# MULTI-DEVELOPER REQUIREMENTS ENGINEERING

Richard Shelton

TRW Inc.

Orlando, Florida

## ABSTRACT

Requirements engineering standards and processes are inadequate in the simulation and training industry to support programs with multiple developers and training objectives. This paper explains how the Joint Simulation System (JSIMS) program corrected that inadequacy by defining and implementing a robust requirements engineering process that maintains a standard, program-wide traceability and test approach, flexible enough to allow multiple development agents, of which there are eight, to use different development processes and requirements management tools. A key to this organized and successful practice was to get common plans, definitions and agreements amongst the development partners.

This paper describes how the requirements engineering process evolved and provided many lessons learned as it grew and improved to support the challenges of a program with diverse requirements and development processes. The requirements engineering process began with a collaborative effort to analyze and consolidate 12 source documents of approximately 6,800 requirements, provided by the various individual development partners, and create a single, common, binding set of top-level, or "system", requirements. These system requirements became the bounding program scope that satisfied training objectives all partners agreed to develop and test against. The next step was to sequence the delivery of these requirements, or rather the capabilities that satisfied these requirements. The sequencing was divided into 5 separate, but common product delivery milestones. A sequencing challenge was that all the partners had dependencies on each other's products and/or deliveries that had to be supported and coordinated. Each development domain became responsible for their "portion" of the system requirements, and was challenged to derive the next level or two of requirements that specifically defined functions unique to their development efforts, yet still allowed for cross domain interactions. The requirements engineering team created a common trace and reporting format, using web technology, to allow users, developers and testers to see the mapping and satisfaction of all system requirements within and throughout all the development domains. It is applicable to any multi-developer, multi-user program.

## AUTHOR'S BIOGRAPHY

Mr. Shelton is the Systems Engineering lead for the Joint Simulation System (JSIMS) program. He defined, coordinated and integrated the requirements engineering processes used to scope and manage the requirements of JSIMS. He authored and published these processes in two JSIMS Technical Notices, as well as the JSIMS Systems Engineering Master Plan document. He successfully facilitated and implemented the processes amongst nine separate JSIMS development partners.

Mr. Shelton has nine years of systems engineering experience, with five years of direct requirements engineering management. He has defined, developed and implemented three large-scale (10,000 or more) requirements database projects, each supporting multiple development partners. He was the project lead for a TRW research and development effort that successfully developed and implemented a unique web-based, graphic-user interface driven software system to categorize and manage data and documents within and across programs.

# MULTI-DEVELOPER REQUIREMENTS ENGINEERING

Richard Shelton

TRW Inc.

Orlando, Florida

## INTRODUCTION

JSIMS is a complex software development program that ties together various development partners, to include branches of military[1] and government intelligence services[2], for a common purpose of providing readily available, operationally valid, computer-simulated environments for use by the Commander in Chiefs (CINCs), their components, other joint organizations, and the Services to train, educate, develop doctrine and tactics, formulate and assess operational plans, assess warfighting situations, define operational requirements, and provide operational input to the acquisition process. That's a very broad brush of functions and capabilities to accomplish, and makes proper and focused requirements engineering effort essential. Requirement engineering needs to provide structured and common processes and formats that all development partners can support and collaborate on. There is some commonality to draw from, as software development programs approach requirements engineering with multiple tiers, or levels, of requirements that evolve from large, global system requirements, to capture the overall scope, down to minute derived requirements that identify unique functions to be coded in software. The requirements engineering program JSIMS put together shows that all of these requirements and code segments weave together and form an integrated web of capability that answers the program scope as a working system.

## GOAL

The goal of any requirements engineering effort should be to properly scope and maintain the requirements baseline of a program. The requirements products and support needs to be process independent, accepted by all as the defining scope for the entire program, regardless of the fact that one may have multiple development partners, with multiple and different development processes. As the requirements flow down to the developers, there needs to be seamless traceability from system requirements to development artifacts, throughout the entire program. A customer, user, developer, tester or any other program partner should be able to look at any requirement in the system and see a forwards or backwards trace from the source data that created the requirement to the development artifact that satisfies the requirement. And lastly, all partners buy into and support the requirements engineering process and maintain a common set of formats and data.

## COMPONENTS

A good requirements product needs proper support from several program disciplines. To properly support the scope analysis, requirements derivation, and traceability through development artifacts to test and integration, a blend of users and/or customers, systems engineers, software developers, database architects and administrators, and test and integration personnel are required. As JSIMS had eight separate development partners, this blend needed to be supported within each partner, as well as for an external collective group.

