The Application of Systems Engineering to Software Development: A Case Study

Robert L. Sweeney, Jeffrey P. Hamman, and Steven M. Biemer

The U.S. Navy's Assessment Division (N81) integrates and prioritizes warfighting capability within resource constraints by using a joint campaign model to represent "what it takes to win" in the complex arena of multiservice regional conflict. N81 commissioned an assessment in the spring of 2006 to determine the feasibility and affordability of adding a maritime capability to the Synthetic Theater Operations Research Model (STORM) to make it an acceptable campaign model for the U.S. Navy staff. The result of this assessment was a partnership between the U.S. Navy's N81 and the U.S. Air Force Air Staff's Studies and Analyses Directorate (A9), under the project name "STORM+." Replacing a legacy campaign model has broad impact on future investment decisions and can attract a wide range of stakeholders with an even wider range of requirements. This article describes how an APL team partnered with both the sponsor and the developer to implement systems engineering concepts to ensure a successful replacement of the existing deterministic U.S. Navy campaign model with a stochastic model created by adding a maritime warfare capability to STORM. The results indicate systems engineering can be successfully applied to a large, complex software development effort as long as the cultures of both the sponsor and the developer are appreciated and accommodated.

THE PURPOSE OF STORM+

The landscape of model development is littered with the abandoned remains of needed analysis tools. They lie discarded, rather than in use, often because they could not achieve a useful set of capabilities constrained sufficiently to maintain a manageable data input load and an acceptably short run time. Such an outcome is generally the result of too many stakeholders with disparate analysis problems and varying granularity being unwilling to compromise for a greater "good enough." They hold out for a sum of all requirements, which often results in a terminated program that fails to meet any requirements at all. Systems engineering provides a disciplined, structured process to keep the effort focused on this greater "good enough"—the minimal set of requirements necessary and sufficient to meet a well-defined, feasible need.[1–4] This is the story of the challenges in getting participants to accept systems engineering even when the sponsor has decreed its use. The story describes a collaboration of different professional cultures that share the community of military modeling and simulation (M&S) but, like most good stories, leads to a happy ending that more than vindicates the trials and tribulations, not to mention the investment of resources.

The sponsor for the STORM+ project was the U.S. Navy's Assessment Division in the Office of the Chief of Naval Operations (OPNAV), known by their office code as N81. In the world of U.S. Navy resource requirements, N81 is OPNAV's "honest broker" and integrator. It produces capability analyses for both warfighting and warfighting support, integrating and prioritizing capabilities within resource constraints and balancing inputs from strong constituencies to recommend a broad, affordable program to ensure that the U.S. Navy can meet its role as defined by the *National Military Strategy*.[5]

N81 analyses take advantage of M&S throughout the modeling hierarchy pyramid depicted in Fig. 1. The ultimate determination is how the program contributes to the joint (all armed services) campaign arena, evaluating "what it takes to win" and the "so what?" of any capability analysis or analysis of alternatives. How a system or concept performs in a campaign with the scope of a Desert Storm or Iraqi Freedom with multiple, competing missions and capabilities from all of the U.S. armed services and against the most capable projected adversary is often the final discriminator for senior decision makers.

Since the early 1990s, N81's campaign model has been the Integrated Theater Engagement Model (ITEM), consisting of air, land, and naval warfare modules that permit realistic representations of capabilities from all armed services in a common computer environment by using a deterministic method to represent uncertainty in outcomes. This deterministic approach can allow greater fidelity of object attributes than a stochastic approach, which requires a large number of model runs and therefore longer run times to generate a distribution of possible outcomes. With the rapid increase of computing power and speed over the last 10 years, run time for stochastic models has become less of a consideration. Stochastic models are gaining broad acceptance for their ability to provide the analyst a solution space rather than a point solution based on an assumed probability. Identifying an area of uncertainty around outcomes increases the credibility of analysis and gives decision makers more contexts with which to make a decision.

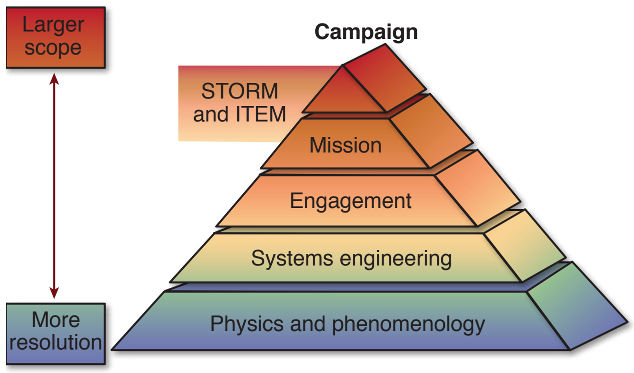


Figure 1. Modeling pyramid, showing the location of the U.S. Air Force's STORM and the U.S. Navy's ITEM in the modeling hierarchy.

When N81 began looking for a stochastic model to replace ITEM, the U.S. Air Force's Synthetic Theater Operations Research Model (STORM) was an attractive choice. First implemented in 2004, STORM is a stochastic, discrete-event, data-driven simulation written in the C++ programming language. STORM is based on an architecture called the Common Analytic Simulation Architecture (CASA), which is designed to reduce development time and life-cycle costs for analytic simulations, while minimizing dependencies between software modules. STORM has an active and ongoing development effort managed by the U.S. Air Force Air Staff's Studies and Analyses Directorate (A9) as well as a growing users' community that includes other U.S. services, the Office of the Secretary of Defense, the staff of the Chairman of the Joint Chiefs of Staff, and international participation. It brings a variety of highly desirable attributes, such as an open architecture, the use of industry standards, low program and ownership costs, and high adaptability because of its data-driven format. N81 commissioned an assessment in the spring of 2006 to determine the feasibility and affordability of adding a maritime capability to STORM to make it an acceptable campaign model for the U.S. Navy staff. The result of this assessment was a partnership between N81 and A9 under the project name "STORM+." Tapping into the convergence of a campaign model that had been embraced by the U.S. Air Force, was used within the Office of the Secretary of Defense, and had captured the interest of the U.S. Marine Corps would bring broad understanding and credibility to future U.S. Navy studies.

