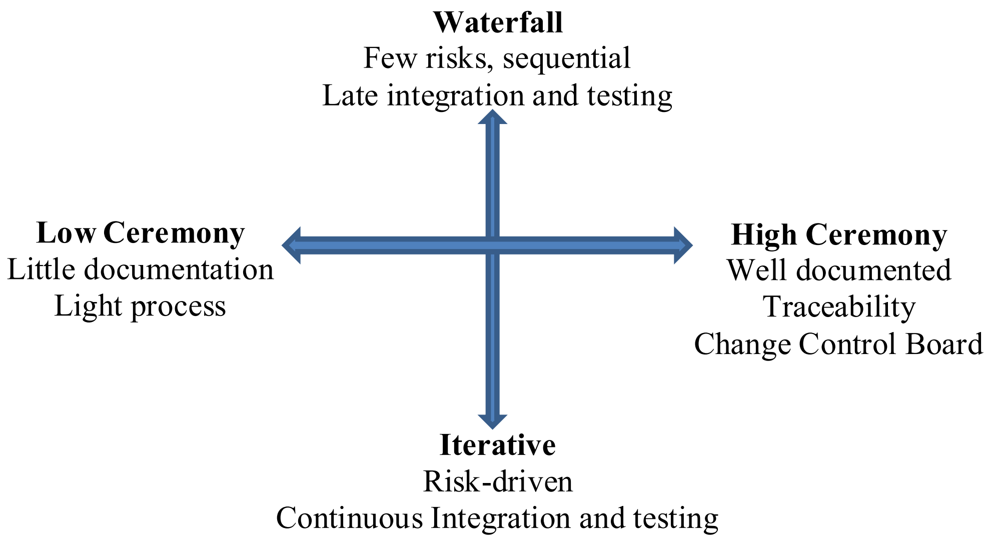

Figure 4: Process Map for Process Comparison[54]

Agile methodologies tend to gravitate in the lower left quadrant for low ceremony and iterative development.[55] CMM and CMMI feature more ceremony but can move toward iterative.[56] The USDP, and more particularly, the RUP, can be anywhere from the lower left quadrant to the lower right quadrant.[57] So, the question Kroll and Kruchten raise is regarding the desired level of iterative and ceremony. Of course, as advocates of the RUP, they recommend a more iterative approach. "Many projects can benefit from taking an iterative approach and using a RUP configuration that is low on the

[54] Kroll, Kruchten, 51.
[55] Ibid., 53.
[56] Ibid., 56.
[57] Ibid., 58.

Waterfall/Iterative scale."[58] Their assumption depends on the organization. Currently, FSC is primarily waterfall-oriented and high on ceremony. A common analogy used to describe how FSC deals with process change is to reference the swing of a pendulum. If FSC is in the upper right quadrant then it follows that the pendulum would swing toward the lower left quadrant. It is interesting that the technical leadership is saying that's where FSC is headed. Those colleagues that have been at FSC for more than a decade or two have all observed that FSC tends to invest heavily in major swings of the pendulum and then after a few years swing back the other way. This trend leaves FSC without the advantage of the middle ground. However, FSC will likely benefit and achieve its strategic goals by using more lean and agile practices. For future growth and improvement opportunities, the Lab expansion is being carefully expanded with watchful eyes from the sponsors and champions. So, if FSC is ready for change why didn't they adopt CMM or CMMI to improve processes or adopt the RUP? Waterfall has been the way projects have always been executed. Analogously, it is a long process to change a large naval vessel's direction. As such, champions for the other methodologies never maintained intensity or focus to ensure adoption. Referencing a recent briefing, CMM didn't work at FSC because it was too much process and documentation. There is uncertainty as to why the RUP was not adopted since IBM is a proponent of it and FSC is heavily invested in IBM products. IBM offered the RUP, but at a price, whereas other methodologies were either available for free, or there already was sufficient resident knowledge as to negate the need for additional professional services. If at all possible, FSC prefers to use a variety of vendors for flexibility in cost management.

[58] Ibid., 64.

Each of these methodologies brings something to offer FSC. From CMM and CMMI there must be discipline and maturity in process in order to improve. The USDP prescribes the use of certain tools such as the UML and use-cases to be effective in requirements and design. Also prescribed in the USDP, but articulated in the RUP, is the need for iterative development and more flexibility in ceremony. The agile manifesto seems very good but can FSC move that far and how fast?