## CHALLENGES

The JSIMS program faced several challenges that strained the goals and components necessary for a successful requirements engineering efforts. Challenges are introduced now so it is understood why some implementation paths were chosen. Some of the work-arounds, issues and more ideal approaches will be further discussed in the Lessons Learned section at the end of this paper. The fundamental, and biggest, challenge JSIMS started with was that a few of the development partners already awarded and working service unique and separate simulation development contracts. In these unique efforts, they had already identified their own requirements management tools and requirements engineering methods, to include unique object oriented development processes. This drove out a difficulty in assimilating diverging and narrowly focused efforts into a common and coordinated requirements engineering effort. We needed to get service unique requirements, specifications, management tools and momentum to converge on a common product. In doing this we discovered a second major challenge with the difference in types, fidelity and resolution of requirements from each of the development partners. Some had very broad, top level training requirements, while others had very detailed, low level requirements. A last major challenge to point out is that a successful, coordinated effort requires complete participation and buy-in by all development partners. Due to cross-domain and/or programmatic issues amongst the multiple developers, JSIMS did not get early or complete participation by all development partners,

[1] Joint staff, Army, Navy, Air Force, Marine Corps

[2] Strategic, Operational and Tactical Intelligence groups

which caused unnecessary delays and multiple iterations of requirements analysis.

## DISCUSSION

## REQUIREMENTS ENGINEERING

Requirements engineering is the starting point of the program to define the development scope and/or baseline. There are several analysis, documentation and management methods that support the conduct of requirements engineering, and with the complexity and multiple developer aspect of JSIMS, we struggled, and lost momentum on a few occasions, but successfully implemented the following sequence of events that define and support our program baseline.

1. Collect all the individual source requirements sets that define both common needs, as well as each development partners unique needs and/or interests.
2. Identify a Common Requirements Allocation Structure into which the complete requirements set can be organized.
3. Employ a common method for defining time-phased delivery of customer expectations.
4. Analyze, consolidate, bin (organize) requirements
5. Establish and maintain the requirements/development baseline (JSIMS RTM)
6. Developers sequence the requirements for time-phased delivery of products
7. Begin detailed traceability from System requirements (baseline) down to development artifacts.

The results of the requirements analysis are captured, baselined, and maintained in the Program Manager (PM) controlled Requirements Traceability Matrix (RTM).

### Collect Source Data and/or Requirements

In order to gain collaboration and cooperation from all development partners, a program must be scoped to surround and support all partners requirements. The primary intent is to take this collection of common and specific requirements that each development partner needs, and generate a common, single set of requirements that all development partners agree to as a baseline. Twelve source documents, ranging from Functional Requirements Documents to System/Subsystem Specifications, came together from all development partners to support the JSIMS requirements engineering efforts. All source documents were laid down as a foundation for the program-wide requirements database, or JSIMS Requirements Traceability Matrix (RTM). The JSIMS RTM content and schema will be discussed in detail later. As requirements engineering efforts evolved, they would all trace back to, or find their origin from, these source documents to allow development partners to see how their components fit in.

### Identify Common Requirements Allocation Structure

As mentioned in the Challenges section, JSIMS started off being behind other service efforts, and the requirements engineering team had to figure out how to bring together multiple requirements and/or source documents into a common specification. There are generically two ways to structure, or view, the requirements: 1) Users or 2) Developers. The two views are rarely similar, and a simplification on one side usually means a complex conversion and mapping effort on the other. For JSIMS, the user perspective revolved around the functional areas, or attributes, that defined their training environment; such as Air Operations, Land Operations, Sea Operation, Mobilization, Physical Environment, Pre-exercise, Exercise Execution, Evaluation, Time, and System Configurations. In contrast, a developer's perspective would dictate a structure that revolves around software objects, or top-level categories; such as movement, sensing, combat, communication, decision and direction, intelligence, and logistics. JSIMS chose to organize their requirements around the user's perspective, as it was considered the most concise way of separating and "binning" the requirements without overlap or redundancy. It was understood that the "fan-out" of requirements trace to development artifacts would become large, complex and likely redundant at times. This will be further discussed in the traceability section.