Campaign models must be capable of joint force structure analysis, strategy assessments, and operational planning while providing metrics useful to decision makers. Given the range of military activities that must be modeled, the diversity of mathematical algorithms contained within, and the interrelated trade-offs of attributes such as run speed, granularity, and ease of use, there can be little doubt that a campaign model is a complex system.

The development of a complex system requires systems engineering throughout its life cycle, from requirements generation through functional definition, development, integration and testing, deployment, and operations and support. Each user (modeler) and stakeholder (user of the model's output) is an advocate for certain model capabilities and attributes. Some capabilities and attributes are true requirements to achieve the overall objective, but others are only "nice-to-haves" that, although desirable to some or even all of the user/stakeholder community, bring unnecessary risk to performance, cost, and schedule.

The U.S. Navy recognized that systems engineering could guide the development process, keeping it headed efficiently toward the objective while coordinating the various disciplines represented by the stakeholders, including M&S code writers, analysts, and decision makers. Inherent to systems engineering also would be a testing plan to verify that requirements were met and, most importantly, to continually inform the sponsor on the level of risk and to recommend courses of action when the risk became too high. Embarking on a multiyear, multimillion-dollar model-development effort to add maritime capability to STORM without a systems engineer to monitor, assess, and report would have invited, at a minimum, schedule delays and cost overruns and ultimately could have led to a final product that did not meet the need originally identified as the reason for replacing the legacy model.

SYSTEMS ENGINEERING PLAN

Systems engineering is built on the principle of maintaining a total system perspective throughout the development of a complex system, resolving design decisions by using the highest context available. This principle requires the systems engineer to continually focus on five activities throughout the development life cycle:

1. Formulating and refining operational, functional, and performance requirements
2. Identifying and decomposing the system's functionality
3. Implementing that functionality into a feasible, useful product
4. Verifying the system's requirements, functionality, and implementation
5. Managing inherent operational, technical, and programmatic risks

Whether in the first or the final stage of development, the performance of these five activities drives design decisions and leads to a structured approach. However, applying a structured approach to software development has always presented a challenge. Creativity, innovation, and rapid response are hallmarks of modern software engineering; this was the case with the STORM+ program.

The STORM+ developers used a form of agile software development, an iterative life cycle model that quickly produces prototypes that the user and developer can evaluate to refine requirements and design. It is especially well suited to small- to medium-sized projects for which requirements are not firmly defined and where the sponsor is willing to work closely with the developer to achieve a successful product. The agile methodology depends on this close sponsor–developer engagement.

As defined by its proponents, the agile methodology is based on the following postulates:

- Requirements (in many projects) are not wholly predictable and will change during the development period. A corollary is that sponsor priorities are likely to change during the same period.
- Design and construction should be integrated because the validity of the design can seldom be judged before the implementation is tested.
- Analysis, design, construction, and testing are not predictable and cannot be planned with adequate levels of precision.

Agile development relies heavily on the software development team conducting simultaneous activities. Formal requirements analysis and design are not separate steps; they are incorporated in the coding and testing of software. In this approach, quality and robustness are evolved attributes of the product. Thus, the iterations are to be built upon, rather than thrown away (see chapters 20–24 in Ref. 4).

Although agile development works well, it is difficult to reconcile with traditional systems engineering methods. Furthermore, the STORM development history has used a series of spiral releases, scheduled approximately every 6 months. It was important that introducing systems engineering into the development of a maritime capability not break this cycle.

To conform to this 6-month periodicity, the STORM+ development effort was initially divided into a set of separate 6-month periods, with five spiral releases (numbered 1 through 5) that culminated in a formal release of STORM v2.0. Each spiral would include additional functionality over the previous one; however, the level of functionality would not be linear. In fact, the first spiral would not involve a software release at all but rather a set of design documents focusing on the model infrastructure necessary to implement maritime capability. The first four spiral releases would be used to measure and evaluate the progress against the maritime requirements. Before the spiral development, however, two early phases would be completed: requirements analysis and conceptual design. Figure 2 depicts this schedule.

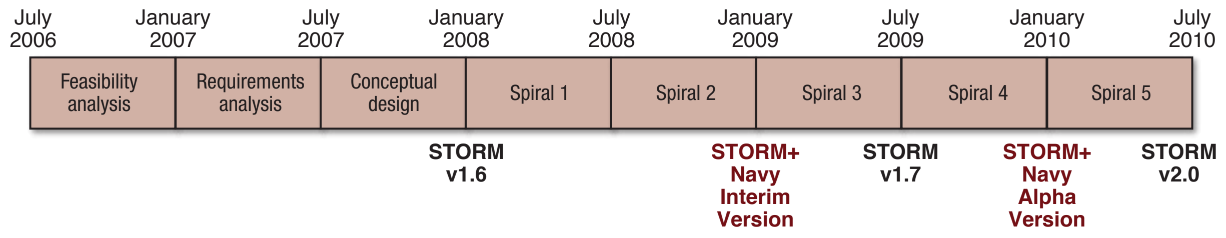


Figure 2. The schedule for the STORM+ development effort.

The version of STORM available to users at the beginning of this effort was v1.6. Two additional versions would become available to general users: v1.7, having limited maritime capability, and v2.0, having full maritime capability.

