Modeling Requirements: The Customer Communication

Dan Matheson
Department of Computer Science
Colorado State University
Fort Collins, CO, USA
dan.matheson.co@gmail.com

***Abstract*—This paper describes the results of six years of experience in eighteen real-world projects in two companies of using models, primarily Unified Modeling Language-based models, to gather, review, record and communicate product definition solution requirements for customers in the medical device industry. The motivation for this approach was a response to ineffective projects that documented requirements in the traditional text format of "The system shall…". In an attempt to both increase the quality of communication and fit with the rapid cycles of an agile-based project structure, a primarily graphical-based technique was used. The use of multiple models for the business requirements versus a single text document made communication across many business groups easier and decreased review time dramatically. The ISO RM-ODP standard was used to keep the focus of the business requirements models at the Enterprise, Information and Computation viewpoints. The use of multiple graphical models emulates the techniques of architects as they extract, document and present a building's requirements. The graphical modeling-based formats and agile-based project techniques of this approach produced successful in-production solutions, which have been validated to FDA compliance standards.**

***Index Terms*—Requirements, Modeling, Agile Process**

I. Introduction

This paper describes the use of models, primarily Unified Modeling Language-based (UML) [1] models, to gather, review and record solution requirements for customers in the medical device industry. The graphical modeling-based formats and techniques have been used in eighteen real-world projects in two companies and have produced successful solutions, which have been validated to US Food & Drug Administration (FDA) compliance regulations. The success is not only measured in solutions used daily in several businesses, but also in a more effective project process.

There were several motivating factors to this work. The primary factor was the experience with many unsuccessful projects with poor and incomplete requirements. Some of the reasons for poor requirements arise from one-dimensional communication with the customer, a documentation format that paints an incomplete picture and requirements at the technology implementation level. It was felt that there must be a better way.

A second motivating factor was the desire to make the solution development process more effective. The term "effective" has both quality of work and efficiency of working implications. A major source of displeasure with past projects was the perceived excessive amount of rework in the end phase. The question of "*Why can't we get it right the first time?*" nagged.

A third motivating factor was to establish a set of first principles and build modeling expertise that can be applied in new areas of consulting work. As one of the senior solution architects it was my responsibility to both develop more effective project processes for our company and to educate the other employees.

Engineers model problems, solutions and designs in order to gain insight, innovate and communicate with the customer [2]. Software engineers should be able to work to the same engineering best practices. Addressing customer communication problems via modeling techniques was a major goal.

The organization of the paper is as follows: the second section covers development of the requirements modeling hypothesis, the third section describes the use of models in the projects and how they were communicated with important lessons learned, the fourth section covers some metrics collected during the projects and comparisons to non-model driven projects, and finally some future research directions will be discussed.

The examples presented in this paper are true to form, but customer proprietary details have been hidden or abstracted.

II. The Requirements Modeling Hypothesis

The hypothesis is that requirements can be modeled for the domain of a Product Lifecycle Management-based solution for the medical device industry. This is a pretty generic statement, therefore some work was needed to setup a software engineering framework in which to test and evaluate the hypothesis. The framework was also needed for the education of the other team members on the project.

A sub-hypothesis is that modeling and the use of graphical requirements specifications should improve communication. One of the more frustrating aspects of prior projects was the

large amount of time wasted in discussions about the priority of one requirement over another.

The development of the software engineering framework for the requirements modeling was started by going back to square one. The basic assumption is that poor requirements caused many of the problems. Some of the characteristics of poor requirements are:

- Incomplete
- Incorrect
- Inconsistent
- Non-essential
- Not at the business level
- Too numerous for a person to comprehend.

In addition to the poor requirements a lack of software engineering First Principles to guide the work was missing. Other engineering disciplines have first principles, encoded into best practices, which can be used by the engineer as a touch stone to monitor the development processes. The establishment of a set of first principles was needed to explain the rationale behind the modeling choices, ensure that the team addressed the more fundamental questions first and to provide a foundation for future work.

A. Software Engineering First Principles

The first component in the software engineering framework is a set of first principles to be used as an external guideline and evaluation criteria to ensure the project was as effective as possible.

First Principle: Conceptual Integrity.

In his book *The Mythical Man-Month* Frederick Brooks [3] describes the difficulties in building large complex solutions. There are two points he makes that are critical to having a successful solution:

"The hardest single part of building a software system is deciding precisely what to build."

"I will contend that conceptual integrity is the most important consideration in system design."

Communicating and recording "what to build" and the solution concept are the primary purposes of the requirements specification.

Establishing the concept of the solution very early will provide a test metric for inconsistent, non-essential and incomplete requirements. The question "Does this support the solution concept?" can be posed for each requirement.

First Principle: Change the language to innovate.

In establishing the software engineering framework a conscious effort was made to reject standard vocabulary to gain new insights, for example using the term "Software CAD Tool" in the project, not a "UML editor".

This principle was used in the customer project to stimulate innovative thinking during the solution development, for example "reusable product definition information entity" rather than "document".

First Principle: Structured Approach.