Chapter 5: Opportunity Analysis and Recommendations

Having presented an adequate foundation, discussion can move to the specific opportunities for FSC to improve. FSC is currently engaged in process improvement initiatives for the various process areas of the project lifecycle. The analysis here provides some additional help for improving those areas discussed.

Software Development Lifecycle Management Revisited

Figure 3 in the last chapter depicted some generic phases of the software development lifecycle and some key artifacts produced in each phase. It can be inferred that there is traceability between the artifacts as one flows through the phases in the project lifecycle. The traceability is not entirely linear though as Figure 3 depicts. Figure 5 below shows the goal of traceability in a managed software development lifecycle.

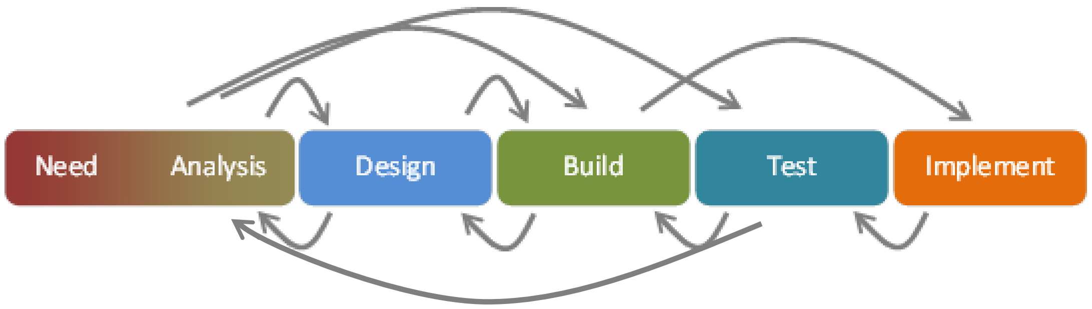

Figure 5: Software Development Lifecycle Traceability

Notice in Figure 5 that Need and Analysis are combined into a single image. This is because the artifacts generated from these phases are business driven and typically business owned, although that is debatable with respect to the analysis documents. Regardless of the viewpoint, the business need is articulated in full through those artifacts. The goal of a managed software development lifecycle (SDLC) is to automate as much as possible in these phases and provide complete traceability. An objective of this system is to have the ability to trace backward and forward amongst the key artifacts. “Forward traceability implies that it is possible to trace a requirement to elements in the outputs of later phases (design, coding) in the lifecycle.”[59] Forward traceability is illustrated in Figure 5 by the arrows on the upper side of the figure. Ideally there should be traceability from the need or requirement to the design, the code and acceptance or system test cases at a minimum. “Backward traceability…implies that is is possible to trace elements in the output to various stages back to requirements back to their origin.”[60]

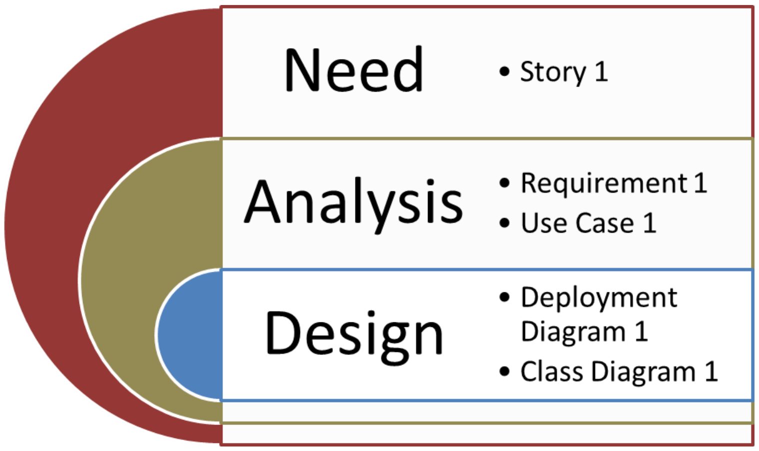

Figure 6: Software Development Lifecycle Artifact Relationship

Figure 6 above, shows how the artifacts might be related in such a managed system. Realistically, it likely would be a database of artifacts all related to each other so that one could pick an artifact and move forward or backward in traceability. In the case of Figure 6, a business need, such as a user story, can be traced forward to the requirements narratives or use-case models or descriptions needed to articulate that need

[59] Jalote, 58.
[60] Ibid.,

and then to design artifacts, which specify system components, modules or services that support and fulfill the requirements. The minimal end-state of an automated and managed software development lifecycle may provide this capability for all the artifacts that should live as part of a system. The user story should be considered just as critical as the code. As mentioned earlier, code is perceived by many at FSC to be the most important deliverable because the working software is what adds business value in reducing costs and generating revenue for the business. However, this perception may indicate laziness and an opportunity for improvement. Should the business have greater ownership and understanding about their systems?