The plan that evolved had to support agile development used by the developer while maintaining the critical aspects of systems engineering. Figure 3 depicts the process that integrates the two. Overlaid in maroon on Fig. 3 are the names of the eight integrated product teams (IPTs) listed next to the activities for which they were responsible. As indicated, the Management, Systems Engineering, Risk Management, and Data Integrity IPTs had responsibilities throughout the process. Although the IPTs had distinct, clearly defined responsibilities, their membership was drawn from all participating organizations, with some people serving on more than one IPT. This mixed membership, along with biweekly teleconferences that included representation from all IPTs, kept "stovepipes" from forming that could lead to inefficiencies, miscommunications, and other problems.

Requirements Analysis

Requirements analysis is a critical component of systems engineering and was at the heart of the STORM+ development effort. The requirements focused on what naval assets (along with their attributes) and what processes would be in STORM+. Initially identifying requirements was not the challenging activity, because they came from the capabilities of the current ITEM. The challenge was scoping this initial set of ITEM capabilities into a manageable and consistent set of requirements. The goal was to establish a stable set of requirements early. Constrained by an ambitious schedule and a budget, the requirements were not allowed to creep beyond the goal of "ITEM-like" capabilities, but they did evolve to enhance clarity.

Functional Analysis

Although the requirements define the assets and processes (the "what") in the model, they do not define how they are implemented (the "how"). Functional analysis focused on the specifics of the naval asset interactions, processes, and information architecture necessary to implement the requirements.

Design, Development, and Unit Testing

The developer followed their agile approach for design and development. Before each spiral development effort, a general road map was published for review and feedback. Once the road maps were understood and agreed upon, a series of module design documents were developed as the design and development progressed. These documents described the general design for each portion of the model addressed in the spiral. Finally, code was engineered, followed by unit testing on each module, which was performed by the developer.

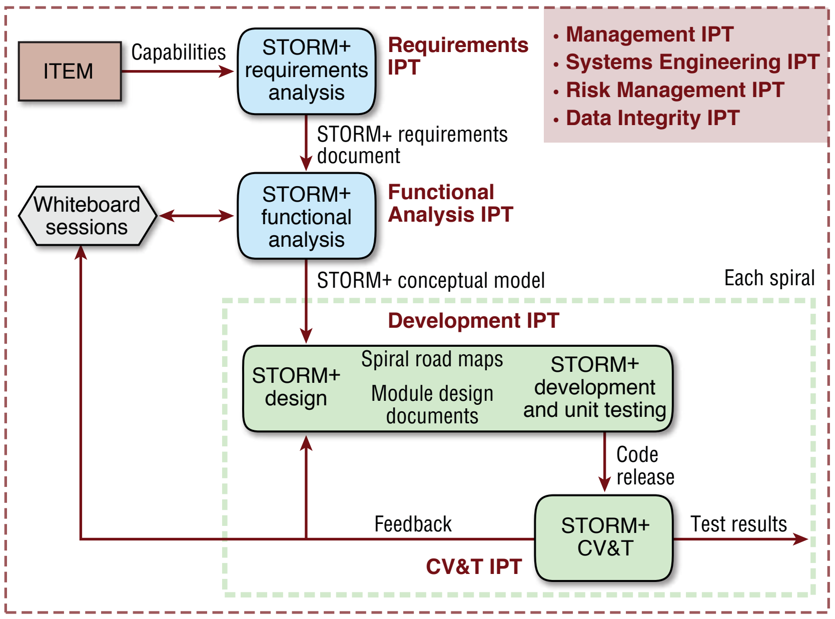


Figure 3. The STORM+ systems engineering process, which incorporated agile development while maintaining critical aspects of systems engineering. Names of the eight IPTs are shown in maroon next to the activities for which they were responsible.

Concurrent Verification and Testing

Concurrent verification and testing (CV&T) involved the planning, execution, and reporting of the STORM+ spiral testing. Activities that were performed included (*i*) verifying the mapping of requirements to conceptual model to development products; (*ii*) assessing developer unit testing; (*iii*) verifying and testing requirement representation, initialization data, and hardware/software integration; and (*iv*) documenting and reporting activities.

As the lead systems engineer, APL was responsible for ensuring that this process was followed as well as identifying and mitigating risks throughout the program. Accomplishing this responsibility meant significant involvement in three of the four primary activities depicted in Fig. 3—requirements analysis, functional analysis, and CV&T—in addition to risk management.

The next section describes the actions performed within these three activities and their outcomes that led to the successful release of STORM v2.0. At the end of the section is a discussion of risk management.

SYSTEMS ENGINEERING CONTRIBUTIONS

Requirements Development and Analysis

The STORM+ maritime requirements development process began during an initial assessment of the ability of the U.S. Air Force's STORM to support OPNAV campaign analysis. The team of subject-matter experts (SMEs) that N81 convened in May 2006 identified broad maritime capabilities that would need to be added to STORM to achieve an analysis capability comparable to ITEM. "ITEM-like" capabilities would be a combination of those demonstrated in STORM plus those implemented in a STORM modification.

The Requirements IPT was responsible for identifying, analyzing, and articulating the model requirements. They surveyed ITEM users and campaign analysts for model requirements necessary to provide maritime capabilities in STORM+. Initially, more than 871 user needs were identified and submitted for requirements analysis.

Proposed requirements went through several systems engineering process steps (depicted in Fig. 4) before being accepted as valid STORM+ requirements. First, user needs were categorized into maritime capability categories (e.g., anti-submarine warfare, sea-based air and missile defense, etc.). Second, a requirements review was completed that identified and deleted duplicate and nonspecific user needs. And third, each requirement was assessed to determine whether it was represented in ITEM and whether it could be represented in STORM.

