

Program execution management

Executive brief



PLM Software

Answers for industry.

SIEMENS

A systems-based approach for managing program risk



Executive summary

Seventy percent of systems development programs and projects fail to meet their established cost and/or scheduling objectives.¹

According to the OMB's Management Watch List and high risk project process, approximately 300 projects totaling nearly \$12 billion in estimated 2007 IT expenditures have been identified as being either poorly planned or performing poorly.²

Today, civilian federal agencies are under increasing pressure to effectively manage their human and budgetary resources. The Office of Management and Budget (OMB) now uses a quarterly scorecard to measure each federal agency in terms of its progress in achieving the objectives established by the five major initiatives in the President's Management Agenda (PMA).

As these initiatives and their related processes are institutionalized by the executive branch, federal agencies must find new and improved ways to execute their missions as efficiently as possible – or face the prospect of significant budget cuts.

Program risk typically is defined as factors that threaten to adversely affect a program in terms of its cost, scheduling and/or delivery-related performance. Traditionally, these impacts have been managed and evaluated independently of one another.

Siemens believes that civilian agencies can more successfully manage government programs of varying complexity – and significantly mitigate their risk – by applying proven system engineering techniques in conjunction with an integrated program execution management toolset.

Siemens is uniquely distinguished by its ability to inject standards-driven systems engineering concepts into a comprehensive program execution management (PEM) solution.

Siemens tools can significantly mitigate program risk by enabling program teams to link critical program elements – including requirements, architecture and testing – to program cost, scheduling and performance criteria.

The resulting PEM solution enables program teams to continuously understand the interdependencies and consequences of changing requirements and systems configurations across the life of the entire program.

Current realities, rising demands

During the 1990s, the emergence of the worldwide web, individual/group productivity tools and enterprise-wide systems spurred an unprecedented rise in human resource and institutional productivity, especially in the private sector. Information systems played a significant role in contributing to these gains.

As the government strives to reduce cost, support the free flow of information and match the productivity gains experienced by the private sector, its managers and decision makers look to further investment in information systems.

However, while investment in these systems often constitutes a sizeable chunk of an agency's annual budget, the development and delivery of these programs carries its share of historically significant hurdles.

As might be expected, the chance of program failure increases rapidly as the program's size and technical challenges grow. However, regardless of size and other challenges, the successful execution of a systems development program is fraught with communications issues and technical challenges for all involved participants. Given the number of "moving parts" and the diversity of skills required for proper program execution, it is small wonder that program success is often elusive.

In fact, 70 percent of today's development programs fail to meet their essential budgetary, scheduling or user community objectives. Given this failure rate, it is easy to understand why program post mortems often reflect a plethora of familiar participant explanations.

- "Requirements were too sketchy and subject to regular change."
- "The customer didn't understand the technology's capabilities and limitations."
- "We didn't set customer expectations properly."
- "We didn't fully understand the depth of interaction required to existing systems."
- "Team communications were poorly managed."
- "Software development didn't take the time to question key systems requirements"
- "We had little or no visibility to changes that impact our requirements, scheduling or project artifacts."

The President's Management Agenda (PMA) is the administration's blueprint for delivering business improvement on a government-wide basis while allowing individual agencies to meet their mission-critical objectives. The agenda's Budget and Performance Initiative is especially important since it links past performance in the area of effective human and budgetary

resource management to future budget allocations.

As these initiatives and their related processes are institutionalized, federal agencies must improve the ways they execute their systems development programs – or face significant budget cuts.



Systems development methodologies

Over the years, methodologies, techniques and tools have evolved to address the need for improving the systems development process. Although evolution has rendered certain methodologies obsolete, more often than not, the choices facing program teams have simply grown. The nature of the system being developed and its operational model make certain choices obvious – such as whether to apply a waterfall, incremental or spiral development methodology. On the other hand, some choices depend entirely on the experience and preferences of the program team.

All projects can be better managed when they are aggregated into hierarchical segments, such as phases, stages, activities, tasks and steps. In systems development projects, the simplest rendition is called the waterfall methodology (shown in the accompanying diagram).

