NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A FRAMEWORK FOR THE MANAGEMENT OF EVOLVING REQUIREMENTS IN SOFTWARE SYSTEMS SUPPORTING NETWORK-CENTRIC WARFARE

by

Linda K Reynolds

June 2006

Thesis Advisor:	Man-Tak Shing
Second Reader:	Richard Riehle

Approved for public release; distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** June 2006	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: A Framework for the Management of Evolving Requirements in Software Systems Supporting Network-Centric Warfare			**5. FUNDING NUMBERS**
6. AUTHOR(S) Linda K. Reynolds			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Network-centric warfare (NCW) has changed the way the Department of Defense addresses technological improvements for its military forces. No longer is the emphasis on enhancing the capabilities of a single platform, but the focus is now on networking people, processes and technology to enable knowledge sharing and rapid decision-making.

The capabilities required to support network-centric operations (NCO) in the NCW environment must be supported by new, innovative networked communication technologies. There are many sources of requirements for these software systems supporting NCO, which may increase in number as the Services continue to develop the capabilities necessary for the transformation to a fully networked military force. Requirements may also emerge and continue to evolve following the fielding of a NCO capability because new technology has the potential to change how warfighters work. Requirements evolution results in requirements engineering challenges associated with the acquisition and development of network-centric software systems. As such, an approach is needed to provide for consistency in elicitation, management and documentation of evolving requirements for technological capabilities supporting NCO.

The purpose of this research is to address the problem of evolving requirements. The requirements engineering framework proposed by this thesis incorporates classification theory and requirements modeling principles, and is supported by the Extensible Markup Language (XML) family of technologies. Particular attention has been paid to the selection of non-proprietary, platform independent technology to ensure data can be exchanged between organizations.

The framework demonstrates a means by which requirements can be classified and structured in a standardized format. The result is a set of requirements that is consistent in structure and content, and that can be easily shared among all stakeholders because it utilizes one standard, non-proprietary format. This approach captures evolving software requirements of fielded network-centric software systems for use in the development of future systems.

14. SUBJECT TERMS Requirements Engineering, Requirements Evolution, Faceted Classification, Extensible Markup Language, XML, Unified Modeling Language, UML			**15. NUMBER OF PAGES** 121
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited.

A FRAMEWORK FOR THE MANAGEMENT OF EVOLVING REQUIREMENTS IN SOFTWARE SYSTEMS SUPPORTING NETWORK-CENTRIC WARFARE

Linda K Reynolds
Lieutenant Commander, United States Navy
B.S., Oregon State University, 1997

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 2006

Author: Linda K Reynolds

Approved by: Man-Tak Shing
Thesis Advisor

Richard Riehle
Second Reader

Peter Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Network-centric warfare (NCW) has changed the way the Department of Defense addresses technological improvements for its military forces. No longer is the emphasis on enhancing the capabilities of a single platform, but the focus is now on networking people, processes and technology to enable knowledge sharing and rapid decision-making.

The capabilities required to support network-centric operations (NCO) in the NCW environment must be supported by new, innovative networked communication technologies. There are many sources of requirements for these software systems supporting NCO, which may increase in number as the Services continue to develop the capabilities necessary for the transformation to a fully networked military force. Requirements may also emerge and continue to evolve following the fielding of a NCO capability because new technology has the potential to change how warfighters work. Requirements evolution results in requirements engineering challenges associated with the acquisition and development of network-centric software systems. As such, an approach is needed to provide for consistency in elicitation, management and documentation of evolving requirements for technological capabilities supporting NCO.

The purpose of this research is to address the problem of evolving requirements. The requirements engineering framework proposed by this thesis incorporates classification theory and requirements modeling principles, and is supported by the Extensible Markup Language (XML) family of technologies. Particular attention has been paid to the selection of non-proprietary, platform independent technology to ensure data can be exchanged between organizations.

The framework demonstrates a means by which requirements can be classified and structured in a standardized format. The result is a set of requirements that is consistent in structure and content, and that can be easily shared among all stakeholders because it utilizes one standard, non-proprietary format. This approach captures evolving software requirements of fielded network-centric software systems for use in the development of future systems.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to give my heartfelt appreciation to my mother who has always given me her unwavering support and who has equipped me with the tools to succeed. I would also like to thank my advisors, Prof Man-Tak Shing and Richard Riehle, for their guidance, patience and words of wisdom and Curt Blais for his help with XML. Finally, I give my thanks to the unsung heroes of Student Services, ITACS, the Dudley Knox Library and the curriculum offices. They keep us all moving in the right direction.

I dedicate this work to my children who are my joy and inspiration.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Since its introduction in 1998, the concept of network-centric warfare (NCW) has changed the way the Department of Defense addresses technological improvements for its military forces. No longer is the emphasis on enhancing the capabilities of a single platform, but the focus is now on networking people, processes and technology to enable knowledge sharing and rapid decision-making. According to Secretary of Defense Donald Rumsfeld in 2003 [DIR05]:

> …we must achieve: fundamentally joint, network centric, distributed forces capable of rapid decision superiority and massed effects across the battlespace. Realizing these capabilities will require transforming our people, processes and military forces.

The capabilities required to support network-centric operations (NCO) in the NCW environment must be supported by new, innovative networked communication technologies. These technologies may be modifications to fully developed legacy software systems that are programs of record (POR). They may also be prototypes, commercial-off-the-shelf (COTS), or government-off-the-shelf (GOTS) systems either introduced as a new capability or proposed as improved replacements for existing systems. In each situation, many sponsors and organizations across the DOD may be involved in the development, testing and evaluation of similar software applications to support the transformation to NCW. These activities may ultimately result in the duplication of efforts, especially those involving the identification and definition of software user requirements.

There are many sources of requirements for software systems supporting NCO, which may increase in number as the Services continue to develop the capabilities necessary for the transformation to a fully networked military force. During the development process, software requirements may originate from many groups of stakeholders with differing viewpoints on what is necessary for warfighters to work effectively and efficiently in the network-centric operational environment. Requirements may also emerge following the fielding of a NCO capability because 1)

new technology has the potential to change how the warfighter works and 2) the operational environment in which it is used may change. As they gain experience with the system, warfighters may be able to identify requirements that would improve their interaction with the technology and, ultimately, their job performance. Warfighters may also find the application lacks certain features or functionalities necessary to support the evolving and possibly unpredictable operational environment in which they work. That is, they may find that the system does not meet the operational requirements of the ever-changing technological environment imposed by the transformation to NCW. Because of these highly unpredictable issues, user requirements for network-centric software systems may continue to evolve long after the system has completed the formal software development process. There are very few mechanisms in place to ensure the resulting requirements are evaluated for implementation in future systems. Requirements evolution, therefore, results in requirements engineering challenges associated with the acquisition and development of network-centric software systems. The following challenges, if not adequately addressed, can influence the extent to which future network-centric software systems meet operational objectives and user requirements:

- Capturing evolving software requirements of fielded network-centric software systems for use in the development of future systems
- Consolidating, standardizing and documenting requirements from multiple informal and formal sources
- Ensuring the requirements reflect high-level strategic and operational goals
- Evaluating a network-centric system against requirements to determine if it will meet its intended purpose and warfighters' needs
- Ensuring the requirements elicitation process includes all stakeholders

Effectively managing these challenges requires a fundamental shift in how requirements engineering practices address evolving requirements. Rather than consider rapid and continuous changes to requirements as impediments to software system development, these changes should be accepted and addressed by a requirements

engineering approach that incorporates processes for handling them, especially following the formal development process.

A. PURPOSE OF STUDY

The purpose of this research is to develop an approach to provide for consistency in elicitation, management and documentation of requirements for technological capabilities that have the highest likelihood of requirements evolution. In conducting this work, we address three questions:

1. How can we collect evolving requirements data from potentially thousands of geographically dispersed network-centric software application users?
2. How can we standardize the evolving requirements data so it can be effectively analyzed and used in the software development process?
3. How can we then make the standardized evolving requirements data accessible across various DOD organizations without relying on proprietary, expensive requirements management software?

To answer these questions, this thesis proposes a requirements engineering framework that incorporates classification theory and requirements modeling principles, supported by the Extensible Markup Language (XML) family of technologies. The intent of this research is not to develop a fully-fledged web application to manage evolving requirements. Rather, it is to suggest an approach, demonstrated with current technologies, the DOD acquisition community could consider for implementation. Because this approach focuses on managing requirements for a technological capability rather than for one specific software system, we recommended it for consideration throughout the life cycle of the capability. A rigorous application of this framework will help prevent the unnecessary expenditure of limited resources for software systems that either do not meet the needs of the warfighters or do not support the implementation of network-centric warfare.

B. THESIS ORGANIZATION

Chapter II provides the background information on requirements and requirements engineering and the role each plays in software development.

Chapter III highlights the proposed framework for managing requirements evolution and outlines the methodology used to develop the framework.

Chapter IV presents the activities of the proposed framework. Each activity is discussed in detail, with an overview of the technology proposed for implementation.

Chapter V details the type of classification scheme used in the framework. This chapter is dedicated to a discussion of the classification scheme because of its importance to the framework.

Chapter VI is the case study. Each activity of the framework is demonstrated using previously collected user requirements of an existing software system that supports NCO.

The concluding chapter of this work is Chapter VII, where there also appears a short discussion of related areas for future work.

II. THE ROLE OF REQUIREMENTS IN SOFTWARE SYSTEMS DEVELOPMENT

A. INTRODUCTION

Practitioners often cite poor requirements as the reason software systems fail to meet the needs of the user [BRU04]. Inadequate, inconsistent or vague requirements can result canceled projects or projects that are delivered late and over budget. As such, it is critical to understand the role requirements play in software development.

Software requirements are commonly defined as the specific capabilities imposed on a system that are a direct reflection of the users' needs and that must be satisfied by the technical solution ultimately designed to address the problem [LEF03]. The users often express their needs in terms of what they expect of the system in relation to their activities. They may also define their needs within the context of the high-level operational goals for the system. Software system developers can then use these goals, in conjunction with the users' needs, to help define and specify the functional and non-functional requirements for the system [ANT98]. The quality of the specified requirements can have a significant impact not only on the quality of the final system, but also on the system's ability to meet the needs of the users. Requirements that are poorly defined or incomplete can prevent system developers from identifying the most appropriate technical solution. Therefore, generating quality requirements is an important activity of software development, and it must include the system users.

B. REQUIREMENTS CLASSIFICATION

1. Introduction

Several characteristics identify quality requirements. For example, requirements must be verifiable and feasible to ensure they can be implemented within the given resource constraints, necessary to meet the needs of the user, complete and correct [WIE03]. Classification of requirements is the first step in developing requirements with these characteristics. It provides developers with a way of understanding and managing the requirements from a technical standpoint; it assists in mapping requirements to

technical specifications necessary for implementation. Typically, requirements are classified either as functional, which describe the system's operation, or non-functional, which express what the user expects from the system.

2. Functional Requirements

The functional requirements describe how the system must operate and how it must interact with its environment. For example, a functional requirement of a chat client may be the capability to log chat sessions.

3. Non-Functional Requirements

The non-functional requirements are constraints imposed on the system and are usually referred to as quality attributes [WIE03], e.g. the "ilities". The following is a partial list of types of non-functional requirements applicable in the design and development of software systems:

- Availability
- Reliability
- Efficiency
- Usability
- Interoperability
- Maintainability
- Portability
- Testability

A non-functional performance requirement for a chat client may be that it must establish a chat session within 30 seconds after it receives the request chat session event.

4. Shortcomings of Traditional Requirements Classification

Although widely used in requirements engineering, traditional requirements classification may not identify requirements that have both functional and non-functional attributes. For example, a requirement may specify a necessary performance factor in addition to a quality attribute. To address this issue, we will introduce a multi-faceted layer approach to requirements classification in Chapter IV.

C. REQUIREMENTS ENGINEERING

1. Introduction

Software systems requirements engineering (RE) involve those efforts required to ensure that a software system is designed with the functionalities necessary to meet the needs of the users. According to [ZAV97], "Requirements engineering is the branch of software engineering concerned with the real-world goals for functions of and constraints on software systems." While there are many proposed taxonomies for RE, one approach suggests RE may be divided into functions associated with development and management of requirements, as shown in Figure 1 from [GSAM03].

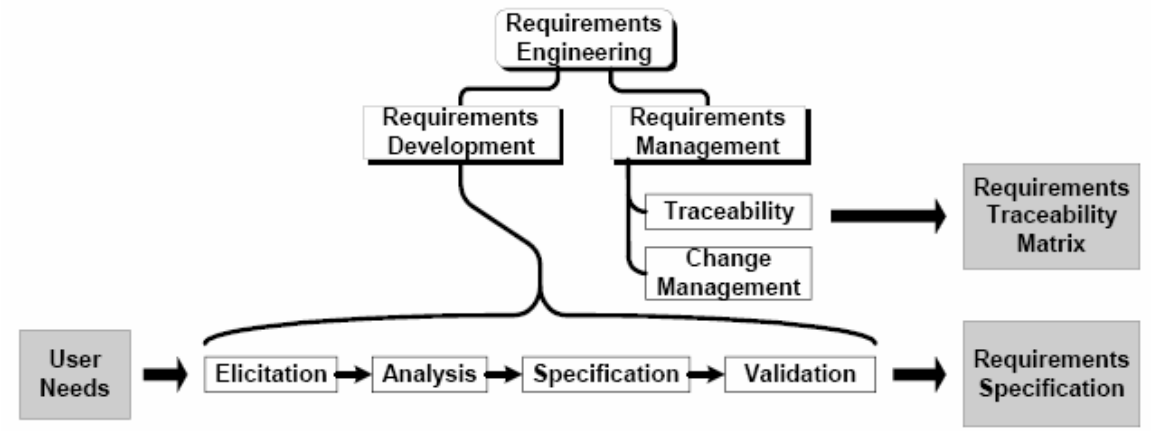


Figure 1. Requirements Engineering Process (From: [GSAM03])

2. Requirements Development

Requirements development involves collecting, analyzing, specifying, validating and documenting the users' needs for the software system. In the traditional software development process, requirements elicitation, generation, and documentation are functions that typically occur at the beginning stages of the development process, as illustrated in Figure 2. These efforts ensure the system under development meets the stakeholders' and ultimately the user's needs and goals. During the initial stages, the development team will, in an attempt to better understand the problem domain, elicit the needs of the customer, user and anyone else with a vested interest in the project, collectively referred to as stakeholders. From this information, the developers, also considered stakeholders, will propose the features necessary to address those needs. The

needs and features form the basis for the software requirements that the development team uses to build the technical solution [LEF03]. As such, it is critical that there are no gaps or overlooked areas in the development activities because the final system will certainly reflect these shortcomings. A series of questions from [GSAM03] may be used to determine if the requirements development efforts have been adequate during the elicitation and generation stages to develop quality requirements:

- Has there been extensive user involvement in developing the requirements?
- Do all stakeholders understand and agree on how the system will be used?
- Are all stakeholders satisfied with the requirements?
- Do the developers understand the requirements?
- Are all requirements clear and unambiguous?
- Have you distinguished between needs and wants? Are requirements relevant?
- Are requirements consistent with each other (i.e., they don't conflict.)?
- Are requirements complete? Do the requirements cover everything that is supposed to be accomplished?

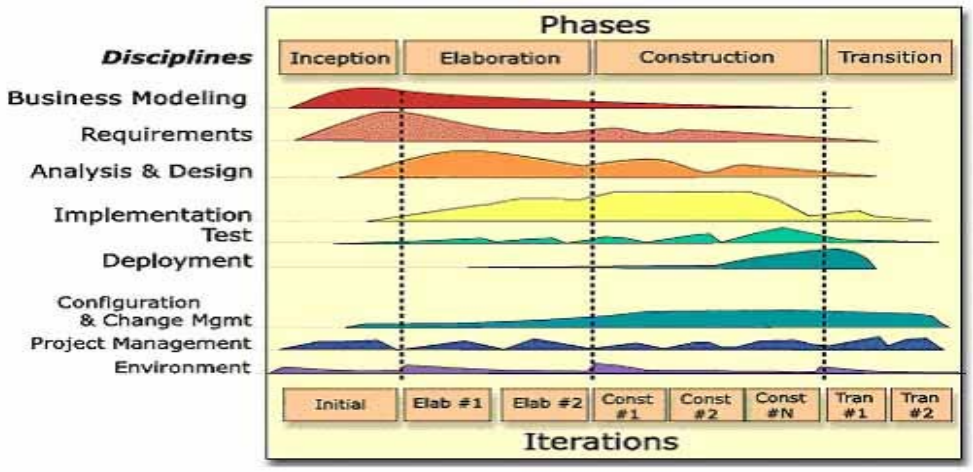


Figure 2. Rational Unified Process lifecycle (From: [CAN03]).

a. Requirements Elicitation and Analysis

Without quality requirements that accurately reflect the needs of the stakeholders, especially users, development of a suitable technical solution is unlikely. As such, an open channel of communication between developers and stakeholders is critical to the success of a project. The failure to communicate is costly and can result in poor quality requirements and ultimately a system that does not meet the needs of its users. [BRU04] states the importance of communication in requirements elicitation:

> Requirements elicitation is about communication among developers, clients, and users to define a new system. Failure to communicate and understand each others' domains results in a system that is difficult to use or that simply fails to support user's work. Errors introduced during requirements elicitation are expensive to correct, as they are usually discovered late in the process, often as late as delivery.

Software developers with a comprehensive understanding of the stakeholders' requirements and the proposed system's operational environment have a greater likelihood of designing, developing and delivering a system within the stated constraints that meets the stakeholders' needs.

There are many approaches to eliciting user requirements, many of which involve direct interaction and coordination between requirements engineers, additional members of the software development team and the stakeholders [LEF03]. Requirements engineers may conduct interviews with users and involve them in scenario development in order to understand the problem domain. However, when the group of stakeholders is large and geographically dispersed, it may not be possible to conduct frequent face-to-face interviews or hold facilitated meetings for the collection of requirements data. As such, they may supplement the primary elicitation techniques with requirements modeling and simple, low-fidelity prototyping.[1] Using pictures and models to represent requirements enhances communication between stakeholders because they do not require an advanced technical background to understand the concepts as they are illustrated [WIE03]. The Unified Modeling Language™ (UML) has become the defacto industry

[1] Low-fidelity prototypes are simple sketches or mockups of the system to be developed.

standard and primary method of visually representing user requirements, supporting software development.

The quality of requirements ultimately affects the quality of the final product, as well as its ability to meet the needs of the users. Users may often submit abstract or vague requirements during the elicitation phase, which make design decisions difficult for software developers [WIE03]. As a result, once the stakeholders' needs are gathered, developers must conduct an analysis to define and scope the requirements. Analysis of requirements is required to [SWEBOK04]:

1. Detect and resolve conflicts between requirements
2. Discover the bounds of the software and how it must interact with its environment
3. Elaborate system requirements to software requirements.

Requirements analysis helps uncover inconsistencies or ambiguities through investigation and analysis of the problem domain. Requirements analysis techniques include goal-based analysis methods [ANT97] [LAM01], as well as structured and object-oriented analysis methods.

b. Requirements Specification and Validation

Following analysis, the development team formally documents requirements to specify what the design of the system must achieve. They may use a Software Requirements Specification (SRS) to document the problem description, system requirements, interface requirements, performance requirements, and design constraints. Aside from an SRS, developers may choose to use alternative methods to record requirements information. One choice may be to organize requirements in sets for subsystems or systems with a particularly large number of requirements and documented using a Vision document and supplemental specifications [LEF03]. Whatever the choice, requirements documentation should be managed and maintained throughout the life of the project and should be available for subsequent iterations of the software system. Thorough documentation supports traceability, change management and can have a significant impact on the quality of the final software system [LUQ04]. This information can be managed either manually using a pen and paper or electronically with any number of commercially available requirements management tools. The choice of documentation

technique may depend on such factors as the size of the project and organization, budget and experience of the developers.

After specification and review by stakeholders, the requirements must be validated. Requirements validation is the final and necessary step of requirements development (as presented in Figure 1) that ensures the specified requirements reflect the wants and needs of the stakeholders. Validation entails examining 1) the requirements to ensure, among other things, they are correct, complete and unambiguous and 2) the SRS to ensure it is consistent, complete, modifiable and traceable [WIE03].

3. Requirements Management

Stakeholders and developers will often make changes to requirements following specification. Controlling these changes is a function of requirements management. Configuration control and traceability functions support the management of requirements throughout the development process. These functions document changes to requirements as well as relationships among requirements. Commercially available requirements management tools, such as Borland© Caliber™ 2005 and Requirements Management Database™, are available to support elicitation and management of requirements.

D. GOALS IN REQUIREMENTS ENGINEERING

Goal-based requirements engineering provides for the identification and specification of requirements using goals that represent the customer's needs for the system. According to [LAM01], "A goal is an objective the system under consideration should achieve. Goals may be formulated at different levels of abstraction, ranging from high-level, strategic concerns to low-level, technical concerns." At the highest level of abstraction, the stakeholders' objectives for the system should remain consistent throughout the development process because they provide the justification for the system's development. However, at the lowest level, the implementation specificities may change radically because, as previously discussed, changes to requirements are common during the development process. Using goals to define requirements at the beginning of the development process may reduce the number of changes to system requirements. In goal-based RE, goals are used to provide the justification for the

requirements, allow for a level of abstraction necessary to evaluate requirements alternatives and are often more stable than requirements [ANT97]. Requirements that are products of (linked to) the high-level objectives for the system are, therefore, probably more consistent and easier to manage than those not (linked).

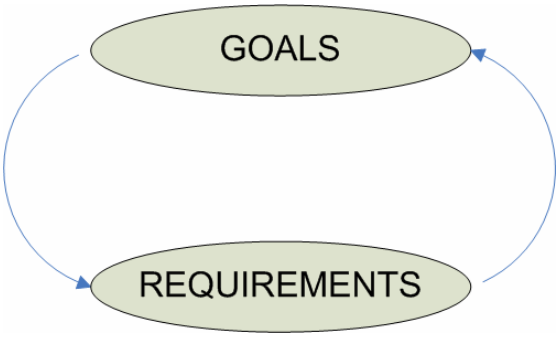


Figure 3. Goal Requirements Feedback

Stakeholders should express their objectives, or goals, for the system under consideration during the requirements elicitation process. Evaluation of these objectives ultimately results in specified requirements. However, as portrayed in Figure 3 above, existing requirements may also be the source of goals. Requirements engineers may derive goals from documented operational and tactical requirements for the system, as well as from the stakeholders' espoused requirements for an existing or legacy system. Use-case specifications, scenarios, and informal requirements documents are also sources of goals [ANT98]. When goals are used to define requirements, it is reasonable to assume they should meet the same quality criteria as requirements; they should be complete, correct and feasible. As such, communication between stakeholders and developers is critical but may be an improvement over non-goal based requirements elicitation because of the evolutionary nature of goals [ANT00]. The use of goals to define requirements may help resolve conflicting points of views among stakeholders; if they can agree on what they want the system to achieve, they may be more likely to agree on the requirements to fulfill those objectives.

