DOMAIN REQUIREMENTS FOR CONSTRUCTIVE WARGAMES FOR COMMAND AND STAFF TRAINING AND C4I STIMULATION

Randy Brasch, Lisa Callahan, and Eytan Pollak
Lockheed Martin Information Systems
Orlando, FL

ABSTRACT

A small team of engineers and subject matter experts at Lockheed Martin Information Systems has recently completed a Domain Analysis focused on constructive simulations used in both command and staff training (CAST) systems and command, control, communication, computer, and intelligence (C4I) stimulation systems. The intent of the domain analysis was to collect typical requirements for such systems and identify a common set of requirements that could be used to drive the development of a framework and toolset to facilitate future development of systems in the domain. This paper describes the results of that domain analysis. Specifically, it highlights the effort to sufficiently abstract and level the requirements to define the constructive simulation framework and toolset and it identifies the areas in which the currently recognized requirements are weak or missing.

ABOUT THE AUTHORS

Randy Brasch is a software engineer at Lockheed Martin Information Systems in Orlando, Florida. For the past four years he has focused on the analysis and design of constructive simulation systems to support commander and staff training. A significant part of that effort has been devoted to investigating and prototyping various techniques for communicating between simulation systems and C4I systems.

Lisa Callahan is currently a program manager at Lockheed Martin Information Systems in Orlando, Florida. She has worked as an engineer in the simulation business for over six years and specifically in the constructive simulation area for the past four years, including three years as the chief engineer for the WARSIM 2000 program.

Dr. Eytan Pollak is currently the IR&D Technical Director at the Lockheed Martin Information Systems Company in Orlando, FL. His responsibilities include the Reconfigurable Simulator, Embedded Training Systems, War Gaming, Simulation Architecture, Synthetic Environment, and other DIS/HLA research programs. He serves as an adjunct professor of Electrical and Computer Engineering at the University of Central Florida. He received his Ph.D. from Purdue University, West Lafayette, IN

DOMAIN REQUIREMENTS FOR CONSTRUCTIVE WARGAMES FOR COMMAND AND STAFF TRAINING AND C4I STIMULATION

Randy Brasch, Lisa Callahan, and Eytan Pollak
Lockheed Martin Information Systems
Orlando, FL

INTRODUCTION

The United States military is in the process of upgrading the simulation systems used for training commanders and staffs in Command Post and Operations Center environments. Systems such as the Joint Simulation System (JSIMS), Warfighters Simulation 2000 (WARSIM 2000), and the National Air and Space Model (NASM) are currently being developed to provide training capabilities well into the 21st Century. The development and deployment of these systems will take several years, and these systems will require a large number of resources to execute. Meanwhile, there is a need for smaller, less resource intensive systems to provide a subset of the training capability of these large systems at much lower cost. For example, the US military units need smaller scale systems such as the Low Overhead Driver (LOD) and Digital Battle Staff Trainer (DBST) to provide local training capabilities. Smaller systems such as these are useful because they require very little support. That is, the soldiers can set it up, operate it, and there is little requirement for role players to support an exercise. Another example of why smaller systems are needed is because they provide the capability for small countries with limited budgets to advance their training capabilities and be able to train with the US military in the future.

Although the development of a small scale training system is less resource intensive than that of a larger system, the specification, design, and development of even a small scale training system is still an ambitious project. Typically, it is more expensive than any single using organization can afford given today's limited military budgets. For that reason, as an alternative to developing a complete, unique system for each using organization, this research explores the idea of developing a production system for generating smaller scale, less expensive training simulations that could be tailored to meet the unique needs of each potential using organization.

The long-term vision of this project is to design and develop the capability to rapidly and inexpensively generate training simulation systems tailored to end users specific needs. This vision requires the development of a set of tools to build simulations as well as the development of a set of simulation components that can be easily adapted for specific applications and used within a general wargaming architectural framework. This led to the definition of a wargame system factory concept that combines the use of tools to support development efforts with a repository of reusable components to enable a streamlined production capability.