This graphic illustrates several key principles that are crucial for a good methodology, including:

- Performing work in stages
- Conducting content reviews between stages
- Leveraging review as quality gates and decision points for go-forward continuation

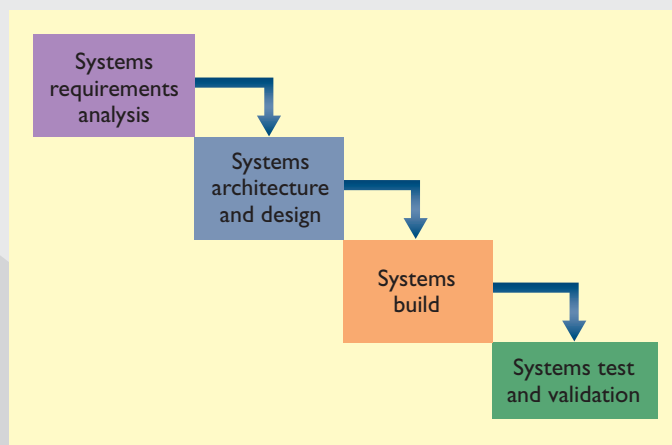
In this methodology, the waterfall provides an orderly sequence of development steps that help ensure the adequacy of the program's design reviews and documentation. In turn, this ensures the quality, reliability and maintainability of the delivered system.

While the waterfall methodology provides an orderly structure for systems development, the demand for faster time-to-market makes its sequential approach cumbersome and needlessly slow. With this in mind, the ultimate evolution from the waterfall is the spiral methodology.

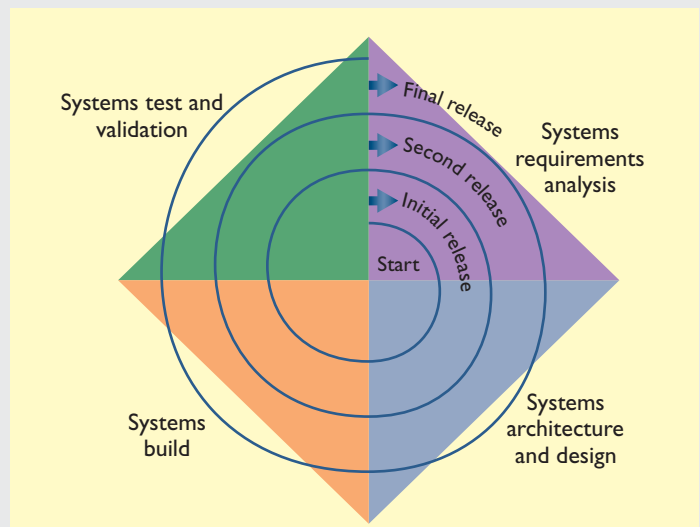
The spiral methodology takes advantage of the fact that development projects work best when they reflect a process that is both incremental and interactive – where project teams start small and benefit from an enlightened process of trial and error.

The methodology leverages tasks that include rapid prototyping, parallel execution and concurrent design/build activities. In addition, the spiral method should be carefully planned with tasks and deliverables identified for each step in the spiral.

What remains consistent in the field of systems development, however, is the need to direct and manage an interdisciplinary team that has to conceptualize, refine and elaborate on the system's requirements, architecture/design, build and test/validation phases. With this in mind, program teams need a systematic and repeatable approach for reducing inherent risk across the entire systems development lifecycle.



Waterfall methodology



Drawing lessons from other sources

Since World War II, the war-fighter side of the U.S. federal government has recognized the value of using a systems engineering approach to manage large-scale programs. By their nature, multi-million dollar federal programs are so complex that individual programs must be managed as “a system of systems” in order for the whole of the program to exceed the sum of its parts.

Systems engineering strives to improve the value of the system as a whole by understanding all of a program’s “moving parts” and their related interconnections so that the program can successfully satisfy the mission-critical requirement of delivering specified value on time and on budget.

