IGI PUBLISHING

701 E. Chocolate Avenue, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.igi-global.com

ITJ 4962

This paper appears in the publication, International Journal of Decision Support System Technology, Volume 1, Issue 2
edited by **Guisseppi A. Forgionne © 2009, IGI Global**

The Design for Tractable Analysis (DTA) Framework:
A Methodology for the Analysis and Simulation of Complex Systems

John M. Linebarger, Sandia National Laboratories, USA

Mark J. De Spain, Sandia National Laboratories, USA

Michael J. McDonald, Sandia National Laboratories, USA

Floyd W. Spencer, Sandia National Laboratories, USA

Robert J. Cloutier, Stevens Institute of Technology, USA

ABSTRACT

The Design for Tractable Analysis (DTA) framework was developed to address the analysis of complex systems and so-called "wicked problems." DTA is distinctive because it treats analytic processes as key artifacts that can be created and improved through formal design processes. Systems (or enterprises) are analyzed as a whole, in conjunction with decomposing them into constituent elements for domain-specific analyses that are informed by the whole. After using the Systems Modeling Language (SysML) to frame the problem in the context of stakeholder needs, DTA harnesses the Design Structure Matrix (DSM) to structure the analysis of the system and address questions about the emergent properties of the system. The novel use of DSM to "design the analysis" makes DTA particularly suitable for addressing the interdependent nature of complex systems. The use of DTA is demonstrated by a case study of sensor grid placement decisions to secure assets at a fixed site. [Article copies are available for purchase from InfoSci-on-Demand.com]

Keywords: *Complex Systems; DSM; Simulation and Modeling; SysML; Systems Analysis Methods*

INTRODUCTION

Systems analysis approaches to persistently challenging problems, which have a variety of stakeholders and scenarios, are traditionally solved using linear or canonical methods. In general, the steps for solving this class of problems include: 1) describing the system in a static model; 2) designing the system to meet the functional requirements; 3) simulating the system to understand how the parts of the system behave; and 4) modifying functional requirements using insights and observations derived from the simulation. This approach to analysis decomposes the overall system into subsystems where most of the analysis effort is applied. Effects due to interdependencies between the subsystems are often not analyzed to the same depth as the subsystems primarily because there is no good closed form analytic method for understanding the contributions of subsystem interactions.

This traditional approach to problem analysis has historically addressed most of the concerns and objectives of the systems of interest. However, as the demand has grown for more capable and autonomous systems to address increasingly complex geopolitical, environmental, security, and combat environments, the problem space has transitioned from linear problems into the "wicked problem" domain, where the solution transforms the problem (Hodge and Weinberger, 2005). Our own work has discovered that interdependencies between subsystems are a primary determinant of system behavior, and can catapult a complex system into the "wicked problem" category.

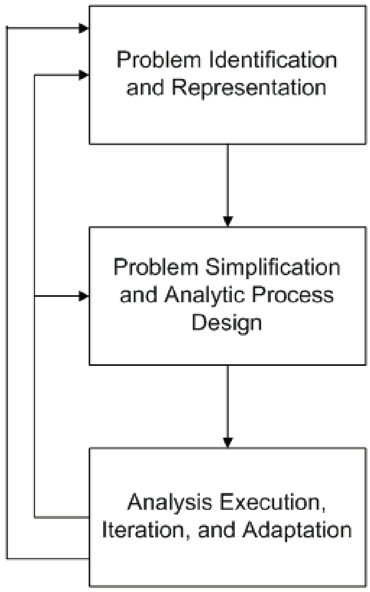
One class of wicked problems is characterized by the interaction of autonomous and semi-autonomous systems within an enterprise. This article proposes that the interactions produce interdependencies that can continuously transform the problem space. A common task for addressing this class of problems is to describe and organize the interactions and interdependencies that cut across the decomposable elements of a system, in order to capture and analyze the functions that the system performs. This becomes even more important when considering an enterprise-wide problem that incorporates potentially-conflicted domains, due to the increased number and complexities of interactions and interdependencies that exist in an enterprise. Examples include climate change, public policy for social systems, and government response to natural disasters. The research questions addressed by this article are how the interdependencies that characterize complex systems and wicked problems can be preserved as the system is decomposed, and how the resulting decomposition can be tractably analyzed and simulated to support a decision. In order to discuss and develop the proposed analytic framework, a case study will address the class of wicked problems associated with security of high-valued assets such as nuclear stockpiles, public utilities, and transportation infrastructures. While the case study focuses on security at the enterprise level, the authors believe the techniques that have been developed apply to a broader range of wicked problems.

The philosophical underpinning of this work is that an analytic process is an artifact and should be designed using formal design methods. As much attention should be paid to the design of the analytic process as to the analysis itself. This article attempts to demonstrate that "designing the analysis" is possible, feasible, and useful.

DESIGN FOR TRACTABLE ANALYSIS (DTA) FRAMEWORK

The vehicle we are proposing for designing the analytic process for wicked problems is an analysis framework called Design for Tractable Analysis (DTA). A distinction of DTA is that the system is initially analyzed as a whole. The analysis of the whole is then used to inform the formal design of an analytic process that efficiently addresses interdependencies within the system under study. This formal design process contrasts with reductionism, which involves the decomposition of the system into constituent subsystems and analyzing them independently of each other. Barton and Haslett (2007) present a good discussion of the tension between reductionism and reverse reductionism in science, which is characterized as a dialectic between analysis and synthesis. DTA strives to avoid reductionism, and is particularly good at untangling the interdependencies that characterize complex systems, and uses abstractions and patterns to exploit the way the analyst's mind works. The method frames the problem and the model in the context of the questions being asked by the stakeholder and the analyst. The nature of the questions the analyst asks about the system will affect the structure and

Figure 1. DTA process flow



execution of the model.

As shown in Figure 1, DTA is an iterative three-stage process that includes 1) problem identification and representation, 2) problem simplification and analytic process design, and 3) analysis execution, iteration and adaptation. The specific steps executed at each stage follow; a detailed explanation is provided in a later section in the context of a case study. Note that as a framework, DTA allows other tools and analytic approaches to be used at each stage, depending on the problem being addressed. The tools listed below were chosen because of their applicability to the problem domain of the case study.

1. Problem Identification and Representation
   - Creation of a top-level Unified Modeling Language (UML) or Systems Modeling Language (SysML) diagram to represent the behavioral aspects of the complex system;
   - Decomposition of that top-level UML or SysML diagram to a level of detail appropriate for the analysis of the system; and
   - Extraction of key decision questions from the diagrams to drive the analysis of the system.
2. Problem Simplification and Analytic Process Design
   - Use the Design Structure Matrix (DSM) methodology to cluster the system tasks from the UML or SysML diagrams into analytic domains;
   - Create a coverage map that maps the clusters into existing analysis or simulation codes;
   - Where gaps appear in the coverage map, develop or purchase software for those analytic domains; and
   - Iterate through step 2, or return to step 1, depending on the results of the DSM analysis and the coverage map.
3. Analysis Execution, Iteration, and Adaptation
   - Create a simulation execution workflow based on the analysis questions and the ordering determined by the clustered DSM output; this will often require a workflow that consists of multiple simulation codes coupled together;
   - Iterate through the simulation execution workflow, especially if the simulations are stochastic in nature and a large number of runs are required for the results to converge;
   - Adapt the simulation results to the analysis questions by adjusting simulation parameters and rerunning the workflow; and
   - If the simulation results do not provide sufficient insight to the key decision questions, return to step 1 or 2.

RELATED WORK

The proposed Design for Tractable Analysis framework touches many categories of related work. Citations of several such categories of related work are presented below, in the following areas: complex problems and wicked systems; patterns, cognition, and mental models; the Design Structure Matrix (DSM) methodology; and multiobjective optimization (which is a goal of the case study that will be presented).

Several characterizations of complex systems exist in the literature. An influential one from Bar-Yam (2004) includes the following characteristics:

- Complex systems are characterized by emergent, self-organizing, collective behavior. Such emergent collective behavior often arises due to the combination of simple individual behavior patterns;
- The components of a complex system are interdependent;
- Complex systems exhibit multiscale variety, in which the structure of the system differs depending on the level at which it is viewed. A common pattern is a mixture of competition and cooperation at different levels of the system;
- Complex systems arise due to evolutionary, not deterministic, processes. Engineered systems should be implemented this way too; and
- Examples of complex systems include the health care system, the education system, and recent military conflict.

Michel Baranger's characterization (2001) is complementary to Bar-Yam's:

- Complex systems contain many constituents interacting nonlinearly;
- The constituents of a complex system are interdependent;
- A complex system possesses a structure spanning several scales;
- A complex system is capable of emerging behavior;
- Complexity involves an interplay between chaos and non-chaos. In other words, "complex systems dance on the edge of chaos;" and
- Complexity also involves an interplay between cooperation and competition.