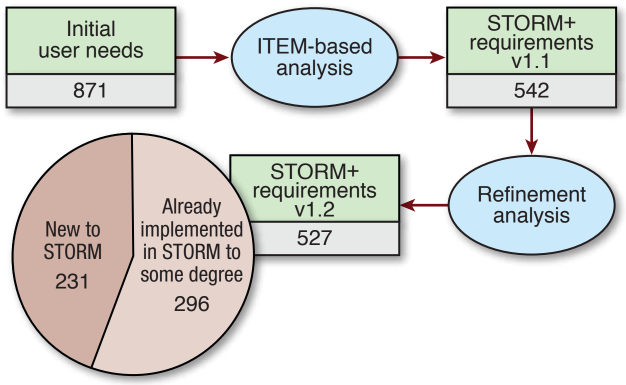


Figure 4. The requirements analysis process.

If a requirement was represented in ITEM, it was designated as "In-ITEM." If it was not, the requirement was designated as "Out-ITEM." If the requirement was already represented in STORM, even if it was not completely met, it was designated as "In-STORM." Finally, if the requirement was not represented in STORM, it was designed as "Out-STORM."

Thus, the requirements were divided into four categories:

1. In-ITEM/In-STORM
2. In-ITEM/Out-STORM
3. Out-ITEM/In-STORM
4. Out-ITEM/Out-STORM

These categories were used to prioritize the STORM+ requirements. Table 1 shows the resulting matrix for the STORM+ requirements and their priorities.

This requirements analysis process produced 542 STORM+ requirements. After further assessment and review by OPNAV and the U.S. Air Force, the number of STORM+ requirements was reduced to 527.

The final product of this process was a STORM+ requirements document. Both the sponsor (N81) and the model manager (A9) signed off on the STORM+ requirements document—N81 from the perspective of what was needed and A9 from the perspective of feasibility. This final set of requirements was split into two categories: The first category represented 231 requirements that the existing STORM could not support and needed to be added as "unique" development efforts; the second category represented 296 requirements that

Table 1. Matrix of STORM+ requirements and their priorities.

ITEM Capabilities	STORM Capabilities	
	In-STORM	Out-STORM
In-ITEM	STORM+ requirement: priority 1	STORM+ requirement: priority 1
Out-ITEM	STORM+ requirement: priority 2	Not a STORM+ requirement; deferred

STORM could support to some degree (those designated In-STORM). The 231 unique STORM+ requirements were provided to the STORM+ Functional Analysis IPT to develop a STORM+ conceptual model for use by the STORM+ developer. The In-STORM requirements were provided to the CV&T IPT to determine whether the STORM implementation was sufficient to fully meet the requirement.

Functional Analysis Development

It is within the functional analysis development that the first integration of systems engineering and agile development occurred. A set of requirements is not sufficient to start software development. The Functional Analysis IPT was responsible for developing and maintaining a STORM+ conceptual model. The conceptual model was a document that described how the unique STORM+ requirements should be implemented, from a real-world operator's perspective rather than from a modeler's perspective. The conceptual model's purpose was to

- Document traceability between requirements and desired capabilities
- Provide a basis for preliminary design and other planning activities by the developers
- Support the associated CV&T activity in relating development plans to requirements

Development of the conceptual model was a collaboration between analysts who currently perform campaign analysis for the U.S. Navy and the developers of STORM. This collaborative approach was designed to accommodate a compressed development timeline. It also ensured that the resulting STORM+ model was responsive to U.S. Navy analysis needs while retaining its architectural integrity, the coherence of its methodology, and the analysis capability on which users of STORM had come to rely. The conceptual model also helped improve the consistency of the analysts' interpretation of model capability before the developer initiated code development.

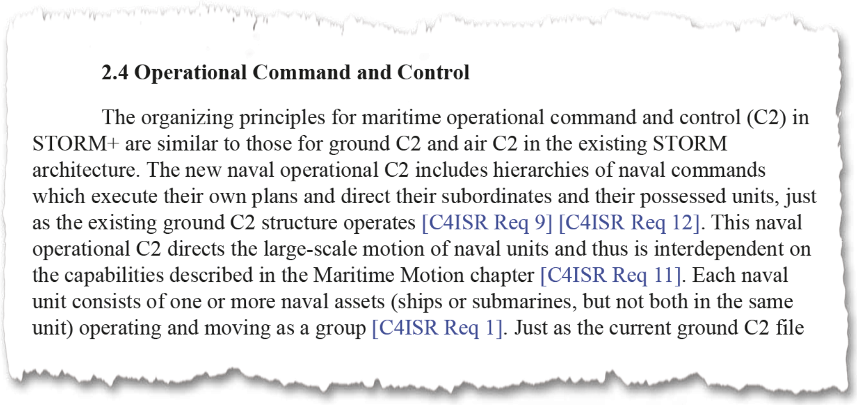
The process for creating the conceptual model was to convene multiple whiteboard sessions comprising selected SMEs from the Functional Analysis, Requirements, and CV&T IPTs as well as the STORM+ developer. These whiteboard sessions were designed to allow the SMEs to interact freely with developers while discussing model functionality to meet a given set of requirements. After each whiteboard session, a Functional Analysis IPT member was assigned to develop a conceptual model description based on the discussions. These conceptual model descriptions were combined into an integrated STORM+ conceptual model that described an agreed-upon method of implementing maritime requirements into the existing STORM software. Figures 5 and 6 show examples of conceptual model products on command and control (C2) that were developed during a whiteboard session.

Although many of the conceptual model sections bore titles related to STORM+ requirement capabilities (anti-air warfare, anti-submarine warfare, etc.), there were also sections that described model functionality, such as "Maritime Motion," "Maritime Sensing," and "Maritime Prosecution, Engagement, and Damage." Each section of the conceptual model referenced the unique and, if needed, the In-STORM requirements for traceability.