From past experience, it was apparent that this application seemed to have many of the characteristics of a classic domain engineering problem. It encompassed a line of specialized products that share many commonalities and offered the potential for gaining production efficiencies through planned reuse. Since domain engineering efforts for both JSIMS and WARSIM 2000 programs had recently been completed, the expertise and lessons learned from those efforts were used to establish this project's baseline.

DOMAIN ENGINEERING

Domain Engineering is an extension of Systems Engineering and is intended to provide a discipline for efficient design and development of related products and systems. It is related to engineering for reuse in that it focuses on identifying commonalities among products and systems and leveraging those commonalities to provide reusable elements for those products and systems. The Domain Engineering process (Figure 1) can be defined as:

> *"… a systematic approach which involves defining high-leverage domains or product-lines, establishing architectures that address the needs of each product within those product lines, and development and management of reusable assets that populate those architectures."* ("Domain Engineering", 1995)

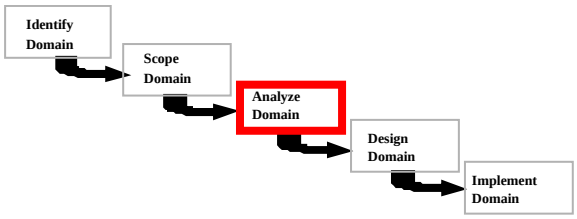

Figure 1: The Domain Engineering Process

This paper presents the first three steps of this process: identification of domain, scope of domain, and analysis of the domain, with the emphasis being placed on the domain analysis phase. That phase is used to collect, organize, analyze and present information in order to identify common requirements within a domain ("Domain Engineering", 1995). These initial phases lay the groundwork for identifying the product line to be developed and provide the basis for the later phases of designing and implementing products. They establish the basis for commonality and reuse and, as such, are the determining factors of the efficiency of the ensuing production efforts. Successfully completing these phases results in the identification of all of the areas that are potentials for planned reuse. Later steps in the process determine which of those areas will actually be developed. Those decisions are typically formulated based on potential return on investment.

Identify and Scope Domain

As indicated earlier, the domain of interest encompassed constructive simulations used for CAST and C4I stimulation systems. This is a subset of the Training, Exercises, and Military Operations (TEMO) domain. One of the three domains for Army M&S applications, TEMO includes most forms of training at echelons from individual simulation trainers through collective, combined arms, joint, and/or combined exercises. TEMO includes mission rehearsals and evaluations of all phases of war plans. ("TRADOC Reg 5-11", 1998)

The focus of this investigation is on collective training at command post and operations center levels, not on crew and team training at the platform level or on individual training. Additionally, the stimulation of C4I systems was included, since interoperability with real world C4I systems has become an essential requirement for commander and staff training systems. Finally, to further scope the domain, focus was placed on the Brigade and Battalion level echelons of commander and staff training systems.

As part of the identification effort, characteristics that might distinguish each of the subdomains were identified (Figure 2). This provided the initial assessment about the feasibility of including each of these in the domain and also served to identify areas in which reuse might not be effective.

Collective Training	C4I Stimulator	Brigade / Battalion
Low cost. Ease of use. Adaptable for language and culture. Supports joint / coalition. Aggregate resolution.	Repeatable. Predictable. C4I native interface. Autonomous simulated units. Visibility of C4I interaction. Control of C4I interactions. Ability to introduce errors. Control load on C4I systems. Unattended, long duration runs.	Platoon / squad / platform resolution.

Figure 2: Distinguishing characteristics of subdomains within the domain of interest.

While analyzing the subdomains, it was apparent that there were some seemingly conflicting characteristics between collective trainers and Brigade / Battalion level systems. These differences centered on the level of resolution. Collective trainers typically represent aggregate units such as companies and above, while Brigade / Battalion level trainers typically require visibility to the platoon or even platform level. However, it was decided to keep both elements in our initial effort because advances in technology are making lower level units visible at higher levels of command. For example, the availability of battlefield video in command posts from remote sensors provides this type of visibility. Thus, it was reasoned that even higher level commander and staff trainers would eventually require platform level resolution for some areas of the virtual battlespace.