So-called "wicked problems" are a proper subset of complex systems. In a seminal article, Rittel and Webber (1973) observed that a whole realm of social planning problems do not respond to traditional linear, analytical approaches. They coined the term "wicked problems" to distinguish these issues from "tame problems." A tame problem is not necessarily trivial—it can be very complicated—but unlike a wicked problem, a tame problem can be readily defined and possesses at least a quasi-stable solution.

Several lists of characteristics of wicked problems have been proposed (Rittel & Webber, 1973; Conklin, 2006; Horn & Weber, 2007). The consensus is that solutions to wicked problems are non-deterministic, meaning that no single correct answer exists. A brief synthesis of these lists follows:

- Every wicked problem is essentially unique;
- There is no definitive formulation of a wicked problem;
- Wicked problems are never solved;
- A wicked problem is not understood until after the formulation of a solution;
- Solutions to wicked problems change the problem itself;
- Solutions to wicked problems are not true or false, but better or worse;
- Stakeholders have radically different world views and different frames for understanding the problem; and
- Solutions to aspects of wicked problems are often contradictory.

A significant roadblock encountered in addressing wicked problems is the inability of the human mind to grasp all the pertinent elements and dynamics of the problem. Because human beings design systems, the design of complex systems is constrained by what humans can understand. Johnson-Laird (1983) notes that when we say we understand something, we are saying that we have created a mental model that we can manipulate to make inferences and predictions about the thing itself. System models provide simpler views that we can understand in a common language. Such models can provide a higher level of abstraction that allows for a working understanding of the system without the need for overwhelming detail (Doyle, 2007).

Cloutier posits that the notion of patterns is almost universal, and that the human mind seems to perceive patterns without conscious thought. Patterns may exist in many different levels of a given system, including the existence of system level patterns, which are abstractions of the structure of several similar systems. A skilled analyst can identify and document these common system structures using abstraction techniques. This observation leads to the conclusion that cognition is contextual, and requires the development of relevant abstractions. These patterns represent models, which are an abstraction of reality (Cloutier, 2006). Consequently, whatever exists in the mind of an analyst and stakeholder is merely a representation of what actually exists in the world. As a result, a primary purpose of modeling complex systems is to make the unconscious use of models explicit and unambiguous, and to better align the soft models that exist in the mind of the analyst with the hard, and sometimes harsh, truth of reality. A further conclusion is that simulations of models of a wicked problem do not provide "the answer" to the problem, but instead inform the mind of the analyst who must make the tough analytical decisions.

The Design Structure Matrix (DSM) is an analysis approach first introduced by Steward (1981). Using a two-dimensional matrix, it maps interactions and interdependencies between entities. DSM is a proven approach and has been used to analyze interdependencies in several domains: product development activities (Eppinger *et al*., 1994), system architecture design (Browning, 2001), project planning and scheduling (Maheswari & Varghese, 2004), organization design (Woodman & Bilardo, 2005), and the creation of flexible designs (Cardin *et al*., 2007). However, to our knowledge no previous work has utilized the DSM to structure and order the analysis and simulation of a complex problem in the way that is proposed below.

Pimmler and Eppinger (1994) demonstrated the use of the DSM on a passenger-climate-control system for the Ford Motor Company. They observed that complex problems are commonly decomposed into small sub-problems to perform analysis for improved understanding. DSM was used to enhance the product architecture of an already well-understood problem. By using DSM, they discovered a superior architecture and team structure that Ford had not considered since it did not fit into their organizational structure.

Yassine published a DSM tutorial that focused on the information flows between teams developing complex engineering products (Yassine, 2004). Yassine called this "activity-based DSM" and showed how it could be used in engineering design management. He began by citing earlier work by Smith and Eppinger (1997), which used DSM to analyze the complexities of an anti-braking system. Yassine observed that patterns can be found in the completed matrices that represent information-exchange patterns.

Tyson Browning (2000) used the DSM to model complex processes as complex systems, where a large number of interdependent processes must be integrated, synchronized and coordinated. He constructed large, complex DSMs by first creating smaller DSMs and then integrating them into larger DSMs. Browning found that using DSM to analyze complex, interdependent processes resulted in reduced variability and execution time.

An alternative to DSM is Nam Suh's more mathematically rigorous axiomatic design formulation (Suh, 2001). However, we chose DSM for its ease of explanation and application to the particular case study chosen. Axiomatic Design and DSM can even be used together; Guenov and Barker (2005) present a design decomposition-integration model called COPE in which Axiomatic Design is used to map functional requirements to design parameters, while DSM provides a structured representation of the system development context. In addition, an alternative static method of examining whole-problem interrelationships (which is a strength of DSM and Axiomatic Design), is proposed by Anderson, Boxer, and Brownsword (2006).

Numerous methods for multiobjective optimization exist; see the excellent survey by Marler and Arora (2004). Coello has provided a comprehensive survey and overview of the use of evolutionary genetic algorithms for multiobjective optimization (2000, 2003), as well as a standard textbook (Coello, Van Veldhuizen, & Lamont, 2002).

DETAILED EXPLANATION OF THE STAGES OF DTA

To motivate and exemplify the DTA framework, a case study is presented in which each of the stages of DTA are exercised.

Problem Identification and Representation

DTA begins by creating a top-level use case diagram (Figure 2), which in the case study represents a security system for a high-valued asset residing within a fixed-site facility. The asset requires that many types of operations be performed, which are categorized either as mission critical operations or as facility-related operations. Additionally, the asset must be protected from adversaries. Note that the use cases are directly derived from the stated purpose of the system, which is to perform operations on a high-valued asset. This case study is a wicked problem because human beings are unpredictable and adaptable. The introduction of an adversary into the operations of the site prevents a deterministic solution to the security questions. The case study exemplifies the multi-scale characteristic of a complex problem as well, in addition to interdependencies, cooperation, and conflict; this is represented in Figure 3.

Figure 2. Top level use case diagram of fixed-site security case study

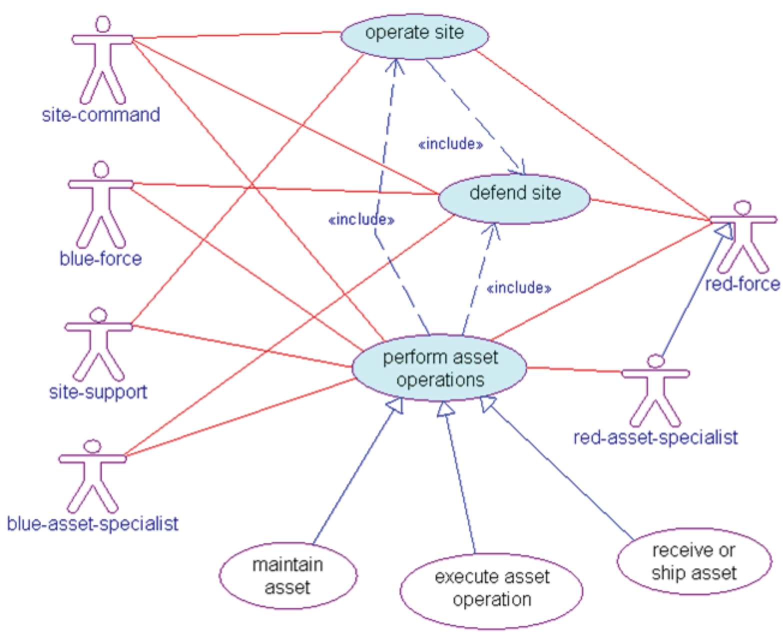


Figure 3. Multi-scale complex system with interdependencies, cooperation and conflict