E. REQUIREMENTS RISK

Risk is the possibility of an event having an adverse effect or outcome and is associated with varying degrees of uncertainty. In recent years, risk management has been identified as a critical function in any project and entails identifying, addressing and eliminating those risks that threaten the success of the project [BOE91]. In particular, risks related to requirements can have a tremendous impact on the cost of developing software systems, the schedule required for its development and the ability of the system to meet the user's needs [FLO02]. Requirements risks can be the result of requirements management efforts that do not adequately address the impact of changing requirements and evolving operational environments. Risks can also result from communication shortcoming between customers and developers [LEI02]. Requirements risks most likely to have an impact on a software project are as follows [WIE98]:

- Lack of a clear product vision
- Lack of agreement of product requirements
- Inadequate customer involvement in requirements process
- Non-prioritized requirements
- New market with uncertain needs
- Rapidly changing requirements
- Ineffective requirements change management process
- Inadequate impact analysis of requirements changes

An extensive risk evaluation effort will be required when the number of requirements is large, potentially resulting in procurement delays and increased project costs [BOE01]. The solution may lie in the level of abstraction provided by defining and assessing risks based on goals. This process has the potential to provide decision makers with an expeditious way to identify significant deficiencies in the software system's ability to meet warfighters' operational requirements. As discussed, stakeholder and developers work to derive the goals for a software system from the overall operational objectives and, to a limited extent, previously specified requirements. These goals help in the specification of requirements—requirements that will be used to either develop or procure a system. Therefore, it is likely a risk management strategy that involves

stakeholders early on in the process will help mitigate the risks of a software system not fulfilling the overall operational objectives, much less the specified requirements. Assigning priorities with stakeholder involvement at the initial stages of requirements development is a critical first step in mitigating risks.

F. FORMAL SOURCES OF REQUIREMENTS

The acquisition community within DOD must adhere to stringent guidelines when developing requirements for software systems. They must meet criteria established by Federal law and DOD directives.

1. DOD Instruction 5000.2

Department of Defense Instruction 5000.2 provides for detailed guidance on the acquisition of software systems. Before the initiation of the contracting process, the operational goals and system requirements are developed, specified and submitted in a series of capabilities documents shown in Figure 4. An Initial Capabilities Document at Milestone A establishes the initial system requirements that are later refined at Milestone B upon finalization of the Capabilities Development Document.

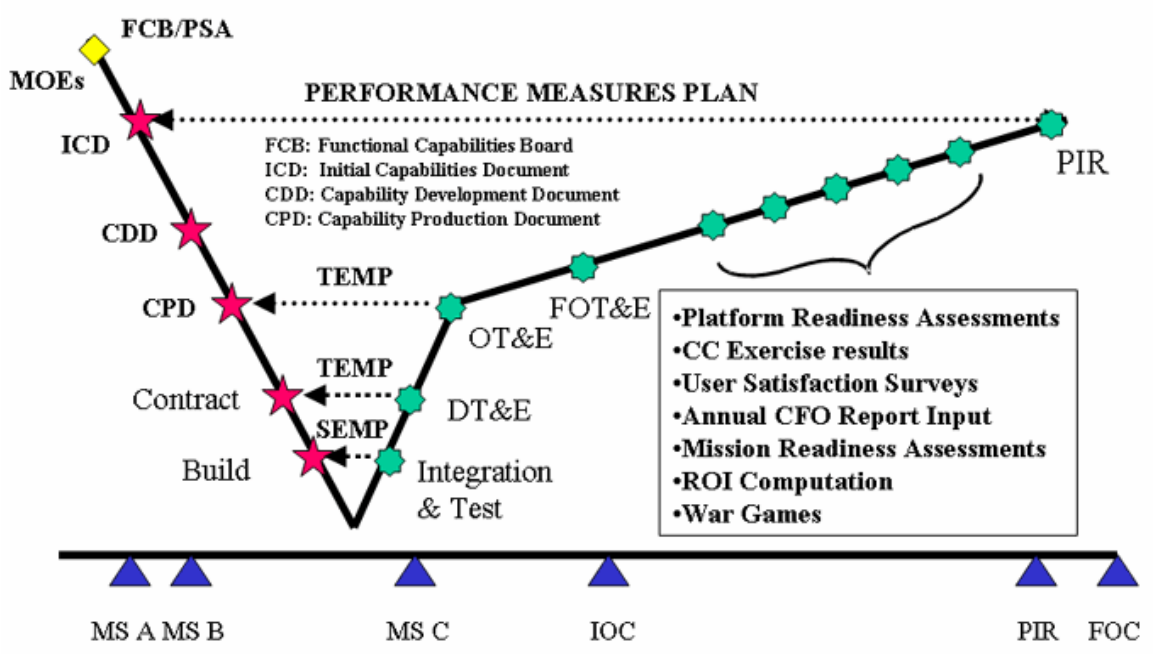


Figure 4. PIR Verification of the Initial Capabilities Document (From: [DAU])

2. Joint Capabilities Integration and Development System

Supporting DOD 5000.2 is the Joint Capabilities Integration and Development System that requires a program sponsor conduct a series of functional analyses to show the system acquisition supports core functions of the Federal Government and to validate the relationship between DOD's mission and the acquisition's function. These analyses are conducted before the instantiation of the Initial Capabilities Document required by DOD Instruction 5000.2.

3. Clinger-Cohen Act

The Clinger-Cohen Act of 1996 (CCA) requires a post-implementation review (PIR) for an IT system that is fielded and is operational in its intended environment. A PIR must be conducted utilizing performance-based measures of effectiveness, otherwise known as Outcome-based Performance Measures of Effectiveness (OPMs), which are established before the formalization of the Initial Capabilities Document. According to the Defense Acquisition Handbook, the overall intent of the PIR is to answer the question, "Did the agency get what it needed as per the ICD?" The PIR may also provide the opportunity for collecting requirements feedback from the users for consideration in an evolutionary acquisition process. [DAU]

However, there can be several obstacles to collecting evolving requirements during a PIR. A PIR may be difficult to plan due to the types of activities (Figure 3) and can take many months to complete [DAU]. As a result, it may be the case the ICD for the next system iteration or a similar software system is being planned well before the PIR for the current system has been completed and the results analyzed. In the event the planned acquisition introduces a new technological concept in an operational domain, the OPMs that were established at the very beginning of the acquisition process may not provide an accurate assessment of the warfighter's current needs due to evolving operational objectives. Again, the introduction of new technology may change how work is performed and there may be no OPM to adequately assess this impact on the warfighter's performance.

G. ADDITIONAL SOURCES OF REQUIREMENTS

1. Warfighter Feedback

There are informal sources of requirements that may be overlooked or may be just too difficult to funnel back into the acquisition and software development process. Well after the PIR is completed, the warfighter may continue to discover the functionalities and intricacies of the new system and find the system in its current configuration does not fully support the mission objectives of the operational unit. In addition, technical difficulties with the system may preclude the warfighter's ability to operate the system as intended by its design. As such, non-traditional sources of requirements may be trouble calls to help desks, casualty reports (CASREPS), Lessons Learned messages, or technical representative (TECHREP) requests. These sources may provide the most valuable requirements feedback because it is often the case the feedback occurs when the system is being utilized in the most arduous operational environment, such as during training exercises or deployment. Capturing this information is critical to ensuring warfighter operational needs, in the form of requirements, are considered either for incorporation in the next software build or during the acquisition process of a follow-on system.

2. Technical Evaluations and Warfighter Demonstrations

User requirements also result from activities associated with testing and evaluating software systems prior to formal acquisition, during the software development process, or following fielding. Proof-of-concept systems and prototypes may be tested during warfighter demonstrations such as Fleet Battle Experiments (FBE), Limited Objective Experiments (LOEs) and Sea Trial exercises to provide users with the opportunity to evaluate these new technologies.

H. SUMMARY

Requirements engineering is an involved process that focuses on developing and managing quality requirements. The elicitation activities are critical components of this process because communicating with stakeholders using language they understand is necessary to develop a system that meets their needs. Modeling is one way to enhance communication between stakeholders, especially when a well-known modeling

specification, such as UML, is used. The use of goals in requirements development can also bridge the gap between stakeholders (and developers) and result in quality requirements because stakeholders often express their needs in terms of what they want the system to achieve.

THIS PAGE INTENTIONALLY LEFT BLANK

III. FRAMEWORK FOR THE INTEGRATION OF REQUIREMENTS EVOLUTION

A. INTRODUCTION

Requirements development efforts do not occur only at the beginning stages of the software development process as system requirements are often defined and refined throughout the entire process. During development, the stakeholders' expectations for the system may change; they may have a better understanding of the required functionalities and request the implementation of additional features. In addition, changes in the operational environment may result in requests for additional system requirements or refinements to the original requirements before the end of the development process. The changes to requirements during development, referred to as "requirements creep", are difficult to manage and can have serious implications on cost, schedule and functionality of the final system [ANT01]. As such, the effectiveness of requirements engineering activities throughout the entire development process largely determines the extent to which the software system ultimately meets the stakeholders' requirements and operational objectives.

Requirements creep can also occur following development and delivery of the software system and then becomes known as requirements evolution [ANTPOT01]. When a software system is fielded in a network-centric operational domain, there is a high probability that requirements for the system will evolve, especially if the system represents a new technological capability. Users will familiarize themselves with the system, explore its technical possibilities and limitations, and determine how it can improve their work processes. As a result, the users may find, after the system is fielded, they require additional features or functionalities in order to perform their work effectively and efficiently.

The problem of evolving requirements then becomes twofold: capturing the requirements that evolve after the system has been fielded, and validating and analyzing the requirements to ensure future software systems are either developed or procured with the functionalities necessary to meet the evolving needs.

B. CRITERIA FOR A FRAMEWORK FOR THE INTEGRATION OF REQUIREMENTS EVOLUTION

We have identified three criteria that must be satisfied by a framework that addresses evolving requirements of network-centric software systems: structured requirements, data exchange capability and standard input format. We propose the Framework for the Integration of Requirements Evolution (FIRE) to satisfy these criteria and address the problems associated with requirements evolution in the NCW domain.

1. Structured Requirements

Problem: *There is no process for standardizing evolving requirements data that is generated independent of a development process.*

With informal requirements collection techniques, such as surveys, usability studies, and field tests, there may be no formal method available to format and manage the resulting information. Requirements data may simply be collected as part of a larger effort, and then analyzed, documented, and distributed in report form in natural language. In order to develop a comprehensive, consistent set of user requirements, a stakeholder must have access to all documentation and devote a considerable amount of time to extracting and reconciling the pertinent information. This approach is extremely ineffective because it is subjective; each stakeholder may have a different interpretation of the information, resulting in dissimilar sets of requirements. The data may not distinguish between a functional or nonfunctional requirement or include descriptions of requirements attributes. As a result, the data generated by these various groups will not result in a reliable, concise set of requirements suitable for use in software development.

Solution: *A structured format, using predefined attributes, that helps stakeholders, including users and developers, formulate their needs into precisely defined requirements. The method used to structure the requirements must also satisfy the remaining criteria.*

2. Data Exchange Capability

Problem: *The RE efforts of traditional software development methods may not adequately provide for the collection, integration and utilization of user requirements after the military component takes delivery of the software system.*

It is likely software requirements will change rapidly and frequently in the NCW operational environment, well after the system has completed the software development process. However, there may not be a standard process for the submission, validation and verification of evolving requirements for inclusion in the next version or build of the system. In addition, organizations may not have the ability to exchange requirements information because each may use a different type of requirements management tool, resulting in dissimilar formats or file types.

Solution: *Data exchange capability of requirements information using nonproprietary, platform independent technology.*

3. Standard Input Format

Problem: *Requirements originate from many different groups of stakeholders who are not involved in the acquisition or development process.*

User requirements will originate from system users, evaluators, testers and developers. As previously discussed, it is difficult to collate information from numerous sources, especially if the results are documented using different formats. In addition, the information collected during field evaluations may reflect high-level requirements that need further development; the collected requirements may only indicate what tasks the warfighter needs the system to support. It may be extremely difficult to identify and track additions, deletions, or modifications to existing requirements due to the inconsistencies in collection and data reporting techniques.

C. OVERVIEW OF FIRE

The framework suggested for managing evolving requirements is based on three basic, yet very important elements of requirements engineering:

1. Stakeholder involvement in the requirements development process
2. Requirements modeling
3. Requirements documentation.

FIRE is an iterative approach, which is necessary to manage the complexities associated with evolving requirements of software systems supporting NCW operations. It is based on requirements development principles, incorporating a layered faceted

classification scheme to define and manage warfighter input. The goal of FIRE is not to replace the RE efforts during the software development process. Rather, it is to suggest an approach for standardizing the data associated with evolving user requirements and making it accessible to any organization involved in developing software systems supporting NCW operations. Our approach supplements the requirements elicitation and generation efforts of the formal development process, which is beneficial when access to those system users who may be constrained by geographical or operational limitations. In addition, it supports requirements reuse because of the accessibility of requirements data and focus on using operational goals as the basis and justification for the abstract, high-level user requirements of a technological capability.

D. RESEARCH AND FRAMEWORK DEVELOPMENT METHODOLOGY

The approach suggested for the management and integration of requirements evolution will be developed and explained in the following steps:

1. Develop a method for structuring requirements:
    a. Research and identify a classification scheme to define and represent evolving requirements.
    b. Research and identify a modeling approach to visually represent the classification scheme.
    c. Research the potential of XML technology to standardize evolving requirements data.
2. Demonstrate requirements data exchange functionality:
    a. Research and identify a data format capable of facilitating production and consumption of evolving requirements data that does not rely on proprietary applications.
    b. Demonstrate transformation of standardized requirements information.
3. Propose a standard input format for requirements data. A fully functional web application is beyond the scope of this thesis and should be the focus of future work. In Chapter VII, we will present our recommendation for an input format suitable to capture evolving requirements data.

E. SUMMARY

We propose a framework to address evolving requirement that focuses on structuring requirements data and providing data exchange capability without relying on proprietary software applications.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. ACTIVITIES REQUIRED TO MANAGE EVOLVING REQUIREMENTS

A. INTRODUCTION

This framework and its activities, as shown in Figure 5, provide for the integration of evolving requirements that are generated during the software development process, and they address the problem of requirements evolution in fielded systems. These activities are recommended either in conjunction with the software development process or as an independent effort.

B. ACTIVITIES TO SUPPORT INTEGRATION OF EVOLVING REQUIREMENTS

The activities in this approach focus on the collection, classification, and documentation of requirements. The specification and verification of requirements should occur during the software development process, leveraging the expertise of the development team. We recommended the activities described in this section as the first steps in addressing the inherent difficulties associated with collecting, classifying and integrating evolving requirements.

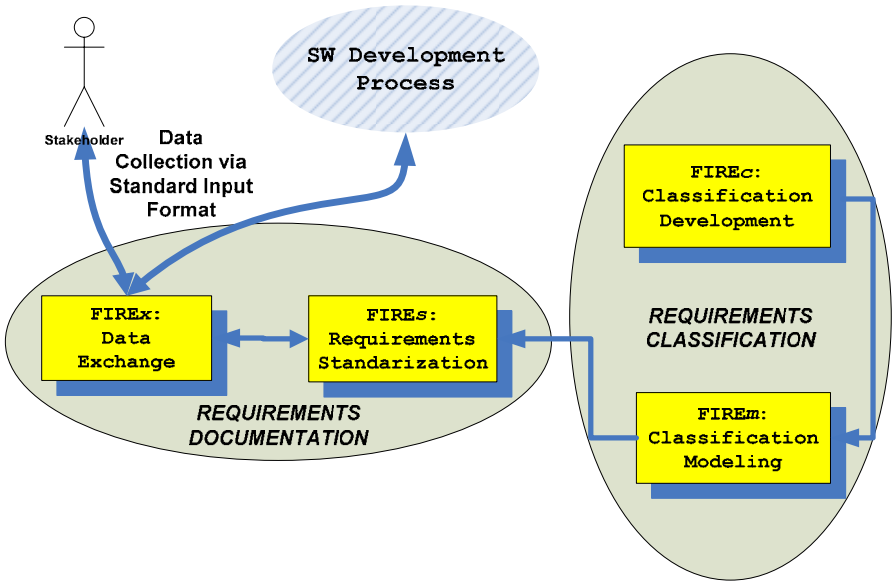


Figure 5. Activities required for the integration of evolving requirements.

1. FIRE*c*: Requirements Classification

Classification of requirements is important because it assists developers in determining the type of technical solution required to address the stated need. Requirements engineers use traditional classification schemes to identify requirements as either functional or non-functional. However, this may not be an appropriate type of classification to use when the requirements are high-level, abstract or informally stated. In addition, warfighters may not be able to explicitly state their needs for the system using terms specific to the RE discipline. Therefore, this work proposes a layered, faceted classification scheme that addresses the natural language requirements of warfighters while decomposing the requirements from high-level objectives to specified, low-level requirements. This is the first of three activities involved in structuring the requirements, the first criterion that must be met by our framework. The classification is developed from domain specific documents that describe the objectives for the capability under consideration. Extracting these objectives is a function of the classification efforts. We will present a requirements classification scheme in Chapter IV.

The use of high-level objectives as the basis for the classification is important primarily because they directly reflect the stakeholders' needs and changes in the operational environment. The high-level objectives espoused by stakeholders for the software systems supporting NCW operations express what the system is expected to achieve in its operational environment. If there are changes in the operational environment, the objectives for the system must change, as well. Therefore, requirements defined by the objectives, or goals, for the system are more apt to meet the changing needs of the stakeholders than requirements not linked to goals. We believe defining and classifying requirements and managing changes at this level of abstraction ensures the resulting requirements support the evolutionary nature of NCW operations.

2. FIRE*m*: Classification Modeling

In the FIRE*m* activity, modeling serves a two-fold purpose. First, we use it to develop and manage the layered classification scheme. Second, it provides a visual representation of the attributes of requirements, facilitating communication between stakeholders.

3. FIREs: Standardize Warfighter Requirements

The importance of standardizing and documenting requirements data cannot be overstated; thorough documentation supports traceability, change management and can have a significant impact on the quality of the final software system [LUQ04]. However, for the purposes of managing evolving requirements outside of the formal development process, it is not realistic to assume an SRS and supporting documentation with fully specified requirements are available at all times. Personnel constraints would preclude such an administrative burden. As such, it may be more practical to utilize the functionality of a commercial-off-the-shelf RE tool that generates working documents and models for internal use and supports traceability and change management. This type of functionality is especially attractive when there are numerous modifications to the requirements or there are a large number of requirements; the application may be able to generate working documents immediately following any changes.

However, there may be several downsides to using a "heavyweight" RE tool; it may be expensive, difficult to learn and impossible to customize [BAN]. The solution is to apply technology that is widely available, easy to use, nonproprietary and inexpensive, such as that offered by the Extensible Markup Language (XML) family of technologies. The Extensible Markup Language is a subset of the Standard Generalized Markup Language (SGML) and, like SGML, is a metalanguage that sets the standard for specialized markup languages. Developed in 1996, XML 1.1 was last published as a World Wide Web Consortium (W3C) recommendation in February 2004. The details of XML can be found in the Extensible Markup Language (XML) 1.1 W3C Recommendation 04 February 2004 [W3C].

The use of XML technology in software engineering has been explored for managing requirements [BAN], developing scenarios [PEN04] and in UML-based verification tools [MAR05]. XML is platform independent, portable, and is capable of representing many different types of data. The intent of this research is to recommend a framework that provides for the standardization and broad accessibility and utilization of evolving requirements data. We propose, therefore, to use XML to transform, standardize and document requirements based on the UML model generated in the

FIRE*m* activity, which is the visual representation of the faceted classification scheme generated in FIRE*c*. Figure 6 illustrates this approach.

In Chapter VI, we will demonstrate our approach for standardizing warfighter requirements using an XML Schema. As defined by [W3Ca], "XML Schemas express shared vocabularies and allow machines to carry out rules made by people. They provide a means for defining the structure, content and semantics of XML documents".

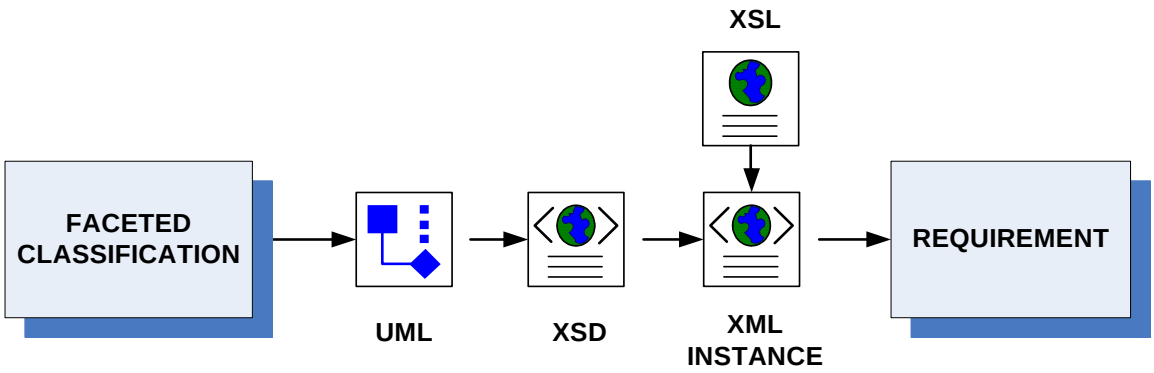


Figure 6. Transformation from Faceted Classification to Requirements using UML and XML

4. FIRE*x*: Data Exchange Capability

Information exchange using the web has become a standard way of doing business; organizations across DOD rely on the web for administrative and operational information, classified and unclassified. This exchange of information occurs using a variety of formats ranging from text documents, Excel spreadsheets, or PowerPoint files. However, in the past few years DOD organizations have begun to adopt the Extensible Markup Language (XML) to exchange information. As a standardized format for shared data exchanges across the web, XML has become one of the most popular and fastest growing text formats for web content since Hypertext Markup Language (HTML) first made its appearance over a decade ago.