As discussed in Chapter 2 business analysts work to ensure the as-is and to-be business process models are documented. Business and information technology personnel work together to produce numerous requirements artifacts. IT personnel work to produce design and code artifacts. Testers and business personnel work to produce test artifacts. All of these artifacts are typically stored in a project SharePoint site, except for the code, test data and executable test cases. During several of these phases, a number of artifacts are ear-marked for maintenance or additional artifacts are produced for maintenance. When the project implements, closes, and the warranty period passes, the project SharePoint site is archived to storage and is not easily retrieved. Those documents that are delivered to maintenance are stored into another SharePoint or some other storage mechanism and typically referenced when something goes wrong, which may not occur for some time. As observed throughout many projects, this cycle repeats itself. The stored documents often get deleted or consolidated or people think the knowledge is ubiquitous. After some period of time the new system becomes an older system. The people that knew its origins have moved on to different tasks. The maintenance teams that support the systems also have people leave and each iteration of movement causes erosion as less information is passed on to each set of new people on the team. The original architecture of the system may become unrecognizable as the original intent was lost or new developers inject, and then follow anti-patterns. The erosion in the architecture stems from project and maintenance teams constantly adding to the system to fulfill short term goals or fulfill requirements with minimal budget or schedule. Individuals struggle to understand past decisions, requirements, and the current state of the design and code. The business may complain that IT is not getting cheaper or that they can't get the enhancement they need fast enough. The original knowledge of the system is with individuals or within personal storage drives with notes and other information they may have documented. Certain components that have never needed changing may never have been studied, and remain difficult to understand and therefore difficult to change. It becomes difficult to determine what the system does. The argument is that the code tells what the system does. This is absolutely true. But, is there certainty the system does what was intended and how can this be verified? At a minimum, the ways projects are managed severely limit the ability to view and analyze diagrams and requirements to describe what the system is and what the system does. FSC seems to manage the lifecycle well enough but in order to achieve the stated time-to-market and quality goals, will the current way of doing things suffice? With the current way of doing business, changing the methodology won't solve the core issues that FSC faces to achieve those goals.

TRACEABILITY AND REUSE

Traceability starts with disciplined requirements management. It is interesting that in the requirements space there are several tools to accomplish the same task of requirements management. There is DOORS, RequisitePro, RSM, SharePoint and Version One. Each of these is being used to support some sort of requirements management in some area of FSC. Why are there so many tools to do the same job? Which tool is best or is a new tool needed? Ultimately a requirements management tool is needed but is FSC willing to do what it takes to realize the stated benefits. For example, DOORS provides a way to document and version requirements. The access and modification of the requirements is done through the tool interface. Like RequisitePro the adoption of the tool is limited to those with the wherewithal to use them. Most recently, FSC is adopting IBM's Rational Team Concert to hopefully be the only requirements management tool. According to Ian Alexander and Ljerka Beus-Dukic there are reasons for and against using an RM tool:

- For: Database manages versions and configuration (sets of documents)
- For: Traceability between related items in different documents may prevent or solve many requirements issues
- For: Status information (e.g. priority, review status) is easy to keep with each requirement
- For: Central database permits many people on different sites to work together.
- For: Project knowledge is held securely
- Against: Additional Training needed for project staff
- Against: Initial setup, design of data structures, and importing of data can slow down project progress
- Against: Exporting and formatting may be needed to create documents for review
- Against: Integration with other tools may be difficult or costly
- Against: Danger of becoming 'locked in' to a specific vendor[61]

[61] Alexander, Beus-Dukic, 420.

Each of these bullets raises a good point for and against a RM tool. Ian Alexander stated, “Tools are no use if they are worn around the neck as amulets. Having the best requirements tool does not of itself solve any part of the requirements problem.”[62] Requirements could be captured without tools and FSC could manage with paper and pencil and filing cabinets; however, it is not practical. Process effectiveness is greatly improved with tooling. One issue is a lack of integration amongst the tools used at FSC. There are tools that can do one thing well but then there is an arbitrary way of correlating all artifacts in the system.