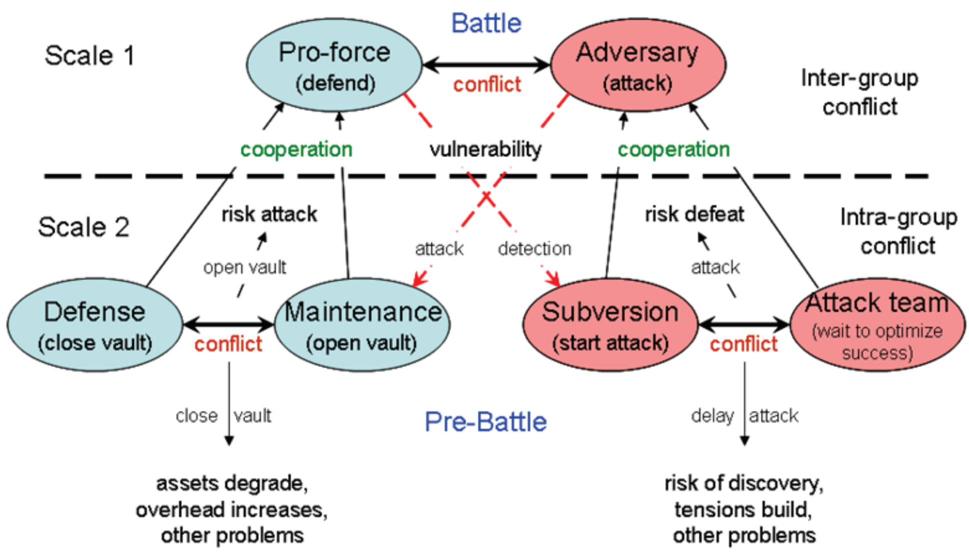


Figure 2 captures two different yet related views of the enterprise of interest on a single top-level UML or SysML (Balmelli, 2007) use case diagram. One view is the operational/design view represented by the "bubble" and "actor" elements and the other is the operational/dependency view represented by the dashed dependency lines that are stereotyped with "include" in the diagram. The design view is necessary to specify the behaviors and structures in the enterprise and their relationships. The use cases in the design view identify the questions and issues that are important to the stakeholders that will ultimately determine the success or failure of the enterprise. The dependency view is useful for identifying the order in which the analysis questions are addressed, which is discussed in more detail later. Note that a parametric SysML diagram, which is a relatively new type of SysML, could be included at the analyst's discretion in order to specify aspects of the system operation to be included in the dynamic analysis. However, for the case study, the UML use case diagrams provided sufficient context to create a dynamic model of the behavior of the system.

After a top-level use case diagram is created, it is hierarchically decomposed into additional lower-level use case diagrams. Figure 4 depicts a first-level decomposition of one of the tasks, Operate Site, from the top-level use case diagram (Figure 2). The case study scenario was decomposed to an appropriate level of detail for analysis, which resulted in a total of five diagrams at the lowest level of decomposition (*i.e.*, five leaf node diagrams).

The key decision questions an analyst asks will affect the structure of the resulting analytic model. In the context of the case study of securing a fixed-site asset, we now show the process by which a good set of key decision analysis questions are derived. In essence, the three tasks or use cases in the top-level use case diagram of Figure 2 form the basis for three questions that will determine the analysis flow. In addition to the use cases, a structure diagram for the enterprise can also highlight relationships and constraints that affect the determination of analysis questions.

A UML/SysML block diagram, which is illustrated in Figure 5 for the case study, provides another view of the problem domain. Several

Figure 4. Second level use case diagram of operate site activity

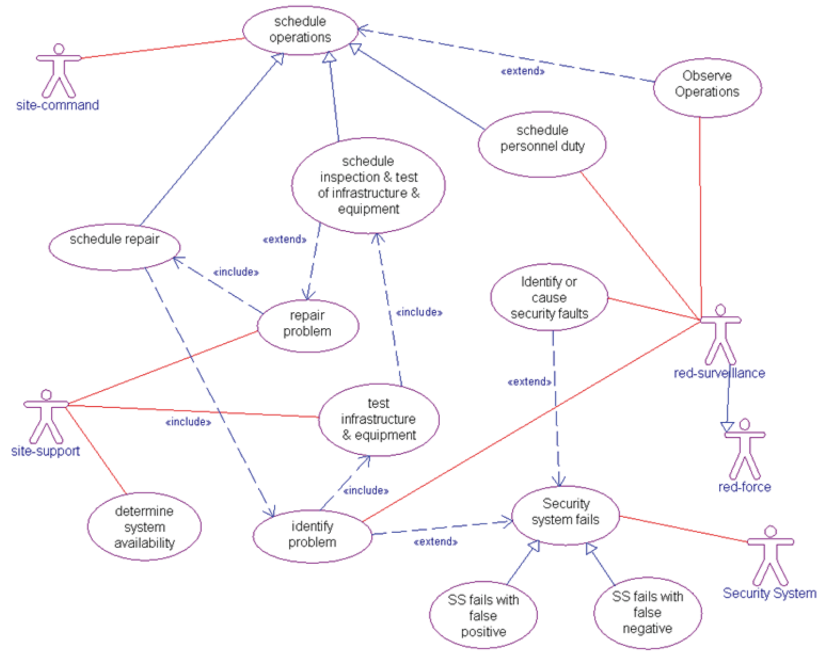
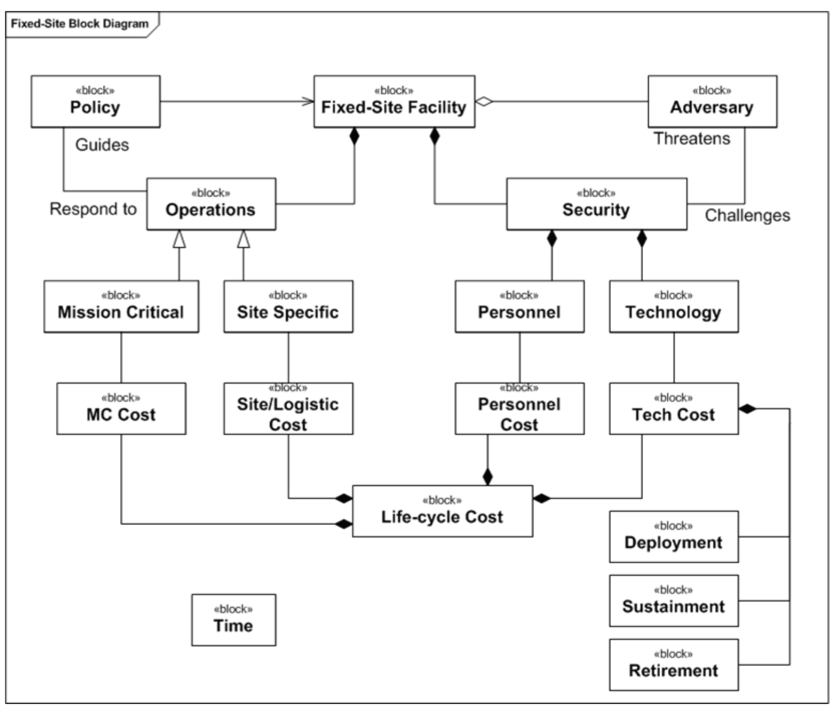


Figure 5. Block diagram of case study



significant elements are included in Figure 5, most notably cost, policy and time. Figure 2 is thus a behavioral view of the system, while Figure 5 is a structural view. The major domains of the system are policy, adversary, operations, security, cost and time. Note that each domain is in conflict or tension with the other elements to some degree. In general, policy constrains operations, adversary and security mutually constrain each other, and cost constrains operations and security. Figure 5 highlights the impact of life-cycle costs on the specific objectives of the use cases, such that the analysis questions are framed to include this specific constraint.

The process of determining the analysis questions begins by identifying the relationships between the use cases, because that governs the order in which the questions must be answered. As will be shown for the case study, often the dependency-driven order may not be the order that is most important to the primary stakeholder.

For the case study, ordering between the tasks, as shown in Figure 2, is indicated by the "include" stereotype on the association lines that link the tasks. In other words, since Perform Mission Critical Operations includes, or is dependent on, Perform Site Operations and Defend Site, the latter two use cases take precedence temporally. Finally, since Perform Site Operations depends on Defend Site, the latter takes precedence. This order of precedence means that Defend Site is a necessary condition for the other two.

Placed in the order of dependence, and taking lifecycle costs from Figure 5 into account, the three use cases of "protect site and asset," "maintain site," and "perform mission critical operations," lead naturally to the following three key analysis questions: 1) What is the optimal balance of cost, technology and performance that maximizes the security of the site and asset while minimizing the deployment costs? 2) What are the optimal maintenance and personnel schedules that maximize site security readiness and minimize sustainment costs? 3) What is the maximum number of mission critical operations that can be executed with acceptable risk for a given budget? Note that answering the third question completes the first iteration of the analysis, which determines the viability of the site as an enterprise.

Each succeeding question is dependent on the previous question, and system interdependencies make it impractical to attempt to answer all questions concurrently. The DTA is designed to restructure the tasks of the analysis model and shift the focus of the analysis as answers are sought for each question. The consequence is that the analysis at the enterprise level proceeds sequentially with an iterative overlay when subsequent questions are not readily answered (see Figure 1). That is, several iterations may be required to balance the conflicting objectives of each question relative to the objectives of the other questions. This implies that the qualitative words used in the above questions, such as "maximizes," "optimal," and "acceptable," do not reflect absolute valuations; instead, each must be balanced across all the objectives for the system.