### Employ a Common Method for Defining Customer Expectations

JSIMS has two formal product deliveries, Initial Operational Capability (IOC) and Final Operational Capability (FOC), with a couple interim product deliveries between the two. Given this fact, the delivery of products to satisfy requirements could be time-phased, or sequenced, over these intervals. The question becomes, what priority of deliveries is appropriate to best support the user's needs? To better understand the time-phasing expected for satisfaction of requirement, we need to recognize our customer's, or end-user's, training expectations.

Training needs for JSIMS are best viewed in terms of classes of JSIMS applications. These applications require differing sets of capabilities. For example, the types of simulations and simulation support functions needed to train Joint Task Force (JTF) commanders and component commanders differ from what is needed to train an Air Force Wing Operations Center (WOC) commander or to develop new doctrine. These potential classes of JSIMS applications are called Functional Capabilities. For JSIMS, the users defined 31 unique Functional Capabilities, in the form of Functional Capabilities Codes (FCCs).

To support the prioritizing of user needs, the FCCs were spread across the delivery intervals. The users further supported the requirements engineering team with the association of these FCCs to system requirements. Knowing this association of requirements to time-phased FCCs, the requirements engineering team could then correctly scope and derive requirements that allowed a sequence of product deliveries to support the FCC, or training, expectations.

## Analyze, Consolidate and Bin (Organize) Requirements

With a common requirements allocation structure, revolving around the user's perspective, as well as a time-phased allocation of training expectations, JSIMS had a proper foundation to begin a comprehensive analysis of the source requirements. The next step was to begin the very large effort of reviewing, analyzing, consolidating, organizing, and/or creating the JSIMS program scope from this huge set of source requirements data. All development partners were requested to provide engineering support for this effort.

The requirements engineering team that was formed began by dividing up the source documents and, one requirement at a time, binning, or allocating, each requirement into an appropriate attribute of the requirements allocation structure. As this effort progressed, a few specific concerns became apparent:

1. The one-size-fits-all requirements allocation structure was insufficient to capture the complete set of source data.
2. The breadth, depth, and/or resolution of the source requirements varied greatly. Some broad requirements gave general statements like "movement and engagement of forces shall be represented down to battalion level", while others specifically defined details like "smoke shall be created by smoke pots, large area smoke generators…"
3. There was a tremendous amount of redundancy in the requirements across the different domains.

The answer to the first concern was obvious, the requirements allocation structure had to be expanded with additional categories to accommodate the unique source requirements. We went back to basics, and pulled additional sections that are standard for requirements specifications, such as: Safety, Security, Reliability, Maintainability, Availability, Design Constraints, Personnel, Training, Logistics, and Packaging.

For the second concern, the team decided to create two level of requirements, System and Detailed, to essentially bin the requirements within the requirements. System requirements are defined[3] as:

> *Top-level requirements that define broad functions or training needs of the JSIMS program. These requirements are verified and/or tested collectively at a system level. They are derived from a collection of the common source documentation normalized or aggregated to a system level.*

And as a subset, scoped within these System requirements, are Detailed requirements, defined[3] as:

> *Lower level requirements that support and are scoped by the System requirements. Detailed requirements are sub-components, or functions, of System requirements and provide unique or specific development items to focus the understanding of program needs.*

The third concern required some domain knowledge of training and simulation needs and expectations to distinguish between subtleties in wording that may or may not have been redundant. The team requested and received additional help from training and simulation subject matter experts (SME). Having identified additional necessary attributes, defined two levels of requirements, and gained support of SMEs, the team continued and completed the task of organizing, allocating, analyzing, and otherwise creating the program's baseline scope.

Through all the analysis and crafting of the program baseline, the team had to remember to include with each System requirement the FCC that provides a user's delivery expectation. The training needs and/or expectations for each System requirement was identified by the collections of applicable FCCs. The lower level Detailed requirements, which were laid underneath and bounded by the System requirements, were not assigned unique FCCs, but rather inherited the FCC of the System requirements they supported.

[3] Per JSIMS Systems Engineering Master Plan, Version 2.0.

### Establish and Maintain the Requirements/Development Baseline

The results of the requirements engineering team's efforts was a large and complex collection of System requirements, with associated FCCs and Detailed requirements. It was not perfect, and in fact, while the requirements analysis team crafted this collection, many of the source requirements had been updated or modified in a parallel effort by the development partners. The issue was how to properly track and incorporate any changes to the current set of data. It was determined that the existing set of System and Detailed requirements needed to be based-lined as the development scope of the program, and begin a controlled process of change management to all aspects of the data, to include: source data/requirements, System requirements, FCCs, and Detailed requirements. The JSIMS program manager, with the support and agreement of all development partners, baselined the System and Detailed requirements in a product called the JSIMS Requirements Traceability Matrix (RTM), and put it under strict change control (see Configuration Management section).