War-fighter programs, such as the Navy/Coast Guard Deepwater, Joint Strike Fighter and DDGI000 initiatives, have used software tools to manage

these “moving parts” including the program’s complex information assets and lifecycle processes. Siemens believes that civilian federal agencies can profit immensely from the experiences and best practices that have been learned from systems-based management.

War-fighter programs typically use systems engineering’s practices to manage “hardware” – such as weapons systems, platforms and supporting infrastructure. However, decision makers, oversight bodies and program managers increasingly understand that all programs have deliverable “information packages” whose planning, execution and sustainment can benefit from systems engineering’s repeatable and auditable process control and data management techniques.

Systems engineering

Systems engineering is an interdisciplinary approach for enabling the realization and deployment of successful systems. It can be viewed as the application of engineering techniques to the engineering of systems, as well as the applications of a systems methodology to engineering projects.³

As the INCOSE web site indicates, “A system is a construct of different elements that produce results not obtainable by the elements alone. The elements can include people, hardware, software, facilities, policies and documents. The value added by the system as a whole, beyond that contributed independently by the parts, is primarily created by the relationship among the parts; that is, how they are connected.”

A home security system is a good example of a system whose components – motion sensors,

door/window contacts, master control module, wiring and alarm speaker – must work together to perform an overall mission.

Successfully applying an interdisciplinary approach to engineering a system is made more complex given the fact that the behavior and interaction of system’s components are not always well defined or clearly understood – at least at the outset of the program. The primary aim of systems engineering is to define and characterize these systems and subsystems and their interactions and dependencies.

The Institute for System Research has established eight criteria for defining systems engineering’s distinguishing characteristics, including the ability to:⁴

- Account for cross-discipline considerations
- Leverage high-level metrics
- Represent hierarchical structures
- Facilitate global and local optimization
- Reflect the role of heterogeneous influences
- Accommodate dynamic behavior
- Apply process and program lifecycle methodologies
- Consider the impact of non-technical components/metrics





Siemens provides solutions that meet all of these systems engineering imperatives and, most importantly, provide the “glue” for bringing these imperatives together. Siemens’ understanding of the role of systems engineering and the implementation of its capabilities starts with the following definition:

Systems engineering is comprised of a broad set of processes and methods that model and analyze the interactions between a program’s requirements, subsystems, constraints and components.

As this definition implies, it is crucial to apply systems engineering at the front end of the program lifecycle (where decisions are made about what is being developed), as well as throughout the entire lifecycle and into each of the disciplines involved in realizing and sustaining the delivered systems.

Siemens solutions integrate systems thinking with systems engineering across the entire program lifecycle by enabling program teams to capture and deliver a systems-level perspective to everyone engaged in the system development process. The idea here is to

recognize that every program decision has implications for everyone who participates in the program lifecycle.

Siemens solutions are distinguished by their ability to foster a system-of-systems approach to program management. Siemens solutions allow program teams to define systems requirements as “requirement objects” that can be associated with system architecture components. In turn, these objects and components can be interconnected to project schedules, budgetary costs, business process workflows and system test plans.

As a result, Siemens solutions enable program teams to continuously understand the interdependencies and consequences of requirement changes, as well as system configuration changes.

It is strategically crucial to understand these implications and communicate their impact so that program stakeholders are able to optimize multiple financial and technical considerations – including the system’s cost effectiveness, scheduling demands, user community requirements, safety, regulatory compliance, ease of use, maintainability and total quality. Just as importantly, program teams need this level of understanding early in the program lifecycle before change becomes too costly.

Several techniques, tools and methodologies in the field of systems engineering enable implementers to arrive at a workable solution. Two of the most crucial capabilities are systems requirements analysis and systems architecture and design. Together these capabilities form a basis for the solution. These elements must go hand in hand. Once they both reach the proper level of maturity, the required foundation for a systems solution will be laid.

After that foundation is in place, its underlying capabilities will enable program teams to perform the program’s design and systems integration phases effectively. In addition, the foundation will be able to support the program’s follow-on test and validation phase – which is critical for bringing the systems development lifecycle to closure.