Making requirements information electronically available in conforming XML documents has the potential to provide all stakeholders with an increased awareness of what is required to meet warfighters' needs. It can enhance their knowledge of the operational environment, its technical limitations and challenges and the usability of the

technological capability. Using XML to represent requirements ensures the content is not only standardized, but it is also accessible from virtually any location on the globe. With XML, evolving requirements information can be available across a common domain in a structured format specifically designed for those who need the information.

Some of the benefits of this approach are:

- Organizations are not required to rely on special software applications to access information because XML is platform independent.
- Developers across organizations can have access to the same set of user requirements.
- The ability to exchange requirements data prevents the duplication of requirements collection, analysis and documentation efforts.
- Knowledge of existing requirements may serve as building blocks for additional requirements that more precisely reflect the needs of the users.

C. SUPPORT FOR REQUIREMENTS MANAGEMENT EFFORTS

Although specific requirements management activities (as depicted in Figure 1) have not been formally included in the proposed framework, the classification of requirements we propose does aid in maintaining traceability of requirements and assessing the impact of changes. These functions are especially important when managing evolving requirements, primarily because of the likelihood there will be an increasing number of requirements changes as the system becomes more mature. Stakeholders will add new requirements as they discover more ways for the technology to expand to fit the operational environment. In addition, there will be an increase in the number of changes to existing requirements as warfighters gain knowledge of the system and can express their requirements in more precise qualitative or quantitative terms. Moreover, change management and analysis functions provide for the identification and elimination of duplicate requirements, as well as the means to clarify ambiguous or ill-defined requirements.

D. SUMMARY

Very few software development projects occur in an environment where requirements engineers and development teams have access to all stakeholders. It is more often the case where the customer who is funding the project represents system users during the requirements development process; the personnel who will be using the system may be geographically separated from the development team or may be unavailable due to operational commitments. For this reason, we believe the web and the XML family of technologies are resources capable of facilitating the development and management of evolving requirements.

V. THE REQUIREMENTS CLASSIFICATION SCHEME

A. INTRODUCTION

In the framework proposed by this research, we use a layered faceted classification scheme to represent evolving NETWORK-CENTRIC requirements by categories, abandoning the traditional classification of all requirements as either functional or non-functional. Commonly used in library science to catalog and organize information, faceted classification uses [WYN92]:

> …clearly defined, mutually exclusive, and collectively exhaustive aspects, properties, or characteristics of a class or specific subject. Such aspects, properties, or characteristics are called facets of a class or subject.

Faceted classification schemes are also used outside of the library science discipline. A recent study discovered 60% of e-commerce web sites use some form of faceted classification to represent information [ADK05]. Popular websites such as Wine.com and Epicurious.com utilize faceted classification schemes to allow customers greater latitude in searching for products and services. In fact, wine is often the subject of demonstrations on faceted classifications because of the many dimensions along which it can be classified, as illustrated in Figure 7. Faceted classifications have also been suggested in software engineering for cataloging reuse components [PRI91] and industrial automation software components [LUC01].

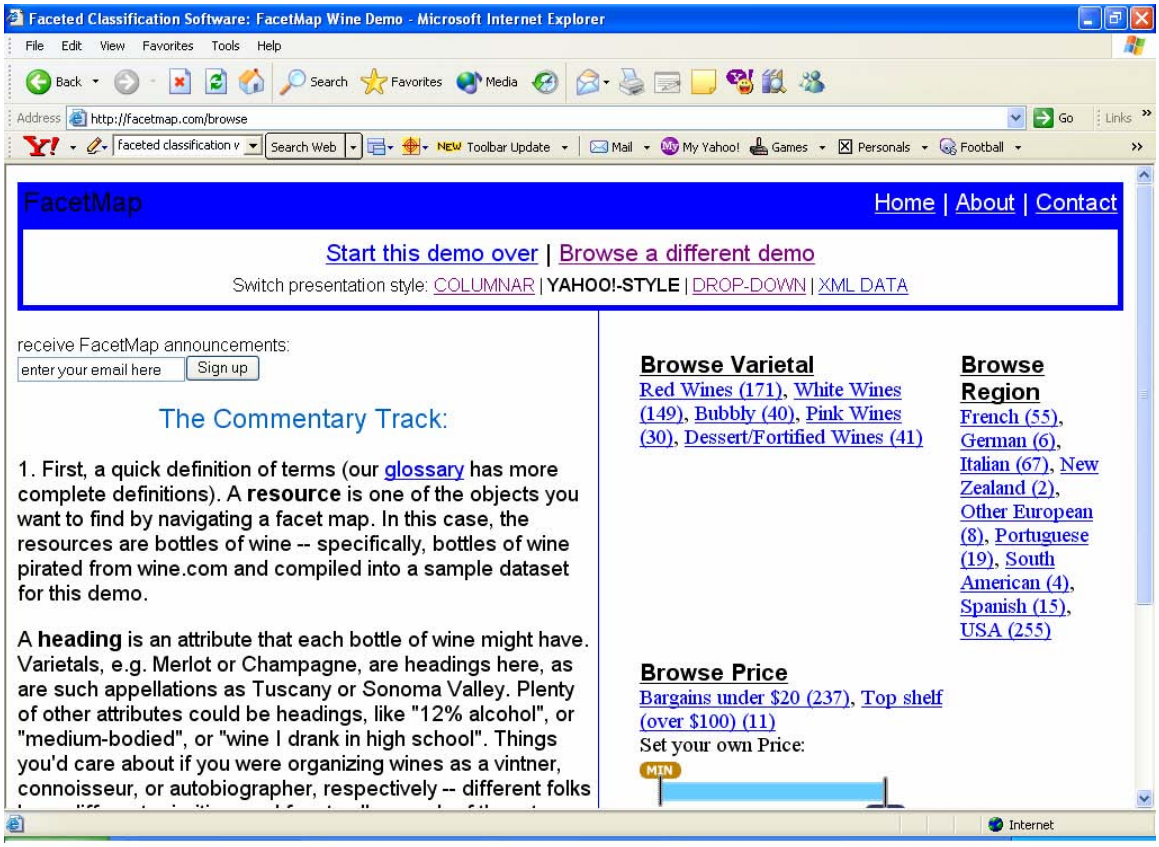


Figure 7. Example of Web-based Faceted Classification (From: www.facetmap.com)

B. A LAYERED APPROACH TO CLASSIFYING REQUIREMENTS

Stakeholders will often have different perspectives on requirements based on their role in the project. Stakeholders who are not users may have requirements for a software system based on fiscal or regulatory constraints, or that they express as functions of their overarching business objectives [LEF03]. Some of these non-user stakeholders may be concerned only with the system's ability to meet the goals of the organization and not with the implementation details. Users, familiar with the operational environment, may be able to express what they want the system to achieve, but they may not know if their requirements are technically feasible or if they conflict with other requirements. Finally, developers must be able to design and implement system features based on verified, specified requirements. As the requirements development process moves from elicitation to specification, the requirements should become more precisely defined, moving from a relatively abstract goal-based need to a requirement that is unambiguous and consistent.

A layered approach to requirements classification is recommended to address the varying levels of abstraction in requirements definitions. It is also beneficial when the classification is composed of many elements. Although this type of fine-grained classification results in a precise definition, it may be difficult to manage. We build upon the concept of levels of requirements discussed in [WIE03]. Requirements become more exactly defined as they traverse from the bottom layer of the classification to the top; each layer of requirements information builds upon the last. We identify the layers developed in FIRE*c* as Strategic (non-user stakeholders), System (user) and Software (developer), as shown in Figure 8. It is important to note the terms 'system' and 'software' appear in [SWEBOK] to refer to user requirements and system requirements, respectively. We find the use of these terms appropriate in this context, as well. In this thesis, we apply our framework at the Strategic and System layers of the classification, with analysis at the Software level reserved for future work. We have chosen to focus our efforts on developing the Strategic and System layers of the classification scheme. It is at this degree of abstraction we can demonstrate how our approach classifies, structures and standardizes evolving user requirements. Classification at the Software layer, while essential, is best implemented during a software development process guided by the expertise of the software development team.

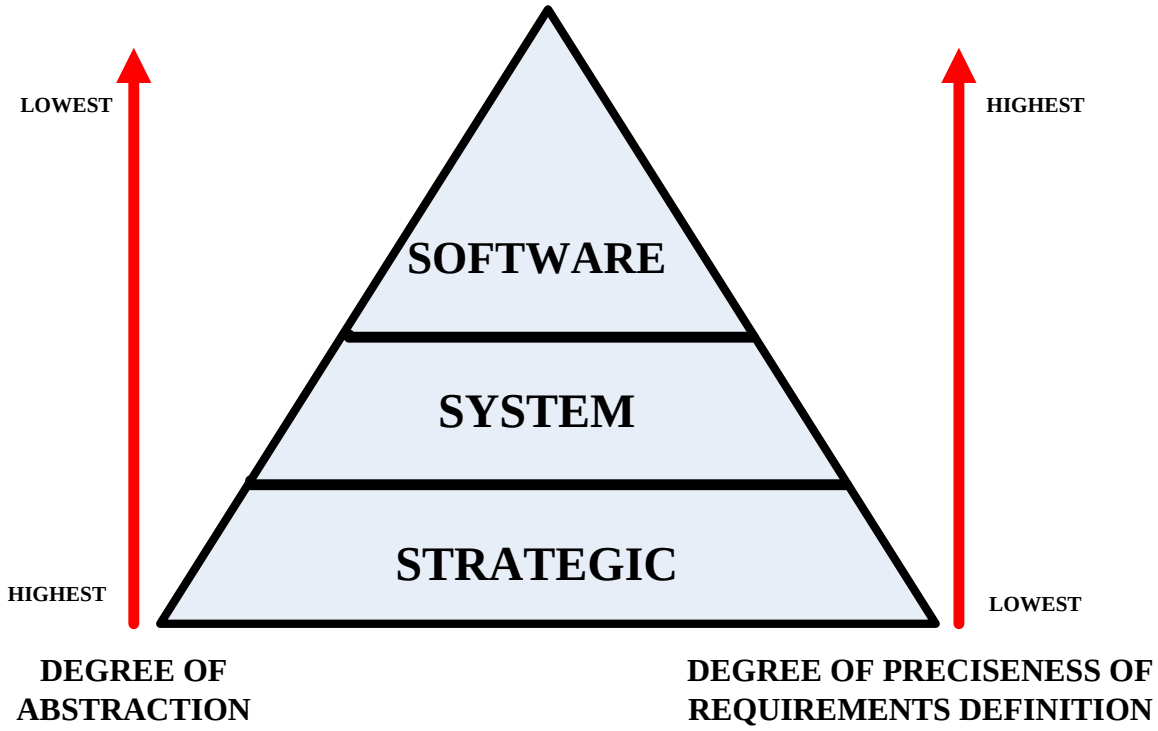


Figure 8. Layers of Software Requirements Classification

At the most abstract layer, called Strategic, we classify requirements according to the non-user stakeholders' high-level operational goals or objectives for the system. Classifying a requirement at the highest level ensures the system meets the organizations strategic goals and focuses on the "why" of system development [OGC]. The System layer then builds upon the classification scheme in the Strategic layer and includes facets for user-defined requirements. This layer focuses on the "what" of user requirements, i.e. what the users want the system to achieve. At the Software layer, the focus is on the "how" of system development and the requirements are even more rigorously defined with a classification scheme based, in part, on the work of [GLI05].

1. Strategic Layer Requirement

A requirement at the strategic layer is in its most abstract form. It is defined in accordance with the high-level objectives, or goals for the software system. As discussed, linking requirements to the underlying goals improves requirements consistency and manageability and ensures the resulting system reflects the stakeholders' needs. Either a requirement is defined initially by the characteristics of the Strategic layer, or it can be

decomposed from a fully specified requirement into the form delineated by the Strategic layer. Either way, the requirement must possess the attributes of the Strategic Layer to be fully defined according to the classification scheme. An example of FORCEnet[2] requirements at the Strategic layer taken from [CNO05]:

1. Provide each decision maker the ability to depict situational information in a tailorable, user-defined, shareable, primarily visual representation.
2. Provide distributed groups of decision makers the ability to cooperate in the performance of common command and control activities by means of a collaborative work environment.
3. Store, catalogue and retrieve all information produced by any node on the network in a comprehensive, standard repository so that the information is readily accessible to all nodes and compatible with the forms required by any nodes, within security restrictions.

The preceding examples are capabilities the FORCEnet architecture must support, i.e., they are requirements for systems operating under the FORCEnet construct. These requirements are expressed as objectives for the system to achieve. For example, the first requirement supports the goal of maintaining situational awareness, the second requirement supports command and control, and the third requirement is necessary for systems to provide information management.

2. System Layer Requirement

The System layer includes attributes that define the requirements from the user's perspective. As discussed, a System layer requirement states what the user expects the system to achieve; the tasks the system must be able to provide support for the user to conduct work. A requirement at this layer is linked to the underlying Strategic layer requirement, which is based on the high-level objectives for the system. For example, the third Strategic requirement in the preceding paragraph refers to the need to store, catalog and retrieve information, a requirement that supports information management. We may state a requirement at the System layer of the classification as follows: "All Navy chat messages must be time stamped with a date and time to allow recreation of events". This

[2] FORCEnet is the architectural framework and operational construct for Naval Network-Centric Warfare. It is "a critical link in network-centric warfare and a transformational architecture for the Navy and Marine Corps that integrates sensors, networks, decision aids, weapons and supporting systems into a highly adaptive human-centric maritime system that operates from the seabed to space and from sea to land." SPAWAR, RADM Slaught, Keynote Address, 2002. Retrieved 16APR06 from: http://www.chips.navy.mil/archives/02_Summer/authors/index2_files/network_centric.htm

requirement meets the high-level objective expressed at the Strategic layer because it supports the cataloging function of information management. It also answers "what" the users expect the system to achieve.

3. Software Layer Requirement

A requirement at the Software layer is the most precisely defined requirement. Not only is it linked to the high-level objectives for the system and the user's needs, but it also includes information necessary for specification and verification during the development process. Using the previous example, a requirement at the Software layer may be, "The system will store a timestamp consisting of ISO 8601 date and time as an attribute of the message data."

C. FIRE*c*: APPLICATION OF FACETED CLASSIFICATION SCHEME TO EVOLVING REQUIREMENTS OF NETWORK-CENTRIC TECHNOLOGIES

The use of a faceted classification scheme to categorize requirements overcomes the difficulties associated with classifying requirements with quality and functional attributes. It provides for the categorization of "fuzzy" non-functional requirements—requirements that possess both functional and non-functional attributes or that have functional or non-functional characteristics depending on how stakeholders express them. Figure 9 from [GLI05] depicts an alternative approach to the typical classification of requirements with four mutually exclusive categories—Representation, Kind, Satisfaction and Role. Each of the categories, or facets, consists of elements commonly used to define and specify requirements. Characterization of requirements is achieved by varying the combinations of facets. Table 1 illustrates examples of requirements characterized using this classification scheme. [GLI05]

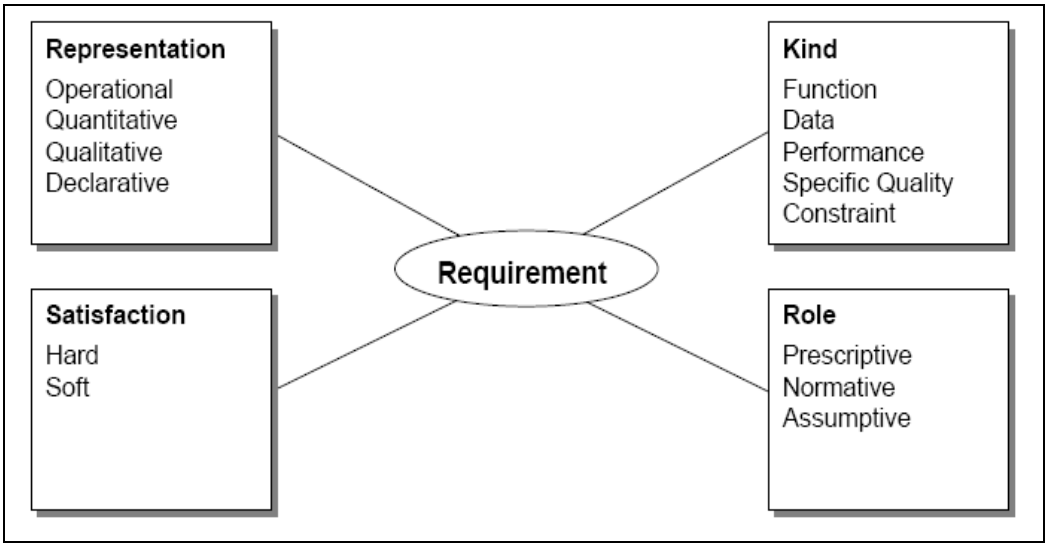


Figure 9. A faceted classification of requirements (From: [GLI05])

REQUIREMENT	*CLASSIFICATION*
"The system shall compute the sum of all applicable deductions."	Kind: function Representation: operational Satisfaction: hard Role: prescriptive
"The system shall be easy to use by casual users."	Kind: specific quality Representation: qualitative Satisfaction: soft Role: prescriptive
"The response time shall be less than 1 s on average"	Kind: performance Representation: quantitative Satisfaction: soft Role: prescriptive

Table 1. Requirements characterized by faceted classification (From: [GLI05])

There are significant benefits associated with using a faceted of classification scheme to characterize requirements in addition to those offered by [GLI05]. It is much

easier to update than a purely hierarchical classification (tree structure); categories, each supporting multiple hierarchies, are added as needed to the faceted classification. However, in a hierarchical classification, rearranging the hierarchy is necessary to accommodate the new information. The faceted classification scheme is evolvable and extensible; there is no limit to the number of categories and each category can accommodate an unlimited number of requirement attributes. In addition, the controlled vocabulary results in improved search capabilities for documented information; the scheme creates a common vocabulary for the requirements, and keywords from this vocabulary define the search parameters. The user has the flexibility to determine along which axis to conduct the search for requirements information. Finally, a faceted classification is a natural fit with XML representation of data; XML can represent each facet of the classification using elements and attributes.

There are also many benefits associated with using a faceted classification to represent evolving requirements for software systems supporting NCW. The ability to tailor and quickly adapt the classification schemes to the domain reflecting technological advances is especially important in the rapidly changing NCW environment. As discussed in the preceding section, two of the key benefits associated with faceted classification schemes are that they are evolvable and extensible. The ability to modify the specification to be either more precise by adding more facets or more abstract by using fewer facets [LUC01] is a critical consideration when the group of stakeholders is diverse, representing not only different organizations, but also different positions of authority in the military command structure. In addition, the ability to rapidly search and evaluate requirements is an important consideration because changes to systems supporting NCW operations can occur quickly and frequently. This is a vast improvement over current methods we have observed of documenting and accessing evolving requirements. Requirements information from informal sources is often submitted and documented in natural language, making it difficult and extremely time consuming to sort and evaluate without the use of parsers. Finally, a faceted classification helps resolve or eliminate ambiguous or poorly defined requirements before submission to the software development process.

The facets, when applied to the software requirements domain, must represent logical divisions of information about the requirements, namely their attributes, behaviors and constraints. However, we submit that facets can also be used to categorize additional attributes of the requirements to provide amplifying information particular to the NCW operational environment, clarification on warfighter input or to address inconsistencies in the data. In addition, we believe facets may be included in the classification to document a requirement's metadata, such as originator, priority, date created or modified, etc. The faceted approach suggested by [GLI05] offers a new way to approach classifying requirements. As suggested by its author, extending the classification scheme with additional facets is a consideration to enhance its usefulness in requirements classification.

D. FIRE*m*: MODELING THE FACETED CLASSIFICATION

As discussed in Chapter II, modeling is used to visually represent requirements information, improving the elicitation efforts between stakeholders and developers. Modeling can produce a picture that needs very little, if any, technical expertise to understand. It presents information at a well organized, high-level of abstraction, effectively isolating stakeholders from the technical details and allowing them to concentrate on the "bigger picture", namely ensuring the modeled requirements reflect their needs.

We submit the same holds true for modeling the classification of requirements. Specifying a model captures all of the information germane to our proposed requirements classification scheme. To support our work, we propose to use UML to model the faceted classification scheme developed in FIRE*c*. With UML modeling, the facets and attributes can be represented using a widely recognized specification language and at high level of abstraction that is understandable, as well as extensible. As an example, [LUC01] demonstrates how UML modeling of a faceted classification scheme improves the understanding and functionality of the components in a reuse environment.

Modeling a faceted classification using UML is not an extremely complicated or time-consuming process. A class is an entity that represents each main facet, or heading,

of the classification. The elements under each heading are attributes of that class. Because we are not concerned with modeling functionality, methods will not appear in the class diagram. We model the relationships between classes with the same notation used when modeling software applications. For the Representation and Kind classes, we have used the commonly used "has a" and "is a" phrases to demonstrate the suitability of the aggregation and inheritance relationships, respectively. The Satisfaction and Role classes are modeling as general associations. Figure 10 illustrates one possible UML model of the requirements classification shown in Figure 9.

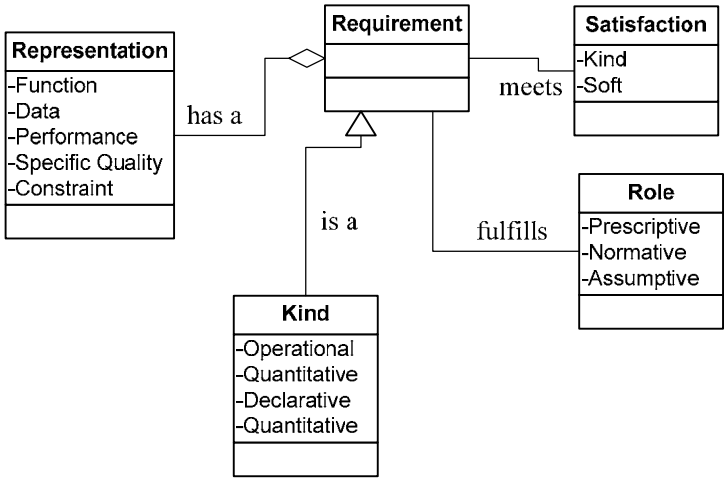


Figure 10. UML Model of Requirements Classification from Figure 9

E. SUMMARY

We propose a layered faceted classification to address the different levels of abstraction in requirements definitions. UML modeling provides a comprehensive, visual depiction of the faceted classification scheme, improving communication between stakeholders during the collection efforts of evolving requirements. Requirements standardization (FIREs), the next activity in the FIRE framework, will be discussed in the following chapter as we present our case study.