Analyze Domain

The next step in the process was to collect and analyze the requirements (Figure 3). In this phase of the project, the requirements are collected (INPUTS), the analysis is performed (METHODOLOGY) and the result is a set of leveled and categorized requirements (PRODUCTS). The following sub-sections address these functions individually.

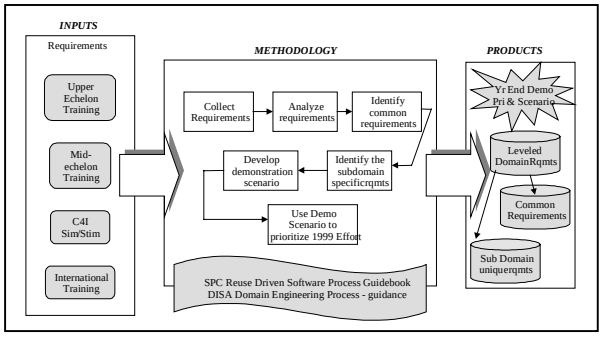

Figure 3: Domain Analysis Task Process Flow

Collect Requirements. The first activity was to identify possible sources of requirements that would be applicable to the domain. Fortunately, a large base of available requirements was available from two major US systems' domain engineering efforts. Additionally, the researchers had worked closely with actual and potential international customers to define requirements for systems of a smaller scale than the US systems. Thus, the requirements base used in this project was a "requirements rich" base.

Eventually, it was decided to work with a set of requirements that covered a broad spectrum of users ranging from large, upper echelon training systems to smaller, lower echelon training systems. These seemed to represent a fairly complete sample of the requirements for the types of systems within the target domain. However, it was apparent that none of the requirements in the base set seemed to address the issue of stimulating C4I systems to provide stand-alone training. Since this seemed to be a natural extension of small scale constructive training systems, the researchers wanted to include those types of requirements in the initial requirements base. This was achieved by using a Capstone Requirements Document written at NSC that provided an initial attempt to "…define high level common requirements for all simulations (tactical, training, analytical, and test) that must link to C4I systems of the future." The requirements in that document were used to provide the inputs for the C4I stand-alone training capabilities in the requirements base.

Tools and Methodology. The Dynamic Object Oriented Requirements System (DOORS) requirements analysis tool was selected as the primary Computer Aided Software Engineering (CASE) tool for recording and allocating requirements. DOORS was already in use on the our WARSIM 2000 program, so using it provided a base of knowledgeable users as well as an extensive database of domain specific requirements. Also, since several of the requirement documents were in Microsoft formats, Microsoft Word and Excel were used. Microsoft Word, in particular, provided a mechanism for preparing requirements for importing into DOORS and for handling DOORS outputs. Lastly, it was discovered that organizing and combining requirements in the Word environment was more efficient than in the DOORS environment.

The basic requirements analysis process incorporated features from the Software Productivity Consortium (SPC) Reuse Driven Software Process Guidebook ("Reuse Driven Software", 1993) and the Defense Information Systems Agency (DISA) Domain Engineering Process ("Domain Engineering", 1995) as well as lessons learned from our previous domain engineering experience. Introducing a spiral development process then enhanced this process. The intent of this enhancement was to maintain focus and incentive by providing a concrete goal at the end of each spiral. This was accomplished by means of defining a scenario to be executed at the end of each spiral and. adding a phase to define demonstration scenarios using the highest priority requirements. In practice, it was necessary to balance both high priority requirements and resource limitations to reach a definition of reasonable scenarios for the spirals.

Analyze Requirements. Once the requirements were identified and collected, the requirements analysis phase began. The goal of this phase was to synthesize a complete set of requirements for systems that would be needed in the domain for the next 5 years or so. Thus, both the combined requirements from existing systems as well as anticipated requirements for systems in the development pipeline were included. Initially, the goal was to combine the requirements, eliminate redundancies, and produce a set of unique, leveled requirements that presented a complete picture of the needs within the target domain. However, this turned out to be far too ambitious for this project. As a result, the effort and expectations were scaled to something that could be accomplished within the resources allocated to the project.