The integration of developers with IPT SMEs in the whiteboard sessions was beneficial in ensuring that the functionality described did not adversely impact the current STORM architecture. Where possible, existing model architecture and functionality for ground and air forces was reused for maritime functionality. If the existing functionality did not support a required maritime function, then the whiteboard session members developed and mutually agreed on a solution.

Concurrent Verification and Testing

CV&T was divided into three phases. Phase I was responsible for testing the naval capabilities that were already in the existing STORM. Phase II was responsible for testing functionality within the U.S. Navy Interim release (spirals 1 and 2). And Phase III was responsible for testing the U.S. Navy Alpha version (spirals 3 and 4). Spiral 5 was released after maritime capability was fully integrated into STORM and therefore was tested under the existing procedures for STORM. Each testing phase followed a similar process, shown in Fig. 7. After each spiral test, results were fed back, in case revisions to



2.4 Operational Command and Control

The organizing principles for maritime operational command and control (C2) in STORM+ are similar to those for ground C2 and air C2 in the existing STORM architecture. The new naval operational C2 includes hierarchies of naval commands which execute their own plans and direct their subordinates and their possessed units, just as the existing ground C2 structure operates [C4ISR Req 9] [C4ISR Req 12]. This naval operational C2 directs the large-scale motion of naval units and thus is interdependent on the capabilities described in the Maritime Motion chapter [C4ISR Req 11]. Each naval unit consists of one or more naval assets (ships or submarines, but not both in the same unit) operating and moving as a group [C4ISR Req 1]. Just as the current ground C2 file

Figure 5. Sample of a C2 product description in the STORM+ conceptual model.

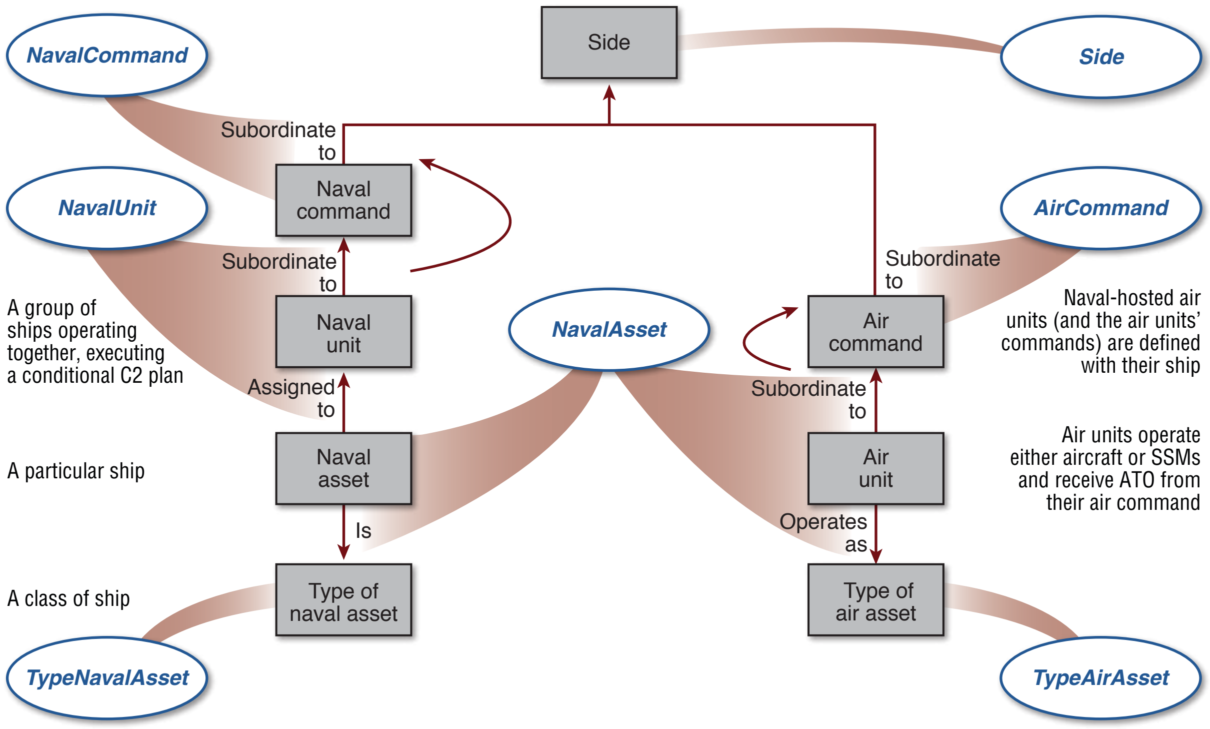


Figure 6. Sample implementation of the C2 product of Fig. 5 in the conceptual model. The sample shows naval unit organizational relationships. ATO, air tasking order; SSMs, surface-to-surface missiles.

the development effort and the conceptual design (the Functional Analysis IPT) were needed.

Testers were selected from organizations that support OPNAV N81 analysis and have extensive experience in campaign modeling. They were challenged to develop test cases that grouped and sufficiently investigated each STORM+ requirement and conceptual model functionality.