VI. CASE STUDY

A. SYSTEM SELECTION

Instant messaging that involves groups of people, commonly called 'chat', has evolved as a critical Network Centric Warfare capability in U.S. Naval operations, providing military commanders and thousands of military personnel with the ability to conduct multiple, real-time conversations. Once confined to social or casual interactions in the general community, Commercial-Off-The-Shelf (COTS) chat applications such as MS Chat and Lotus Sametime are often used by surface Navy personnel in lieu of radio transmitter (RT) circuits to coordinate and conduct command and control operations, as well as logistical and administrative functions. As such, chat has become a primary collaborative tool used in establishing and maintaining situational awareness (SA) through knowledge sharing particularly in the maritime domain.

However, as the chat technology becomes more mature and embedded in tactical operations, Navy personnel discover the applications do not fully meet their requirements. Evaluations of chat applications during Fleet exercises, such as Trident Warrior in support of FORCEnet, have determined that chat applications currently in Navy ships do not meet all of the warfighters' needs [CAT05a]. Although there has been much work in identifying these shortfalls and replacing or modifying the tools to support the requirements, there continue to be difficulties in meeting the rapidly changing needs of the warfighter. To date, there is no existing methodology or process to include evolving requirements in the development process of the follow-on system to ensure all user requirements are satisfied. The development and subsequent utilization of such a method will ensure applications are developed and that meet not only the formally identified requirements, but also those that result from day-to-day operational use.

We have chosen to focus our efforts on applying the framework proposed by this research to the U.S. Navy's real-time, online communication systems, commonly referred to as 'chat' tools. We will henceforth refer to these tools collectively as Navy Chat.

B. PRELIMINARY ANALYSIS OF CHAT REQUIREMENTS

Since 2001, several organizations have been involved in collecting and analyzing data on Navy Chat use, usability, architecture, and required functionality, as listed in Appendix A. The compilation of the resulting information is part of a larger attempt to develop a source of information for the facilitation of discussions, to build consensus and to identify gaps in user requirements [CAT05]. It is important to note this is not an exhaustive listing of all Navy Chat requirements. We chose to limit the focus of our study to the requirements in [CAT05] because they represent the needs of naval maritime chat users, the focus of our study.

We added the Navy Chat user requirements from [CAT05] to an Excel spreadsheet to facilitate an initial examination of the data. We then sorted the requirements by date and graphed them as a function of time to illustrate the number of requirements collected from system users during field evaluations and from surveys over a four year period. As shown in Figure 11, the number of documented user requirements has increased dramatically from 2001 to 2005. The problem of managing evolving requirements initially appeared to be determining how to handle this significant increase.

Although we expected an increase in requirements over time indicating users have become more familiar with the technology, the sharp increase from 2003 to 2005 was thought provoking. A closer look at the requirements data resulted in some interesting conclusions. For one, there appeared to be several duplicate requirements; the authors identified similar requirements from one year to the next, but used slightly different wording, e.g., the requirements “provides logging capabilities” and “logs chatroom conversations”. Two, there was no standardized way of representing requirements; the requirements information was extremely high-level and abstract with no identified attributes, behaviors, or constraints. Three, all requirements appeared to be created equally; priorities were not assigned to the requirements. We recognize that [CAT05] has been developed as a quick reference for Navy Chat user requirements and is not intended to document fully specified software requirements. However, we believe our framework will be useful for creating a standardized source that can serve as both a quick reference to the growing number of requirements and as input to the software development process.

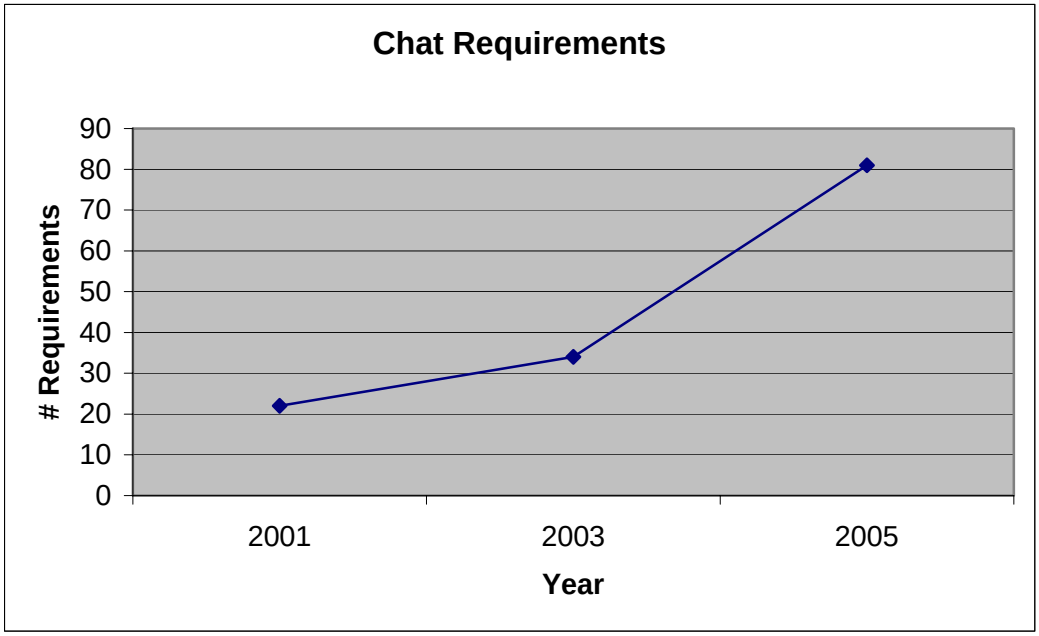


Figure 11. Chat Requirements 2001 – 2005 (From: [CAT05]

D. FIRE*c*: BUILDING THE CLASSIFICATION SCHEME

Developing a classification scheme for evolving requirements entailed creating facets based, as closely as possible, on goal and requirements-oriented terminology used by the stakeholders in the NCW operational environment. This ensured the classification scheme represented requirements information particular to the NCW domain.

There are several recommended procedures for building a faceted classification scheme [RAN62][SPI98][DEN03]. The critical steps of these procedures are summarized as follows:

1. Content analysis: Investigate the subject domain to retrieve common terms.
2. Group common terms: Combine similar terms identified in the first step.
3. Facet Creation: Identify major categories that can be represented as relevant, mutually exclusive facets.
4. Facet ordering: Place terms in appropriate facets. Order facets and terms in a manner appropriate to the planned retrieval.

The development of each layer of our proposed classification scheme will follow the above steps. However, rather than perform one content analysis for the entire

requirements domain, we will perform a content analysis for the Strategic and System layers in the classification scheme.

1. Strategic Layer

a. Content Analysis

A content analysis at the Strategic layer required an examination of documents that could answer "why" the software system is required. In particular, documents that describe the types of high-level objectives the stakeholders expect the system to meet. The documents selected for the content analysis at this layer appear in Appendix A. With Navy Chat applications as the focus of our case study, the analysis must include NCW, Navy Chat, and FORCEnet related documentation. We found reoccurring terms and concepts throughout the available, unclassified documents on the aforementioned topics. Because the intent at the Strategic layer is to define requirements in accordance with high-level goals, we focused on identifying goal-related terms or phrases. The terminology we extracted from the documentation and used for our Strategic layer content analysis is shown in Table 2.

We found the process at this level required subjective analysis. For example, we discovered the term 'coordination' frequently used in the documentation, as well as 'situational awareness'. Based on experience, 'situational awareness' can also refer to 'real-time coordination'. Thus, one of our objectives for the content analysis was to select terminology that was the least likely to result in facets with dual meanings. A possible solution may be to develop an associated list of synonyms for each facet, carefully selecting terms to support the mutual exclusivity requirement. Again, the content analysis at this layer was largely subjective due to the abstract nature of the topic under investigation.

NCW/NCO	FORCENET	CHAT
Increased battlespace awareness	Reliable communication (comms)	Information and knowledge mngmt
Improved command and control	Store, catalog, retrieve information	Interoperability
Rapid, superior decision making	Readily accessible information	Enhanced situational awareness
Coordination of complex military ops	Visual display of situational information	Bandwidth efficient comms
Self-synchronization	Share situational information	Secure, authentic comms
Improved understanding of higher command's intent	Collaborative environment for command and control activities	Reliable comms
Improved understanding of operational situation	Information Assurance	Support collaboration
Reduce uncertainty of fighting	Multiple security domains and levels	Real-time conversation
Shared knowledge	Tracking and engagement information on environmental, neutral, hostile elements	Up-to-the-minute command and control
Speed of command	Process, sort, analyze information	
Increasing responsiveness	Accessibility to raw data	
Dissemination of commander's intent	Survivability during network outages	
	Standard repository of compatible information	

Table 2. Strategic Layer Terminology

b. Facet Development

The intent of using facets to classify requirements is to provide a more precise definition than that achieved by using natural language. The facets developed from the content analysis must be representative of the domain under consideration, easily understood, and sufficient in quality and quantity to provide the desired level of precision in requirements definition. Developing the Strategic layer of the classification is the first step in this process, linking high-level goals to requirements. To that end, we combined the common terms and phrases identified in the content analysis associated with goals and designated the primary facet as 'Goal'. Within this facet, the goal-related attributes are shown in Figure 12.

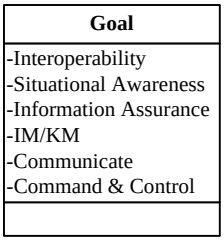


Figure 12. Goal Facet at Strategic Layer

We identified three additional facets at the Strategic layer—Behavior, StrategicAgent, and Dimension (Figure 13). The Behavior facet identifies the general capabilities of all FORCEnet systems, and the StrategicAgent facet refers to NCW/NCO constituents affected by the objective. The force and command facets are modeled using inheritance and appear as subclasses of the StrategicAgent superclass. The elements of these facets are examples chosen to illustrate our approach; there may be additional elements appropriate to the NCW/NCO and FORCEnet domain. The Dimension facet reflects the areas upon which developmental efforts of FORCEnet systems must focus [CNO05] and serves to define the general domain for the Strategic layer requirement.

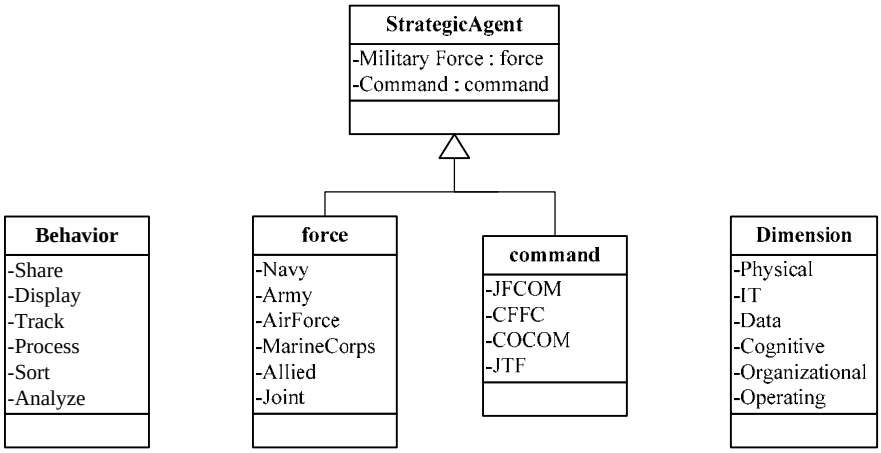


Figure 13. Strategic Layer Facets

The remaining facets classify additional supporting information, or metadata, about the requirement such as the author, its descriptive data and the system to which it applies (Figure 14). Included in the supporting metadata is a Priority facet to

allow stakeholder-assigned requirements priorities, supporting preliminary risk management efforts. There is technically no limit to the number of facets in this type of classification scheme, but we limited our focus to a number sufficient to demonstrate our approach. It is important to note the metadata can be included at either the Strategic layer or the System layer; we will illustrate both approaches. The UML diagrams in Figures 16 and 17 illustrate all of the facets developed for use at the Strategic layer. We will discuss these diagrams in the following section.

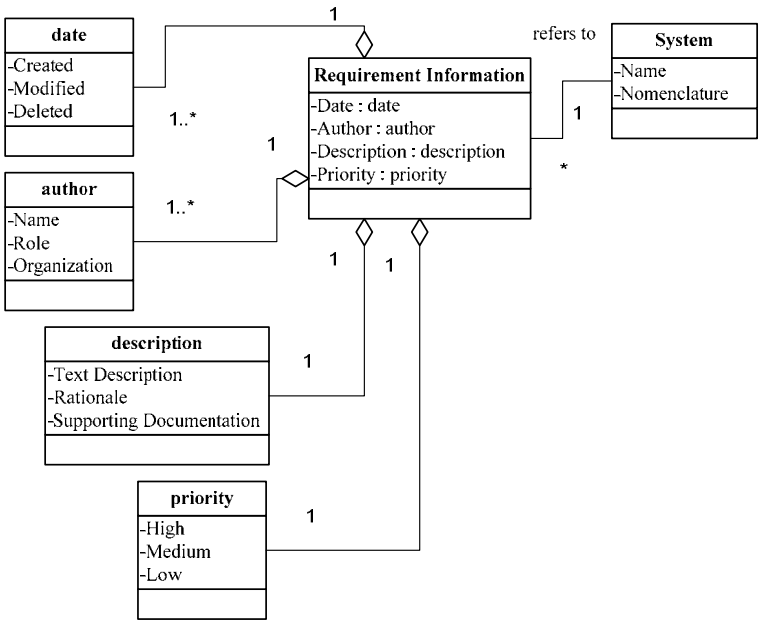


Figure 14. Supporting Requirements Information Metadata

The facets chosen for the Strategic layer provide the most abstract classification of requirements, one based on high-level strategic goals, as well as metadata about the requirements themselves. The facets meet the mutual exclusivity requirement. For example, a goal can never be an agent or a service.

2. System Layer

a. Content Analysis

The content analysis at the System layer entails evaluating documentation pertaining specifically to Navy Chat. The purpose of the content analysis at this level is to build upon the results of the content analysis at the Strategic Layer and establish the common vocabulary to help answer "what" the user needs the system to do. We

evaluated several unclassified documents, listed in Appendix A, and the results are shown in Table 3. Again, it is important to note the terminology in Table 3 is only a sample from the available documentation used to demonstrate our approach.

During the content analysis at the Strategic level, we found a considerable number of technical concepts used in conjunction with operational objectives. For example, it was clear the phrase "visual display of situational information" referred to the importance of maintaining situational awareness. It also referred to the use of a visual display, which underscored the importance of human-computer interface (HCI), or user-computer interface, technology to goal achievement. At the System level, we also discovered terminology consistent with the need for HCI technology support; the content analysis terminology at this layer refers to message characteristics implemented through the user interface, such as font, text color, etc. Reflecting the subjective nature of the content analysis and facet selection, we chose to incorporate a facet called 'Supporting Technology' at the System layer that is based on terminology found at both the Strategic layer and the System layer.

		CHAT		
Chat	Text-based Message	Persistent	Unix/Windows compatible	Control Access
Authenticate	Search	View	Send	Network stability
Support multiple applications	Communicate to limited listeners	Listen	Receive	Configurable
Broadcast to multiple users	Semi-permanent chatrooms	Distributed server architecture	Client-to-server architecture	Organic chat capability
Private chatrooms	Standing chatrooms	Supports emoticons	Monitor	Alerts users to new message
Support low data rates	Public chatrooms	Server-server data compression	Servers Auto-Reconnect	Font is tailorable
Message is easy to read	Supports good usability practices	Whisper	Share files	Supports multiple layouts
Tile windows	Resize windows	Alert users to lost comms	View multiple rooms	Filter

Table 3. System Layer Terminology

b. Facet Development

As with the Strategic layer, the facets reflect the grouping of like terms. We identified the first facet for the System layer as 'Supporting Technology'; each requirement is associated with a general technology. Additional categories derived from the content analysis and grouping of like terms include the Process, Component and System Agent facets (Figure 15). The Process facet includes system capabilities in support of the Navy Chat user's common functions, such as chat, whisper, send and view. The Component facet includes the category of Navy Chat system components to which the requirement applies, such as hardware, software, server, or client. The System Agent facet identifies the specific stakeholder that the requirement must address at the System layer, to include the user. Because of the subjectivity inherent in this process, there may be redundancy in the Supporting Technology and Component facets. However, any redundancy or inconsistency can be addressed by either removing or modifying facets, illustrating how our approach meets the changeability principle as discussed in Chapter III.

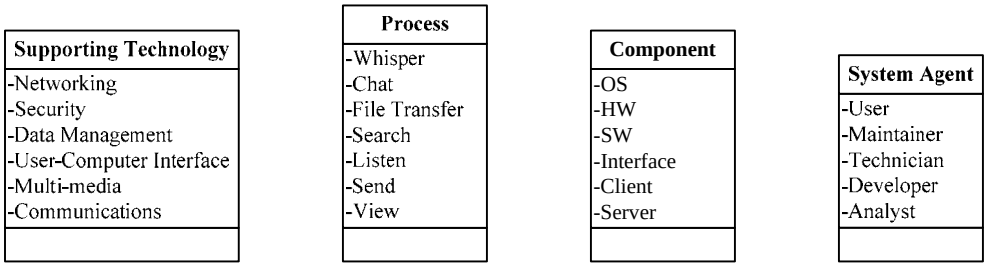


Figure 15. System Layer Facets

As with the Strategic layer, all facets appear to meet the mutual exclusivity requirement for faceted classification. However, there may be one possible exception. We identified Communicate as a high-level goal; the chat systems supporting NCW operations must satisfy the communication requirements. We also identified Communications as a Supporting Technology; communications technology usually refers to the equipment supporting the act of communicating.

E. FIRE*m*: MODELING THE FACETED CLASSIFICATION SCHEME

The UML models shown in Figures 16 through 19 represent the Strategic and System layers of the faceted classification scheme developed in FIRE*c*. UML notation denotes the relationships between facets, as well as the multiplicity of the associations. A reference identification attribute identifies the requirement at every layer of the classification. We show alternative methods for modeling the supporting requirements metadata, in part to demonstrate the ease of change associated with this approach. Changing the degree of requirements definition entails adding, deleting or modifying facets in any layer of the faceted classification and representing these changes in the respective UML models. Not only is the classification scheme easy to change and tailorable to any specific domain, but using UML to model it means its visualization is easy to change, as well.