The Open Distributed Processing - Reference Model (RM-ODP) (ISO 10746-1) [4] provides a useful framework for organizing the progression from business requirements through solution realization. The RM-ODP model provides five levels of viewpoints that partition the thinking about a solution. The levels are consistent with the modularization guidelines from Parnas [5]. Each of the viewpoints provides a different focus for looking at the solution for a specific area of essential understanding. The five viewpoints are:

- Enterprise viewpoint: A viewpoint on a system and its environment that focuses on the purpose, scope and policies for that system.
- Information viewpoint: A viewpoint on a system and its environment that focuses on the semantics of information and information processing.
- Computational viewpoint: A viewpoint on a system and its environment which enables distribution through functional decomposition of the system into objects which interact at interfaces.
- Engineering viewpoint: A viewpoint on a system and its environment that focuses on the mechanisms and functions required to support distributed interaction between objects in the system.
- Technology viewpoint: A viewpoint on a system and its environment that focuses on the choice of technology in that system.

The Enterprise, Information and Computation are the viewpoints that can be found in business requirements.

Frist Principle: Work from stability to variability.

Catalog the problem according to the stability of the business situation. We know that business conditions and technology change over time. However, each situation has areas of higher stability. Identify those areas and start the requirements modeling in those places first.

Experience has shown that brainstorming requirements only leads to an unstructured list. The unstructured list is often organized on a personal perception of must, high want and want classification. This principle is intended to short-circuit the extended and wasteful arguments that happen when prioritizing requirements.

First Principle: Symmetry of Action.

This principle is used to help create a complete set of requirements. For example if there is an action to create something, then the action to delete it should be listed as a requirement. It can be determined that the delete action is improper, but then that rationale is captured to prevent wasted time revisiting the question.

First Principle: Apply Patterns.

Apply patterns at all levels of the work. There are patterns within the requirements to help organize and communicate. Patterns help to ensure completeness of the requirements gathering process. Be prepared to recognize and use new patterns applicable to the domain.

First Principle: Model.

As with all engineering disciplines, model as much as possible. Use as many modeling techniques as needed to fully document and communicate the requirements, design and

realization of the solution. The UML can handle many of the needs of modeling requirement, but not all so additional techniques were used.

B. Solution Domain Constraints

There are many different business domains for which software-based solutions or products are developed. The differences are reflected in the requirements. This paper addresses one specific business domain. In the final section both the generalization of the approach described here and additional domains will be discussed.

The solution domain that is the target of this approach is the management of product definition knowledge for medical devices across the full product life-cycle.

Upon closer inspection several constants, constraints and limitations arise. These can be used to select the modeling techniques most appropriate for the requirements and design needs for the solution. Some of the previous listed first principles can help with the identification.

Solution Constraint: Information Centric.

The most significant constraint is that the solution is centered on the creation, evolution and management of product knowledge. Most of the companies in this business domain view themselves as product leadership companies [6]. This means that their product knowledge is held as extremely valuable intellectual property. Their objective is to produce the best products in the world, thereby ensuring their business success and the satisfaction of the employees through helping people.

The recognition that information is centric to the solution forms the primary stability point for this domain.

Solution Constraint: Regulated Industry.

This work was developed for solutions in the medical device industry. The focus was on managing the information created during product development and continuing on until end of sales and support. Much of the product definition information is subject to review and constrained by laws and regulations by various health agencies such as the FDA. The laws and regulations form a set of business requirements, such as information that must be collected and processes that must be followed.

C. The Modeling Process

Of equal importance to the models is the process by which the models are developed and verified with the user. An incremental and iterative process (agile) was used to develop the models as well as get verification feedback. The process approach leveraged ideas from extreme programming, scrum development and other agile software development processes [2][7].

A key component of the communication with the customer is the order in which the requirements are developed. The requirements gathering process used on the projects described in this paper made use of the work from stability to variability first principle to order the development of the requirements models.

III. The Models and The Process

The above first principles and constraints were used to select the modeling artifacts used to capture the business requirements and to guide the process. The models governed by the UML provided a starting point for the set of models to be considered, although the UML does not cover all the needs.

The RM-ODP standard was the foundation for the process to generate the modeling artifacts for the requirements. Figure 1 shows how the RM-ODP framework was used to select the modeling approaches.

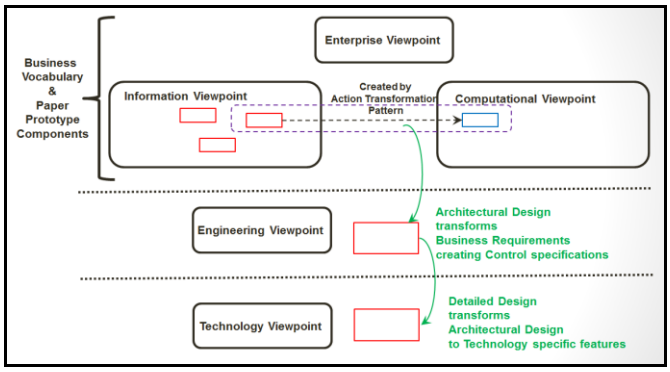

Figure 1: RM-ODP Framework for Model Artifact Selection

A. Paper Prototypes as Light-weight Artifacts of The Models

A waterfall-style project with a large text-based requirements specification does not allow an iterative and incremental gathering of the requirements. An iterative and incremental requirements gathering process was desired, so that as requirements stabilized the design of the realization could be started. The project process was divided into three sub-phases for development of the solution with additional sub-phases for testing and deployment (not shown), as shown in Figure 2.

Figure 2 was used with the customers to explain the process. There was a natural convergence on the paper prototype artifacts that enabled the decision to move forward. The sub-phases for development have the following purposes:

- Paper Prototype Purpose
  - Quickly creates a general consensus on the direction and boundaries of the business problem to be addressed.
  - Establish conceptual integrity and provide a fast litmus test that the project remains on course.
  - A light weight method to capture the major business requirements in a graphical manner that promotes communication.
  - Allows for fast iterative incremental requirement specification.
  - Provides for incremental verification of primary business requirements.