While traceability starts with requirements management it must extend beyond the requirements artifacts. FCS has tools to handle requirements management but what is lacking is the ability to easily trace design, code and test artifacts to requirements in a way that these artifacts live for the life of the system not just for a particular project. FSC needs to focus on those artifacts that actually add value and are adequate enough to convey intent and meaning for the various parts of the system. For analysis, user stories, business process flow diagrams, use-cases and their corresponding narratives and models, and user interface models should be sufficient. For design, FSC should adopt the UML as the standard for communicating key design models. At a minimum FSC should adopt the deployment diagram, the component diagram and continue use of the class and sequence diagrams. This would help alleviate variability in design reviews particularly in architecture at FSC. Supporting word documents, power point presentations and custom diagrams will be used as needed but should only support the primary models which should be versioned along with the requirements and the code. For code, it is a bit more difficult to achieve traceability back to design artifacts and requirements. The

[62] Ibid., 411.

requirement should not necessarily be documented with the code through an arbitrary requirement name or some id as a comment but it would not be detrimental. The version control system must store comments and change sets such that they are tied to a requirement much like a request for change in the current system. Test case code, test data and supporting narratives would also be checked into a version control system that would likewise connect to a requirement. Once again, this calls for an integrated system rather than separate independent tools to manage the artifacts and process.

If FSC invested in such an integrated system much could be accomplished. Imagine a system that allowed projects to annotate the business process and requirements for those things they are changing or adding. Parallel projects would then be able to identify and handle collision at the onset of the project rather than at integration or implementation. Architects could quickly identify the needed components to be modified and the interfaces or those things they consume and provide to likewise be modified. The impact assessment would be a relatively trivial task. The analysis and design tasks would be laser focused on the to-be state of the project rather than on re-discovering the current state. How much time could be eliminated from the analysis and design phases of the project lifecycle? It would be significant but the investment to attain such an integrated system is by no means a small thing. The cost to implement systematic traceability is of course a concern. First, there is the cost to initialize the traceability. Second, there is the cost to maintain the traceability over time. As the size and complexity grows for an application and business problem being addressed, there is a greater granularity and effort needed in the traceability strategy. The advantages can more than offset any costs of implementation, helping FSC to:

1. Align changing business needs with current projects
2. Reduce risk by capturing knowledge that's vital to a project's success

3. Insulate against staff turnover
4. Perform impact analysis over new, changed or deleted items
5. Track projects as inputs into repeatable process improvement

As traceability is not a new concept, FSC should revisit how it could benefit from a more integrated system approach to help achieve its strategic goals for lower time to market and increased quality in project execution.

Renewed Emphasis and Commitment for Analysis and Analysts

Good design, code, and test cases rely on good requirements. Good requirements rely on good requirements elicitation. "Requirements elicitation remains one of the most important and challenging steps in systems analysis and design."[63] A difficulty at FSC is that how well requirements elicitation is executed is variable across the organization. "Clearly, as the range of information systems applications and development methodologies, tools, and techniques expands, effective communication becomes an increasingly critical imperative for effective requirements elicitation."[64] The role of an analyst is dependent upon who the person is and what area they are in. In some areas, an analyst is almost a glorified scribe and therefore the role is greatly diminished in the eyes of management. In other areas, the analyst is a purist who owns the requirements engineering discipline through and through. As discussed in the introduction, there is a need for highly skilled people in the role of analyst at FSC whereas the current trend is diminishing the role's importance. This erosion of the role and discipline occurs because of consistently lowered expectations for what could and should be done. With Agile, the desire is to minimize even more what documentation is created in order to move faster, but this limitation is precisely what gets FSC into trouble in the long term. There is no

[63] Chiang, Hardgrave, Siau, 21.
[64] Ibid., 22.

value in creating a bunch of documents for the sake of a process; FSC needs to create the right amount. Without highly skilled analysts this will not be accomplished. Left up to developers, no documentation would get created, except maybe until after the project is complete, because documentation requires motivated team members and developers are usually not highly motivated to create documentation.