### Developers Sequence the Requirements for Time-Phased Delivery of Products

The development scope for the program was defined and baselined in the System and Detailed requirements set. The user's time-phased training expectations were defined in the associated FCCs. This defined "what" the program "needed". This set was handed to the collective development partners with the question, "can you support these needs?" Schedule, resources, technology constraints, and programmatic collaboration constraints were a few of the factors that affected the answer to that question. Another analysis effort between requirements engineers and developers was required to characterize the developers ability to support the program scope and delivery expectations. The documentation mechanism created to support this time-phased characterization of delivery support was called Sequencing.

The JSIMS RTM schema was expanded to include sequencing columns for each development partner. Each development partner read every System and associated Detailed requirement, and allocated a product sequencing, or delivery, date to the System requirement. Each development partners did not have to provide an entry to every requirement, as many would not apply to products they would deliver or support, but every System requirement did have to have at least one sequencing entry. The sequencing versions (V) choices matched the FCC defined training intervals; specifically, JSIMS had seven delivery sequencing choices: blank, V1.0 (IOC), V1.1, V1.2, V1.3, V2.0 (FOC), and NP. Blank means that no entry was provided because the requirement did not apply to that development partner. NP means "not planned", which applied when a development partner agreed and understood they were responsible for supporting some portion of a particular requirement, but for whatever programmatic reasons they could not support it. Combinations of these indicators are allowed, as complete satisfaction of many requirements spans multiple builds and/or versions.

After several long and challenging meetings, the JSIMS RTM program baseline was updated with a complete set of sequencing data to support the System requirements. Detailed requirements were not uniquely sequenced because, like FCC allocations, the Detailed requirements inherited the sequencing allocations of the System requirements to which they were linked, or associated. As with everything this big and complex, there were exceptions and notes added to some Detailed requirements that, although within scope of their linked System requirements, could not be satisfied at the same time. These were handled and updated on a case by case basis.

### Begin Detailed Traceability from System Requirements (Baseline) Down to Development Artifacts

Requirements traceability applies to a wide spectrum of requirements engineering needs. It could apply to a single thread of a tracing source data to a System or Detailed requirement, or is could apply to the process that shows the connectivity of all requirements to the development artifacts produced to satisfy the requirements. This paper will discuss the latter definition, as the connectivity, or traceability, of all System requirements, across multiple development partners, to development artifacts that satisfy the requirements, became a complex and critical process that all facets of the program became involved or interested in. Traceability became a necessary thread, or glue, that connected the users expectations, as defined in the System requirements, through the design elements, through the integrated products, to developed code, and allowed the system test group to verify all the answers (see Figure 1).

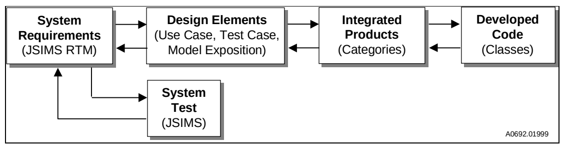

System Requirements (JSIMS RTM)
Design Elements (Use Case, Test Case, Model Exposition)
Integrated Products (Categories)
Developed Code (Classes)
System Test (JSIMS)
A0692.01999

Figure 1. Functional View of System Requirements Traceability

The products discussed thus far in this paper, incorporated and maintained in the RTM, provided the scope and expectations management for this multi-developer program. Each development partner then took their allocated and/or accepted set of System and Detailed requirements, as defined by the sequencing inputs they committed to, and began a process to further refine and derive requirements that focused on software objects and development. The aforementioned challenge (see Identify Common Requirements Allocation Structure) of a complex conversion and mapping of user-oriented requirements into software development oriented products emerged quickly.

Conventionally, in a program with a single developer, the traceability is easily maintained in the program's baseline RTM. In a multiple developer program, this traceability, though potentially more complex, could still be accomplished in a single database, if the development partners all used the same development processes and requirements/database management tools. JSIMS, as mentioned previously in the Challenges section, has neither of these luxuries. Several of our development partners, having individually started their programs prior to JSIMS, had unique and different object-oriented development methodologies (Texel/Williams[4], Shlaer-Mellor, and a derivative of Booch-Rambaugh[5]), as well as different requirements databases.