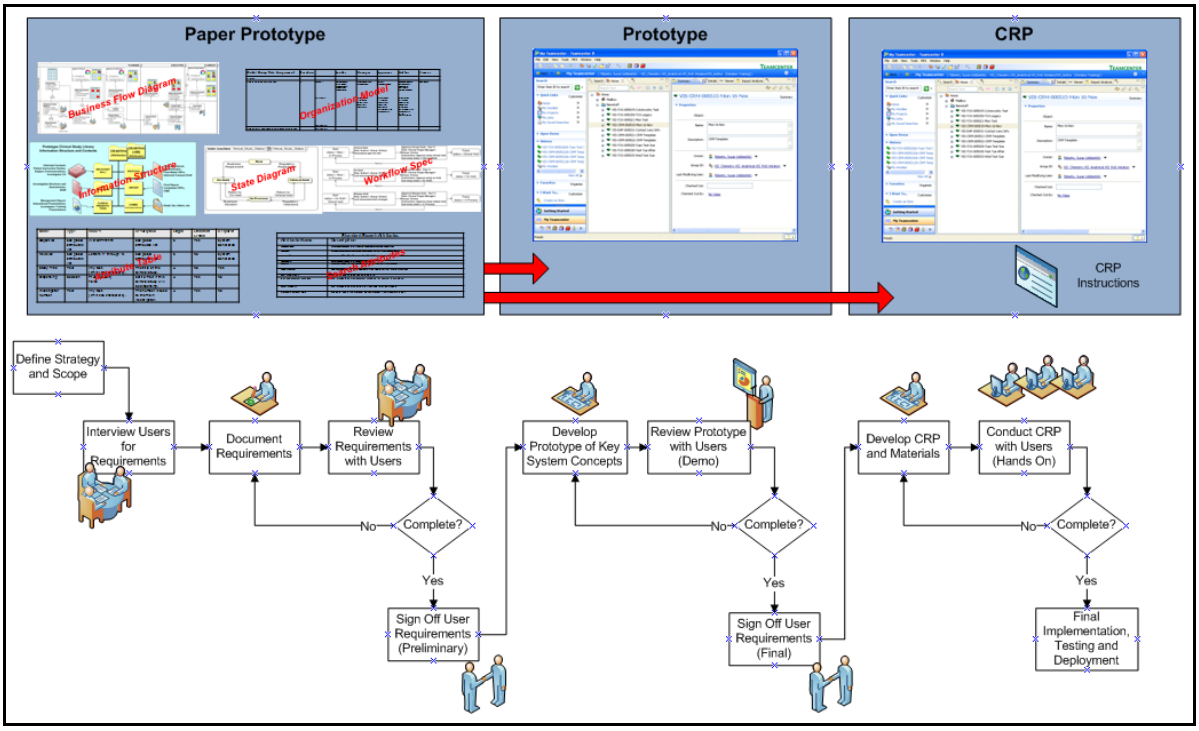

Figure 2: Project Development Phases Overview

- Conference Room Pilot (CRP) Purpose
  - Hands-on verification of the solution realization by the business users.
  - Final verification before completion of the implementation, testing and start of deployment to production.

The paper prototype consists of seven artifacts that specify the business requirements. The name comes from the light-weight representation that could be updated in a few minutes and often was real-time updated during the verification review meetings. The specific paper prototype artifacts will be described in the sections below with the corresponding RM-ODP viewpoint.

B. The Enterprise Viewpoint and Conceptual Integrity

The RM-ODP Enterprise Viewpoint establishes the primary business purpose or benefit for the solution. The primary business purpose is equivalent to the principle of Conceptual Integrity as espoused by Brooks [3]. The Enterprise Viewpoint becomes the touchstone during design and development for determining whether the project is on track to deliver the desired goals. When in doubt the question "Is this feature essential to realizing the goal and ensuring the conceptual integrity?" can be asked of any task or proposed feature.

The conceptual integrity is captured, expressed and communicated in two formats. The first is via a Project Mantra. The mantra for the medical device projects where this approach was used is "*File it, Find it, Reuse it*". Most medical device companies have a business discipline of Product Excellence. This means a combination of supplying both the best products in their segment and delivering innovative products faster.

Since many new products are evolutionary rather than revolutionary much of the work of the predecessor product can be reused in the successor, if managed properly. The business value concept is the value of reuse over recreating. In a paper-based business world this is extremely difficult to accomplish, but when the business is supported electronically, then much can be accomplished.

When the product definition information is *Filed* electronically, the business can *Find* it quickly, then effectively determine whether it can be *Reused* in the new product. For example, if the business can reuse a material Cytotoxicoloy study rather than recreate it, the business can save about $50,000 and four months in the development cycle.

The mantra for medical devices implies the importance of what is being created: modeling the information.

The second part of the Enterprise Viewpoint and format for supporting, expressing and communicating the conceptual integrity is the Business Flow. The business flow describes how the business operates to create and manage some part of the product definition, example in Figure 3. The flow starts with nothing and completes with an approved piece of the product definition. In the business flow the work of creating, verifying, approving, filing, finding and determining reuse can be seen.

The flows or processes that a business uses to develop and manufacture a medical device are carefully reviewed and monitored by the FDA and other health authorities. Beyond meeting the FDA constraints with process models that can be easily reviewed, a good modern business is always trying to improve through continuous process improvement.