Finally, the main interest of the primary stakeholder will generally revolve around how much value he derives from his dollar; that is, his focus is going to be on the last question regarding the number of operations that can be performed, given the associated risk and the overall cost of operations. The primary stakeholder may not be interested in the particulars of the first two questions; however, the concerns of the primary stakeholder cannot be adequately addressed without resolving the issues identified in the first two questions. In other words, the three analysis questions are sequentially dependent on each other.

Completing the UML/SysML analysis and identifying the key decision questions prepares the analyst to begin problem simplification and analysis. This is performed using the Design Structure Matrix to identify and order the interdependencies between tasks.

Problem Simplification and Analytic Design

In this step, the analyst designs an analytic process that will tractably address the key decision questions. UML/SysML has already informed the analyst of the processes and relationships that exist in the modeled enterprise. Using the DSM approach, the analyst starts to simplify the problem and design the analytic process. We call this step "designing the analysis" because not only does it identify the analysis tools needed, but it also indicates the order in which they should be applied. An implementation of DSM created by MIT—which uses Microsoft Excel macros to implement partitioning, tearing, and banding—was used for this step; it is freely available from The Design Structure Matrix Homepage (http://www.dsmweb.org).

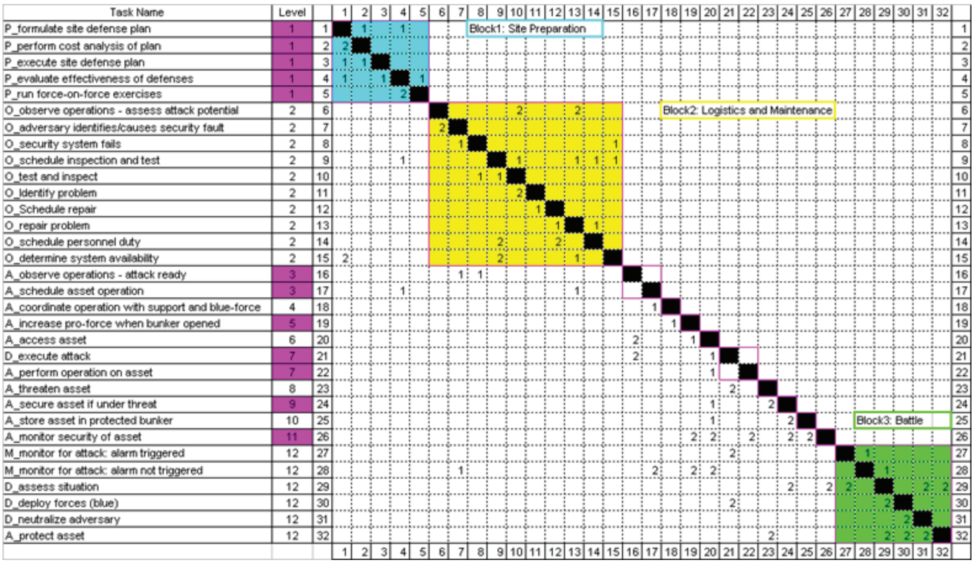
Figure 6 depicts the input to the DSM for the fixed-site case study. Each of the entries on the left is a task from a leaf-node (*i.e.,* fully decomposed) use case diagram for the case study. However, only leaf-node tasks that interacted with other tasks in some way, either in a predecessor/successor relationship or in terms of a data dependency, were included in the DSM input matrix. The prefix in the DSM task name is a code for one of the five fully-decomposed use case diagrams from which the task came: O – Operate Site, D – Defend Site, P – Prepare Site Defenses, M – Monitor for Attack, and A – Perform Asset Operations. Note that there is no specific order required for entering the values into the matrix. However, it is important to ensure that the relationships between tasks are completely identified. The authors found that several iterations were required to verify that the information was correct and complete. The distinction between '1' and '2' in the cell entries is that '1' represents a sequential ordering where the initiating task must be completed prior to the start of the trailing task, and '2' indicates a data dependency where the trailing task can start any time after the start of the initiating task.

In Figure 7 the input DSM of Figure 6 has been transformed by reordering the tasks to minimize the number of feedback relationships. Those that do remain are grouped into self-con-

Figure 6. Fixed-site DSM input

Project Name		<NOTE>			1. Enter the name of a project in cell '2A'.																												
Fixed Site Security Analysis					2. Enter all task names in column 'A'.																												
Task Name		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
O_observe operations - assess attack potential	1					2			2																								
O_adversary identifies/causes security fault	2	2																															
O_security system fails	3		1																														1
O_schedule inspection and test	4					1			1	1				1																			1
O_test and inspect	5			1	1																												
O_Identify problem	6					2																											
O_Schedule repair	7						1																										
O_repair problem	8							1		1																							
O_schedule personnel duty	9				2			2																									
P_formulate site defense plan	10											1		1																			
P_perform cost analysis of plan	11										2																						
P_execute site defense plan	12										1	1																					
P_evaluate effectiveness of defenses	13										1		1		1																		
P_run force-on-force exercises	14													2																			
M_monitor for attack: alarm triggered	15																1	2															
M_monitor for attack: alarm not triggered	16		1																1					2		2						2	
D_execute attack	17																								2	1							
D_assess situation	18															2					2	2					2			2			
D_deploy forces (blue)	19																	2	2														
D_neutralize adversary	20																			2													
A_protect asset	21																		2	2	2								2				
A_coordinate operation with support and blue-forc	22																															1	
A_increase pro-force when bunker opened	23																						1										
A_observe operations - attack ready	24		1	1																													
A_access asset	25																							1	2								
A_monitor security of asset	26																							2		2		2		2	2		
A_perform operation on asset	27																									1							
A_threaten asset	28																	2															
A_secure asset if under threat	29																									1			2				
A_store asset in protected bunker	30																									1				2			
A_schedule asset operation	31								1					1																			
O_determine system availability	32				2				1		2																						

Figure 7. Fixed-site DSM output



tained iterative operational blocks, which have been labeled Site Preparation, Logistics and Maintenance, and Battle. The tasks included in each block represent a set of related activities that must usually be iterated in some fashion to derive a useful result that can be employed by the downstream tasks. Those tasks that are not contained within an iterative block are essentially sequential, although some may be performed in parallel if they fall within the same level of the transformed matrix. For example, tasks 21 and 22 appear to be out of order, but they are actually executed in parallel; in other words, based on the scenario for the fixed site, the adversary can launch the attack when the operation on the asset has begun.

The advantage of using DSM is that a relatively complex enterprise with many interdependent activities can be succinctly described, grouped into related subgroups, and globally ordered at the lowest level based on dependencies. The subgroups identified represent analytic domains (or functions) against which existing analysis tools can be applied, or for which analysis tools will need to be developed if they do not currently exist. In addition, the grouping of the tasks provides an indication of the specific tasks that should be performed by the analysis application. For example, as seen in Figure 6, an analytic capability for the Battle function is needed to complete the analysis of the total system, as well as an analytic capability for Logistics and Maintenance. Observe also that Logistics and Maintenance should be analyzed prior to Battle.

It must be stressed that the SysML use case diagrams analyze the system as a whole, at an enterprise or cross-enterprise level. This prevents premature decomposition along subsystem lines, which can obscure interdependencies between subsystems. Subsystems are not identified until iterative blocks are created by the DSM analysis step.

Analysis Execution, Iteration, and Adaptation

In the case study, the analytic iterative blocks identified by the DSM analysis in Figure 7 are well delineated. This is not surprising given the marked difference in the domains of Site Preparation, Logistics and Maintenance, and Battle. While Site Preparation is a necessary activity, the other two are of greater importance to a dynamic analysis of the system; consequently the case study will focus on the domains of Logistics and Maintenance and Battle. Once the analytic domains are identified the analysis proceeds to the construction of the analytic process.

SIMULATION EXECUTION WORKFLOW

For several years Sandia National Laboratories has been developing world class simulation applications to evaluate such enterprise-level issues as agent-based force-on-force battle and the field availability of sophisticated combat technology based on reliability and logistics. Until recently, a software system to couple multiple simulations in the analysis of a common problem did not exist. However, several commercial applications are beginning to fulfill this need. Among these applications are Phoenix Integration's ModelCenter; Engineous's iSIGHT, SAMTECH's BOSS Quatro, and MSC's SimManager. These products integrate multiple simulation applications and provide an optimization framework with a broad spectrum of techniques. The case study used iSIGHT to manage the integration and execution process.

The case study consists of a scenario for a notional force-on-force battle simulation at a notional fixed-site facility that could be an armed forces base, a nuclear power plant, an embassy compound, or another secured facility. The analysis uses a software application developed at Sandia called Dante (Design Analysis of Neutralization Technologies Evaluator), which supports the analysis of physical security systems. Sophisticated stochastic models are employed to represent the uncertainties associated with battle, which require numerous iterations to converge on a result. Dante simulates force-on-force engagements, and uses batch mode processing to analyze the large amount of data generated by a given scenario. Not only can the data be analyzed to compute the probability of neutralization, but it can also be used to gain additional insight into the effectiveness of the physical security system and potential options for improvement.