In order to have highly skilled analysts, FSC needs to have people with understanding of the business domain or the systems or, better yet, both. Figure 7 below illustrates this need. In section (a), "requirements are known and understood by both the user and the analyst due to their shared prior experiences in the domain of interest."[65] In section (b), the analyst has knowledge that the user does not.[66] In Section (c), the user has knowledge that the analyst does not.[67] In Section (d), neither the user nor the analyst has the knowledge.[68] In the ideal state the user makes requests as requirements; this accounts for section (a) and (c). The analyst will ask the right questions to also account for section (a) and cover section (b). Section (d) is the risk that needs attacking, but how can FSC do this without highly skilled analysts?

[65] Ibid., 26.
[66] Ibid.,
[67] Ibid.,
[68] Ibid.,

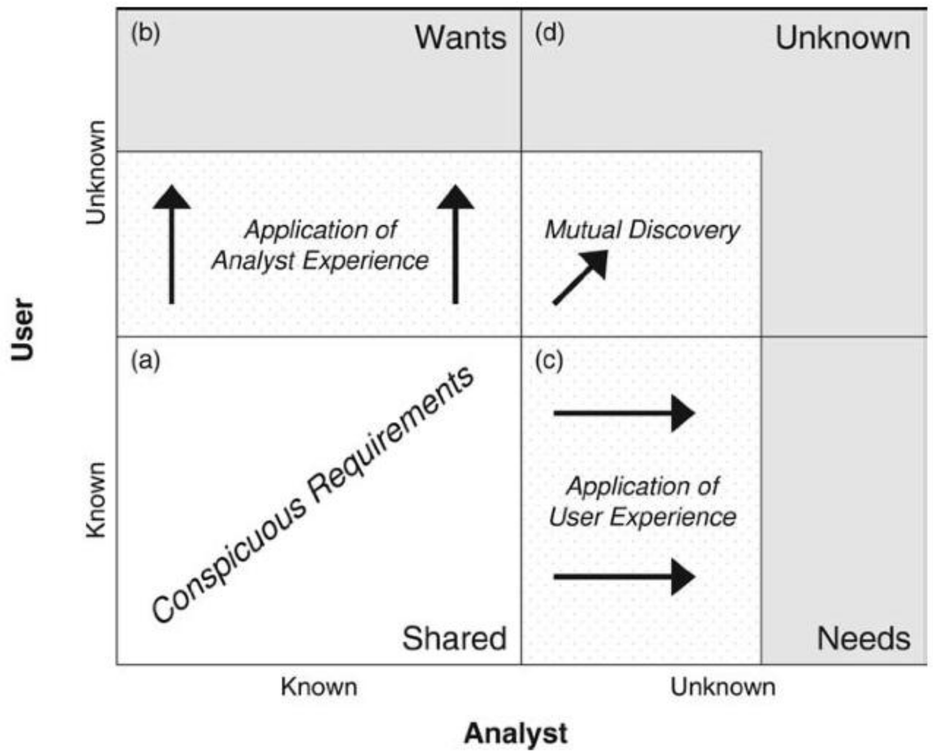

Figure 7: The Requirements Appreciation Model

FSC needs highly skilled systems analysts. They need experience. "No architectural firm would dream of putting a new graduate to work alone designing a bridge or an office building, yet these systems are in many ways far simpler than the information systems the systems analyst/designer is routinely called upon to design."[69] FSC should not put people in an analyst role on either IT or business if they have not first apprenticed on some projects and gained some domain and system knowledge. In other words, no one should be an analyst who has not been in the area for some time and gotten their feet wet and their hands dirty on some projects before taking the role. They need training. If FSC is to ever have a managed software development lifecycle, the analyst will have to be fluent in the tools and methods available. Based on current staff, this will

[69] Gerald M. Weinberg, *Rethinking Systems Analysis and Design*, (Dorset House Publishing, 1988). 23.

not be easy, particularly if FSC is going to adopt a new tool, but there are some basics that analysts should be able to do with or without tools such as:

- Have a command of both written and verbal English
- Know and understand Use-cases
- Have a basic knowledge of the UML (at least for technical analysts)
- Have a working knowledge of FSCs processes
- Actively participate in an analyst community

Systems analysts should have technical backgrounds. They need to be participatory in the design as well as the requirements management process. FSC may choose to make the analyst the project librarian or it could realize the potential benefits in having skilled analysts. No team is repeatedly successful without good analysis. No team is successful without delivering quality software. Let FSC allow developers to be good developers and analysts to be good analysts and stop asking developers to be really good at both disciplines.