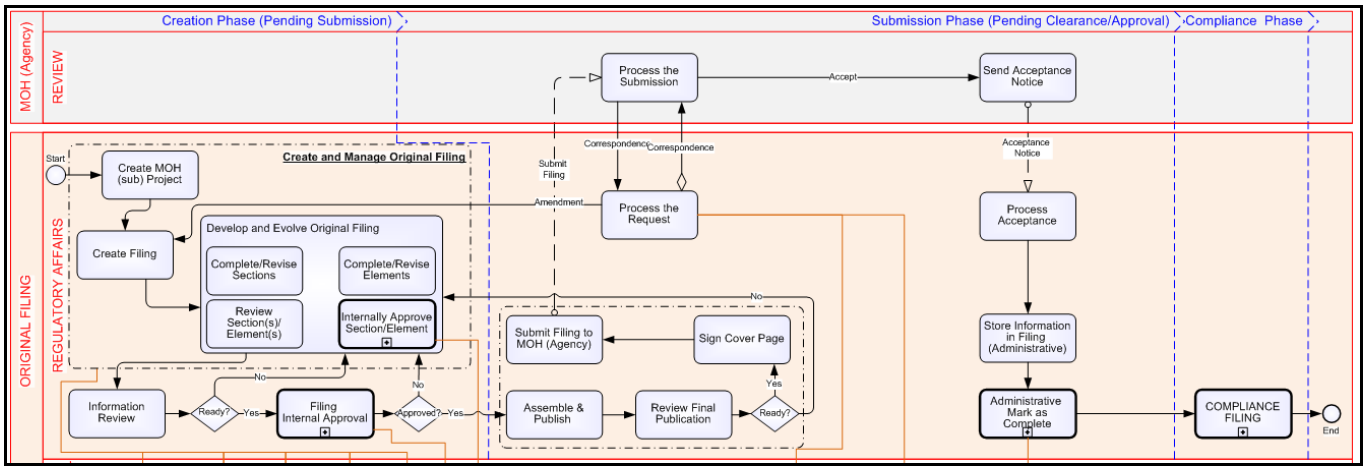

Figure 3: Example of the Business Flow for a Regulatory Filing

In every project done to date the business flow or its equivalent did not exist. The creation of the business flow brought people together, often for the first time, to discuss how the business operated and how it should operate in the future.

The business flow becomes the touchstone for the proper enterprise-level behavior of the solution: modeling the processes.

The business flow is one of the main paper prototype artifacts. It is based on the UML Activity Model. The customer presentation is rendered in Microsoft Visio® and softened from the output of a UML Computer Aided Design tool. The presentation softening was an important learning early on and applies to other UML–based models. Internally the project team used the more robust UML format.

1) The 1-pager: The full Enterprise Viewpoint is presented in an engineering drawing called the **1-pager**. The 1-pager was incrementally developed and became the stake in the ground for starting all review meetings. A quick review of the 1-pager was a verification that all the team members were on the same page.

A 1-pager example is shown in Figure 4. There are five components to the 1-pager: the business flow, the associated information model (abstract view), state machines for the information model components, use cases related to the business flow and text notes for clarification.

Experience showed that formatting the 1-pager to fit on an 11x17 inch page was the most useful. Copies would be left with the customer for review while the team was at home. Because the 1-pager was a single sheet, the edits were quickly incorporated and conflicting edits easily high-lighted for discussion. An unexpected reviewing benefit occurred when the 1-pager was laminated. We could walk the laminated version to individuals, record their comments with whiteboard makers and move on. This resulted in more flexibility in scheduling reviews, the one-on-one reviews were often faster and the business reviewer felt their concerns were taken more seriously.

C. The Information Viewpoint

The obvious choice for modeling the information requirements in the Information Viewpoint is the UML Class Model. The UML Class Model allows all of the important information components to be captured: information units, relationships between information units and the properties that both distinguish one unit from another, but also making finding a unit efficient.

With the customer the terms ”information unit” or “information entity” were used rather than a software centric term like “class” or “object”.

In gathering requirements the first principle of “work from stability to variability” applies strongly for the Information Viewpoint. Experience in developing solutions for the medical device industry has shown that the product definition information is the most stable point of requirements agreement. Developing the information model becomes the initial requirements stake in the ground.

There is interaction between the information model and the business flow. As the information components of the product definition are discussed, it is natural to discuss the order in which they are developed. The information development order leads to positioning tasks in the business flow.

The graphical display of the information units in the UML Class Model has proven to be a powerful communication mechanism. There is no ambiguity on the units and their properties. The relationships between the units are clear, including the type of relationship such as peer-to-peer or parent-child. A single image can replace many tens of textual pages. There are reasons why engineers use engineering drawings to specify parts.

However, some people have difficulties with the UML Class Models when they are delivered as UML Class Diagrams. To overcome UML diagram difficulties the graphical display softened the geometric shapes of UML Class Diagram classes, added color and other representative images. The modeling integrity and rigor behind the presentation is maintained internal to the development team.