Dante is used to answer the first analysis question regarding optimization of site defense. The assumption is made that there are no issues related to availability of equipment and personnel, and that all required mission critical operations can be fulfilled as scheduled, which obviates the need for Logistics and Maintenance analysis. This assumption is an approximation of reality, which the analysis can iteratively improve with later simulations, but it does make the initial analysis "spiral" tractable.

The decision that drives the case study is where to place grids of sensors in a sensor network, in order to cost-effectively protect the fixed site. The sensor grid contains a variety of sensor types, with a minimal amount of sensor fusion capability, placed at a single location.

The simplified simulation execution workflow for the case study is implemented and managed using iSIGHT, and is shown in Figure 8. The goal of the simulation for the case study is to discover the topology in both overall cost and battle effectiveness with respect to the number and placement of sensor grids, which monitor the fixed-site facility and its assets. A design-of-experiments module drives a loop which consists of the Dante application and a cost model currently implemented in Microsoft Excel. The Dante model parameters (independent variables) are the number and placement locations of the sensor grids (shown in Figure 9). The primary response variable is the percentage of engagements won by the protective force (also known as the pro-force or the blue team; the attack force is called the red team). The deployment cost model for this analysis is somewhat simplistic, since the deployment cost is essentially linear with each added sensor grid.

The basic attack scenario involves three red force snipers at the base of the hills near locations three and four as shown in Figure 9 (which is a screen shot from Dante), as well as

Figure 8. iSIGHT simulation execution workflow

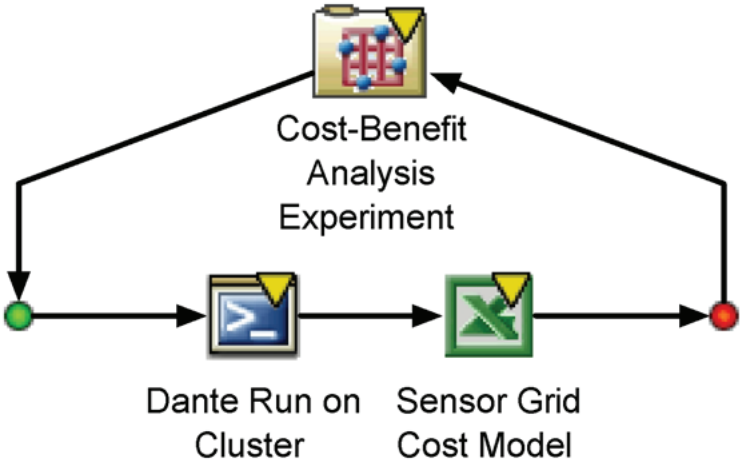
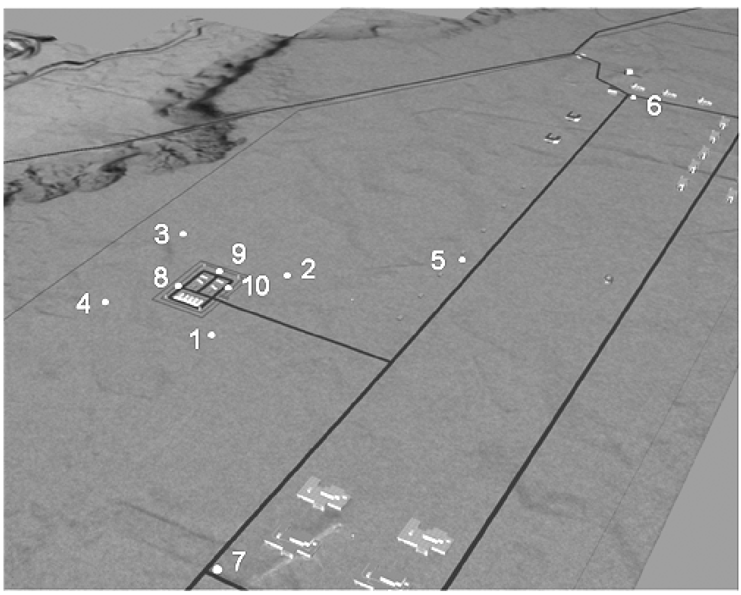


Figure 9. Numbered positions for possible sensor grid locations



a suicide convoy of four vehicles that enters the facility at the outer perimeter gate near location six and moves down to the road past location five. The convoy then moves into the inner perimeter at the gate in the fence near location ten. The objective is to gain entrance to an open bunker near location eight and take possession of an asset in the target bunker. There are pro- force personnel, vehicles and sensor grids at various locations throughout the facility that attempt to neutralize the attacking force, and to close the open bunker before the red force can penetrate the defenses.

The first key decision question involves the determination of the optimum security configuration, based on the deployment cost and the level of security provided by the sensor grids. If the cost is too high or the security inadequate, both of which are judgment calls dependent on stakeholder objectives, the analysis may have to start over and consider other changes to the system. These changes could involve adding new sensors, reconfiguring the facility, changing procedures, or making other alterations. The second key decision question is concerned with the availability of the various security subsystems. While seemingly straightforward, this issue involves a number of very thorny issues having to do with reliability, maintenance logistics, and even the potential for subversion. With this question, the analysis begins to move into the real world where the situation is not always in an optimal state. This second question begins to address enterprise-level concerns, where security may be affected by day-to-day operations. The issue is how to structure the operations of the site to ensure the maximum availability of the critical security subsystems. Again, costs play a role, because unlimited sustainment resources are not available to ensure that every piece of equipment and all personnel are always at their peak performance.

At this point the analysis is entering a phase where possible closure could be achieved; at the very least, additional data is now available which could enhance the approximations used in the first question and allow refinement of the results in the next analysis spiral. Proceeding with the final (third) question also addresses the purpose for implementing the fixed site, which is the performance of mission critical operations. The analysis of this question could be straightforward, depending on the conditions that drive the scheduling of mission-critical operations. For example, if scheduling is flexible then the optimum security could be realized if all such operations were performed immediately after maintenance is completed on the most critical elements of the security system. (This conclusion assumes that the asset or site is most vulnerable to an attack during a mission critical operation due to the exposure created by an authorized access of the asset.) However, if the mission-critical schedule is governed by external programmatic drivers, then the security risk may vary depending on the maintenance cycle. A judgment of whether the added risk is acceptable would have to be made by the stakeholder. If the risk is deemed to be too great then the analysis would have to revert back to either the prior logistics analysis, or even all the way back to the initial security configuration study. These are the tensions and tradeoffs that Figure 3 attempts to capture.

DECISION ANALYSIS PROCEDURE

The goal of the procedure that was developed to answer the first analysis question was severalfold: to determine the effect on blue team win percentage of the presence of a grid at a given location; to determine the best locations for a given number of sensor grids, from one to ten; to do a cost-benefit analysis of the resulting list of best grid locations for each number of grids; and to complete the analysis in a reasonable amount of time. Since a "brute force" full-factorial approach to determining the optimum number of grids at the best locations would be prohibitive in terms of time and computational cost, the following analysis procedure was created:

1. A baseline simulation is performed with no sensor grids in order to establish two things: the blue team win percentage in that particular scenario with no sensor grids present, and the number of runs that are necessary in order for the simulation results to converge.
2. A subject matter expert (SME) examines the terrain of the simulation scenario and selects several suitable locations for sensor grid sites.
3. A statistician creates a balanced set of runs (in which each location occurs either the same number of times or a similar number of times) to use for the initial simulation experiment.
4. The initial experiment is performed on a parallel computer, since the problem

is embarrassingly parallel. For the case study, the experiment was performed on a four-machine cluster, with four processor cores per machine.