1. Strategic Layer UML Model

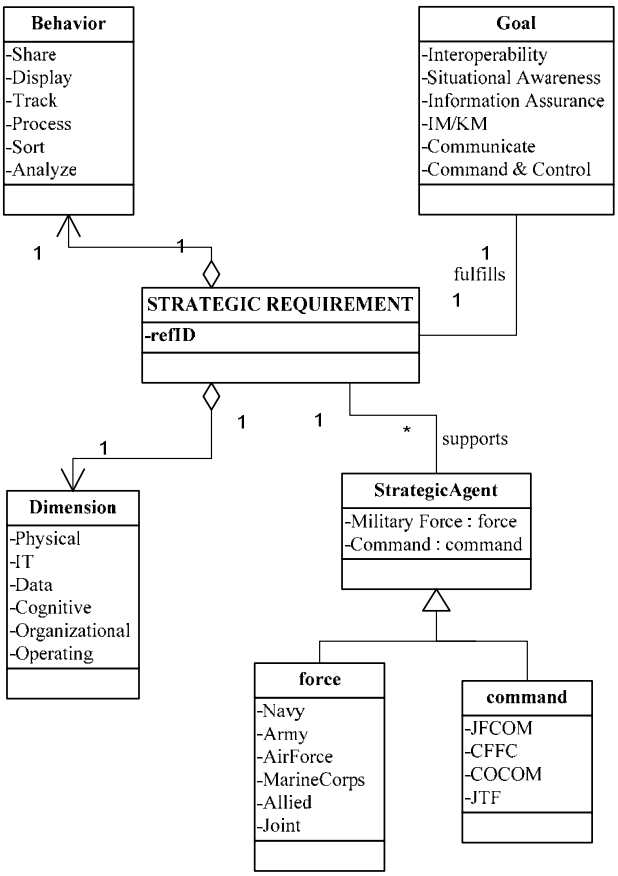


Figure 16. UML Model of Strategic Layer Classification

2. Strategic Layer UML Model with Supporting Metadata

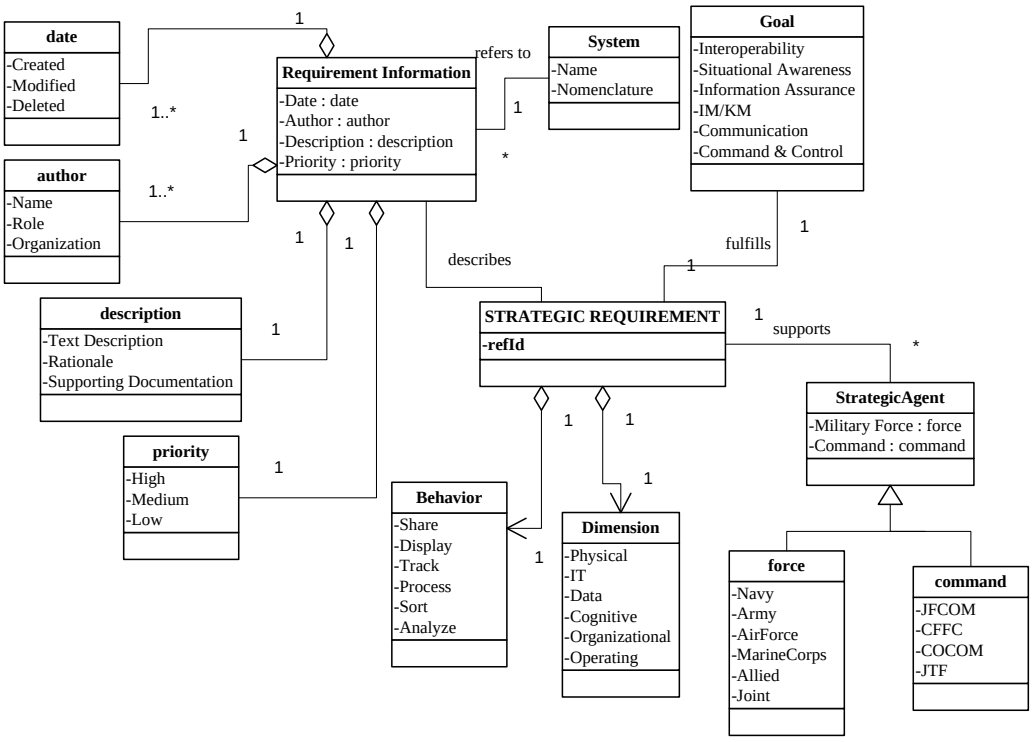


Figure 17. UML Model of Strategic Layer Classification with Supporting Metadata

3. System Layer UML Model

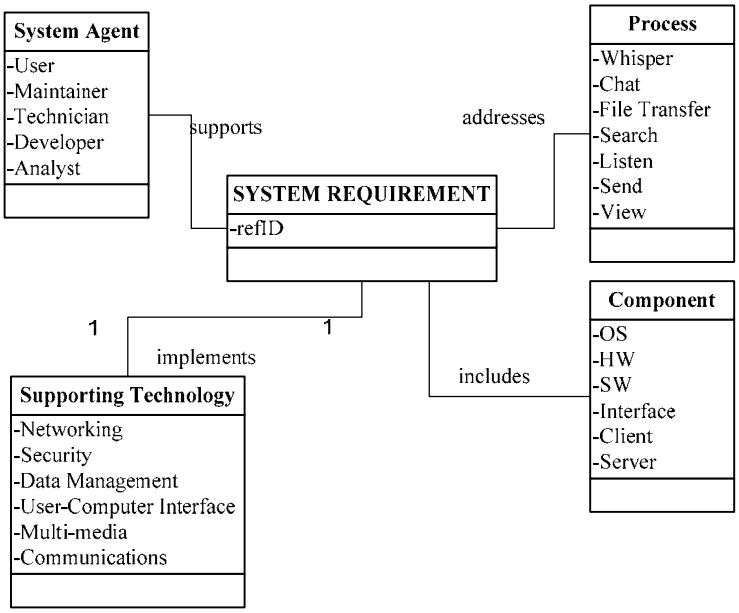


Figure 18. UML Model of System Layer Classification

4. System Layer UML Model with Supporting Metadata

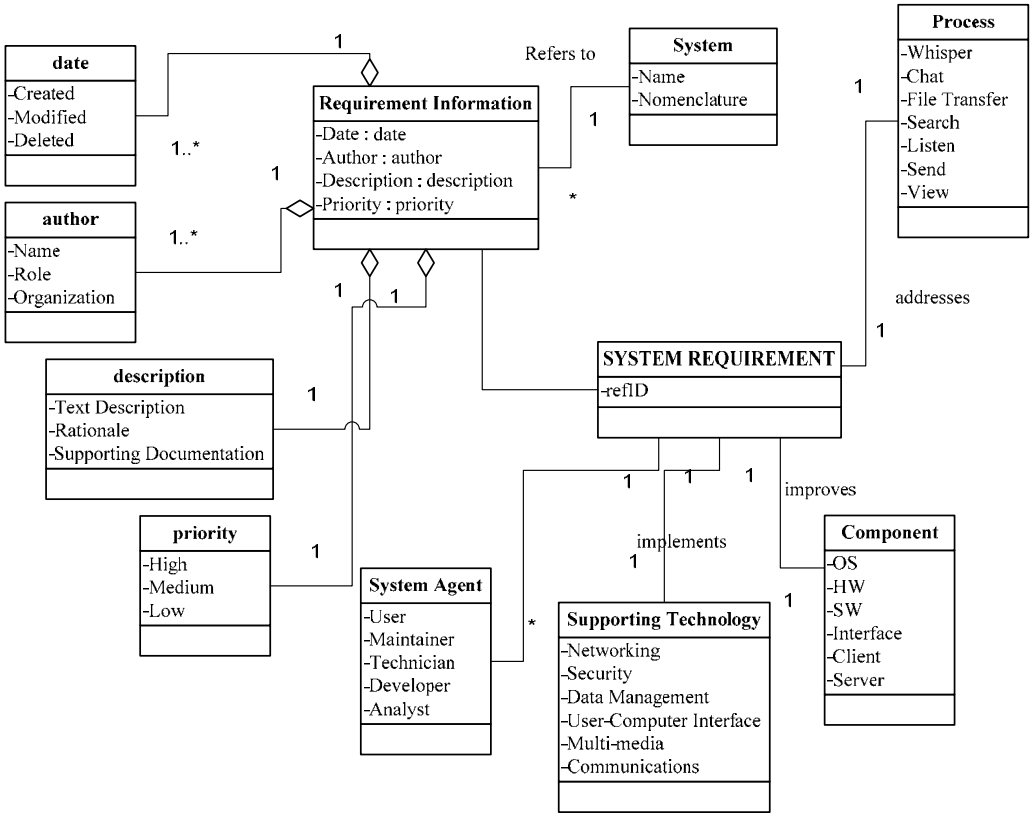


Figure 19. UML Model of System Layer Classification with Supporting Metadata

F. OVERVIEW OF CLASSIFICATION APPLICATION

To implement our approach, we determined we needed to link the requirements to the high-level goals (objectives) for the system and we required a standardized approach for classification. We selected the two primary facets—Goal and Dimension—based on our content analysis for the first layer of the classification scheme. As discussed, the first layer of facets allows for rapid sorting and evaluation of requirements based on keywords defined by the highest, most abstract goals in the classification. Evaluation at the first layer helps to eliminate the most obvious discrepancies, such as duplicate requirements.

To support a preliminary evaluation of the requirements data by our classification scheme, we developed a simple Access relational database, incorporating selected facets at the Strategic layer. The preliminary requirements data from the generated Excel spreadsheet was imported and each requirement was evaluated and categorized by the facets, beginning with the Strategic level. Appendix B illustrates the classification of

requirements by Strategic facets, Dimension and Goal, and the System layer facet Supporting Technology.

The first evaluation of the documented requirements indicated we could perform the classification using the Strategic level facet Goal. However, using one facet only provided minimal refinement because of the degree of abstraction in the documented requirements. As a result, we determined we could achieve an even more refined classification of the requirements at the Strategic level by including the Dimension facet in the classification scheme. Taking the classification one step further using the Access database, we included the System layer facet Supporting Technology in our analysis.

The initial analysis of the requirements defined by the selected Strategic layer and System layer facets of our classification indicated there were indeed duplicate requirements. The inconsistencies were apparent when the requirements were categorized according to the classification scheme suggested above. The first set of requirements data we analyzed included those requirements categorized by the goal IM/KM (information management/knowledge management). The data is shown in Table 4. The requirements in italics are those we found to be inconsistent (duplicates) upon applying the faceted classification scheme using the two primary facets of the Strategic layer and one System layer facet. In both cases, the requirements originated three years before their modification. This analysis substantiated our use of the faceted classification scheme to improve the process of defining and classifying requirements, as well as the searchability of requirements data.

Requirement	Goal	Dimension	Supporting Tech
Automatic download of logs	IM/KM	IT	Networking
Logs should be controlled centrally at the server	IM/KM	IT	Networking
Messages should be sent to server	IM/KM	IT	Networking
Logs chatroom conversations	*IM/KM*	*IT*	*Data Management*
Provides logging capabilities	*IM/KM*	*IT*	*Data Management*
Logs non-permanent chatrooms	*IM/KM*	*IT*	*Data Management*
Ability to control how much of the historic log is downloaded when user logs on	IM/KM	IT	Data Management
Logs should be searchable within a certain time segment	IM/KM	IT	Data Management
Logs must be readily available to users for review of past events	IM/KM	IT	Data Management
Logs the entry/exit of members in the room	IM/KM	IT	Data Management
Ability to search for specific information since last logon	IM/KM	IT	Data Management
Timestamp Messages	*IM/KM*	*Data*	*Data Management*
Timestamp messages should include date and time	*IM/KM*	*Data*	*Data Management*
Supports file transfer	IM/KM	Data	Data Management
Logging of private messages should be configurable	IM/KM	Cognitive	User-Computer Interface

Table 4. Analyzed IM/KM Requirements

G. FIRE*s*: STRUCTURED REQUIREMENTS--DEVELOPMENT OF THE XSD

As discussed in Chapter IV, the purpose of the XML Schema is to define and constrain the content and structure of XML documents. We followed an object-oriented approach to develop the XML Schema (XSD) necessary to define both the vocabulary and structure as represented by our layered faceted requirements classification. This approach supports extensibility, allowing for changes to the type and scope of requirements data. Using the UML diagrams developed in FIRE*m*, we associated each layer of the classification scheme with a set of XML Schemas; each set is comprised of a schema representing the class structure and a schema defining the attributes. Each classification layer's set of schemas inherits the composition and attributes of the preceding one, as shown in Figure 20. An association class represents the supporting metadata, or requirements information that enhances the classification. As discussed, this

metadata (requirements information) is not unique to any particular layer in the classification. This decomposition allows additional facets to provide amplifying administrative information without complicating the classification scheme and allows for reusability. Because we use the UML representation of the classification to derive the XML Schema, the instances (XML documents) that conform to the schema contain the information needed to define evolving requirements.

The tools used to develop the XSD and supporting XML documents are MDXSYS Limited XMLobjective 1.2© [MDX] and Altova XMLSpy v2006 sp2 [ALT], both Integrated Development Environments that support the visual design of XML Schemas and editing and validation of XML documents. It is important to note the schemas developed in this work only represent functional concepts; they do not conform to the Department of the Navy XML Naming and Design Rules Version 2.0 [DONCIO]. Implementation of the approach recommended in this thesis would require full compliance with [DONCIO].

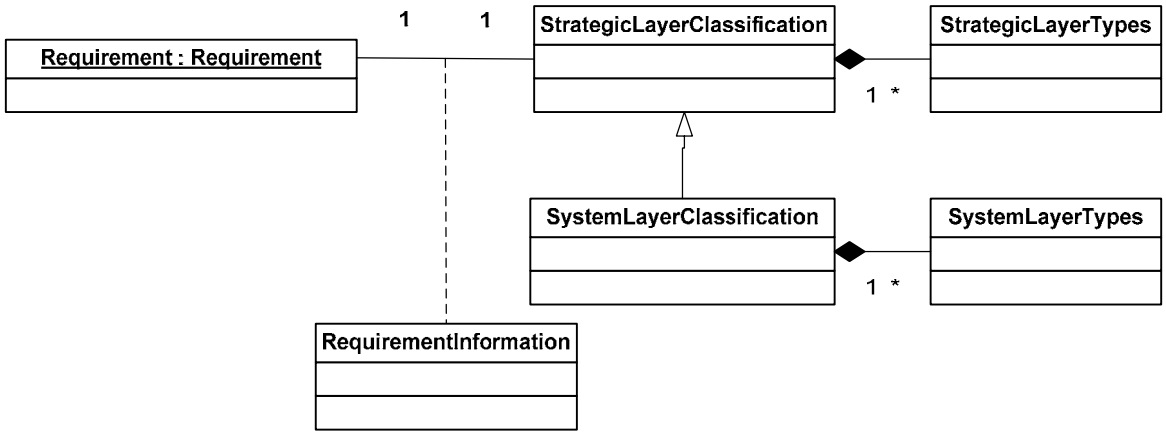


Figure 20. XSD Structure of Faceted Classification

1. Strategic Layer XML Schema

The Strategic layer XSD incorporates the facets developed in FIRE*c* and is structured according to the UML model developed in FIRE*m*. The design of the Strategic layer XML Schema, with supporting requirements metadata, is shown in Figure 21. The requirement defined at the Strategic layer, named "StrategicRequirementType", is a

complex element, composed of a sequence of child elements that represent the facets of the classification (Figure 22). Each child element must occur in the order they are defined and at least one time in the conforming XML document, as denoted by the occurrence indicators “sequence” and “minOccurs”. Each element refers to the name of the corresponding complexType or simpleType element that defines it. For example, the element below is named “Goal” of type “GoalType”, which is a primary facet of the Strategic layer of the classification (refer to Figure 16).

```
<xs:element minOccurs="1" name="Goal" type="req:GoalType"></xs:element>
```

The simpleType element “GoalType” must occur at least once in the conforming XML document. This ensures the resulting requirement is linked to a high-level objective, or goal, for the system. The default value for the occurrence “minOccurs” is 1, so we chose not to include it in the schema.

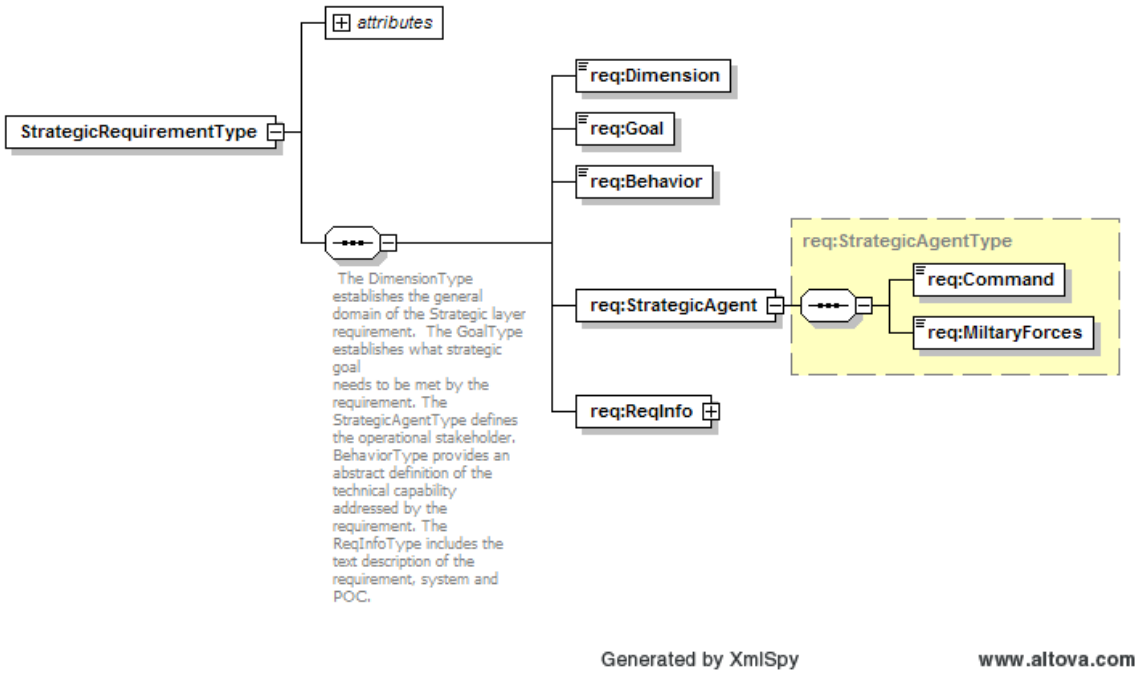


Figure 21. Strategic Layer XSD Design

The namespace declaration `xmlns:req="http://www.nps.edu/requirements` allows us to define data elements unique to our domain of interest by using the prefix “req”.

```
       <xs:element name="StrategicRequirementType">
            <xs:complexType>
                  <xs:sequence>
                        <xs:annotation>
                              <xs:documentation> The DimensionType
establishes the general domain of the Strategic layer requirement.
The GoalType establishes what strategic goal needs to be met by the
requirement. The StrategicAgentType defines the operational
stakeholder. The BehaviorType provides an abstract definition of the
technical capability addressed by the requirement. The ReqInfoType
includes the text description of the requirement, system and POC.
</xs:documentation>
                        </xs:annotation>
                        <xs:element name="Dimension"
type="req:DimensionType"/>
                        <xs:element name="Goal" type="req:GoalType"/>
                        <xs:element name="Behavior"
type="req:BehaviorType"/>
                        <xs:element name="StrategicAgent"
type="req:StrategicAgentType"/>
                        <xs:element name="ReqInfo"
type="req:ReqInfoType"/>
                  </xs:sequence>
                  <xs:attribute name="strategicRefID"
type="req:strategicRefIDType" use="required"/>
            </xs:complexType>
      </xs:element>
```

Figure 22. Example of XSD Strategic Requirement Element from Strategic.xsd

As shown in Figure 23, the requirement defined by the XML Schema Strategic.xsd is identified by a four-digit identification number, `<xs:simpleType name="strategicRefIDType">`, and restricted to the long number format, as defined by `<xs:restriction base="xs:long">`. The reference identification number is required for the resulting XML document to conform to the Strategic layer schema, and it must match the specified pattern. In this example, the "strategicRefID" of a conforming XML document must have four-digits, each digit a number from 0 to 9.

```
<xs:simpleType name="strategicRefIDType">
     <xs:restriction base="xs:long">
         <xs:pattern value="[0-9]{4}"></xs:pattern>
     </xs:restriction>
       </xs:simpleType>

<xs:attribute name="strategicRefID" type="req:strategicRefIDType"
use="required"></xs:attribute>
```

Figure 23. Strategic Layer Reference Identification Number

The simple element types “GoalType”, “DimensionType”, “StrategicAgentType” and “BehaviorType” are defined in the XML document StrategicLayerTypes.xsd. The “ReqInfoType” complex element represents the requirements metadata and is defined in the XML document ReqInfo.xsd. Both documents appear in Appendix C. The StrategicLayerTypes.xsd and ReqInfo.xsd are included in the Strategic layer schema Strategic.xsd by using <xs:include</xs:include>, as shown in Figure 24.

```
<xs:include schemaLocation="StrategicLayerTypes.xsd"></xs:include>
<xs:include schemaLocation="ReqInfo.xsd"></xs:include>
```

Figure 24. Including Multiple Schemas in the Strategic Layer Schema

The simpleType element “GoalType”, as shown in Figure 25, is defined by a string of characters, whitespace, tabs and carriage returns, as indicated by the string datatype shown in <xs:restriction base="xs:string">. In addition, the content of the element is restricted to the value indicated by the enumeration constraint. In the example shown in Figure 25, the conforming XML document must specifically include “Interoperability”, “SituationalAwareness, “InformationAssurance”, etc. as an entry of the simpleType “GoalType”.

```
<xs:simpleType name="GoalType">
        <xs:annotation>
            <xs:documentation>
    The GoalType links the requirement to the highest level strategic
concern; restricted to
enumerated values</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:enumeration value="Interoperability"></xs:enumeration>
            <xs:enumeration
value="SituationalAwareness"></xs:enumeration>
            <xs:enumeration
value="InformationAssurance"></xs:enumeration>
            <xs:enumeration value="CommandControl"></xs:enumeration>
            <xs:enumeration value="Communication"></xs:enumeration>
            <xs:enumeration value="IMKM"></xs:enumeration>
        </xs:restriction>
    </xs:simpleType>
```

Figure 25. Example from StrategicLayerTypes.xsd

The design of ReqInfo.xsd follows the UML model that is shown in Figure 14. The diagram in Figure 26 represents the XML structure of the ReqInfo UML model, as it is included in the Strategic layer schema. As previously discussed, the number of elements required to capture the data can be changed depending on the desired level of abstraction in the requirements definition.

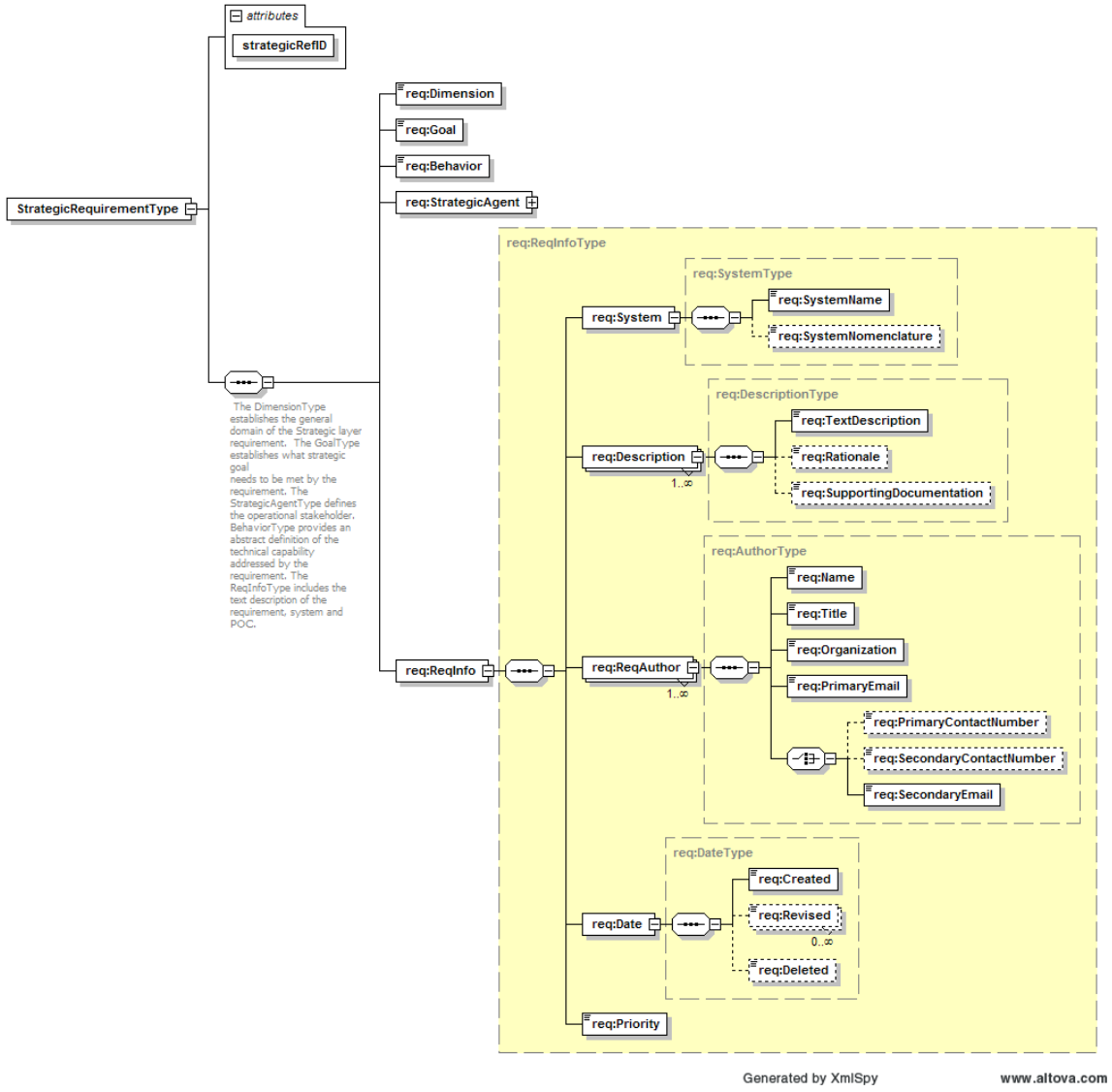


Figure 26. XML Diagram of Strategic.xsd illustrating the inclusion of ReqInfo.xsd.