OVERALL PROCESS

With regards to process, the methodology doesn't really matter as long as it is managed with more maturity and automation. Standardization should be a primary goal for FSC's process organizations. The issue is that different areas of the company do things differently. There is the bank way and the insurance way and the investments way and the enterprise way. Each has variations and each has its people inculcated into the "right" way of doing business. The truth is, each pocket has just gotten seasoned at doing things their way. As mentioned earlier, FSC does not document requirements the same way across the company. The role that business and systems analysts play on a project vary by what team or area they are in. Currently some projects run under Agile methodology but more projects run under Waterfall methodology. The technical leadership wants to move toward agile and yet some believe there will always be pockets in the company for running according to their preferred methodology. Diversity in process will not lead to consistency in execution.

Process implications are paramount to the success of project execution. With the growing support of Lean-Agile and the Lab Agile concepts at FSC, there should be concern that the lure of minimal documentation and light process will not provide a strategic competitive advantage. One of the tenants of Agile is that there is a dedicated and highly motivated team. There is also the assumption of the ability to improve estimation and delivery performance with Agile. The assumption is that team composition is fairly constant and consistent and that that the team has domain and or system knowledge. In the first case, it is impossible for team members to rely on a velocity if there is a lot of churn in the core team membership, particularly developers for the build phase of a project. Velocity is the relative speed by which the team can deliver function points or stories for a period of time. In the second case, a highly successful team will take an unknown period of time to become as successful if they were transplanted into a new subject area or unfamiliar system. Once again, domain or system knowledge is essential to be effective in execution. Waterfall, which is the dominate methodology at FSC, has its shortcomings, but the business does like it. Waterfall can provide a clearer picture to the business. It can tell what will be delivered, when and relatively for how much. The issue in the near term operating state is that FSC is going to have agile project teams and waterfall project teams, processes for waterfall and processes for agile. The human issue will require support teams to bounce back and forth between methodologies. This may only be a temporary issue; really FSC needs to have one way of doing things with various ways to tailor the process for unique project situations.

People and Organization

Regardless of tools or processes, FSC needs the right people doing the right things at the right time. It seems every few years there is an executive at FSC who champions an effort to bring in a new process or tool that will drastically improve the performance of the company. Management needs to champion efforts but the efforts need constant and consistent championing and grass roots support. When the management champion leaves, efforts have a tendency to die rather quickly as people slip back into what is comfortable and what known. But, FSC cannot put all faith in a process such as Agile to solve its problems. FSC needs to attack some basic fundamentals in how it executes projects.

Concluding Thoughts

The software quality assurance (SQA) team at FSC fills their time with checking project shares to ensure that projects have followed the process. They are checking for whether certain artifacts were created but they are not checking the quality of the artifacts. They don't have the tools or time to do that. When FSC says a project meets SQA controls what is really being said? They checked all their checkboxes but were their requirements good? Were their design documents good? Was the code good? Are the test cases good? FSC relies on defect counts, project burn rate and scope delivered at the projected dates to determine if a project was successful. While working software in production is a success, will that system continue to be successful or will it become a metaphorical nightmare? If FSC develops great systems but provides no context through the many artifacts needed to execute projects to build systems, how can FSC continue to ensure their greatness? In the many improvement efforts over recent years, FSC has first focused on minimizing production defects. Now FSC focuses on minimizing test defects. FSC is working from the end of the lifecycle to the beginning. But, to achieve higher quality much faster, FSC has to start at the beginning.

> If we did an accurate accounting, we'd find our investment in all this knowledge on paper is far greater than our investment in software or hardware. But greater still—and still less recognized as part of the system—is our investment in the knowledge tucked away in people's heads…With a broadened concept of the system, and a consequently broadened concept of development, we might see this thrashing period grow shorter and less violent.[70]

The thrashing period Weinberg refers to is the analysis and design activities on a project. On every project, there is an assessment on the current state, the future state, and how the team will get there. If FSC wants to achieve faster time to market and higher quality it needs to better manage the software development lifecycle and it should start with how it manages analysis.

[70] Ibid., 40.

Bibliography

Alexander, Ian, Beus-Dukic, Ljerka. *Discovering Requirements: How to Specify Products and Services*. England: Wiley, 2009.