Since a good cross section of the systems in the domain had been captured in the existing requirements base, its use was continued as the project proceeded through the phases of the system engineering process. However, not all of the requirements in that base were used during this initial effort. Instead, focus was placed on those requirements that dealt directly with Brigade / Battalion level training systems. It was believed that the greatest potential return could be achieved by focusing on those requirements since they seemed to generally to apply to all segments of the domain (Figure 4).

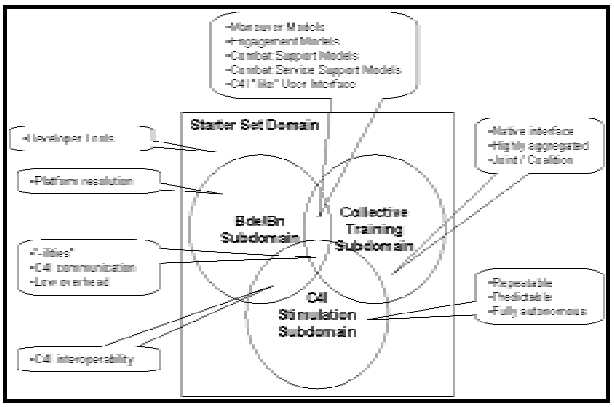

Figure 4: Requirements distribution across subdomains.

One of the biggest challenges and an initial goal of the requirements analysis effort was to abstract the input requirements to a common level that would be applicable for system level design and reuse analysis. Using the wide variety of inputs available resulted in a large base of requirements that included broad generalizations as well as detailed implementations.

For example, the following requirement essentially provides a high level definition of a complete training system:

> *The system shall provide an interactive, multi-sided, force-on-force, real-time modelling and simulation for Joint Service tactical operations.*

In contrast, the following requirement provides a specific list of class attributes for a model:

> *The system shall model the cargo capacities with the following characteristics: weight, volume, length, width and height.*

It quickly became obvious that abstracting the requirements to a consistent level would be a prodigious task and probably would not support the project's goals. It was necessary to insure that the common requirements were consistent and that those requirements that applied at the system and software architecture levels were clearly identified. However, eliminating the details that perhaps were more applicable to the design level was not essential. Instead, the researchers focused efforts on identifying those requirements that would influence the common product architecture and on partitioning the requirements based on the architectural components.

Based on previous experiences in domain and systems engineering, it was apparent that a target system architecture would be needed to effectively partition the requirements. There has been a lot of work done over the past few years in exploring architectures for constructive simulation systems used in collective training. Consequently, it was decided to use that base of knowledge and not “invent” a new architecture. Also, a commercial simulation infrastructure and framework to use in the initial implementations had already been selected. The combined effect of these decisions led to a structure that resembles the JSIMS architecture and fits into the framework provided by a commercial product. Since, at this point efforts are primarily targeted at the software aspects of training systems, a model to provide the "bins" for partitioning the requirements was created (Figure 5). The software categories that map to the elements of the software model are:

- General Software Requirements
- User Support Toolset
- User Interfaces
- Message Models
- Environment Models
- Developer Support Toolset
- Interface Handlers
- Process Models
- Equipment Models