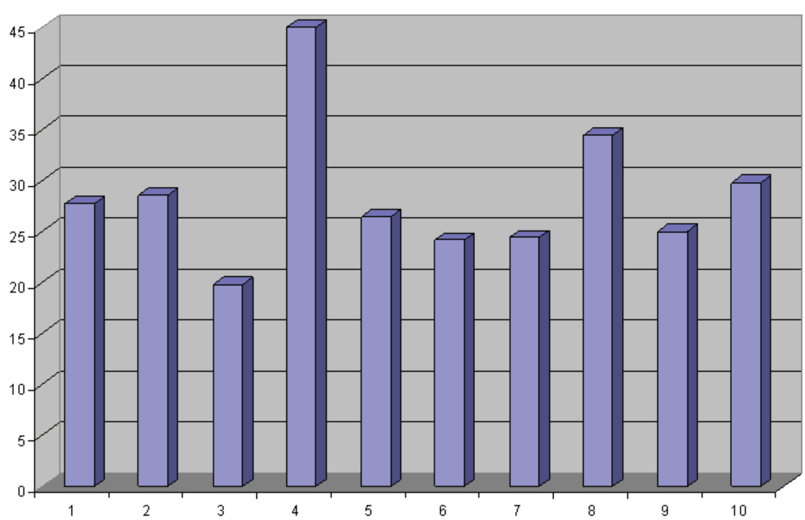
5. A statistician analyzes the results and creates a predictive model. The factors are formatted as a vector of sensor grid locations and the results analyzed based on the presence or absence of a sensor grid at a particular location.
6. The predictive model is used to generate a list of other sets of sensor grid locations to simulate in order to validate the model.
7. The validation experiment is performed on a parallel computer.
8. A statistician analyzes the results of the validation experiment to determine the extent to which the predictive model is validated.
9. A cost-benefit analysis is performed based either on the experiment results or (preferably) on the validated predictive model.

PRESENTATION OF EXPERIMENT RESULTS

Since the purpose of the analysis of the case study is to evaluate the effectiveness of the sensor grids, the analysis began by determining the baseline performance when there are no grids present. In 480 runs of the baseline configuration (no sensor grids), the blue team stopped the red team 98 times for a "Percent Blue Team Wins" of 20.4%. The results when a single grid is added at any of the ten locations are shown in Figure 10. Except for the sensor grid at location 3, each sensor grid does improve the win percentage of the blue team when compared with the baseline condition, regardless of location; however, there is obvious room for improvement in the win percentage, which can only be achieved by performing additional simulations with a larger number of sensor grids. It is important to note that as part of the simulation, sensor grids can be identified and destroyed by the red team.

However, the single sensor grid analysis represented in Figure 10 is not sufficient; answering the first analysis question requires the evaluation of a larger number of sensor grids

Figure 10. Percent blue team wins for a single sensor grid at one of ten different locations



distributed throughout the ten locations of Figure 9. While all possible two grid configurations (45 in total) can be analyzed with a full-factorial set of simulations, full factorials of larger sensor grid configurations cannot be analyzed in a reasonable amount of time. Consequently, a balanced fractional factorial design matrix for the simulations was constructed to enable completion of the analysis in a reasonable time period that would also provide sufficient information to be able to estimate the performance of configurations not actually simulated.

Cost data was also calculated using a very simple notional cost model that took several factors into account: the cost of the sensor grid; the lifetime of the sensor grid; and the annual cost of operating the sensor grid. This model was implemented in a Microsoft Excel spreadsheet. After each Dante run, iSIGHT would pass the relevant output parameters to the Excel spreadsheet to calculate the lifetime cost of the number of sensor grids used in that run.

The results of the so-called "balanced experiment" were analyzed statistically. A predictive model was generated that estimates the highest probability of blue team wins for each number of sensor grids, and suggests which configuration of sensor grid locations yields the highest probability of wins for that number of sensor grids. To validate this predictive model, several sensor grid location configurations were generated, one each for the projected best and worst combination of sensor grid locations for two through six sensor grids. An experiment was run using this set of sensor grid location configurations, and the results of this predictive model validation experiment are presented in Table 1.

STATISTICAL ANALYSIS OF EXPERIMENT RESULTS

The following section presents a statistical analysis of the results of the case study discussed above.

Logistic Model for Characterizing Probability

Logistic regression is a model used for prediction of the probability of occurrence of an event by fitting data to a logistic curve. The regression equation makes use of predictor variables that may be either numerical or categorical. In the case study, the event being modeled is the

Table 1. Results of predictive model validation experiment

Sensor Grid Locations	*A Priori* Assessment	Number of Blue Team Wins	Percentage of Blue Team Wins
1,3,4,5,9,10	Best 6	440	92
2,4,5,6,8,10	Worst 6	354	74
1,3,4,9,10	Best 5	420	88
1,2,4,7,10	Worst 5	365	76
3,4,9,10	Best 4	392	82
1,4,9,10	Best 4 without location 3	363	76
4,9,10	Best 3	303	63
3,6,7	Worst 3	121	25
4,10	Best 2	244	51
3,6	Worst 2	106	22

success of the blue team against a given attack scenario. The predictor variables under study are the availability or absence of sensor grids at specific locations.

Logistic regression is based on the logistic function given by

$$p(z) = \frac{1}{1+e^{-z}}.$$

The logistic function is useful because it can take as an input any value from negative infinity to positive infinity, whereas the output is confined to values between 0 and 1 and can therefore represent probabilities. The variable z represents the influence of some set of explanatory factors, while *p(z)* represents the probability of a particular outcome given that set of explanatory factors. The variable z is a measure of the total contribution of all the factors used in the model and is known as the logit. A different expression of the logistic function is provided by

$$\ln\left(\frac{p(z)}{1-p(z)}\right) = z,$$

which shows why the model is sometimes referred to as a linear log odds model when the logit, z, is expressed in terms of the explanatory variables. Specifically, the variable z is expanded in terms of the explanatory factors as

$$z = \beta_0+\beta_1\cdot x_1+\beta_2\cdot x_2+\ldots+ \beta_{10}\cdot x_{10}+\beta_{1,2}\cdot x_1\cdot x_2+\ldots+ \beta_{9,10}\cdot x_9\cdot x_{10}$$

In the logit equation, β_0 is called the "intercept;" β_1, β_2, β_3, and so on, are called the "regression coefficients" of x_1, x_2, x_3 respectively, where $x_i = 0$ or 1 according to whether sensor grid i ($i = 1,2\ldots,10$) is available ($x_i = 1$) to the blue team, or not available ($x_i = 0$). The combination of the x_i terms enables an adjustment of the probabilities when the availability of two sensor grids has effects that are not predicted by the additive model for the individual effects. Each of the regression coefficients describes the size of the contribution of that explanatory factor. A positive regression coefficient means that the associated factor increases the probability of the outcome, while a negative regression coefficient means that the associated factor decreases the probability of the outcome. The relative size of the coefficient reflects how strongly the probability of the outcome is influenced by the factor, while a near-zero regression coefficient means that the factor has little influence on the probability.

The logit, z, is shown in terms of only first and second order interactions. It should be clear that the representation can be expanded to include even higher order interactions. Including all higher order interactions, up to and including a single tenth order term, is equivalent to modeling the probability with 1024 parameters (ten – first order, 45 – second order, 120 – third order, etc.) and would thereby provide for perfect prediction of all probabilities associated with the possible subsets for sensor grid inclusion. The intent, however is to have a more parsimonious equation utilizing only the necessary lower order terms.

Check on Predictions

We expected that fitting 52 run conditions with a 45 parameter model would result in good fits. That this is the case is demonstrated in Figure 11, which plots the results of the predictive model against the simulation results of the 52 sensor grid location configurations. One case, the three sensor grid configuration with SensorGrid05, SensorGrid08, and SensorGrid10 being available, was over-predicted by a greater amount than all the other cases, possibly due to software hangs encountered with this particular configuration.

A further test of the prediction equation was provided by choosing another ten configurations of 2, 3, 4, 5, and 6 sensor grids in a sensor grid network, and comparing the proportion of blue team successes from the simulations to the success rate that is predicted using the model that was created by fitting the results of the original experiments (which simulated 52

Figure 11. The predicted probability from the equation fit versus the simulation probability from the runs