In this scenario, program teams leverage the systems engineering foundation to establish and execute comprehensive test plans and test specifications that comply with quality-driven defect levels consistent with the program’s requirements and user community expectations.



Systems requirements analysis

In both systems engineering and software engineering, requirements analysis encompasses all of the tasks that go into the instigation, scoping and definition of a new or altered system. Requirements analysis is an important function in the systems design process, whereby requirements engineers and business analysts – along with systems engineers and software developers – identify the needs or requirements of a client.

Once the client's requirements have been identified, systems designers are then in a position to design a solution. During most of the history of engineering, requirements analysis was considered to be a relatively easy part of the process.

However, in the last decade, requirements analysis has grown in stature and is now recognized as the most important element in the systems development process. This is driven by the fact that failure to properly identify program requirements makes it virtually impossible for the finished system to meet the client's functional needs,

budgetary constraints and/or scheduling deadlines.

However, it is a major challenge to successfully complete the requirements analysis task. For starters, it is difficult to identify all program stakeholders, provide them with an appropriate form of input and document this input clearly and concisely.

And then there are the program's constraints, which the requirements engineer has to understand in order to determine whether or not the system is:

- Feasible
- Schedulable
- Affordable

With new projects, there often is a rush of enthusiasm that tempts the program team to downplay the importance of requirements analysis. However, case studies reveal that cost, scheduling and technical risks can be significantly affected, in a positive way, through rigorous and thorough up-front requirements analysis (which, of course, is an essential component of systems engineering).

Requirements analysis can be an extended and iterative process. Requirements specialists do their work by talking to people, documenting their findings, analyzing the collected information to discover inconsistencies/oversights and then talking to people some more. This process can go on for anywhere from a week to a year

or longer – and it often continues throughout the system's lifecycle. Although several current techniques are used to record system requirements, the current and dominant methodology involves the use case.

A use case defines the interactions between external actors and the system with respect to accomplishing a business goal. Actors are parties outside the system that interact with the system. An actor can be a class of users, a role that users can play or another system.

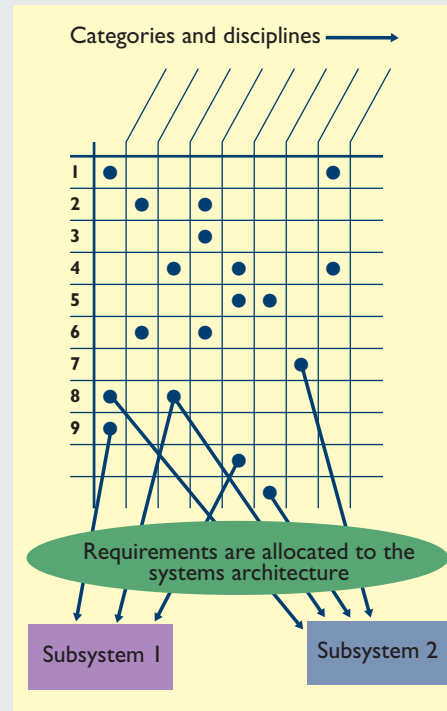
Use cases treat the system as a "black box" and perceive interactions with the system (including systems responses) as being outside the system. This policy is deliberate; it simplifies the requirements' descriptions without making assumptions about how the requirement's functionality will be accomplished.

Systems architecture and design

The systems architecture is an abstract representation of a system. Architectures can be viewed as partitioning schemes that divide all of the system's present and foreseeable requirements into a workable set of clearly bounded subsystems with nothing left over. In other words, the partitioning scheme is exhaustive.

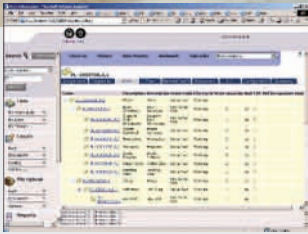
A major goal of partitioning is to arrange the elements in the subsystems so as to minimize the need for communications between them. In both hardware and software systems, a good subsystem tends to be seen to be a meaningful "object."