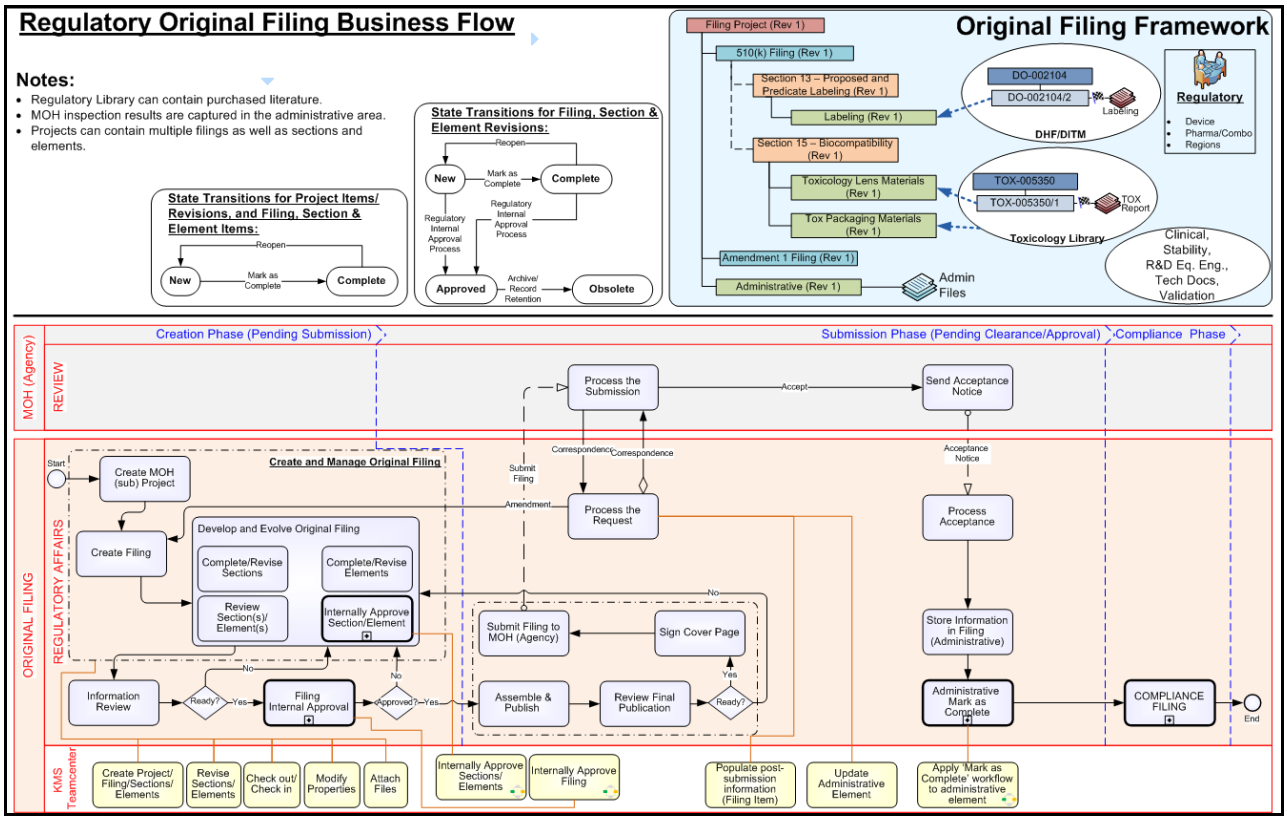

Figure 4: 1-pager Example for a Regulatory Filing

There are two features of a UML Class Model that are not used when modeling requirements; the generalization relationship and methods. For requirements modeling only the classes (units of information) that are instantiated are of importance. Any higher-level abstract classes that result from generalization will only appear via design activities. Experience early in this approach only resulted in confusion for the business users when generalization was used.

The second feature that is not used for requirements are method definitions. The class models at the requirements level have no need of methods. The method names and signatures are code or close to code specific. At the requirements level the coding language design decision has yet to be made. In the case of the experiences with PLM products and medical devices, code might not be created.

Figures 5 and 6 illustrate two styles of recording and displaying the information models. Each supports different discussions and each is a paper prototype artifact.

The UML Class Models show the properties for each class, but do not have the details needed for good requirements specifications. The class models for each type are supplemented with a Microsoft Word™ table as shown in Figure 7. The table defines additional characteristics for each property beyond the type. The key column in the table is the "Description" column, which holds the rationale for the existence of that attribute. Experience has found this to be the clearest way to communicate with the business users on this important data.