As shown in Figure 27, the complex type “ReqInfo” is defined by the order indicator `<xs:sequence>` as predefined sequence of child elements. Both the

“DescriptionType” and “ReqAuthorType” elements can have unlimited occurrences as defined by the occurrence indicator `maxOccurs="unbounded"`. This ensures the conforming XML document allows for an unlimited number of changes to a requirement or the addition of amplifying information by multiple authors. The changes appear as additional elements in the conforming XML document, supporting the evolution of requirements. Again, because of the modularity of the ReqInfo.xsd, the requirements information may also be included at any layer or all layers of the classification.

```
<xs:complexType name="ReqInfoType">
        <xs:annotation>
            <xs:documentation>Sequence of elements defines the
requirement metadata.  Description and ReqAuthor can occur an unlimited
number of times to support revisions to a requirement. System, Date
and Priority can only occur once.</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element minOccurs="1" name="System"
type="req:SystemType"></xs:element>
            <xs:element maxOccurs="unbounded" minOccurs="1"
name="Description" type="req:DescriptionType"></xs:element>
            <xs:element maxOccurs="unbounded" minOccurs="1"
name="ReqAuthor" type="req:AuthorType"></xs:element>
            <xs:element minOccurs="1" name="Date"
type="req:DateType"></xs:element>
            <xs:element minOccurs="1" name="Priority"
type="req:PriorityType"></xs:element>
        </xs:sequence>
</xs:complexType>
```

Figure 27. Example from ReqInfo.xsd illustrating requirements information elements.

The “DateType” is a datatype that specifies the date the requirement has been created, modified or deleted (Figure 28). While there is no limit to the number of occurrences of the “Revised” element, and no requirement for an occurrence of the “Deleted” element, the “Created” element must occur at least once. This ensures a modification to, or deletion of, an existing requirement includes the respective date. The format for the date datatype is included in the XML Schema definition of the form YYYY-MM-DD.

```
 <xs:complexType name="DateType">
        <xs:annotation>
            <xs:documentation>Sequence of elements gives the date each
requirement is created,revised and deleted.  Created and Deleted can
only occur once, but Revised can occur an unlimited number of times to
capture changes to requirements.</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element minOccurs="1" name="Created"
type="xs:date"></xs:element>
            <xs:element maxOccurs="unbounded" minOccurs="0"
name="Revised" type="xs:date"></xs:element>
            <xs:element minOccurs="0" name="Deleted"
type="xs:date"></xs:element>
        </xs:sequence>
   </xs:complexType>
```

Figure 28. Example from ReqInfo.xsd illustrating “DateType” element.

In the ReqInfo.xsd, a primary email address of the author is required, but there are also options for either a primary or secondary phone number or a secondary email address, as shown in Figure 29. The email address is defined by the simpleType element “EmailType” and the phone number is defined by the simpleType element “PhoneType”, and both are restricted to a pattern value as shown in Figure 30. In the “EmailType” and “PhoneType” simpleType elements, the `<xs:restriction base="xs:token">` is used to accommodate the a phone number and email address, removing the white space characters, carriage returns and tabs. The pattern value can be tailored for any desired format in either datatype.

```
<xs:complexType name="AuthorType">
        <xs:annotation>
            <xs:documentation>Sequence of elements specifies POC
information.</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element minOccurs="1" name="Name"
type="xs:string"></xs:element>
            <xs:element minOccurs="1" name="Title"
type="xs:string"></xs:element>
            <xs:element minOccurs="1" name="Organization"
type="xs:string"></xs:element>
            <xs:element minOccurs="1" name="PrimaryEmail"
type="req:EmailType"></xs:element>
            <xs:choice>
                <xs:element minOccurs="0" name="PrimaryContactNumber"
type="req:PhoneType"></xs:element>
                <xs:element minOccurs="0" name="SecondaryContactNumber"
type="req:PhoneType"></xs:element>
                <xs:element minOccurs="1" name="SecondaryEmail"
type="req:EmailType"></xs:element>
            </xs:choice>
        </xs:sequence>
    </xs:complexType>
```

Figure 29. Example from ReqInfo.xsd illustrating "AuthorType" element

```
<xs:simpleType name="EmailType">
  <xs:restriction base="xs:token">
      <xs:pattern value="([\.a-zA-Z0-9_])+@([a-zA-Z0-9_])+(([a-zA-Z0-
9_])*\.([a-zA-Z0-9_])+)+"></xs:pattern>
  </xs:restriction>

</xs:simpleType>

<xs:simpleType name="PhoneType">
  <xs:restriction base="xs:token">
      <xs:pattern value="(\-[0-9]{3})+(\-[0-9]{3})+(\-[0-
9]{4})"></xs:pattern>
  </xs:restriction>

</xs:simpleType>
```

Figure 30. Example from ReqInfo.xsd illustrating "EmailType" and "PhoneType" elements.

2. System Layer XML Schema

We used a similar approach used to develop the schemas at both the Strategic and System layers. Again, we relied on the extensibility and modularity of XML to represent requirements data, merging and building upon the data structured at the Strategic layer, as shown in the schema diagram in Figure 31.

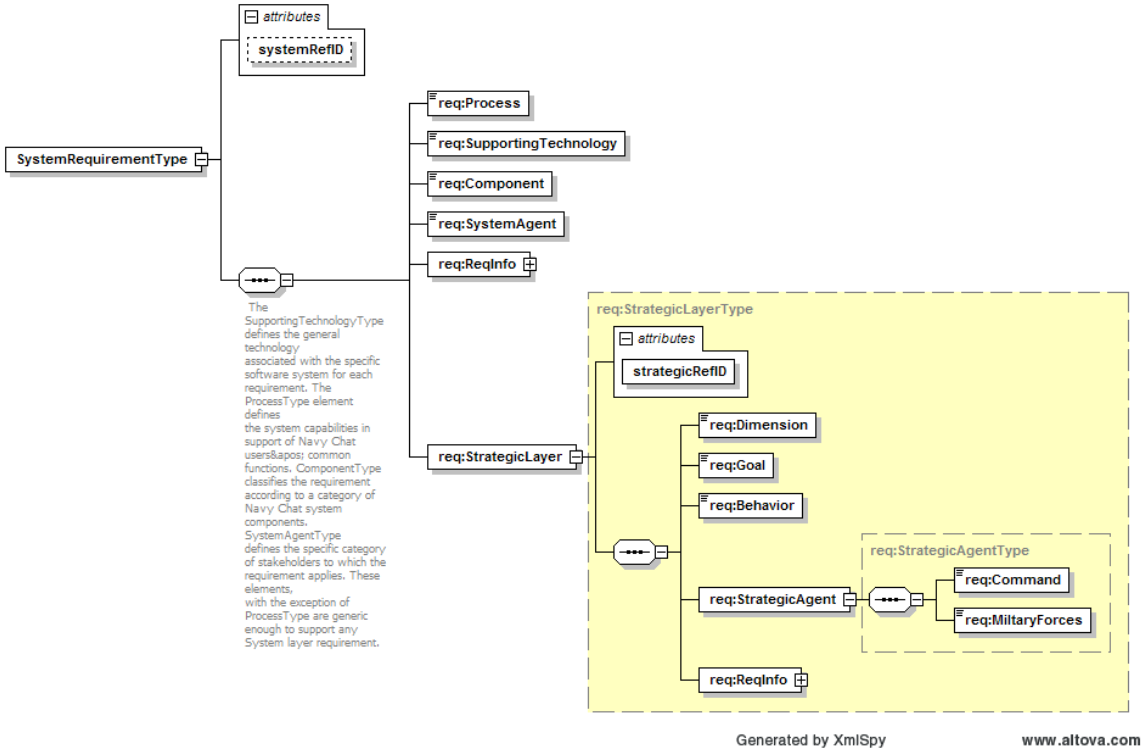


Figure 31. Schema diagram of System.xsd

The requirement defined at the System layer is represented by the complexType element “SystemRequirementType”, which is composed of a sequence of child elements including the elements of type “StrategicLayerType” and “ReqInfoType” that implement the elements and datatypes defined by Strategic.xsd and ReqInfo.xsd, as illustrated in Figure 32. A attribute is assigned at each layer, identified as “strategicRefId” and systemRefId, that serves as a unique reference identification number for each instance of Strategic or System layer requirement. As previously discussed, the ReqInfo.xsd can be implemented at either the Strategic layer, the System layer or both with the `<xs:include</xs:include>`.

```
   <xs:element name="StrategicRequirementType">
            <xs:complexType>
                  <xs:sequence>
                        <xs:annotation>
                              <xs:documentation> The DimensionType
establishes the general domain of the Strategic layer requirement.
The GoalType establishes what strategic goal needs to be met by the
requirement. The StrategicAgentType defines the operational
stakeholder. The BehaviorType provides an abstract definition of the
technical capability addressed by the requirement. The ReqInfoType
includes the text description of the requirement, system and POC.
</xs:documentation>
                        </xs:annotation>
                        <xs:element name="Dimension"
type="req:DimensionType"/>
                        <xs:element name="Goal" type="req:GoalType"/>
                        <xs:element name="Behavior"
type="req:BehaviorType"/>
                        <xs:element name="StrategicAgent"
type="req:StrategicAgentType"/>
                        <xs:element name="ReqInfo"
type="req:ReqInfoType"/>
                  </xs:sequence>
                  <xs:attribute name="strategicRefID"
type="req:strategicRefIDType" use="required"/>
            </xs:complexType>
      </xs:element>
```

Figure 32. XML Elements of the "SystemRequirementType" from System.xsd

H. FIRE*x*: PROVIDING DATA EXCHANGE CAPABILITY

Being able to access and utilize requirements information without relying on proprietary software is one of the key factors in managing evolving requirements. XML is a platform neutral and programming language independent technology, which offers a flexible way to represent and manage requirements data. Because the XML document conforms to a schema that defines its structure and content, it contains standardized data suitable for use by applications designed to process the data. Notably, one of the design goals for XML is, "It shall be easy to write programs which process XML documents" [W3C].

The XML documents that conform to defining schemas contain requirements data that can be interchanged across the Internet. The data within these documents can be retrieved using the XML Document Object Model (DOM) in the case of small

documents.[3] Or, the data in large XML documents can be parsed using the Simple API for XML (SAX) that is an open source project hosted by SourceForge [SAX]. Large amounts of XML data can be stored using native XML databases or XML-enabled enterprise database systems, providing the ability to add, modify, or search XML documents. The XML Query (XQuery) language can be used to extract and process the data contained in XML documents. Most commonly, the XML Path language (XPath) and Extensible Stylesheet Language Transformations (XSLT) are used together in a stylesheet to process specific parts of XML documents and convert the data to another form of output. These technologies, which belong to the XSL family of recommendations [W3Cb], operate on the hierarchical elements of the source XML document to transform the data into a result tree with a different structure. This transformation may be necessary simply so the data can be presented in such a way that is more easily read than its associated XML document. In addition, it may be used to restructure an existing XML document by selecting, removing, or renaming the source elements for more efficient, effective data exchange.

In this section, we present the sample XML documents representing the requirements data generated from the first three activities of our framework. In addition, we present a simple stylesheet to demonstrate how XSL and XML can be used to transform this data without relying on expensive, proprietary software applications.

1. XML Documents Representing Evolving Requirements

Classification at the Strategic layer results in a requirement in what is probably its most abstract form. However, even at this layer, the facets of the classification can provide information that is useful in reducing inconsistencies in requirements, as demonstrated in the preceding section. In Figure 33, we provide an excerpt from an XML document that conforms to the XML Schemas Strategic.xsd and ReqInfo.xsd and contains preliminary information for a Navy Chat requirement at the Strategic level of the classification. The XML requirement document, in its entirety, is shown in Appendix D.

[3] "The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents." W3C Architecture Domain, Retrieved 19APR06 from: http://www.w3.org/DOM/#what

```
<Dimension>Cognitive</Dimension>
    <Goal>SituationalAwareness</Goal>
    <Behavior>ShareInformation</Behavior>
    <StrategicAgent>
        <Command>JFCOM</Command>
        <MiltaryForces>Navy</MiltaryForces>
    </StrategicAgent>
    <ReqInfo>
        <System>
            <SystemName>Navy Chat</SystemName>
            <SystemNomenclature>not available</SystemNomenclature>
        </System>
        <Description>
            <TextDescription>Chat technologies on afloat units must
provide warfighters with visual cues/indicators to aid in maintaining
situational awareness </TextDescription>
```

Figure 33. Sample from conforming XML Document generated from Strategic.xsd

The sample requirement for Navy Chat, shown in Figure 33, is structured according to the classification scheme and the associated XML schema. The elements <Goal>, <Behavior>, <StrategicAgent>, and <ReqInfo> are the XML representations of the corresponding facets of the classification. We have included sample entries in each of the elements to illustrate the use of XML to standardize the format of data.

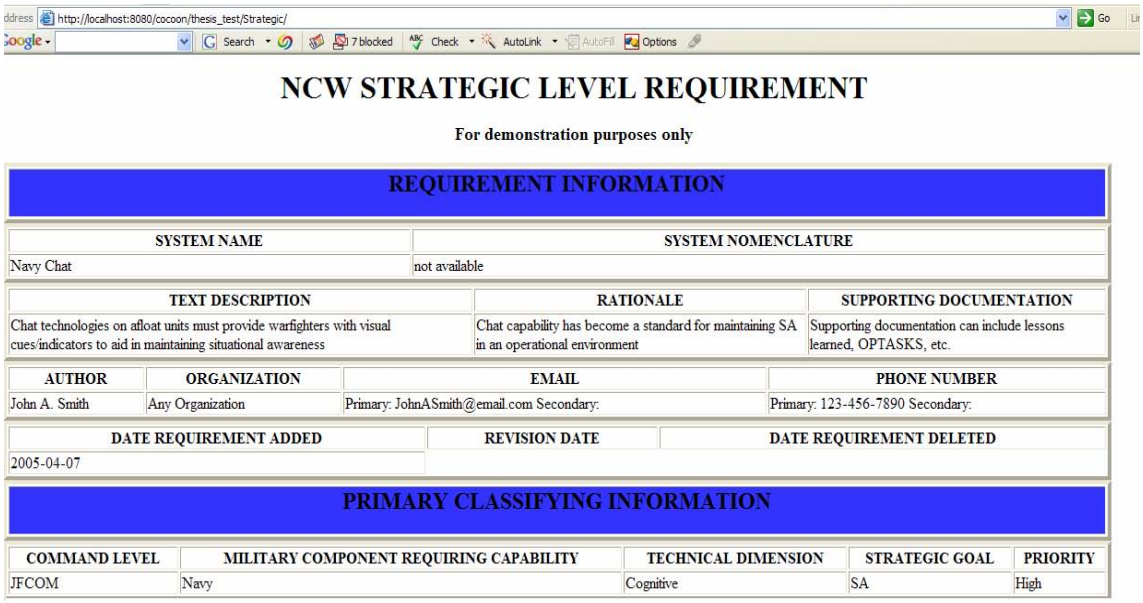
The conforming XML document based on the System layer schema contains data that defines the requirement at both the Strategic and the System layer. At each layer of the classification, the requirement is defined according to the layer-specific facets, defined by StrategicLayerTypes.xsd and SystemLayerTypes.xsd, as well as by the facets that capture the requirements metadata, as defined by ReqInfo.xsd. The XML requirement document, as it is defined at the System layer, can also be found in Appendix D. Again, we have populated the elements of the System layer XML requirement document with example data based on [CAT05] appearing in Appendix B to demonstrate our approach.

There are many benefits of using XML to represent requirements information. First, all data particular to one requirement is captured in one XML document. In the XML requirement document, each layer of the classification is associated with a reference identification number that is unique to that layer, aiding in traceability. This

document also illustrates the capability of the classification scheme and associated schemas to capture revisions to requirements complete with author, point of contact information and date of revision. In addition, a priority is assigned at each layer; inconsistent priority assignments between layers may indicate there is a mismatch that exists between the stakeholders' goals (StrategicAgent) and the warfighers' needs (SystemAgent). Secondly, XML documents are searchable. Data across a collection of XML requirement documents can be retrieved by searching by element, attribute or text content because each XML requirement document has exactly the same structure as defined by the schemas. This means, for example, a stakeholder could retrieve all requirements that are classified by the child element "Interoperability" defined by the Strategic layer complexType element "GoalType".

2. Using XSLT to Transform XML Requirements Documents

A crucial part of being able exchange standardized data is having the capability to view it and extract that which is required. As previously discussed, XPath and XSLT are used to 1) transform an XML document into an easy-to-read form such as HTML and 2) manipulate the elements of the source XML document. In both of these cases, the XSLT stylesheet is applied to an XML document to produce the desired output. Using XMLobjective and the open source web development framework Apache Cocoon [APA], we developed a stylesheet to transform and view the Strategic layer XML document that appears in Appendix D. The web content is shown below in Figure 34 and the complete stylesheet appears in Appendix E. Again, this is a very simple example of how XML, XSL and XPath can be used to structure and transform requirements data.



NCW STRATEGIC LEVEL REQUIREMENT

For demonstration purposes only

REQUIREMENT INFORMATION

SYSTEM NAME	SYSTEM NOMENCLATURE
Navy Chat	not available

TEXT DESCRIPTION	RATIONALE	SUPPORTING DOCUMENTATION
Chat technologies on afloat units must provide warfighters with visual cues/indicators to aid in maintaining situational awareness	Chat capability has become a standard for maintaining SA in an operational environment	Supporting documentation can include lessons learned, OPTASKS, etc.

AUTHOR	ORGANIZATION	EMAIL	PHONE NUMBER
John A. Smith	Any Organization	Primary: JohnASmith@email.com Secondary:	Primary: 123-456-7890 Secondary:

DATE REQUIREMENT ADDED	REVISION DATE	DATE REQUIREMENT DELETED
2005-04-07		

PRIMARY CLASSIFYING INFORMATION

COMMAND LEVEL	MILITARY COMPONENT REQUIRING CAPABILITY	TECHNICAL DIMENSION	STRATEGIC GOAL	PRIORITY
JFCOM	Navy	Cognitive	SA	High

Figure 34. Screenshot illustrating transformation of the XML requirement document StrategicRequirement.xml using the XSL stylesheet Strategy.xsl

I. SUMMARY

In this chapter, we presented our case study to demonstrate how the Framework for Requirements Evolution (FIRE) approach can be used to classify and manage evolving requirements. We demonstrated a method for classifying user requirements that is based on a stakeholder defined faceted classification scheme. In addition, we illustrated how UML, XML and XSL can be used to model, standardize and transform requirements data ultimately providing structured requirements and universal data exchange capability.

VII. CONCLUSION AND RECOMMENDATIONS FOR FUTURE WORK

A. CONCLUSION

All software systems are designed and built to meet a set of requirements as set forth by the stakeholders. These requirements define the qualities and functions that must be addressed by a technical solution. Requirements, however, have a propensity to change, both during and after the software development process. This is particularly true in a fast-paced operational environment because users will put a system to the test and in doing so discover its capabilities and limitations. Additionally, as the DOD undergoes the transformation to NCW, new technical solutions supporting network-centric operations may change how warfighters perform their jobs, potentially resulting in new requirements. Ultimately, unless there are mechanisms in place to capture and integrate evolving requirements, software systems will not meet all of the users' needs.

There are no boundaries associated with evolving requirements of software systems supporting NCW/NCO. User requirements originate from every group of stakeholders, from the warfighters who use and maintain the systems to the organizations responsible for acquiring them. In addition, frequently occurring warfighter demonstrations may result in the collection and documentation of user requirements for new technologies. Unfortunately, there is no formal process for collecting and integrating evolving requirements from all of these sources. The result is a set of requirements that is neither consistent in structure nor content and cannot be easily shared among all stakeholders due to varying data formats. In addition, the warfighters' needs may be expressed in terms of what operational goals need to be supported, resulting in requirements that are defined a high level of abstraction. Managing evolving requirements that span countless organizations and many groups of stakeholders requires an approach that, as a starting point, defines requirements according to operational goals and that utilizes technology that is neither platform dependent nor proprietary. As such, we recommend a framework incorporating a non-traditional method of classifying requirements, which is modeled in UML to provide visualization and implemented with

XML to provide structure and content. We believe this approach is necessary to develop requirements that are standardized and easily exchanged across organizations without relying on expensive software applications.

We recommend a faceted classification scheme to provide a way of defining and classifying requirements without relying on subjectively interpreted and difficult to manage natural language requirements. Predefined attributes serve to guide the user during submission of requirements information. The faceted classification scheme is tailorable to be domain specific, a characteristic not found in current tools. It is also easy to change; updating or changing the classification simply means adding, deleting, or modifying facets as required. We believe UML is a natural choice for modeling this classification because it is widely used, easily understood and can represent a scheme that is mostly non-hierarchical.

We base our approach on the premise the web is available to manage requirements discovered, changed or deleted based on warfighters' experience with the technology. Access to the web is assumed for any operational unit that is part of the NCW transformation, as well as for any organization within DOD that needs access to requirements information for research and development of software systems supporting NCW. Clearly, using the XML standard for data exchange over the web supports our objectives of providing 1) structured requirements and 2) non-platform specific data exchange.