Each test case described not only the requirements to be tested but also a scenario, including technical,

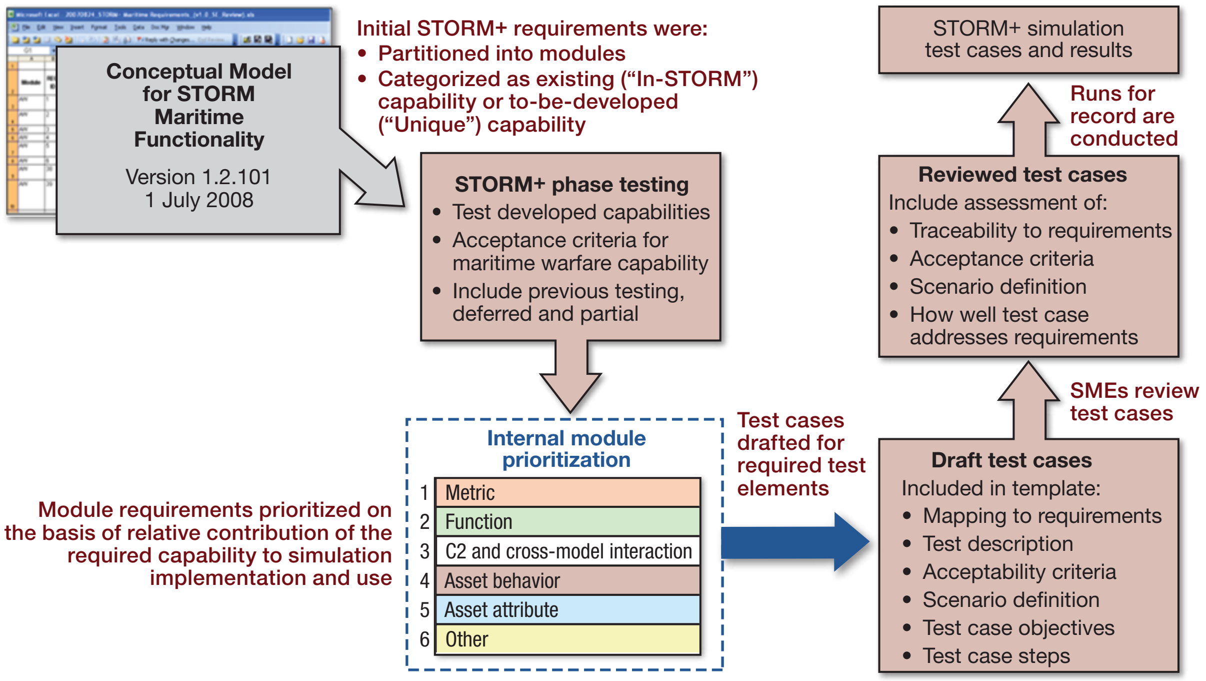

Conceptual Model for STORM Maritime Functionality
Version 1.2.101
1 July 2008
Initial STORM+ requirements were:
• Partitioned into modules
• Categorized as existing ("In-STORM") capability or to-be-developed ("Unique") capability
STORM+ simulation test cases and results
Runs for record are conducted
STORM+ phase testing
• Test developed capabilities
• Acceptance criteria for maritime warfare capability
• Include previous testing, deferred and partial
Reviewed test cases
Include assessment of:
• Traceability to requirements
• Acceptance criteria
• Scenario definition
• How well test case addresses requirements
SMEs review test cases
Internal module prioritization
1 Metric
2 Function
3 C2 and cross-model interaction
4 Asset behavior
5 Asset attribute
6 Other
Test cases drafted for required test elements
Module requirements prioritized on the basis of relative contribution of the required capability to simulation implementation and use
Draft test cases
Included in template:
• Mapping to requirements
• Test description
• Acceptability criteria
• Scenario definition
• Test case objectives
• Test case steps

Figure 7. Defining and scoping the spiral testing process.

operational, and environmental data as well as acceptability criteria. The Data Integrity IPT was responsible for identifying data transformation algorithms needed to convert existing scenario databases within the U.S. Navy to STORM+ data files and for developing specific test databases that met the needs of individual test cases. The test cases were subsequently reviewed and accepted by OPNAV SMEs before use.

Each test case explicitly defined acceptance criteria that established the measures against which to judge the appropriateness of a simulation for an intended use. These criteria were developed by testers and reviewed by the Requirements and Functional Analysis IPTs as needed. Some fundamental properties of good acceptability criteria that were applied included the following:

- Criteria should map to the documented requirements.
- Criteria should be quantitative when practical but may be supplemented by qualitative values provided by the user and SMEs.
- Criteria should reflect the planned uses of the simulation.
- Criteria should support the assessment of statistical confidence in simulation results for intended uses.

Test Case Development Process

Each test suite followed a similar process for developing the underlying test cases. This process consisted of the following steps.

- **Step 1: Define test cases.** A detailed description of each test case was developed and documented with a standard template. Each test case was mapped to one or more requirements and applicable conceptual model sections and contained a description of the test scenario, the testing procedure, the tester-defined expected result, the test acceptability criteria, and the overall test results. All test cases required model runs; however, some portions of a test case, specifically those that related to asset attributes, could be verified by inspection of STORM+ input files and output reports.
- **Step 2: Review of test cases by Functional Analysis and Requirements IPTs.** Initial test case descriptions were reviewed by members of the Functional Analysis and Requirements IPTs to ensure the testing procedures met the intent of the STORM+ requirement or requirements.
- **Step 3: Modify test cases as needed.** Test cases were modified as needed based on the review results.
- **Step 4: Refine/debug test cases.** Test cases were implemented and prepared for model runs.
- **Step 5: Run for record.** Official test runs were conducted, and the results were documented.
- **Step 6: Develop test results matrix.** A results matrix summarizing the results of the test cases was produced.
- **Step 7: Archive test cases.** Completed test cases were archived to capture all of the test case descriptions, associated input data, reviewer comments and associated responses, test case results, test case traceability matrices, and test case results matrices. Testers defined the test procedures for each test case and provided detailed documentation of the steps that were taken to implement the test (e.g., which input files were modified, which output files were reviewed, etc.). This level of detail served two purposes. First, it provided testers with an understanding of how STORM is structured and what is required to run the simulation. Second, it provided a basis for follow-on regression testing.