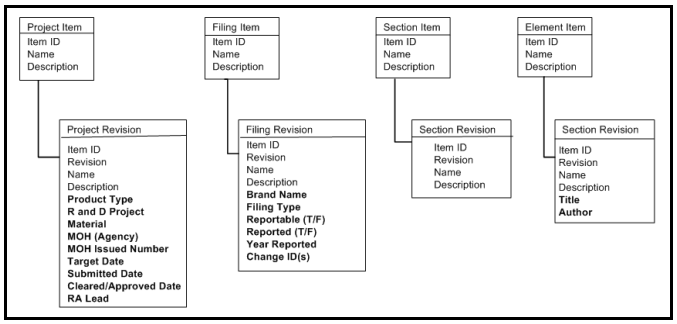

Figure 5: Information with UML-style Basic Classes

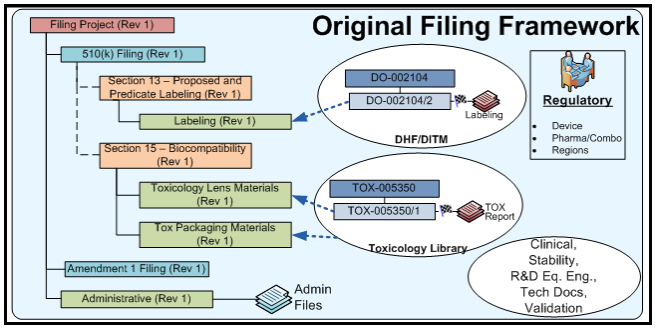

Figure 6: Information Model Softened for 1-pager

Table 6: Regulatory Project Revision Properties						
Name	**Type**	**Values**	**Description**	**Origin**	**Standard Search**	**Required**
Item ID	*Text*	*Same as Item*	*Unique revision identifier, inherited from the item*	*S*	*Yes*	*Sys Gen*
Revision	*Text*	*Numeric starting at 0*	*Revision number*	*S*	*No*	*Sys Gen*
Name	*Text (Limit 240 characters)*	*Any text*	*Also referred to as a 'nick name' for this revision*	*U*	*Yes*	*CR*
Description	*Text (Limit 256 characters)*	*Any text*	*Simple description*	*U*	*Yes*	*No*
MOH Issued Number	**Text (Limit 256 characters)**	**Any text**	**The filing number issued by the Ministry of Health**	**U**	**Yes**	**No**
Target Date	**Date**	**Date**	**Date when the submission is due to the MOH**	**U**	**No**	**No**
Submitted Date	**Date**	**Date**	**The date the filing was submitted to the MOH office**	**U**	**Yes**	**No**
Cleared/ Approved Date	**Date**	**Date**	**The date on which the filing was cleared or approved by the MOH**	**U**	**No**	**No**
RA Lead	**User**	**User**	**Regulatory Affairs group member who is mainly responsible for this project information**	**U**	**No**	**No**
Owner	*User*	*User ID*	*The revision-owning user*	*S*	*No*	*Sys Gen*

Figure 7: Example of Object Property Table

The information model graphics and the property table are paper prototype artifacts and components of the requirements specification.

In addition to the UML Class Models of the Business Information Model, UML State Machine Models are created, as shown in Figure 8. The state machines are used to model the maturation of the information from newly created to fully stable and approved for use. The Business Flow from the Enterprise Viewpoint describes the business tasks or activities that create the information units and move them to the approved state.

Feedback from the development of the state machines can also add or re-order tasks in the business flow.

There can be multiple state machine models needed to cover all the information units. The state machine for the leaf components of a structure can be different from the state machine of the structure. Between the new state and the approved state there can be intermediate states, such as design reviewed. The states are used as indicators to the organization as to the fitness for use or the doneness of the information, therefore its reliability for reuse.

The state machine model is one of the paper prototype artifacts and a component of the requirements specification.

Three major components comprise the Information Model: 1) the UML-based Class Model for types, properties and relationships, 2) the properties details table and 3) the UML State machines.

D. The Computational Viewpoint or Solution Behavior

The required behavior of the solution comes from two main sources. The first source of the behavior is the Business Flow. The second source is the application of the CRUD (Create Retrieve Update Delete) pattern to all the components of the information model.

The Business Flow is the highest level abstraction of the solution behavior. Developing the business requirements specification means analyzing and refining the flow. The first step is examining all the tasks in the flow and determining if the computer-based solution will support the task in full or in part. Some tasks in the regulatory example are done outside the business, so they are not supported by the solution. For tasks that are supported, a Use Case is defined and linked to the task in the PLM swim lane, as shown in Figure 9.

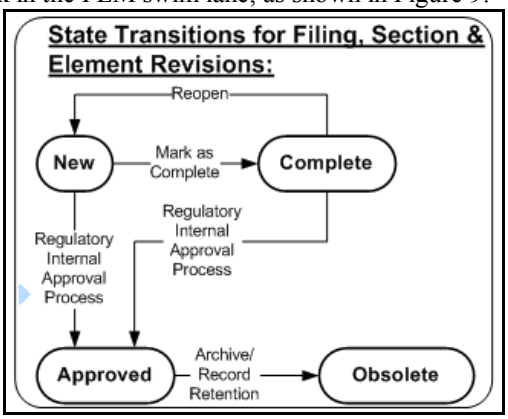

Figure 8: Example of a State Machine

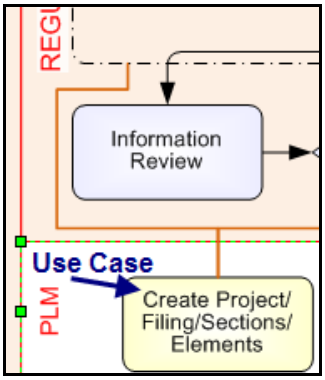

Figure 9: Example of Use Case Related to Business Flow Task

The Use Case is not the UML Use Case Model as it provides insufficient detail for requirements specification. Rather the Use case details are captured in a form of a table, which has common usage, as shown in Figure 10. The table format has been easy for the business customers to understand.

Form:	U_REG_10
Process Name	Create Filing Item
Object	Filing Item
Actors (Who can access)	Any member of the R&D Regulatory Affairs group.
Preconditions	User must have appropriate system access.
When Access	Any time.
Begins With	A new filing needs to be initiated.
Ends With	A new Filing Item exists in the KMS with a unique identifying number.
Definition/Steps	1. User selects the OOTB function in Teamcenter for creating a new Filing Item. 2. User assigns the new item number. 3. User populates the rest of the required properties. • → Name • → Filing Type 4. KMS creates the new item and automatically assigns it to the R&D Regulatory Library.
Exceptions	Exceptions due to infrastructure failure during the create process. User attempts to create a filing item without having the privilege to do so.
Post Conditions	A new Filing Item exists in the KMS and is automatically assigned to the R&D Regulatory Library. The new Filing Item exists in the owner's Newstuff folder or the active folder.
Traceability/Audit	Audit entries will be created according to the system audit policies defined in VS-0310(13).
Constraints	None
Manage/Monitor	None
Notes	None

Figure 10: Example of a Use Case Table

The goal to making this table effective is to use business terms and no system terms. In some cases that cannot be helped. The Definitions/Steps row lists the order in which the business user does the lower level steps to realize the objective of the Use Case.