It is important to note the intent of this approach is not to add a complicated procedure for collecting and managing requirements. Rather, it is to ensure evolving requirements accurately reflect the warfighters' true needs, are as consistent and complete as possible, and can be integrated with the formal software development process.

B. FUTURE WORK

1. Developing a Requirements Domain Model

The facets of our classification scheme and associated XML elements were the result of a very subjective interpretation of the information we collected for the domain

analysis. In practical application, formal development of an ontology or taxonomy would be necessary to establish the requirements domain. An ontology would define the semantic structure of the domain, which could them be used to further develop and refine the structure and content of the XML documents. Open source tools such as Protégé OWL [PRO] are available to develop the concepts and relationships of the knowledge base for the software requirements domain.

2. Web-based Application Supporting Standard Input Format

The most effective way to capture evolving requirements is to provide stakeholders (in particular users/warfighters) with a highly accessible, user-friendly way to contribute their ideas and results of their experiences with the technology. Although requirements engineers often use surveys, questionnaires and interviews to elicit requirements during the software development process, these tools are not practical when the group of stakeholders is large, geographically dispersed and possibly involved in military operations. As such, a web-based application is probably the most suitable tool to collect requirements directly from the warfighter. This type of tool has the potential to reach all stakeholders, even those with limited communication capabilities or who cannot participate in traditional elicitation processes due to operational commitments. It would also serve as a data collection medium for requirements previously documented in text files or spreadsheets, as well as those generated from informal sources such as trouble calls or CASREPs. A fully-functional web-based application utilizing a standard input format would be the focus of future work to implement our recommended approach.

A standard input format for XML content can be achieved using the XML application XForms. Recently added as a W3C recommendation, XForms offers several advantages over commonly used HTML forms, such as device independence and reuse of existing schemas to define data elements and maintain validation constraints [W3Cc]. More information can be found at [W3c] regarding application and implementation. One of the major hurdles to implementing XForms has been lack of native browser support. However, support is growing; the browser plug-in formsPlayer and open source JavaScript FormFaces are both available to support the XForms standard.[4]

[4] formPlayer available at http://www.formsplayer.com/content/index.html and FormFaces is available at http://sourceforge.net/projects/formfaces/

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A: SOURCES USED FOR DOMAIN ANALYSIS

Bentrup, John A. and Sunoy N. Banerjee. Memo: Fleet Chat Requirements. Center for Naval Analysis. June 2003.

Galdorisi, George, Jeff Clarkson, Jeff Grossman, and Mike Reilley. Composable FORCEnet Command and Control: The Key to Energizing the Global Information Grid to Enable Superior Decision Making. The Ninth International Command & Control Research and Technology Symposium. September 2004. Retrieved 05May06 from: http://www.dodccrp.org/events/2004/ICCRTS_Denmark/CD/track01.htm

Heacox, Nancy J., Ronald A. Moore, Jeffrey G. Morrison, and Rey F. Vturralde. Real-time Online Communication: 'Chat' Use in Navy Operations.

FORCEnet Capabilities: 15 Capabilities Necessary to Implement the FORCEnet Concept. FORCEnet website. Retrieved 06May06 from: http://forcenet.navy.mil/

Jara, Timothy and Matt Lisowski. Don't Silence Navy Chat. Proceedings, September 2003.

Network Centric Warfare: Background and Oversight Issues for Congress. CRS Report for Congress, June 2, 2004. Retrieved 06May06 from: http://www.fas.org/man/crs/RL32411.pdf

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B: NAVY CHAT USER REQUIREMENTS [5]

Requirement	Goal	Dimension	Supporting Tech
Maintain an accurate list of chat participants to reduce ping rate (or, reduce ping rate to maintain accurate list)	SA	IT	Networking
Maintain chat history during network outages	SA	IT	Networking
Maintain screen contents on reconnect	SA	IT	Networking
Automatic system reconnect	SA	IT	Networking
Distributed server architecture	SA	IT	Networking
Receive transcript of chat messages upon entering chat room.	SA	IT	Data Management
Monitor chat sessions without participating	SA	IT	Communications
Ability to rejoin chatrooms immediately and automatically	SA	IT	Communications
Locate specific person	SA	IT	Communications
Locate chat rooms	SA	IT	Communications
Ability to rejoin chatrooms several times an hour	SA	IT	Communications
Ability to send message to one user	SA	IT	Communications
Ability to use the client when the server is offline	SA	IT	Communications
Chat room access should be controlled by individuals or groups	SA	IT	Communications
Supports hidden rooms	SA	IT	Communications
Supports multiple chatroom types	SA	IT	Communications
Ability to launch private chat via a room	SA	IT	Communications
Allows temporary chat room	SA	IT	Communications
Supports use of multiple user ID's	SA	IT	Communications
Participate in multiple rooms simultaneously	SA	IT	Communications
Supports functional account/user names	SA	IT	Communications
Allows users to join or leave a chatroom	SA	IT	Communications
Indication user is reading a private message	SA	Cognitive	User-Computer Interface

[5] Source: [CAT05]

Requirement	Goal	Dimension	Supporting Tech
Provides indication of members joining and leaving	SA	Cognitive	User-Computer Interface
System alerts should be hidden on demand	SA	Cognitive	User-Computer Interface
Provides alert for new messages	SA	Cognitive	User-Computer Interface
Provides audio alert to keywords	SA	Cognitive	User-Computer Interface
Alert modality is configurable by the user	SA	Cognitive	User-Computer Interface
Supports tiled windows	SA	Cognitive	User-Computer Interface
Legibly display information on message	SA	Cognitive	User-Computer Interface
Ability to turn off join/depart messages related to other users	SA	Cognitive	User-Computer Interface
Ability to visually monitor at least 10 rooms at once	SA	Cognitive	User-Computer Interface
Supports visual alerts	SA	Cognitive	User-Computer Interface
Provide indication that someone wants to chat	SA	Cognitive	User-Computer Interface
Supports audio alerts	SA	Cognitive	User-Computer Interface
Ability to monitor several chat rooms at once	SA	Cognitive	User-Computer Interface
User nicknames should be flexible	SA	Organizational	N/A
Provides a standardized naming convention	SA	Organizational	N/A
Functional account can remain online during watch turnover	SA	Organizational	N/A
Automatic download of logs	IM/KM	IT	Networking
Logs should be controlled centrally at the server	IM/KM	IT	Networking
Messages should be sent to server	IM/KM	IT	Networking
Logs chatroom conversations	IM/KM	IT	Data Management
Provides logging capabilities	IM/KM	IT	Data Management
Ability to search for specific information since last logon	IM/KM	IT	Data Management
Ability to control how much of the historic log is downloaded when user logs on	IM/KM	IT	Data Management
Logs should be searchable within a certain time segment	IM/KM	IT	Data Management
Logs must be readily available to users for review of past events	IM/KM	IT	Data Management
Logs the entry/exit of members in the room	IM/KM	IT	Data Management
Logs non-permanent chatrooms	IM/KM	IT	Data Management
Timestamp Messages	IM/KM	Data	Data Management

Requirement	Goal	Dimension	Supporting Tech
Timestamp messages should include date and time	IM/KM	Data	Data Management
Supports file transfer	IM/KM	Data	Data Management
Chatroom layout is configurable	IM/KM	Cognitive	User-Computer Interface
Logging of private messages should be configurable	IM/KM	Cognitive	User-Computer Interface
Provides server interoperability	Interoperability	IT	Networking
Provides support for WAP or mobile device	Interoperability	IT	Networking
Provides support for handheld devices	Interoperability	IT	Networking
UNIX and Windows compatible	Interoperability	IT	Communications
Single version of chat software	Interoperability	IT	Communications
Standardized chat server policies	Interoperability	Organizational	N/A
Supports tactical platforms/units with capability of 10Kbps per enclave	Comms	Physical	Networking
Limits number of users in chat sessions to 2000	Comms	IT	Networking
Minimizes the initial bandwidth cost to connect to server	Comms	IT	Networking
Chat server is scalable to support approx 700 rooms simultaneously	Comms	IT	Networking
Bandwidth efficient	Comms	IT	Networking
Shipboard servers maintain onboard chat during outages	Comms	IT	Networking
User restricted to 40 concurrent chat sessions	Comms	IT	User-Computer Interface
Messages are in real-time	Comms	Data	Networking
Provides server-to-server compression of data	Comms	Data	Networking
System supports low data rates	Comms	Data	Networking
Messages are text-based	Comms	Data	Data Management
User authentication can be set to optional	IA	IT	Security
Supports temporary password protections on chatrooms	IA	IT	Security
Ensures security by avoiding computer-to-computer file transfers	IA	IT	Security
Provides user authentication	IA	IT	Security
User authentication is lightweight	IA	IT	Security
Broadcast chat	C & C	IT	Communications
Provides alert to users of TAO action	C & C	Cognitive	User-Computer Interface
Track new information	C & C	Cognitive	User-Computer Interface

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C: CASE STUDY XML SCHEMAS

STRATEGIC.XSD

```
<!-- edited with XMLSpy v2006 sp2 U (http://www.altova.com)
and XMLObjective 1.2 mdxsys (http://www.xmlobjective.com) by
Linda Reynolds (Naval Postgraduate School) April 200 -->


<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:req="http://www.nps.edu/requirements"
targetNamespace="http://www.nps.edu/requirements"
elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:include schemaLocation="StrategicLayerTypes.xsd"/>
      <xs:include schemaLocation="ReqInfo.xsd"/>
      <xs:simpleType name="strategicRefIDType">
            <xs:annotation>
                  <xs:documentation>This is a unique reference
identification number for a
requirement defined at the Strategic layer of the
classification</xs:documentation>
            </xs:annotation>
            <xs:restriction base="xs:string">
                  <xs:pattern value="[0-9]{4}"/>
            </xs:restriction>
      </xs:simpleType>
      <!--The StrategicRequirementType is referenced when the
requirement is only defined
at the Strategic layer.-->
      <xs:element name="StrategicRequirementType">
            <xs:complexType>
                  <xs:sequence>
                        <xs:annotation>
                              <xs:documentation> The DimensionType
establishes the general domain of the Strategic layer requirement.  The
GoalType establishes what strategic goal needs to be met by the
requirement. The StrategicAgentType defines the operational
stakeholder.BehaviorType provides an abstract definition of the
technical capability addressed by the requirement. The ReqInfoType
includes the text description of the requirement, system and POC.
</xs:documentation>
                        </xs:annotation>
                        <xs:element name="Dimension"
type="req:DimensionType"/>
                        <xs:element name="Goal" type="req:GoalType"/>
                        <xs:element name="Behavior"
type="req:BehaviorType"/>
                        <xs:element name="StrategicAgent"
type="req:StrategicAgentType"/>
                        <xs:element name="ReqInfo"
type="req:ReqInfoType"/>
                  </xs:sequence>
                  <xs:attribute name="strategicRefID"
type="req:strategicRefIDType" use="required"/>
```

```
            </xs:complexType>
      </xs:element>

      <xs:complexType name="StrategicLayerType">
        <xs:annotation>
    <xs:documentation>
    The StrategicLayerType element is referenced when the requirement
is defined at System layer.In this example, the elements contained in
the complexTypes "StrategicRequirementType and "StrategicLayerType" are
the same.</xs:documentation>
    </xs:annotation>
            <xs:sequence>
                  <xs:element name="Dimension"
type="req:DimensionType"/>
                  <xs:element name="Goal" type="req:GoalType"/>
                  <xs:element name="Behavior" type="req:BehaviorType"/>
                  <xs:element name="StrategicAgent"
type="req:StrategicAgentType"/>
                  <xs:element name="ReqInfo" type="req:ReqInfoType"/>
            </xs:sequence>
            <xs:attribute name="strategicRefID"
type="req:strategicRefIDType" use="required"/>
      </xs:complexType>
</xs:schema>
```

STRATEGICLAYERTYPES.XSD

```
<!-- edited with XMLSpy v2006 sp2 U (http://www.altova.com)
and XMLObjective 1.2 mdxsys (http://www.xmlobjective.com) by
Linda Reynolds (Naval Postgraduate School) April 2006-->

<xs:schema
        attributeFormDefault="unqualified"
        elementFormDefault="qualified"
        targetNamespace="http://www.nps.edu/requirements"
        xmlns:req="http://www.nps.edu/requirements"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <xs:simpleType name="DimensionType">
        <xs:annotation>
            <xs:documentation>
    The DimensionType defines the general domain for FORCEnet
developmental efforts
as defined by CNO N6[CNO05]; restricted to enumerated
values</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:enumeration value="Physical"></xs:enumeration>
            <xs:enumeration value="IT"></xs:enumeration>
            <xs:enumeration value="Data"></xs:enumeration>
            <xs:enumeration value="Cognitive"></xs:enumeration>
            <xs:enumeration value="Organizational"></xs:enumeration>
            <xs:enumeration value="Operating"></xs:enumeration>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="GoalType">
        <xs:annotation>
            <xs:documentation>
    The GoalType links the requirement to the highest level strategic
concern; restricted to
enumerated values</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:enumeration value="Interoperability"></xs:enumeration>
            <xs:enumeration
value="SituationalAwareness"></xs:enumeration>
            <xs:enumeration
value="InformationAssurance"></xs:enumeration>
            <xs:enumeration value="CommandControl"></xs:enumeration>
            <xs:enumeration value="Communication"></xs:enumeration>
            <xs:enumeration value="IMKM"></xs:enumeration>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="BehaviorType">
        <xs:annotation>
            <xs:documentation>
    The BehaviorType defines the general domain of the technical
capability; restricted to
enumerated values</xs:documentation>
```

```
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:enumeration value="ShareInformation"></xs:enumeration>
            <xs:enumeration
value="DisplayInformation"></xs:enumeration>
            <xs:enumeration value="TrackTargets"></xs:enumeration>
            <xs:enumeration value="ProcessData"></xs:enumeration>
            <xs:enumeration value="SortData"></xs:enumeration>
            <xs:enumeration value="AnalyzeData"></xs:enumeration>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="StrategicAgentType">
        <xs:annotation>
            <xs:documentation>
    The StrategicAgentType links the requirement to the highest level
strategic command and Service
supported</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="Command"
type="req:CommandType"></xs:element>
            <xs:element name="MiltaryForces"
type="req:ForcesType"></xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="CommandType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="JFCOM"></xs:enumeration>
            <xs:enumeration value="CFFC"></xs:enumeration>
            <xs:enumeration value="COCOM"></xs:enumeration>
            <xs:enumeration value="JTF"></xs:enumeration>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="ForcesType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Navy"></xs:enumeration>
            <xs:enumeration value="Army"></xs:enumeration>
            <xs:enumeration value="AirForce"></xs:enumeration>
            <xs:enumeration value="Joint"></xs:enumeration>
            <xs:enumeration value="Allied"></xs:enumeration>
        </xs:restriction>
    </xs:simpleType>


</xs:schema>
```

SYSTEM.XSD

```
<!-- edited with XMLSpy v2006 sp2 U (http://www.altova.com)
and XMLObjective 1.2 mdxsys (http://www.xmlobjective.com) by
Linda Reynolds (Naval Postgraduate School) April 2006-->
<xs:schema
        attributeFormDefault="unqualified"
        elementFormDefault="qualified"
        targetNamespace="http://www.nps.edu/requirements"
        xmlns:req="http://www.nps.edu/requirements"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <xs:include schemaLocation="SystemLayerTypes.xsd"></xs:include>

    <xs:include schemaLocation="Strategic.xsd"></xs:include>

    <xs:include schemaLocation="ReqInfo.xsd"></xs:include>

    <xs:simpleType name="systemRefIDType">
        <xs:annotation>
            <xs:documentation>This is a unique reference identification
number for a
requirement defined at the System layer of the
classification</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:pattern value="[0-9]{4}"></xs:pattern>
        </xs:restriction>
    </xs:simpleType>

    <xs:element name="SystemRequirementType">
        <xs:complexType>
            <xs:sequence>
                <xs:annotation>
                    <xs:documentation> The SupportingTechnologyType
defines the general technology associated with the specific software
system for each requirement. The ProcessType element defines the system
capabilities in support of Navy Chat users&amp;apos; common functions.
ComponentType classifies the requirement according to a category of
Navy Chat system components. SystemAgentType defines the specific
category of stakeholders to which the requirement applies. These
elements, with the exception of ProcessType are generic enough to
support any System layer requirement.
</xs:documentation>
                </xs:annotation>
                <xs:element minOccurs="1" name="Process"
type="req:ProcessType"></xs:element>
                <xs:element minOccurs="1" name="SupportingTechnology"
type="req:SupportingTechnologyType"></xs:element>
                <xs:element minOccurs="1" name="Component"
type="req:ComponentType"></xs:element>
                <xs:element minOccurs="1" name="SystemAgent"
type="req:SystemAgentType"></xs:element>
                <xs:element minOccurs="1" name="ReqInfo"
type="req:ReqInfoType"></xs:element>
```

```
                <xs:element minOccurs="1" name="StrategicLayer"
type="req:StrategicLayerType"></xs:element>
            </xs:sequence>
            <xs:attribute name="systemRefID"
type="req:systemRefIDType"></xs:attribute>
        </xs:complexType>
    </xs:element>


</xs:schema>
```

SYSTEMLAYERTYPES.XSD

```
<!-- edited with XMLSpy v2006 sp2 U (http://www.altova.com)
and XMLObjective 1.2 mdxsys (http://www.xmlobjective.com) by
Linda Reynolds (Naval Postgraduate School) April 2006-->
<xs:schema
        attributeFormDefault="unqualified"
        elementFormDefault="qualified"
        targetNamespace="http://www.nps.edu/requirements"
        xmlns:req="http://www.nps.edu/requirements"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <xs:simpleType name="ProcessType">
        <xs:annotation>
            <xs:documentation>
    The ProcessType defines the technical capability supported by the
System requirement; restricted to enumerated values.  This element is
domin specific, in this example defining Navy Chat</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:enumeration value="Whisper"></xs:enumeration>
            <xs:enumeration value="Chat"></xs:enumeration>
            <xs:enumeration value="FileTransfer"></xs:enumeration>
            <xs:enumeration value="Search"></xs:enumeration>
            <xs:enumeration value="Send"></xs:enumeration>
            <xs:enumeration value="View"></xs:enumeration>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="SupportingTechnologyType">
        <xs:annotation>
            <xs:documentation>
    The SupportingTechnologyType links the System layer requirement to
a technical domain; restricted to enumerated values.
</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:enumeration value="Networking"></xs:enumeration>
            <xs:enumeration value="Security"></xs:enumeration>
            <xs:enumeration value="DataManagement"></xs:enumeration>
            <xs:enumeration
value="UserComputerInterface"></xs:enumeration>
            <xs:enumeration value="MultiMedia"></xs:enumeration>
            <xs:enumeration value="Communication"></xs:enumeration>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="ComponentType">
        <xs:annotation>
            <xs:documentation>The ComponentType classifies the
requirement according to category of software system components;
restricted to enumerated values.</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:enumeration value="OS"></xs:enumeration>
            <xs:enumeration value="HW"></xs:enumeration>
```

```
            <xs:enumeration value="SW"></xs:enumeration>
            <xs:enumeration value="Interface"></xs:enumeration>
            <xs:enumeration value="Client"></xs:enumeration>
            <xs:enumeration value="Server"></xs:enumeration>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="SystemAgentType">
        <xs:annotation>
            <xs:documentation>The SystemAgentType classifies the
requirement according to a
                        System layer category of Navy Chat
stakeholders; restricted to enumerated values.</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:enumeration value="User"></xs:enumeration>
            <xs:enumeration value="User"></xs:enumeration>
            <xs:enumeration value="Technician"></xs:enumeration>
            <xs:enumeration value="Developer"></xs:enumeration>
            <xs:enumeration value="Analyst"></xs:enumeration>
        </xs:restriction>
    </xs:simpleType>


</xs:schema>
```