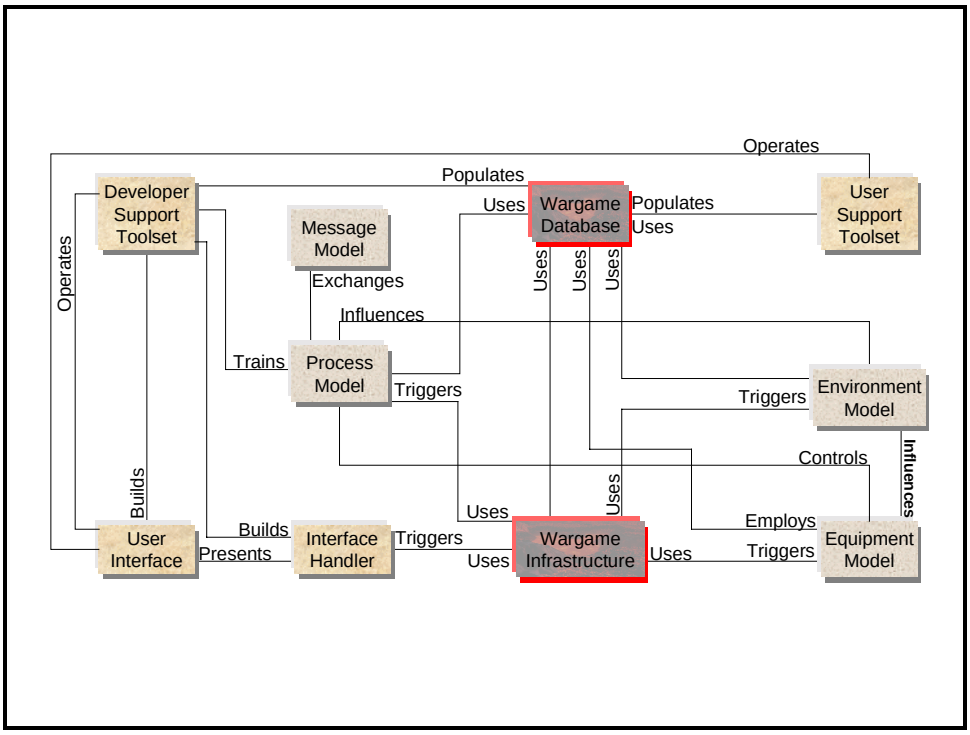

Figure 5: Software model for the wargame system factory.

General Software Requirements: This category captures those requirements that applied to the overall software capabilities, but did not fit into one of the other categories. These include requirements that deal with general system concepts (e.g., open, scalable, etc) and that deal with overall system performance (e.g., run faster than real time, provide spare memory, etc).

Developer Support Toolset: This category represents capabilities that would be used by system developers according to this project's concept of how the systems would be produced. This category includes some specific tools unique to the software technologies that are intended to be incorporated in the products. For example, this category includes a tool to support generation of rules for cognitive models. Also included in this category are tools that might be delivered with the system, but primarily be used by the developers of the system, as opposed to its end users in the target markets. For example, it includes tools for populating the detailed parameter databases used by the simulation.

User Support Toolset: This category probably represents a more common view of the applications that would be used to support a developed simulation system over its life cycle. It includes the tools that the end users will need to use their system in their environment. These include such tools as a scenario generator and network monitor.

Interface Handlers: This category was used to capture requirements that define the capabilities that the simulation has to provide to the user. The system architecture for external interfaces is based on the Model-View-Controller (MVC) design pattern (Gamma, Helm, Johnson & Vlissides, 1994). This provides a separation of the user interface from the application and allows either to be changed independently of the other. The interface handlers provide the mechanism for mapping interfaces to applications.

User Interface: This category is used to capture the requirements that define the user's view of the system. This includes the view of the simulation as well as that of the system tools.

Process Models: This category is intended to capture the requirements dealing with the behaviors of the entities that are modeled in the simulation, and on the cognitive capabilities included in the system to provide automation of those behaviors. The separate identification of the "thinking" part of the simulation is typical of current simulation architectures.

Message Models: This category is a result of the architecture adopted based on the structure of the

commercial framework being used. It also fits with a basic concept in modeling real world entities – "model the way it works in the real world." As a requirements category, it was used to catalog the requirements that define the information that is transferred among entities in the simulated battlespace.

Equipment Models: This category is used to capture the requirements that cover the physical characteristics of the entities that are to be simulated.

Environment Models: This category is used to capture the requirements that describe which aspects of the physical environment being simulated must be modeled.