Moreover, a well planned and thought out architecture provides easy mapping to user requirements and user validation/acceptance tests. When completed, mapping will exist from every least element to every requirement and test.



Requirements mapping across the systems architecture





tools enable program teams to trace requirement-related change and its impact throughout the program lifecycle.

Systems artifacts

Throughout the systems development lifecycle, participants in the program team usually are busy generating documentation for which they are responsible. Today, the Microsoft Office suite of applications – including Word, Excel, Project and Visio – is typically used to perform the documentation task.

This documentation can include project plans, progress reports, business process flows, requirements documents (e.g., use cases), systems design documents, component design specifications, software design specifications, hardware specifications, interface control documents, test plans and test specifications, as well as other documents.

This documentation set is generally referred to as the systems artifacts, which collectively describe the who, what, when, where and how of systems use.

As might be expected, a myriad of relationships exist between these systems artifacts, including use case-to-use case, use case-to-systems design, use case-to-software design specification, use

case-to-project plan, use case-to-test plan, project plan-to-test plan, systems design-to-component design, systems design-to-interface control specification, systems design-to-hardware design, systems design-to-software design specification and many more.

Challenges for effective systems development

Although artifact relationships can be readily implied, creating and maintaining these relationships in an effective manner is almost never done.

To make matters worse, the highly iterative nature of the systems development lifecycle necessitates the natural evolution and progression of artifacts – thereby compounding the problem of relationship maintenance between various artifact versions.

Ideally, a change control system should be in place to identify all affected artifacts and their related revisions. Historically, this has rarely been done in a controlled fashion. As a result, artifacts become dated – unable to reflect the program's most recent activity. In addition, development costs rise and scheduling milestones deteriorate as the

testing and validation phase reveals various shortcomings.

In ideal conditions, system requirements would rapidly mature to the point where they describe the intended system in complete and unambiguous terms that would completely satisfy all business and user communities. Similarly, systems and software architects would produce near perfect designs and software – and test groups would generate comprehensive test plans that identify defects in time to facilitate swift resolution.

However, the ideal world is far removed from the real world.

The real world is a place where users might be technically unsophisticated or not know exactly what they want. As a result, they might insist on new requirements long after budget and scheduling constraints have been fixed.

Initial ideas about what is needed are frequently incomplete, wildly optimistic or rigidly entrenched in the minds of participants leading the acquisition process. In addition, requirements sometimes are gathered by unseasoned individuals who cannot bring the balance of technical, communications and negotiation skills



Similarly, a robust architecture is one that exhibits an optimal degree of fault tolerance, backward compatibility, forward compatibility, extendibility, reliability, maintainability, serviceability and usability – along with other targets deemed necessary by the program's requirements.

Siemens solutions leverage an integrated suite of systems architecture, requirements analysis and testing documentation capabilities. This integration allows requirements to be associated with specific validation tests.

As a result, program teams are able to make certain that their test protocols are exhaustive and complete and thereby mitigate program risk. Siemens' unified

necessary for producing a realistic, unambiguous, clear and concise set of requirements.

The end result: needless change well into the system development lifecycle that, in turn, causes management dissatisfaction, excessive attention to the wrong kind of detail, lack of strategic focus – and ultimately, a runaway program.

What can be done?

While many program challenges are difficult to control, other challenges can be met and mastered.

First and foremost, program teams can – and must – plan for the changes that inevitably occur by implementing structured and repeatable change management procedures.

Secondly, oversight reporting is a fact of life that can be planned so that it consumes no more cycles than is absolutely necessary.

Thirdly, a streamlined method should be established to link, maintain and effectively manage the relationships between the program’s systems artifacts.

The disciplines of project management, requirements analysis, systems architecture and test planning and execution traditionally have been a loosely connected set of activities leading to built-in overhead, out-of-sync data and ultimately, misguided decisions.

The toolsets favored by these disciplines mostly followed a best-of-class approach that did nothing to ensure effective cross-discipline change control. This omission is especially damaging during the early program stages when changes occur more frequently and result in more radical program impact.

With this in mind, Siemens now provides a program execution management solution that enables program teams to focus their energies on mastering challenges they actually can control.