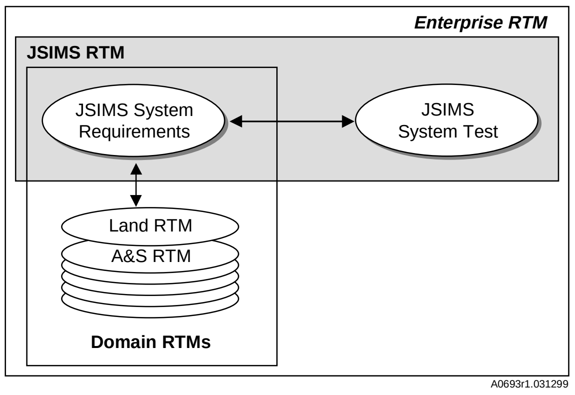
Theoretically, we still could have created a common and centralized RTM that maintained all requirements and traces to development and test artifacts, but that was programmatically unsupported by all development partners. The management and maintenance of this RTM would be very difficult and time-consuming, and would require all development partners total support and acceptance. Also, due to the programmatic issues, it would also be very redundant with data maintained by the individual development partners. Instead, we chose to implement a common method of reporting traceability to allow the same visibility into products, while still providing each development partner the programmatic flexibility they required.

JSIMS implements an RTM concept that consists of multiple linked elements. This RTM, which JSIMS calls Enterprise RTM, consists of two parts, a single JSIMS RTM and multiple Domain RTMs. The JSIMS RTM contains the JSIMS System requirements, defining and scoping the program development baseline, and linkages between the System Test and Domain RTMs (see Figure 2 for a logical view of the Enterprise RTM schema).

Figure 2. Enterprise RTM

## System Requirements Centric

Each development partner is responsible for creating and maintaining a database that supports its requirements tracing from the JSIMS RTM System requirements to its development products. The format of this database is at development partner's discretion, with the exception that all software products developed



for JSIMS be traced from JSIMS System requirements. The System requirements trace must include the Enterprise-wide unique key identifier of System requirement ID as defined in the JSIMS RTM. Development partners must maintain traceability from JSIMS System requirements down to Category and/or Class levels, or equivalent levels of implementation. This traceability may be accomplished according to the individual development partner's process.

## Domain RTM Reports

Domain RTM reports are common reports provided to and posted at an JSIMS program level. These reports provide the traceability from JSIMS RTM System requirements to each development partners development artifacts. These reports are unclassified and must be provided electronically, to the JSIMS Webmaster, in the form of Excel spreadsheets. These reports contain the following columns of data:

a. System Requirement ID.

b. Detailed Requirement ID.

c. Derived or Design Requirement ID.

d. Derived or Design Requirement text.

e. Build/Version of Derived/Design Requirements.

[4] Use Cases Combined with BOOCH/OMT/UML; P. Texel and Charles B. Williams, Prentice Hall, Inc., 1997.

[5] Object-Oriented Modeling and Design, Rumbaugh, Blaha, Premerlani, Eddy and Lorensen, 1991, Prentice Hall

f. Use Case ID, Test Case ID, or Exposition Model ID.

g. Category (or equivalent) Name.

h. Class (or equivalent) Name.

The Class Name data comprises a unique column, in that it is only filled in when a Derived or Design requirement is not directly supported in development by a Category. An example would be if a requirement were a very simple function that will be implemented by a single class or method, rather than through a Category to Class hierarchy.

### Scope and Development Partner Responsibility

JSIMS RTM System requirements are sequenced for time phased deliveries. For each requirement where a development partner has added sequencing data for its domain, the development partner is responsible to show that trace in the Domain RTM down to a development artifact. A many-to-many mapping in the traceability is expected, as a single development artifact will likely be used to satisfy several broad System requirements, but all System requirement entries for a development partner must be accounted for. In some cases, there will be some System requirements that are global, such as for safety or MIL-STD compliance, which must be noted in the Domain RTM as not requiring unique development artifacts.

### Constraints Traceability and Satisfaction

Many of the requirements that all development partners must support are software constraints that do not specifically trace to unique development products. The Domain RTM reports, however, still need to recognize and show a trace for satisfaction of all System requirements against which a development partner provided a sequencing input. For example, development partners could place a “C”, for constraint, in place of a Use Case, where a software constraint requirement has been derived for the development effort. This shows recognition of the requirement in the development effort, as well as a completion of the trace from a System requirement into the development effort.