Assess Requirements. It was evident that most of the requirement sets that had been reviewed included at least three types of requirements:

- Conceptual requirements that apply to the system as a whole and that attempt to define overall characteristics. These included concepts such as open, scalable, usable, and composable.
- Performance requirements that define metrics such as how many, how fast, and how much. Typical requirements in this category included such things as numbers of entities simulated, relationship of processing time to real time, and percentage of spare memory and storage.
- Functional requirements that define what the system is supposed to do. These tended to range from very broad (e.g., provide a realistic environment) to very specific (e.g., the maximum ground speed of an M1 tank shall be 100 kph).

As mentioned above, the conceptual and performance requirements generally apply to the system or the software in a broad sense and could not be allocated to any specific system component at this time. They provide a set of guidelines for designing and developing systems and may even require that the developers design systems in such a way as to foster reuse principles.

The functional requirements provide a rich environment for discovering reuse potential. Many of the functional requirements were either identical or very similar across the systems. This was as expected, since the perceived commonalities among the systems stimulated the initial interest in performing this effort.

During this phase, it became apparent that the Internet has somewhat changed the technical side of the procurement process and, perhaps, led to more commonality than might be expected from a diverse set of requirements writers. Many of the international customers have picked up information from US documents available on the Internet and incorporated it into their requests for proposals. Many requirements were either exactly the same or very similar to the WARSIM and JSIMS requirements that have been published on the Internet. However, this has led to international customers requesting a set of capabilities that would be "challenging" to provide within their budgetary constraints. It also possibly sets a level of expectation that may be difficult for low cost systems to meet.

As a result of this analysis, it was discovered that specifications typically went into great detail about the equipment to be simulated, provided reasonable information about the interactions between units, and defined the environmental effects that were to be simulated. They also did a fairly complete job of defining the interfaces to external systems.

Identify Key Requirements. With respect to the domain of constructive simulations, the key requirements were deemed to be those that would influence the basic architecture and that would remain part of any product that was produced as a domain application. Those requirements tended to be the more general requirements that really provided guidelines for a flexible system design. They included requirements that would be applicable to any product as well as those that were more specifically oriented to products in the target domain.

Extensible Design. Because there are many potential customers with several different needs, an underlying requirement is an architecture that is extensible both in content and scope. The flexibility to tailor the war game simulation to the customer's requirements without major architectural changes is one of the design goals.

Machine Independence. Until the specific customer requirements are identified, this must be a generic solution to constructive simulation requirements.

Low Cost. One of the underlying benefits of a simulation is that it costs less than a real war. Because potential customers are seeking a less expensive and faster alternative for conducting war games, the solution must require minimal support for setting up and running an exercise as well as provide a low maintenance environment. There is a real concern that development, maintenance, and operational costs be as low as possible.

Turnkey solution. For these smaller scale systems, end users really want to get a product that they can set up and operate on their own. They do not want

systems that require contractor support for day to day use.

Comprehensive. The simulation must be capable of providing a training environment that encompasses all of the activities that modern military forces are being asked to perform and that incorporates the effects of all of the resources that affect the outcomes.

WARGAME SYSTEM FACTORY

The ultimate vision for this project is a production system that is capable of generating the applications that meet the needs of specific customers. As such, the basic architecture has evolved into a production line concept that provides capabilities to efficiently extend a common simulation framework and customize off the shelf components to meet the requirements of any domain specific application. This results in a "factory" which has a composable simulation system at its core, an inventory of reusable simulation components, and a customization facility consisting of several tools to address specific product needs (Figure 6).

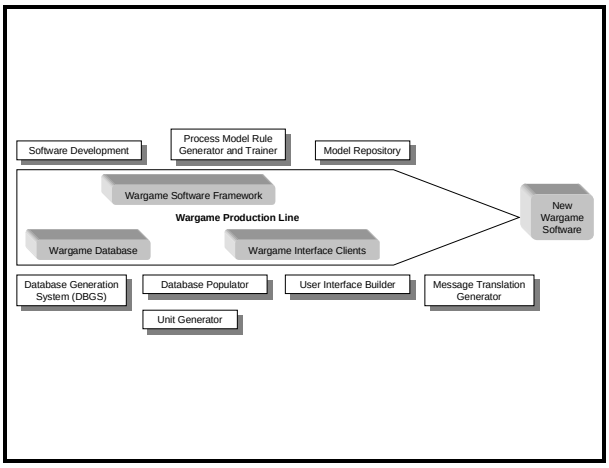

Figure 6: Starter Set production line architecture.