The Use Case table is not one of the paper prototype artifacts, but is a key component of the requirements specification.

1) Behavior Pattern for Information Components: The second source for developing the required business behavior is applying the CRUD pattern to the information model. The CRUD pattern helps to round out the business behavior. For this work the CRUD pattern was extended and joined with ideas from the Volere Requirements Management Methodology [8]. The combination of the CRUD pattern and the Volere methodology results in a 2-dimensional matrix that is more compact than the list of single requirements. There are characteristics of each action that limit or constrain the behavior. These constraints are part of the requirements needed to complete the pattern, often referred to as the "*ilities".

The matrix is called the Action Transformation Matrix (ATM). The Extended Action column is used to refine the basic CRUD actions and can be extended as needed, a template example is shown in Figure 11. Columns can also be added as needed and Figure 11 shows an example.

In the ATM the rows are the behaviors on the information and the columns the constraints. The cells of the ATM are filled in with references to other places in the requirements specification, for example with a reference to a Use Case template as described above. The ATM is used as a 2-dimensional guideline to elicit business requirements and as a completeness check. Any empty cell indicates a missing requirement. A cell might have a "Not Applicable" (N/A) entry, to show that the question was considered.

The ATM is not one of the paper prototype artifacts, but is part of the requirements specification. The table has proven very useful as a check list to ensure that all the requirements have been addressed.

Main Action	Extended Action	Business Need	Who can access	When can access	Initial value	Value Range	Pre-Conditions	Action Results	Audit Action	Mgmt / Monitor Action	Display Format
Create	Item										
	Duplicate Item										
	Structure										
	Duplicate Structure										
	Reference (add)										
Retrieve	Values										
	Search / Find item										
	Where used in structure										
	Where referenced										
	Structure										
Update	Value										
	Structure add										
	Structure re-org										
	Reference										
	Status/State										
	• Approve										
	• Other state? extend matrix										
	Revise Item										
Delete or Obsolete	Item										
	Revision										
	Structure Membership										
	Whole Structure										
	Reference										

Figure 11: Action Transformation Matrix Template Example

2) Approval Workflows: Besides the use of a UML Activity Model for the business flow, UML Activity Models are used at a lower level, and referred to as Approval Workflows. The state transitions in Figure 8 are business events and often correspond to a task in the business flow such as "Approve Report". The requirements of the business events, in most cases are represented in more detail with an approval workflow. The workflow has the order of the approval tasks and signoffs by various people in the business to move the information to the next maturity state. The information approvals are something that is regulated by the FDA.

Supporting both the business flow and the approval workflow is an Organization Model. This model consists of groups and roles that represent subject matter expertise, business authority or a combination of both. People are assigned to the groups and roles. The full organizational hierarchy is not modeled as that often exists in another system. The group model is often a simple graphic of an organization chart. As a starting point the number of roles has been reduced to four: Viewer, Author, Manager and Leader.

The documenting of the requirements behavior corresponding to the RM-ODP Computation Viewpoint is accomplished with the Business Flow model, the ATM, Approval Workflow models and the Organization model.

The approval workflows and organization model are paper prototype artifacts and components of the requirements specification.

3) UML Models not used for Requirements: The UML Use Case Models are not used, because they are too high-level and have no concrete content. UML Sequence Models are not used as there are no methods defined at the business requirements level.

IV. Project Success Measurements

The requirements modeling approach described above has been very successful over the eighteen projects, six years and two customers, success for both the customer and my company producing the solutions. The success is measured along several dimensions: team size, speed to requirements closure, quality of requirements, customer satisfaction and business benefit of the solution. During the first four years a parallel project was

running that used the more traditional requirements gathering methods. At the end of four years the size and complexity of the business solutions were comparable. Both projects conducted After Action Reviews at the end, both internally and with the customer. Comparisons between the two projects will support the claims.

A. Team Size

The development team size of the modeling project was smaller by a factor of four. The smaller team size reduced the team communication overhead. The smaller team could send the same people, all the significant project roles, to meet with the customer, while the larger team needed to rotate people for visits. Rotating people produces misunderstandings in the development team as each person has a partial view of the customer needs. The major impact was more consistent communication between the team and customer, which resulted in a better shared vision of the solution needed (conceptual integrity).

B. Speed to Requirements

Speed to requirements closure was much faster for the modeling team versus the traditional team. The modeling team would have a fairly stable set of business requirements in about six weeks, while the traditional team took close to eight months. There were a couple of major factors that influenced the time to closure.

It was much faster to produce the graphical models, the initial model was often done on a whiteboard with the customer, and they could be reviewed easier and faster by the customer. Once a good relationship was established and the initial model was developed at a face-to-face meeting, many reviews happened via web-based meetings. This technique is called the Paper Prototype Process.

Another speed factor was the better communication within the modeling team and with the customer. There was significantly less rework. The models reduced the ambiguity that is present in text–based requirements. The traditional text-based approach had many more rework cycles because requirements were poorly expressed, often as simple declarative sentences, which demanded that the reviewer mentally combine several requirement statements to achieve a complete and hopefully correct understanding.