## CONFIGURATION MANAGEMENT

One of the harder questions to answer is, “how much configuration management (CM) can I have without slowing down or constraining the momentum and progress of the engineers and developers?” The answer varies depending on whether all development partners are following the same development processes, with common requirements tiers and traceability paths. A well-defined CM process, and adherence to that process, became a key to gaining acceptance and support from all development partners of the program-wide development baseline documented in the JSIMS RTM and traced further into the Domain RTMs. JSIMS configuration management was directed by the JSIMS Configuration Control Board (JCCB). The JCCB was chaired by the JSIMS program manager, and was responsible for configuration managing the JSIMS RTM, or program development baseline, consisting of the System requirements, and their associated Detailed requirements, Functional Capability Codes and sequencing data. Once the JSIMS RTM development baseline was established (December 1997), the JCCB tracked and evaluated all change requests via formal System Problem Reports (SPR). The SPR process provided a multi-stage method of analyzing and controlling changes, while at the same time providing the necessary visibility to all development partners.

This centralized control, with common visibility, gave the development partners confidence in the stability and management of the program baseline. Below the JSIMS RTM level, however, the multiple development process and efforts made common control complex and cumbersome. As stated in the Domain RTM discussions, all development partners were scoped by and had to support and trace to the System requirements, but they were also given the flexibility to configuration manage the derived requirements and products they developed in support of those System requirements. Regular updates and reporting of the Domain RTMs became the means of programmatic visibility into each partners development efforts in support of the program baseline and common needs.

## LESSONS LEARNED

Creating mulit-developer requirements engineering processes and answering the challenges faced in JSIMS provided many lessons learned. Many of the lessons learned, such as mandating that all development partners follow the same development process, may not be correctable or controllable by the requirements engineering teams, but rather are programmatic. Nonetheless, this section presents the lessons we learned from various impacts, delays, and/or problems we were with our requirements engineering efforts.

- Full participation – All development partners must completely participate in all aspects of the programmatic requirements engineering efforts. In the very beginning of collecting and analyzing the source data to create a common baseline, some of the development partners chose not to participate. One could speculate many reasons for this, as some had already started and were obligated to their own program contracts, but it was a visible problem. The analysis team that did conduct the baseline development effort were unable to adequately support the position and needs of the missing partners. This led to issues later when the baseline was established and these missing partners would argue that their requirements were not properly scoped and supported. Many SPRs and man-hours were used to correct these deficiencies.

- Full buy-in – All development partners must accept the program baseline as scoping and binding to their development efforts, and use that as their controlling data. When the JSIMS program baseline was established, many of the development partners agreed to it for JSIMS, but did not accept it as a scoping and controlling baseline for their development efforts. Several of the partners still maintained their own RTMs and system requirements specifications. This became a conflict of interest and configuration management on several occasions. The development partners need to throw away their individual documents and embrace the commonly developed and controlled data as a universal scope that all adhere to and support.
- Common RTM tools – All development partners need to work out of and/or support a common RTM. Again, this stems from some partners having started their programs prior to JSIMS being established, and having chosen their unique RTM tools and development processes. It became difficult to import, export, or share RTM data amongst partners because of tool incompatibility. Complete traceability is a very cumbersome job because we cannot incorporate and create a common, single database with all Domain RTM report data. The Domain RTM report standard established was via Excel spreadsheet reports generated be the different RTM tools each DA used. Had JSIMS controlled a single decision for RTM tools and control, then commonality across all development partners would have allowed us to maintain an integrated traceability.
- Common Development Process (or at least common terminology) – All DAs need to have a common, or compatible, approach to their development efforts. Multiple concurrent development efforts are fine, but when they use different processes, with different terms for each phase, product, and development effort, then it is very difficult to create, standardize and maintain a common baseline and requirements traceability effort. For example, throughout the various development efforts, the partners were associating and defining system, detailed, derived and design requirements differently. Further, when associating and tracing down to development products, confusion and concerns were surfaced by trying to correlate and normalize use cases, model exposition, test cases, categories, classes, methods and so on.

## CONCLUSION

The requirements engineering challenges of a multiple developer program like JSIMS are certainly not unique, but this author was not aware of any published answers that would have made the path easier. This paper is intended to explain the path JSIMS took and identify the components that helped and hindered the progress and success we achieved. JSIMS ran into many challenges that were new to all requirements engineers involved. Common, yet unique, solutions, standards and processes were created to manage the challenges. We learned several lessons that cost the effort valuable time and support, and hope this paper provides others necessary understanding and insight to gain efficiency and strength in their efforts.