Reporting Test Case Results

The results of a test case provided evidence of how well the required capability was implemented in STORM+. Test results were placed into one of five categories:

1. **Requirement met.** The simulation fully supports the required capability and meets all the acceptability criteria.
2. **Requirement partially met.** The simulation supports some elements of the required capability but does not provide the complete functionality and/or does not meet all the acceptability criteria.
3. **Requirement not met.** The test results indicate that the simulation does not provide the required capability and either there is no trace to future development or the results do not meet any of the acceptability criteria.
4. **Testing deferred.** The required capability does not currently exist in the simulation but is planned in future STORM+ development.
5. **Not tested.** Requirement not selected for testing on the basis of a risk assessment (probability of a problem and overall impact to the model).

Test Results

Figure 8 shows the three test phases and the categorization of requirements through the phases. Phase I testing focused exclusively on requirements that were at least partially implemented in STORM. Each requirement was tested within one or more test cases and placed into one of the five categories defined above. The last category, not tested, was not used in Phase I.

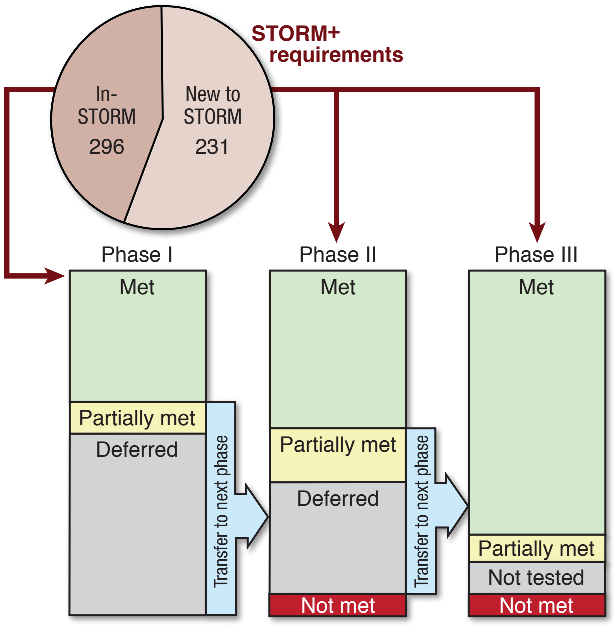


Figure 8. Requirements transition through test phases.

Requirements that did not meet the conditions to be declared met were carried forward into Phase II. Additionally, new functionality implemented in development spirals 1 and 2 was added to the requirements set to be tested in this phase. The results of the testing placed this new set of requirements into one of four categories. Again, the not tested category was not used in Phase II.

The cycle was repeated for the final Phase III, with the four categories becoming met, partially met, not met, and not tested. The not tested category applied to requirements that, for one of many possible reasons, were not included in the Phase III testing. Reasons included that the testing effort was descoped on the basis of a risk assessment or that the requirement was deleted from the original list, meaning that the sponsor no longer considered it a requirement in the initial version of STORM with maritime capability. The number of requirements in this category was ~16%, and they dealt almost exclusively with model usability rather than with functional representations.

Requirements deemed as not met by the test team were also few, at only 3%. After Phase III testing, the sponsor chaired a final adjudication process that included the leads from the Systems Engineering, Requirements, Functional Analysis, CV&T, and Development IPTs. The purpose was to review and obtain final sponsor interpretation of all requirements categorized as either partially met or not met. In addition to accepting the results of Phase III testing, the sponsor dropped one requirement as being beyond the original scope, characterized others as coding errors to be fixed immediately before release of STORM v2.0, and designated one requirement for future implementation and/or correction in subsequent releases of the simulation.

Risk Management

Finally, the Risk Management IPT was responsible for identifying, tracking, and reporting on all risks throughout the project. Risk management consisted of identifying, planning, mitigating, and retiring risks to the program. Risks were handled by a combination of methods. Any team member could identify and propose a risk. The Risk Management and Systems Engineering IPTs reviewed all proposed risks, and if they were accepted, a risk manager from outside of the Risk Management IPT was assigned to develop, execute, and monitor a mitigation plan. The Risk Management IPT was responsible for managing and documenting the process, while identifying and mitigating specific risks was spread across all of the IPTs. This process ensured that the appropriate IPT took ownership for each significant risk.

Once the risk manager was convinced that the risk had been successfully mitigated, he could apply to the Risk Management IPT to retire the risk. When the Risk Management IPT was convinced the risk was mitigated, the IPT chair would petition the Systems Engineering IPT for final risk retirement.

Figure 9 shows the number of risks tracked by program phase; the correlation between testing and risk mitigation is obvious. Identification of program risk tends to precede each test phase as test team members focus on development products and potential testing issues. Risk mitigation comes about through a variety of activities

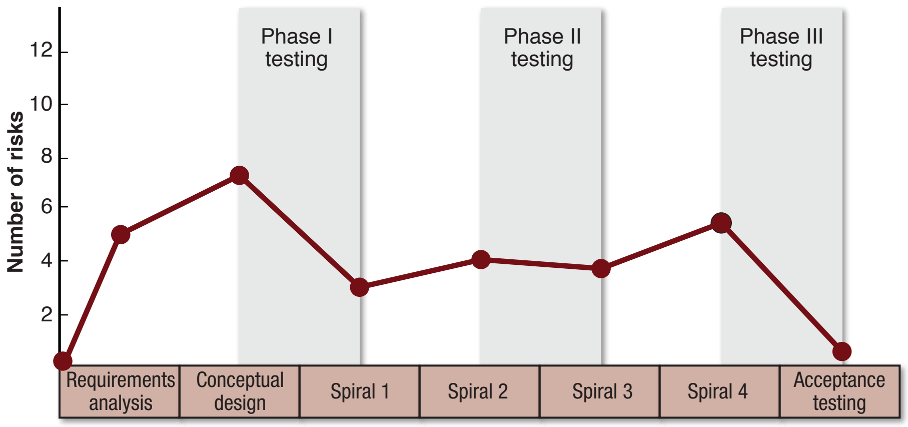


Figure 9. Number of program risks, by phase, tracked by the Risk IPT.