The third speed factor was that models allowed the project to be more agile. The models enabled easier agreement with the customer on partitioning the solution into sub-solutions when faced with unforeseen schedule or budget constraints. This was easily achieved by going to the graphic of the information model and drawing a line around the sub-solution. This kind of partitioning is very difficult to do with many text-based requirements because of the linear structure.

C. No Requirements Prioritization

One of the goals of modeling requirements rather than using a text-based approach was the elimination of the wasted time spent prioritizing requirements and arguing over the priorities.

Not a single minute over the eighteen projects was spent on traditional requirements prioritization. The model-driven approach with the graphical communication allowed clear expectation setting with the customer and within the development team.

There were times when the scope of a project was changed to respond to external changes in budget or other business pressures. This was always done within the scope of the business flow and information model and maintained the conceptual integrity.

D. Quality of Requirements

Quality of requirements produced by modeling was significantly better. In this business situation an important quality metric is the stability of the requirement specification during implementation and verification. The specifications that used models were about one third of the length of text-based specifications. This resulted in more thorough reviews and the ability of a wider range of people to be involved in the reviews. While either approach can produce a high quality requirements specification, the modeling approach spent about 10% of the time in final specification review and rework compared to the text-based approach.

The stability metric used was the number of issues identified during the Conference Room Pilot (CRP) verification exercise that were traceable back to a requirement. For the model-based requirements the number of issues per project was in the 15 – 20 range. About half of those issues needed to be resolved by process improvements in the business or resetting the requirement specification. The other issues were resolved by implementation changes that averaged about one engineering-week of effort. In the more traditional project the CRP issue list was over 200 and several engineering-months of reimplementation and requirements rewriting were needed.

E. Customer Satisfaction

Customer satisfaction with the solutions was about the same. Two aspects of the model-based approach garnered special praise. The model-based project had more visibility into the progress for the solution sponsors. The graphical nature of the requirements was part of visibility, but the feedback from the participants on how fast things were moving was also an important factor.

The graphical nature of the models allowed the creation of the 1-pager, Figure 4. The 1-pager is a software engineering solution drawing. It fit easily onto an 11x17 inch page and contained the essential models. It formed the basis of the start of all the meetings and customers could show it to colleagues and easily explain what the solution was going to be. One was posted on the wall of the R&D vice president's office, which was most unusual.

F. Business Benefits

The primary business benefits of modeling were in the earlier delivery of parts of the solution. Starting with the information models allowed us to partition the solution easily. In looking at a picture of the solution information components, lines could be drawn around sets of information units, which

became the realization partitioning. The partitioning made it easy to explain to the customer why a certain realization order was best, as some of the partitions were foundational.

Although often difficult to quantify, organization change in response to a business change is a critical factor of success. An unexpected benefit of the partitioning was the ability to evaluate whether the groups involved with the partition were more or less open to change. When we had the flexibility, targeting groups that were more open to improvement in their operating procedures, we minimized organization resistance and created examples of success.

The smaller team was a financial benefit to the customer, as it kept the costs down. This helped with the satisfaction as the customer felt they got a good return on the investment.

With several years of experience with the solutions in place there is some data available. One customer ran an internal survey of where the R&D people spend their time in generating, finding, checking and organizing information. Before the new solution the result was 16.5% of their time, of which 11.5% was considered non-value added. The goal of the project was to reduce that by 8.0%. The results after a couple of years indicate that the R&D people are spending about 3% on the non-valued added information tasks. Other benefits are reduction in the time to prepare for regulatory audits from an average of five people over three to four weeks to one person over two days.

V. Conclusions and Future Directions

The model-based requirements gathering approach is superior to the text-based approach. However, certain requirements, such as some constraints, are better expressed as text, so a combination of the two specification techniques is needed. The ATM was designed to support relating multiple requirement specifications in a single location. Using models definitely allows for a more agile requirements gathering process and with better customer review.

One area for future research is the application of the first principles to another solution domain. This would verify that the first principles really are first principles of software engineering and would enable the discovery of other first principles and patterns.

While the models and their use for the medical device product development domain and a solution based on Product Lifecycle management (PLM) technologies is proven, another domain or realization technology might contribute other modeling artifacts. Refinement to the model artifacts presented is also possible.

A more comprehensive set of mutually supporting software engineering Computer Aided Design (CAD) tools is needed. Borrowing the experiences of other engineering disciplines, a CAD tool models the solution. The model can be rendered in various formats, individual views can be combined into a software engineering drawing then, models can be passed to a Computer Aided Engineering analysis (CAE) tool, and later to a Computer Aided Manufacturing (CAM) tool. The mindset of a UML *editor* blocks progress in this area.

References

[1] OMG Unified Modeling Language 2.4.1 http://www.omg.org/spec/UML/2.4.1.

[2] F. P. Brooks, Jr., The Design of Design: Essays from a Computer Scientist, Addison-Wesley, 2010.

[3] F. P. Brooks, Jr., The Mythical Man-Month Essays on Software Engineering 20th Anniversary Edition, Addison-Wesley, 1995.

[4] Open Distributed Processing - Reference Model (RM-ODP) ISO 10746-1.

[5] D. L. Parnas, "On the Criteria to be Used in Decomposing Systems into Modules", Communications of the ACM, volume 15, number 12, December 1972, pages 1053 – 1058.

[6] J. W. Ross, P. Weill and D. C. Robertson, Enterprise Architecture As Strategy, Harvard Business School Press, 2006.

[7] T. Winograd, Bringing Design to Software, ACM Press, 1996.

[8] S. Robertson and J. Robertson, Mastering the Requirements Process, Addison-Wesley, 1999.