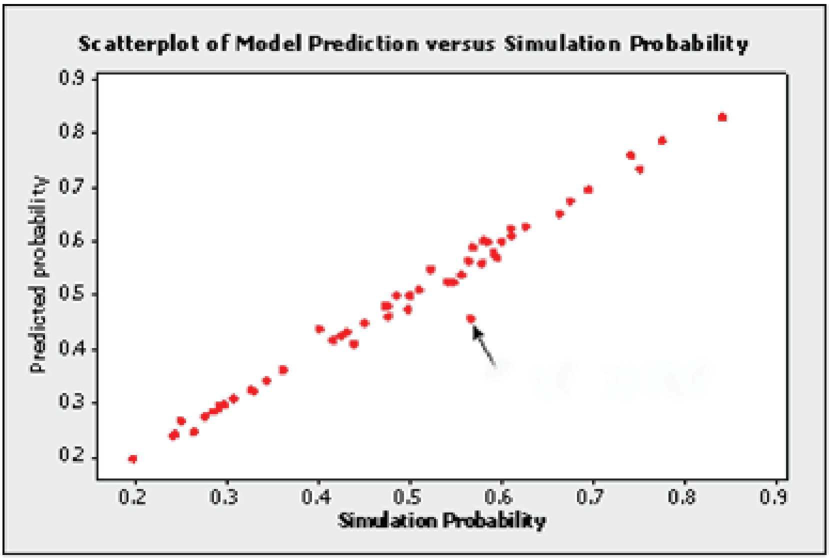
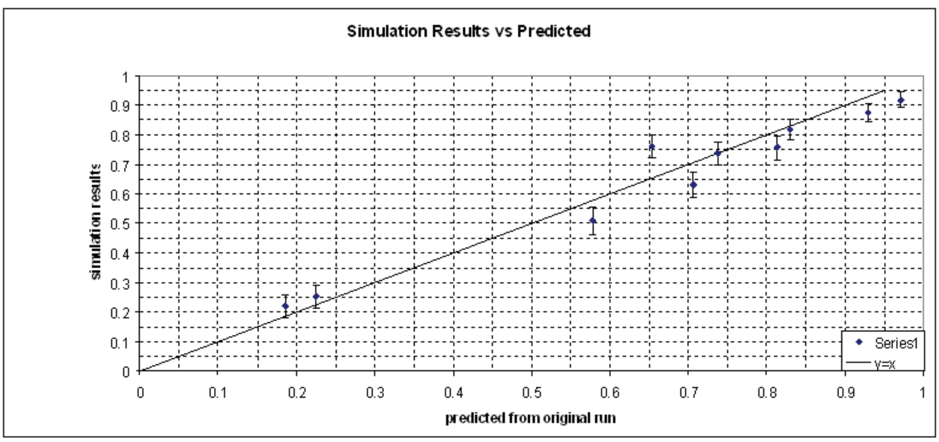


Figure 12. The simulation proportion of blue team successes versus that predicted from the equation fitted to the results of the original experiments



sensor grid location configurations). The results are shown in Figure 12, which plots the results of the predictive model against the results of the predictive model validation experiment. The error bars reflect two-sigma uncertainty for the estimate of the proportion based on 480 individual runs.

As expected, the amount of departure of the simulation probabilities compared to the predicted probabilities increases as compared to the residual differences in the data used to fit the prediction equation. However, the prediction equation does a good job of reflecting the general structure of the probabilities. The fact that the equation does a fair job with configurations of 5 and 6 sensor grids using only some of the second order interaction parameters indicates that the logistic regression model can serve as a stand-in for studying and understanding the configurations analytically, without requiring additional time consuming simulations apart from those used to create the prediction equation.

Use of Prediction Equation to Characterize Likely Outcomes

Using the prediction equation, the five configurations of four sensor grids that yield the highest probability of success for the blue team are easily determined. All five of these configurations have SensorGrid04 and SensorGrid10 present. The prediction model allows for the local sensitivity to the absence or presence of a sensor grid to be readily determined. For example, if it is known that either SensorGrid04 or SensorGrid10 (or both) have a significant chance of not being available when needed, thereby effectively reducing the number of sensor grids to three or even two, what is the effect on the probability of success for the blue team? Table 2 gives the outcomes and indicates that although the configuration of sensor grids at locations 3, 4, 9, and 10 would be expected to give the best results when four sensor grids are present, the potential of losing sensor grids 4 and/or 10 would lead one to the conclusion that a network of sensor grids at locations 1, 4, 9, and 10 is preferable.

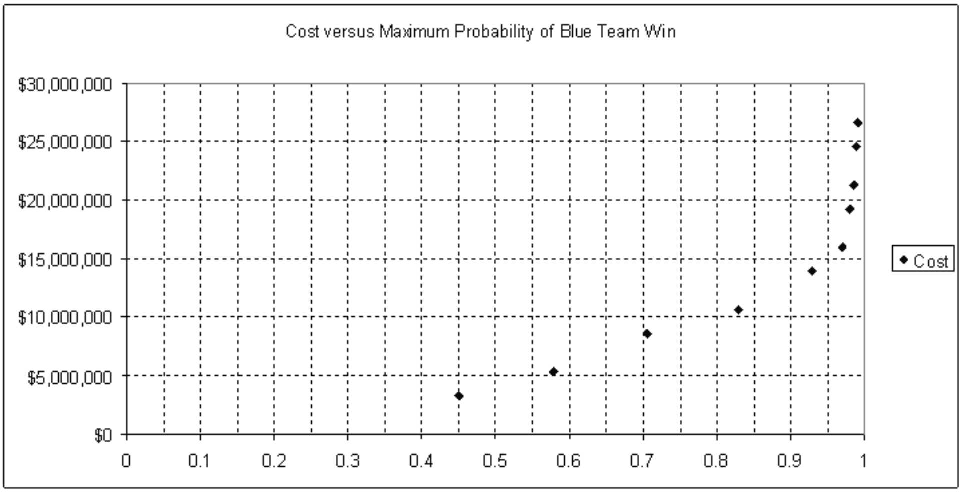
Cost-Benefit Analysis

At least two ways of approaching cost-benefit analysis are possible. The first graphs the experimental data and identifies when the return on investment from adding additional sensor grids starts to fall off. The second (preferable) way is to use the prediction equation to extrapolate beyond the experimental data and analyze the return on investment. The second approach is preferable if the predictive model is a good one. Figure 13 adopts the second approach, and shows the relationship of the costs associated with having one through ten sensor grids available versus the maximum probability of blue team success predicted for that number of sensor grids. The X axis indicates the blue team win percentage; the Y axis the lifetime cost of the sensor grids in the configuration; and the dots in the plot indicate the highest win percentage predicted for that number of sensor grids (from 1 to 10, going from left to right). The plot indicates that the incremental costs are buying a steady increase in win probability up to five, possibly six, sensor grids. At that point there is a sharp knee in the curve due to the probability getting sufficiently close to one. The added dollars are incapable of increasing the probability at the same rate beyond six sensor grids.

Table 2. Probability of Blue Team success with top five four-grid configurations with the effect of losing one or the other or both of the two most critical sensor grids

Sensor Grids included	Probability of success			
	All 4 present	Loss of SensorGrid04	Loss of SensorGrid10	Loss of both SensorGrid04 and SensorGrid10
3,4,9,10	0.829	0.548	0.548	0.274
1,4,9,10	0.813	0.585	0.620	0.347
3,4,7,10	0.807	0.398	0.615	0.239
1,3,4,10	0.795	0.500	0.584	0.310
4,7,9,10	0.793	0.431	0.603	0.272

Figure 13. Cost of having 1 through 10 sensor grids available versus the maximum win probability achievable with that number of sensor grids



FUTURE WORK

Several items of future work can be identified:

1. Complex systems, and the subset of complex systems that are truly wicked problems, simply cannot be understood using a single analytical approach or simulation tool. Instead, multiple analytical approaches and associated simulation tools should be used to see which ones give the best insight into the essence (or certain aspects) of the problem. It is for this reason that we call DTA an "analysis framework" instead of an "analysis methodology," because DTA is broad enough to encompass a number of analytical tools. The particular methodology presented above (UML/SysML modeling and decomposition followed by DSM dependency reordering) is simply one possible collection of analytical tools within the overall DTA framework; it was chosen because of its amenability to the particular problem used for the case study. Additional tools and analytical approaches could be integrated into the DTA framework, depending on the characteristics of the system under study.
2. The DTA should be applied to complex systems and wicked problems in other domains in order to validate and refine the framework methodology. Two such domains have been identified as suitable candidates—critical infrastructure protection and information assurance.
3. Additional work in the "gap analysis" aspect of DTA is needed, in order to refine the procedure for determining when the development of new simulation tools is required, and exactly what functionality those new tools should provide.
4. For the case study, simulation tools that perform site preparation and logistics and maintenance should be identified and integrated into the execution workflow.
5. A weakness of the case study experiments is that only one adversary attack scenario is considered. For a sensor grid location recommendation to have any validity, the analysis should consider a complete ensemble of attack scenarios, from most probable to most potentially devastating. A methodology for developing such attack

scenarios is needed, as well as a way to weight the sensor grid location recommendations for each attack scenario, in order to provide a robust set of sensor grid location recommendations.
6. With regard to the case study, other potential applications of the approach to the evaluation of surety systems include the prevention of vandalism against public property, the control of borders to prevent unauthorized entry, and the protection of secure domestic facilities.
7. In the simulations performed for the case study, the cause of the software hangs when a small set of particular sensor grid configurations were simulated, which prevented a full set of simulation iterations from being executed for those configurations, should be determined and fixed.

SUMMARY AND CONCLUSION

In the context of the case study presented, the DTA framework allowed the complex problem to be decomposed, key decision analysis questions to be identified, the task interdependencies to be disentangled and clustered, and a simulation analysis workflow to be constructed. The results of the simulation workflow were statistically analyzed to answer the question of the optimum number and placement of sensor grids to protect a fixed-site. In short, the use of the DTA allowed the key decision analysis question that was the focus of the case study (which was a cost-benefit optimization question) to be successfully answered.