and must be complemented with testing to provide essential insights into the risk and confirm the effectiveness of the mitigation. Although risk was assessed across all aspects of the project—including schedule, resources, and model performance—the dominant risk came from the availability of qualified testers and its impact on the number of requirements that could be tested and retested if necessary. Mitigation for this risk involved close monitoring of the progress, productivity, and availability of the testers to provide early visibility to management when testing organizations were experiencing personnel turnover or resource issues. Through timely reviews and management intervention, this risk was not realized; sufficient experienced personnel were available throughout the testing phases.

CONCLUSION AND LESSONS LEARNED

The APL systems engineering support to N81's STORM+ project was an unqualified success. The process of taking a single-service campaign model and modifying it to be embraced by another service was a daunting task. The systems engineering methodologies employed ensured that (*i*) requirements were identified, verified, and controlled in scope; (*ii*) requirements were translated into conceptual models that could be integrated into existing code; (*iii*) verification of the implemented code was based on previously vetted acceptability criteria; and (*iv*) feedback mechanisms were in place to identify emergent modifications to requirements and risk. Several critical lessons were learned from the project:

- Systems engineering concepts are critical to managing software modification to existing applications. This is especially true for projects of the scope and size of STORM+.
- Systems engineering is compatible with and enhances the relationship between software development concepts (e.g., agile programming) and traditional requirements and concept development.
- Flexibility in applying systems engineering concepts is key to maintaining active participation of project participants that may have a broad range of experience with systems engineering.
- Systems engineering provides a framework and environment for the various divergent supporting organizations to collaborate in integrating and delivering a quality product.

Project participants were quick to recognize the role of sound systems engineering tenets in keeping the project on schedule and within budget, mitigating risk and delivering the desired campaign modeling capability to N81. We hope this example will provide future software project teams the confidence to embrace systems engineering as a dynamic framework for proactive project management.

ACKNOWLEDGMENTS: We thank Robbin Beall, Head of OPNAV's Campaign Analysis Branch, for having the vision to insist on having systems engineering involved from the beginning, as well as Sunny Conwell and Jerry Smith of OPNAV's World Class Modeling Initiative for empowering systems engineering within the project organization at the right level to be effective. We also thank the following APL staff members, each of whom applied systems engineering principles in a practical way to achieve the necessary results: Joseph Kovalchik, the first STORM+ systems engineer, who established the role systems engineering would play; Ngocdung Hoang, who was instrumental in the structure and rigor of the requirements process; Peter Pandolfini, who led the Risk Management IPT; Simone Youngblood, who led the CV&T IPT; and Alicia Martin, who led the Test Team. Systems engineering support for the STORM+ program was sponsored by OPNAV.

REFERENCES

[1]Kossiakoff, A., and Sweet, W. N., *Systems Engineering Principles and Practice: Wiley Series in Systems Engineering and Management*, A. P. Sage (ed.), John Wiley & Sons, Hoboken, NJ (2003).
[2]Blanchard, B. S., and Fabrycky, W. J. (eds.), *Systems Engineering and Analysis* (5th Ed.), Prentice Hall, Upper Saddle River, NJ (2010).
[3]Kendall, K. E., and Kendall, J. E., *Systems Analysis and Design* (8th Ed.), Prentice Hall, Upper Saddle River, NJ (2010).
[4]Pressman, R. S., *Software Engineering: A Practitioner's Approach* (7th Ed.), McGraw Hill, New York (2009).
[5]Chairman of the Joint Chiefs of Staff, *The National Military Strategy of the United States of America: A Strategy for Today; A Vision for Tomorrow*, http://www.defense.gov/news/mar2005/d20050318nms.pdf (2004).

The Authors

Robert L. Sweeney

Jeffrey P. Hamman

Steven M. Biemer

Robert L. Sweeney is a member of the Senior Professional Staff in APL's National Security Analysis Department. A retired U.S. Navy officer with experience in warfighting analysis with both the U.S. Navy and Joint staffs, his current assignment is as the Program Manager for the U.S. Navy's Resources, Requirements, and Assessment Directorate (OPNAV N8), which includes their World Class Modeling initiative. He has been an instructor in the Systems Engineering Program of The Johns Hopkins University Whiting School of Engineering since 2006. **Jeffrey P. Hamman** is a member of the Senior Professional Staff in APL's National Security Analysis Department and the Systems Engineering IPT lead for the STORM+ task. He is an accomplished operations research analyst with more than 20 years of experience in DoD acquisition, M&S, test and evaluation, program management, operations research, systems analysis, and military aviation weapon systems. **Steven M. Biemer** is a member of the Principal Professional Staff in APL's National Security Analysis Department. He is currently the coordinator for APL's Systems Engineering Competency Advancement program, with the goal of educating and training the technical staff in the latest systems engineering principles and practices. Before taking on this role, he was the Program Area Manager for Naval Analyses and Assessments, where he worked with U.S. Navy and U.S. Marine Corps organizations to define and conduct analytical assessments of warfighting systems, platforms, architectures, and networks. Mr. Biemer has 25 years of systems engineering and analysis experience with APL. Additionally, he is a curriculum developer and instructor for the Systems Engineering Program of The Johns Hopkins University Whiting School of Engineering. For further information on the work reported here, contact Robert Sweeney. His e-mail address is robert.sweeney@jhuapl.edu.