Alexander, Ian, Neil Maiden, Suzanne Robertson, "What Influences the Requirements Process in Industry? A Report on Industrial Practice," *Requirements Engineering: 13th IEEE International Conference* (2005): 411-415.

Arsanjani, Ali. "Empowering the business analyst for on demand computing," *IBM Systems Journal* 44.1 (2005): 67-80.

Babcock, Jonathan, "Finding a Home for Business Analysts," *PracticalAnalyst: Practical insight for Business Analysts and Project Professionals* (blog), July 23, 2007, http://practicalanalyst.com/2007/07/23/finding-a-home-for-business-analysts/.

Bell, Simon, Trevor Wood-Harper, *How to Set Up Information Systems: A Non-specialist's Guide to the Multiview Approach,* Sterling, Virginia: Earthscan Publications, 2003.

Bentley, Lonnie D., Jeffrey L. Whitten, Kevin Dittman, *Systems Analysis and Design Methods* 6. ed., McGraw-Hill, 2004.

Beyer, Hugh R., Karen Hotzblatt, "Requirements gathering: the human factor," *Communications of the ACM* 38.5 (1995): 31-32.

Biffl, Stefan, Michael Halling, Paul Grunbacher, "An economic approach for improving requirements negotiation models with inspection," *Requirements Engineering* 8.4 (2003): 236-247.

Blash, Deen, Shane Mcgraw, "CMMI on the web," (webcast), *SEI*, October 8, 2011, http://www.sei.cmu.edu/library/abstracts/presentations/20080925webinar.cfm.

Broache, Anne. “IRS trudges on with aging computers,” *CNET News*, April 12, 2007. August 14, 2010. http://news.cnet.com/IRS-trudges-on-with-aging-computers/2100-1028_3-6175657.

Chiang, Roger, Bill C. Hardgrave, Keng Siau, *Systems Analysis and Design*, Armonk, New York: M.E. Sharpe, Inc. 2009.

Cobb, Charles G., *Making Sense of Agile Project Management*, Hoboken, New Jersey: John Wiley & Sons Inc., 2011.

Jacobsen, Ivar, *The road to the unified software development process*, Cambridge, United Kingdom: Cambridge University Press. 2000.

Jalote, Pankaj. *CMM in practice: process for executing software projects at Infosys*, Reading, Massachusetts: Addison Wesley Longman Inc. 2000.

Krebs, Jochen, *Agile Portfolio Management, Redmond, Washington: Microsoft, 2009.*

Kroll, Per, Phillippe Kruchten, *The Rational Unified Process Made Easy: A Practitioner’s Guide to the RUP* Boston, Massachussetts: Addson Wesley. 2003.

Kruchten, Phillippe, *The Rational Unified Process An Introduction*, 2. ed., Reading, MA: Addison Wesley Longman Inc., 2000.

McMenamin, Stephen M., John F. Palmer, *Essential Systems Analysis,* New York, New York: Yourdon Inc., 1984.

Rinzler, Ben. *Telling Stories: A Short Path to Writing Better Software Requirements*. England: Wiley. 2009.

Etien, Anne, Colette Rolland, Camille Salinesi, “Eliciting gaps in requirements change,” *Requirements Engineering* 9:1 (2004): 1-15.

Schaffner, Michael. “Let’s Get Down to Business,” *Beyond Blinking Lights and Acronymns Mike Schaffner on Managing Information Technology and Your IT Career* (blog), February 14, 2007, July 16, 2010.

http://mikeschaffner.typepad.com/michael_schaffner/2007/02/lets_get_down_t.html.

Tattersall, Glenn. "Supporting Iterative Development Through Requirements Management," *IBM developerWorks*, October 15, 2002, August 25, 2011, http://www.ibm.com/developerworks/rational/library/2830.html.

United States, "Internal Revenue Service Business Systems Modernization Program Progress Report," *Internal Revenue Service*, September 1, 2000, http://www.irs.gov/pub/irs-utl/bsm-prog.pdf.

United States, "Computer Systems Analysts," *Bureau of Labor Statistics,* June 2010, http://www.bls.gov/oco/ocos287.htm.

Wasson, Kimberly S., "A Case Study in Systematic Improvement of Language for Requirements," *Requirements Engineering: 14th IEEE International Conference* (2006): 9-18.

Weinberg, Gerald M., *Rethinking Systems Analysis and Design*, Dorset House Publishing. 1988.