The DTA methodology that we propose in this article is obviously not the only way that wicked problems can and should be addressed. However, the DTA is a structured, repeatable, and defensible analysis option that holds the promise of resolving the unsolvable, especially by untangling interdependencies. As mentioned in the introduction, wicked problems tend to be heavily influenced by policy decisions. Therefore when developing the use case description of the system, policy constraints should be included as part of the stakeholder objectives and goals, or at least the results of the analysis should be vetted against the policy constraints.

As an analysis framework, DTA is particularly good at simplifying the interdependencies characteristic of complex systems and wicked problems. The authors believe that the transition between a SysML decomposition and a DSM dependency partitioning, and between that DSM partitioning and a simulation execution workflow, is a novel contribution of the work presented in this article. The application of DTA to sensor grid placement decisions in order to secure a fixed site is also valuable, and can be extended to several domains.

ACKNOWLEDGMENT

Sandia National Laboratories is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.

REFERENCES

(2001). Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions. *IEEE Transactions on Engineering Management 48*(3), 292-306.

(2003). Guest Editorial: Special Issue on Evolutionary Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation, 7*(2), 97-99.

Anderson, W., Boxer, P., & Brownsword, L. (2006). *An Examination of a Structural Modeling Risk Probe Technique*. Software Engineering Institute/Carnegie Mellon Special Report CMU/SEI-2006-SR-017, Pittsburgh, PA.

Balmelli, L. (2007). An Overview of the Systems Modeling Language for Products and Systems Development. *Journal of Object Technology, 6*(6), 149-177.

Bar-Yam, Y. (2004). *Making Things Work*. Cambridge, MA: NECSI/Knowledge Press.

Baranger, M. (2001). *Chaos, Complexity, and Entropy*: *A Physics Talk for Non-Physicists*. Wesleyan University Physics Department Colloquium. Retrieved 9 September 2008 from http://necsi.org/projects/baranger/cce.pdf.

Barton, J., & Haslett, T. (2007). Analysis, Synthesis, Systems Thinking and the Scientific Method: Rediscovering the Importance of Open Systems. *Systems Research and Behavioral Science, 24*, 143-155.

Browning, T. R. (2000). Using the Design Structure Matrix (DSM) for Process Integration. *Proceedings of the 34th Annual Government Electronics and Information Technology Association (GEIA) Engineering & Technical Management Conference*, 25-29 September, Dallas, TX, 131-140.

Cardin, M-A., Nuttall, W. J., de Neufville, R., & Dahlgren, J. (2007). Extracting Value from Uncertainty: A Methodology for Engineering Systems Design. *Proceedings of the 17th Annual International Symposium of the International Council on Systems Engineering (INCOSE)*, (pp. 25-28). San Diego, CA.

Cloutier, R. J. (2006). *Applicability of Patterns to Architecting Complex Systems*. PhD Dissertation, Department of Systems Engineering & Engineering Management, Stevens Institute of Technology, Hoboken, NJ.

Coello, C. A. (2000). An Updated Survey of GA-Based Multiobjective Optimization Techniques. *ACM Computing Surveys, 32*(2), 109-143.

Coello, C. A. Van Veldhuizen, D. A., & Lamont, G. B. (2002). *Evolutionary Algorithms for Solving Multi-Objective Problems*. New York: Kluwer Academic Publishers.

Conklin, J. (2006). *Dialogue Mapping: Building Shared Understanding of Wicked Problems*. Chichester, West Sussex, England: John Wiley & Sons.

Doyle, L. (2007). *Cognitive Theory of System Models*. PhD Dissertation, Systems Engineering and Engineering Management Department, Stevens Institute of Technology, Hoboken, NJ.

Eppinger, S. D., Whitney, D. E., Smith, R. P., & Gebala, D. A. (1994). A model-based method for organizing tasks in product development. *Research in Engineering Design, 6*(1), 1–13.

Guenov, M. D., & Barker, S. G. (2005). Application of Axiomatic Design and Design Structure Matrix to the Decomposition of Engineering Systems. *Systems Engineering, 8*(1), 29-40.

Hodge, N., & Weinberger, S. (2005). *A Nuclear Family Vacation: Wicked Things at Sandia*. Posted 14 July 2005 on Slate.com. Retrieved 14 May 2008 from http://www.slate.com/id/2122382/entry/2122434/.

Horn, R. E., & Weber, R. P. (2007). *New Tools for Resolving Wicked Problems: Mess Mapping and Resolution Mapping Processes*. MacroVU, Inc., and Strategy Kinetics LLC. Retrieved 4 June 2008 from http://www.strategykinetics.com/New_Tools_For_Resolving_Wicked_Problems.pdf.

Johnson-Laird, P. N. (1983). *Mental Models: Towards a Cognitive Science of Language, Inference, and Consciousness*. Cambridge, MA: Harvard University Press.

Maheswari, J. U., & Varghese, K.. (2004). Project Scheduling using Dependency Structure Matrix. *International Journal of Project Management, 23*(3), 223-230.

Marler, R. T., & Arora, J. S. (2004). Survey of Multi-Objective Optimization Methods for Engineering. *Structural and Multidisciplinary Optimization, 26*(6), 369-395.

Pimmler, T. U., & Eppinger, S. D. (1994). Integration Analysis of Product Decompositions. *Proceedings of the ASME Sixth International Conference on Design Theory and Methodology*, 11-14 September, Minneapolis, MN.

Rittel, H. W. J., & Webber, M. M. (1973). Dilemmas in a General Theory of Planning. *Policy Sciences, 4*, 155-169.

Smith, R. P., & Eppinger, S. D. (1997). Identifying Controlling Features of Engineering Design Iteration. *Management Science, 43*(3), 276-293.

Steward, D. V. (1981). The Design Structure System: A Method for Managing the Design of Complex Systems. *IEEE Transactions on Engineering Management, EM-28.3*, 71-74.

Suh, N. P.(2001). *Axiomatic Design: Advances and Applications*. New York: Oxford University Press.

Woodman, K. L., & Bilardo Jr., V. J. (2005). *Applying the Design Structure Matrix (DSM) Technique to NASA Organizational Design*. Presentation at the

NASA APPL 2nd Project Management Challenge Conference, March, NASA Langley Research Center, Hampton, VA.

Yassine, A. (2004). *An Introduction to Modeling and Analyzing Complex Product Development Processes Using the Design Structure Matrix (DSM) Method*. Retrieved 14 May 2008 from http://www.iese.uiuc.edu/pdlab/Papers/DSM-Tutorial.pdf.

John M. Linebarger is a principal member of technical staff at Sandia National Laboratories. He has over 25 years of experience in the areas of distributed systems, collaborative virtual environments, semantic technologies, cognitive systems, and modeling and simulation. He has an MBA from New York University, an MS in computer science from the University of New Mexico, and a PhD in computer science from Lehigh University. Linebarger is a member of IEEE and the ACM.

Mark J. De Spain has been an engineer at Sandia National Laboratories for over twenty years. He has worked in weapons components including sensing devices, firing sets and use control. Currently he is working as a weapon systems engineer. De Spain has a BSME from Oregon State University and an MSEE from the University of Portland. He is currently enrolled in the doctoral systems engineering program at Stevens Institute of Technology, and is the current president of the INCOSE Enchantment Chapter in New Mexico.

Michael J. McDonald is a principal member of technical staff at Sandia National Laboratories. He is recognized as an expert in modeling and simulation-based systems engineering. His work focus is on science-based systems engineering and analysis, which uses design theory, computer science, mechanical engineering, and modeling and simulation to develop new technologies for security, networking, military systems, robotics, human systems integration, and manufacturing applications. McDonald's current work assignment is to develop new effects-analysis techniques for Sandia's Information Systems Analysis Center.

Floyd W. Spencer retired as a distinguished member of technical staff at Sandia National Laboratories after thirty years of experience in statistical analysis. He received his PhD and MS degrees in applied probability and statistics from Cornell University and a BS in mathematics from Harvey Mudd College.

Robert Cloutier is a research associate professor in the School of Systems and Enterprises at Stevens Institute of Technology. He has over 20 years experience in systems engineering and architecting, software engineering, and project management in both commercial and defense industries. His interests include systems engineering patterns, systems architecting, MBSE, SysML, and architecture management. Cloutier belongs to the International Council on Systems Engineering (INCOSE), IEEE and ACM. He received his PhD in systems engineering from Stevens Institute of Technology, an MBA from Eastern University, and a BS from the United States Naval Academy.