REQINFO.XSD

```
<!-- edited with XMLSpy v2006 sp2 U (http://www.altova.com)
and XMLObjective 1.2 mdxsys (http://www.xmlobjective.com) by
Linda Reynolds (Naval Postgraduate School) April 2006-->

 <xs:schema
        attributeFormDefault="unqualified"
        elementFormDefault="qualified"
        targetNamespace="http://www.nps.edu/requirements"
        xmlns:req="http://www.nps.edu/requirements"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <xs:complexType name="ReqInfoType">
        <xs:annotation>
            <xs:documentation>Sequence of elements defines the
requirement metadata.  Description and ReqAuthor can occur an unlimited
number of times to support revisions to a requirement. System, Date
and Priority can only occur once.</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element minOccurs="1" name="System"
type="req:SystemType"></xs:element>
            <xs:element maxOccurs="unbounded" minOccurs="1"
name="Description" type="req:DescriptionType"></xs:element>
            <xs:element maxOccurs="unbounded" minOccurs="1"
name="ReqAuthor" type="req:AuthorType"></xs:element>
            <xs:element minOccurs="1" name="Date"
type="req:DateType"></xs:element>
            <xs:element minOccurs="1" name="Priority"
type="req:PriorityType"></xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="SystemType">
        <xs:annotation>
            <xs:documentation>Sequence of elements supplies system name
and nomenclature.</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element minOccurs="1" name="SystemName"
type="xs:string"></xs:element>
            <xs:element minOccurs="0" name="SystemNomenclature"
type="xs:string"></xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="DateType">
        <xs:annotation>
            <xs:documentation>Sequence of elements gives the date each
requirement is created, revised and deleted.  Created and Deleted can
only occur once, but Revised can occur an unlimited number of times to
capture changes to requirements.</xs:documentation>
        </xs:annotation>
        <xs:sequence>
```

```
            <xs:element minOccurs="1" name="Created"
type="xs:date"></xs:element>
            <xs:element maxOccurs="unbounded" minOccurs="0"
name="Revised" type="xs:date"></xs:element>
            <xs:element minOccurs="0" name="Deleted"
type="xs:date"></xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="AuthorType">
        <xs:annotation>
            <xs:documentation>Sequence of elements specifies POC
information.</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element minOccurs="1" name="Name"
type="xs:string"></xs:element>
            <xs:element minOccurs="1" name="Title"
type="xs:string"></xs:element>
            <xs:element minOccurs="1" name="Organization"
type="xs:string"></xs:element>
            <xs:element minOccurs="1" name="PrimaryEmail"
type="req:EmailType"></xs:element>
            <xs:choice>
                <xs:element minOccurs="0" name="PrimaryContactNumber"
type="req:PhoneType"></xs:element>
                <xs:element minOccurs="0" name="SecondaryContactNumber"
type="req:PhoneType"></xs:element>
                <xs:element minOccurs="1" name="SecondaryEmail"
type="req:EmailType"></xs:element>
            </xs:choice>
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="EmailType">
        <xs:annotation>
            <xs:documentation>Email address restricted to format
established by pattern value</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:token">
            <xs:pattern value="([\.a-zA-Z0-9_])+@([a-zA-Z0-9_])+(([a-
zA-Z0-9_])*\.([a-zA-Z0-9_])+)+"></xs:pattern>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="PhoneType">
        <xs:annotation>
            <xs:documentation>Phone number restricted to format
established by pattern value</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:token">
            <xs:pattern value="([0-9]{3})+(\-[0-9]{3})+(\-[0-
9]{4})"></xs:pattern>
        </xs:restriction>
    </xs:simpleType>
```

```
    <xs:complexType name="DescriptionType">
        <xs:annotation>
            <xs:documentation>Provides textual description,
justification and supporting documentation in string
format</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element minOccurs="1" name="TextDescription"
type="xs:string"></xs:element>
            <xs:element minOccurs="0" name="Rationale"
type="xs:string"></xs:element>
            <xs:element minOccurs="0" name="SupportingDocumentation"
type="xs:string"></xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="PriorityType">
        <xs:annotation>
            <xs:documentation>Priority of requirement restricted to
values specified by enumeration</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:enumeration value="High"></xs:enumeration>
            <xs:enumeration value="Medium"></xs:enumeration>
            <xs:enumeration value="Low"></xs:enumeration>
        </xs:restriction>
    </xs:simpleType>


</xs:schema>
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D: CASE STUDY XML EVOLVING REQUIREMENTS DOCUMENTS

STRATEGICREQUIREMENT.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Information contained within this document is for demonstration
purposes only  -->
<StrategicRequirementType
        strategicRefID="0001"
        xmlns="http://www.nps.edu/requirements"
        xmlns:requirements="http://www.nps.edu/requirements"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.nps.edu/requirements
I:\Strategic\Strategic.xsd">
    <Dimension>Cognitive</Dimension>
    <Goal>SituationalAwareness</Goal>
    <Behavior>ShareInformation</Behavior>
    <StrategicAgent>
        <Command>JFCOM</Command>
        <MiltaryForces>Navy</MiltaryForces>
    </StrategicAgent>
    <ReqInfo>
        <System>
            <SystemName>Navy Chat</SystemName>
            <SystemNomenclature>not available</SystemNomenclature>
        </System>
        <Description>
            <TextDescription>Chat technologies on afloat units must
provide warfighters with visual cues/indicators to aid in maintaining
situational awareness </TextDescription>
            <Rationale>Chat capability has become a standard for
maintaining SA in an operational environment</Rationale>
            <SupportingDocumentation>Supporting documentation can
include lessons learned,OPTASKS,CASREPS, etc.</SupportingDocumentation>
        </Description>
        <ReqAuthor>
            <Name>John A. Smith</Name>
            <Title>Program Manager</Title>
            <Organization>Any Organization</Organization>
            <PrimaryEmail>JohnASmith@email.com</PrimaryEmail>
            <PrimaryContactNumber>123-456-7890</PrimaryContactNumber>
        </ReqAuthor>
        <Date>
            <Created>2005-04-07</Created>
        </Date>
        <Priority>High</Priority>
    </ReqInfo>
</StrategicRequirementType>
```

SYSTEMREQUIREMENT.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- The information contained in this document is for demonstration
purposes only -->
<SystemRequirementType
        systemRefID="0001"
        xmlns="http://www.nps.edu/requirements"
        xmlns:requirements="http://www.nps.edu/requirements"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.nps.edu/requirements
I:\Strategic\System.xsd">
    <Process>Chat</Process>
    <SupportingTechnology>UserComputerInterface</SupportingTechnology>
    <Component>Client</Component>
    <SystemAgent>User</SystemAgent>
    <ReqInfo>
        <System>
            <SystemName>Navy Chat</SystemName>
            <SystemNomenclature>Applicable
SystemNomenclature</SystemNomenclature>
        </System>
        <Description>
            <TextDescription>Alert modality for new messages on chat
client must be configurable by user via the mouse or the
keyboard</TextDescription>
            <Rationale>Watchstanders in operational environment must be
able to set the alert status manually depending on level of
activity</Rationale>
            <SupportingDocumentation>Applicable Supporting
Documentation</SupportingDocumentation>
        </Description>
        <Description>
            <TextDescription>Alert modality for new messages on chat
client must be configurable by user via the mouse or the keyboard to
include sound and text highlighting</TextDescription>
            <Rationale>Watchstanders in operational environment must be
able to set the alert status manually depending on level of
activity</Rationale>
            <SupportingDocumentation>Applicable Supporting
Documentation</SupportingDocumentation>
        </Description>
        <ReqAuthor>
            <Name>John B Smith</Name>
            <Title>Navy Sailor</Title>
            <Organization>USS AnyShip</Organization>
            <PrimaryEmail>SailorsPrimaryEmail@email.com</PrimaryEmail>
            <PrimaryContactNumber>123-456-7890</PrimaryContactNumber>
        </ReqAuthor>
        <ReqAuthor>
            <Name>John C Smith</Name>
            <Title>Another Navy Sailor</Title>
            <Organization>USS AnotherShip</Organization>

<PrimaryEmail>AnotherSailorsPrimaryEmail@email.com</PrimaryEmail>
            <PrimaryContactNumber>555-666-7777</PrimaryContactNumber>
```

```
        </ReqAuthor>
        <Date>
            <Created>2005-05-19</Created>
            <Revised>2005-06-05</Revised>
        </Date>
        <Priority>High</Priority>
    </ReqInfo>
    <StrategicLayer strategicRefID="0001">
        <Dimension>Cognitive</Dimension>
        <Goal>SituationalAwareness</Goal>
        <Behavior>ShareInformation</Behavior>
        <StrategicAgent>
            <Command>JFCOM</Command>
            <MiltaryForces>Navy</MiltaryForces>
        </StrategicAgent>
        <ReqInfo>
            <System>
                <SystemName>CHAT</SystemName>
                <SystemNomenclature>Applicable
SystemNomenclature</SystemNomenclature>
            </System>
            <Description>
                <TextDescription>Chat technologies on afloat units must
provide warfighters with visual and audio cues/indicators to aid in
maintaining situational awareness</TextDescription>
                <Rationale>Chat capability has become a standard for
maintaining SA in operational environments</Rationale>
                <SupportingDocumentation>Supporting Documentation to
include CASREPS, OPTASKS,lessons learned,
etc.</SupportingDocumentation>
            </Description>
            <ReqAuthor>
                <Name>John A. Smith</Name>
                <Title>Program Manager</Title>
                <Organization>DOD Organization</Organization>
                <PrimaryEmail>PMPrimaryEmail@email.com</PrimaryEmail>

<SecondaryEmail>PMSecondaryEmail@email.com</SecondaryEmail>
            </ReqAuthor>
            <Date>
                <Created>2005-04-07</Created>
            </Date>
            <Priority>High</Priority>
        </ReqInfo>
    </StrategicLayer>
</SystemRequirementType>
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E: SAMPLE XSL STYLESHEET

```
<?xml version='1.0' encoding='ISO-8859-1'?>

<!-- Developed using XMLobjective 1.2 mdxsys by Linda Reynolds at Naval
Postgraduate School
April 2006 -->
<!-- For demonstration purposes only-->

<xsl:stylesheet
        version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
        <html>
            <body>
                <h1>
                    <center>NCW STRATEGIC LEVEL REQUIREMENT</center>
                </h1>
                <h3><center>For demonstration purposes
only</center></h3>
                <table border="4" width="97%">
                    <tr bgcolor="#3333FF">
                        <th>
                            <h2>REQUIREMENT INFORMATION</h2>
                        </th>
                    </tr>
                </table>
                <table border="4" width="97%">
                    <tr bgcolor="#FFFFFF">
                        <th>SYSTEM NAME</th>
                        <th>SYSTEM NOMENCLATURE</th>
                    </tr>
                    <xsl:for-each
select="StrategicRequirementType/ReqInfo/System">
                        <tr>
                            <td>
                                <xsl:value-of
select="SystemName"></xsl:value-of>
                            </td>
                            <td>
                                <xsl:value-of
select="SystemNomenclature"></xsl:value-of>
                            </td>
                        </tr>
                    </xsl:for-each>
                </table>
                <table border="4" width="97%">
                    <tr bgcolor="#FFFFFF">
                        <th>TEXT DESCRIPTION</th>
                        <th>RATIONALE</th>
                        <th>SUPPORTING DOCUMENTATION</th>
                    </tr>
```

```
                    <xsl:for-each
select="StrategicRequirementType/ReqInfo/Description">
                        <tr>
                            <td>
                                <xsl:value-of
select="TextDescription"></xsl:value-of>
                            </td>
                            <td>
                                <xsl:value-of
select="Rationale"></xsl:value-of>
                            </td>
                            <td>
                                <xsl:value-of
select="SupportingDocumentation"></xsl:value-of>
                            </td>
                        </tr>
                    </xsl:for-each>
                </table>
                <table border="4" width="97%">
                    <tr bgcolor="#FFFFFF">
                        <th>AUTHOR</th>
                        <th>ORGANIZATION</th>
                        <th>EMAIL</th>
                        <th>PHONE NUMBER</th>
                    </tr>
                    <xsl:for-each
select="StrategicRequirementType/ReqInfo/ReqAuthor">
                        <tr>
                            <td>
                                <xsl:value-of
select="Name"></xsl:value-of>
                            </td>
                            <td>
                                <xsl:value-of
select="Organization"></xsl:value-of>
                            </td>
                            <td>
                                <xsl:text> Primary: </xsl:text>
                                <xsl:value-of
select="PrimaryEmail"></xsl:value-of>
                                <xsl:text> Secondary: </xsl:text>
                                <xsl:value-of
select="SecondaryEmail"></xsl:value-of>
                            </td>
                            <td>
                                <xsl:text> Primary: </xsl:text>
                                <xsl:value-of
select="PrimaryContactNumber"></xsl:value-of>
                                <xsl:text>  Secondary:     </xsl:text>
                                <xsl:value-of
select="SecondaryContactNumber"></xsl:value-of>
                            </td>
                        </tr>
                    </xsl:for-each>
                </table>
```

```
                <table border="4" width="97%">
                    <tr bgcolor="#FFFFFF">
                        <th>DATE REQUIREMENT ADDED</th>
                        <th>REVISION DATE</th>
                        <th>DATE REQUIREMENT DELETED</th>
                    </tr>
                    <xsl:for-each
select="StrategicRequirementType/ReqInfo/Date">
                        <tr>
                            <td>
                                <xsl:value-of
select="Created"></xsl:value-of>
                            </td>
                            <td>
                                <xsl:value-of
select="Revised"></xsl:value-of>
                            </td>
                            <td>
                                <xsl:value-of
select="Deleted"></xsl:value-of>
                            </td>
                        </tr>
                    </xsl:for-each>
                </table>
                <table border="4" width="97%">
                    <tr bgcolor="#3333FF">
                        <th>
                            <h2>PRIMARY CLASSIFYING INFORMATION</h2>
                        </th>
                    </tr>
                </table>
                <table border="4" width="97%">
                    <tr bgcolor="#FFFFFF">
                        <th>COMMAND LEVEL</th>
                        <th>MILITARY COMPONENT REQUIRING
CAPABILITY</th>
                        <th>TECHNICAL DIMENSION</th>
                        <th>STRATEGIC GOAL</th>
                        <th>PRIORITY</th>
                    </tr>
                    <xsl:for-each
select="StrategicRequirementType/StrategicAgent">
                        <td>
                            <xsl:value-of
select="CommandType"></xsl:value-of>
                        </td>
                        <td>
                            <xsl:value-of
select="MilitaryForces"></xsl:value-of>
                        </td>
                    </xsl:for-each>
                    <xsl:for-each
select="StrategicRequirementType/Dimension">
                        <td>
                            <xsl:value-of select="*"></xsl:value-of>
```

```
                        </td>
                    </xsl:for-each>
                    <xsl:for-each
select="StrategicRequirementType/Goal">
                        <td>
                            <xsl:value-of select="*"></xsl:value-of>
                        </td>
                    </xsl:for-each>
                    <xsl:for-each
select="StrategicRequirementType/ReqInfo/Priority">
                        <td>
                            <xsl:value-of select="*"></xsl:value-of>
                        </td>
                    </xsl:for-each>
                </table>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

LIST OF REFERENCES

[ADK05] Adkisson, Heidi P. Use of Faceted Classification. webdesignpractices. Retrieved 19APR06 from: http://www.webdesignpractices.com/navigation/facets.html

[ALT] Altova. XMLSpy® 2006. Retrieved 19APR06 from: http://www.altova.com/

[ANT97] Anton, Annie. I., "Goal Identification and Refinement in the Specification of Software-Based Information Systems," Ph.D. Dissertation, Georgia Institute of Technology, Atlanta, GA, 1997.

[ANT98] Anton, Annie I., and Potts, Colin. The Use of Goals to Surface Requirements for Evolving Systems. In Proceedings of the IEEE International Conference on Software Engineering (ICSE), pp. 157 - 166, 1998.

[ANT00] Anton, Annie I, Dempster J., and Seige, D. Managing Use Cases During Goal-Driven Requirements Engineering: Challenges Encountered and Lessons Learned. IEEE International Conference on Software Engineering (ICSE 2000), Limerick, Ireland, 2000.

[ANT01] Anton, Annie I, Carter, Ryan A., Williams, Laurie and Dagnino, Aldo. Evolving Beyond Requirements Creep: A Risk-Based Evolutionary Prototyping Model, Fifth IEEE International Symposium on Requirements Engineering (RE'01), 2001

[ANTPOT01] Anton, Annie I. and Potts, Colin. Functional Paleontology: System Evolution as the User Sees It, icse, p. 0421, 23rd International Conference on Software Engineering (ICSE'01), 2001.

[APA] The Apache Cocoon Project. Retrieved 05MAY06 from: http://cocoon.apache.org/2.1/features.html

[BAN] Bannerman, Steve W. *A Middleweight Requirements Management Framework.* Retrieved 19APR06 from: http://reqs.tigris.org/pdfs/mrmf.pdf

[BOE91] Boehm, Barry. Software Risk Management: Practices and Principles. IEEE Software, 1991.

[BOE01] Boehm, Barry and Port, Dan. Risk-Based Strategic Software Design: How Much COTS Evaluation is Enough? Third International Workshop on Economics-Driven Software Engineering Research, Toronto, Canada, 2001.

[BRU04] Bruegge, Bernd and Dutoit, Allen H. Object-Oriented Software Engineering. Pearson Prentice Hall, 2004.

[CAN03] Cantor, Murray. Organizing RUP SE Projects. IBM developerWorks. Retrieved 19APR06 from:
http://www-128.ibm.com/developerworks/rational/library/814.html

[CAT05] Catanzaro, Jean Ph.D. and Gwynne, John Ph.D. Compiled Chat User Requirements, Pacific Science and Engineering Group, August 19, 2005

[CAT05a] Catanzaro, Jean PhD, Gwynne, John PhD and Mitchell, Craig PhD. Usabilty of Chat in the Maritime Coalition Environment: Empirical Findings from a Limited Objective Experiment for Trident Warrior 05. Pacific Science and Engineering, September 2005 (Limited distribution).

[CNO05] Chief of Naval Operations, FORCEnet Functional Concept. February 7, 2005. Retrieved 19APR06 from: http://cno-n6.hq.navy.mil/Director_Net-Centric_Warfare/OPNAV_N71/FORCEnet/index.htm

[DEN03] Denton, William. How to Make a Faceted Classification and Put It On the Web, Nov. 2003. Retrieved 19APR06 from: http://www.miskatonic.org/library/facet-web-howto.html.

[DIR05] Director, Force Transformation, Office of the Secretary of Defense. The Implementation of Network-Centric Warfare, Washington, DC, 2005.

[DONCIO] Department of the Navy, Office of the DON, Chief Information Officer. XML Naming and Design Rules, January 2005.

[FLO02] Florence, Al. CrossTalk; The Journal of Defense Software Engineering, Reducing Risks Through the Proper Specification of Software Requirements, April 2002. Retrieved 19APR06 from: http://www.stsc.hill.af.mil/crosstalk/2002/04/florence.html

[GLI05] Glinz, M. Rethinking the Notion of Non-Functional Requirements. Proceedings of the Third World Congress for Software Quality (3WCSQ 2005). Munich, Germany, Vol. II, pp. 55-64, 2005.
Retrieved 19APR06 from: http://www.ifi.unizh.ch/groups/req/staff/glinz/activities.html

[GSAM03] Software Technology Support Center. Guidelines for Successful Acquisition and Management of Software-Intensive Systems: Weapon Systems, Command and Control Systems, Management Information Systems - Condensed Version 4.0 February 2003.
Retrieved 19APR06 from: http://www.stsc.hill.af.mil/resources/tech_docs/gsam4.html

[LAM01] van Lamsweerde, Axel. Goal-Oriented Requirements Engineering: A Guided Tour. Invited mini-tutorial paper 5th IEEE International Symposium on Requirements Engineering. Toronto, Canada. August 2001.

[LEF03] Leffingwell, Dean and Widrig, Don. Managing Software Requirements; A Use Case Approach. Addison-Wesley, 2003.

[LEI02] Leishman, Theron and Cook, Dr. David A. CrossTalk; The Journal of Defense Software Engineering, Requirements Risks Can Drown Software Projects, April 2002. Retrieved 19APR06 from: http://www.stsc.hill.af.mil/crosstalk/2002/04/leishman.html

[LUC01] de Lucena, Vicente Ferreira Jr. Facet-Based Classification Scheme for Industrial Automation Software Components. Sixth International Workshop on Component-Oriented Programming at ECOOP 2001, Budapest, Hungary, 19 June 2001. Retrieved 19APR06 from: http://research.microsoft.com/~cszypers/events/WCOP2001/

[LUQ04] Luqi, Lin Zhang, Berzins, Valdis and Qiao, Ying. Document Driven Development for Complex Real-Time Systems, IEEE Transactions on Software Engineering, Vol 30, No 12, Dec 2004.

[MAR05] Martinez, J., del Mar Gallardo, M., Merino, P. and Pimentel, E. ed. Hongji Yang. Abstracting UML Behavior Diagrams for Verification. Software Evolution with UML and XML. Idea Group Publishing, Inc., 2005.

[MDX] MDXSYS Limited, XMLobjective 1.2©. Retrieved 19APR06 from: http://www.xmlobjective.com/

[OGC] Office of Government Commerce, OGC Successful Delivery Toolkit™ 2005, Version 5.03. Retrieved 19APR06 from: http://www.ogc.gov.uk/sdtoolkit/reference/documentation/p11_busreqts.html

[PRI91] Prieto-Diaz R. Implementing Faceted Classification for Software Reuse. Comm. of the ACM, 34(5):pp. 89 - 97, May 1991.

[PEN04] Penna, Giuseppe D., Benedetton I., Laurenzi, Anna R., and Orefice, S. A Methodology for Scenario Development Proceedings of Sixteenth International Conference on Software Engineering and Knowledge Engineering (SEKE) , 20-24/6/2004, Banff, Alberta, Canada, pp. 7 - 12, Knowledge Systems Institute, 2004.

[PRO] Protégé Owl. An Ontology Editor for the Semantic Web. Stanford University. Retrieved 05MAY06 from: http://protege.stanford.edu/plugins/owl/index.html

[RAN] Ranganathan, S.R. Elements of Library Classification. Asia Publishing House, 1962.

[SAX] SAX: Simple API for XML. SourceForge.net. Retrieved 19APR06 from: http://sourceforge.net/project/showfiles.php?group_id=29449

[SPI] Spiteri, Louise. A Simplified Model for Facet Analysis. Original publication: Canadian Journal of Information and Library Science V23, pp 1-30, April-May 1998. Reprinted: The Information Architecture Institute. Retrieved 05MAY06 from: http://iainstitute.org/pg/a_simplified_model_for_facet_analysis.php

[SWEBOK04] IEEE Computer Society Professional Practices Committee. Guide to the Software Engineering Body of Knowledge, 2004.
Retrieved 19APR06 from: http://www.swebok.org/

[WIE98] Wiegers, Carl E. Software Development, Know Your Enemy: Software Risk Management, October 1998.
Retrieved 19APR06 from: http://www.processimpact.com/articles/risk_mgmt.html

[WIE03] Wiegers, Karl E. Writing Quality Requirements. Process Impact. Retrieved 05MAY06 from: http://www.processimpact.com/articles/qualreqs.html

[WYN92] Wynar, Bohdan S. and Taylor, Arlene. Introduction to Cataloging and Classification. 8th ed. Libraries Unlimited: 1992 .

[W3C] W3C, Extensible Markup Language (XML) 1.1 W3C Recommendation 04 February 2004. Retrieved 19APR06 from:
http://www.w3.org/TR/2004/REC-xml11-20040204/#sec-intro

[W3Ca] W3C, Architecture Domain. XML Schema.
Retrieved 21APR06 from: http://www.w3.org/XML/Schema

[W3Cb] W3C, Extensible Stylesheet Language (XSL) Version 1.0. W3C Recommendation 15 October 2001.
Retrieved 26APR06 from: http://www.w3.org/TR/xsl/

[W3Cc] W3C, XForms 1.0. W3C Recommendation 14 March 2006. Retrieved 05MAY06 from: http://www.w3.org/TR/2006/REC-xforms-20060314/

[ZAV97] Zave, P. Classification of Research Efforts in Requirements Engineering. ACM Computing Surveys, 29(4): pp. 315-321, 1997.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Man-Tak Shing
   Computer Science Department
   Naval Postgraduate School
   Monterey, StateCA

4. Richard Riehle
   Computer Science Department
   Naval Postgraduate School
   Monterey, CA

5. Peter Denning
   Computer Science Department
   Naval Postgraduate School
   Monterey, CA

6. Linda Reynolds
   Space and Naval Warfare Systems Center
   Det Norfolk
   Norfolk, VI