The factory concept and its associated architecture as depicted here represents the initial work in the next phase of the domain engineering process, viz., Design Domain. In the model, there are three major system components that move along the production line: the software framework, the database, and the interface clients. A common software framework is extended for a specific product at the software development, process model rule generator and trainer, and model repository stations. The database for the product is generated at the database generation, database populator, and unit generation stations. Interface clients are constructed at the user interface builder and message translation generator stations. Finally the three system components are brought together at the end of the line to form the customized product for a specific user.

LESSONS LEARNED

In doing the domain analysis we learned several things and confirmed several of concerns that had been expressed going into the task. Probably the key lesson learned was that more advanced tools are really needed to efficiently do the domain requirements analysis on a large scale, e.g., identify common or related requirements, identify redundant requirements, sort requirements by category. DOORS promises to be able to perform some of these tasks to some level, but those advanced capabilities were not tried in this effort. DOORS was used to record and categorize requirements on a large scale and that proved to be very tedious. DOORS over a busy network with lots of requirements and links can be painfully slow.

Collecting the requirements and setting up the requirements database surfaced some problems with the magnitude of the effort that had been undertaken. Gathering requirements from many different sources introduced many redundancies in the database. An initial goal was to eliminate those redundancies to come up with a database of unique requirements. In practice this proved to be more of an effort than anticipated. A lexical analysis tool to automatically at least suggest requirements that appeared to be duplicative was sorely needed. Even after sorting the requirements by system component, the resulting datasets were still too large to evaluate manually in the time available.

No procedural mechanism was used for identifying common requirements across systems. Essentially the analyst had to either remember that a system had a similar requirement or do a manual search for similar requirements across systems. Some automated tool to identify requirements that were similar to reduce the scope of some of the searches would have been a great benefit. Another handy tool would have been something that could be used to record a common version of a requirement and link the common version to all of the originals that went into it. Again, this could be done using DOORS, but was not pursued in this effort.

Floating licenses: We were using a pool of licenses for the DOORS tool and, although we were not directly involved with other programs sharing the same licenses, we were impacted by their use. Whenever one of the programs would go into a requirements analysis phase, the licenses would inevitably be used up. Because we were not a

contract program, we did not use the tool when it became evident that programs were running out of licenses. Fortunately, we were never at a loss for other things to do, but this did occasionally interrupt our workflow. I can only imagine the impact of the shortages on the programs.

SUMMARY

We have embarked on a Domain Engineering effort initially targeted at battle staff training at the Brigade and Battalion level. During the course of this study, the domain definition was extended to include the stimulation of real world digital command and control devices. The domain engineering effort performed to date has identified the broad domain of interest, reduced its scope to a target subset that appeared to provide a high potential for return on investment, and completed a domain analysis which focused on the collection and assessment of requirements from existing and potential future systems within the domain. The requirements set was extended to fill in missing and weak areas and to incorporate ideas pertaining to a factory concept. The resulting requirements were categorized based on their applicability across the domain and partitioned based on a general component architecture for a commander and staff training system production factory.

CONCLUSION

The work we have done so far is a start towards implementing our vision of a commander and staff training system factory. We will continue to refine our requirement base as we proceed through the rest of the phases of the Domain Engineering process. At this point we have a good start, a fairly complete requirements set, a solid simulation infrastructure and framework, and an initial set of support tools.

REFERENCES

Domain Engineering Process, Version 2 (1995). Defense Information Systems Agency.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J., (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley.

Reuse-Driven Software Processes Guidebook (1993). Software Productivity Consortium Services Corporation.

TRADOC Reg 5-11 (1998). US Army Training and Doctrine Command (TRADOC) Models and Simulations (M&S) and Data Management.