Siemens’ Program Execution Management solution

Siemens provides a program execution management (PEM) solution that supports the entire systems development program lifecycle – from defining project concepts through operation, maintenance and retirement. The Siemens solution addresses the eight phases of the systems engineering process defined by the International Council on Systems Engineering (INCOSE) fellows.⁵

Siemens’ PEM solution enables program teams to establish a seamless environment populated with powerful, integrated web-enabled tools for:

- Program, project and portfolio management
- Systems engineering, including systems architecture, requirements management and test plan/execution management capabilities
- Business process management
- Document and records management
- Oversight reporting
- Team collaboration

This PEM solution is powered by Siemens’ Teamcenter® software, the world’s de facto standard for deploying digital lifecycle management applications. The solution also includes Teamcenter’s intuitive collaborative workspace, which is ideal for meeting the needs of civilian federal agencies and their IT contractors.

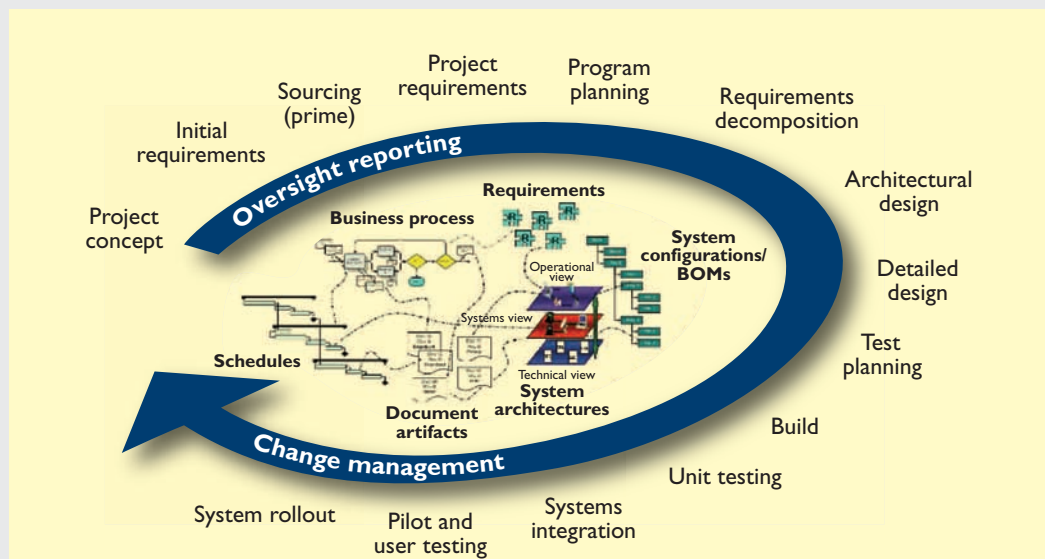
The PEM solution provides program teams with total, up-to-date visibility into the status of the program spend and completion percentages at agreed upon levels of granularity. Agencies are

able to view the current status and health of their programs from an easy-to-use collaborative workspace that displays numerous earned value management (EVM) indicators. Equally important, this viewing is facilitated with little or no effort required on the part of the contractor.

Siemens’ overall PEM solution supports the systems development program lifecycle from project concept through operations, maintenance and retirement. By supporting and linking critical and evolving program knowledge through their inevitable cycles of change, the Siemens solution enable program teams to:

- Conceptualize, refine and elaborate on the program’s systems requirements, systems design and systems validation phases
- Ensure that all requirements have been fully allocated to one or more elements in the systems architecture
- Make certain that all requirements are covered in the testing and validation phase of the program lifecycle

Siemens’ systems development program lifecycle



The Siemens solution differs from traditional program management capabilities by placing project schedules in the context of virtual teams. This enables program managers to assign project-related tasks to individual team members across an enterprise through the use of workflow processes and configuration-controlled program-related data.

As a result, program teams can efficiently plan, execute, audit and report on continuous program activities that engage and coordinate the performance of their stakeholders. These activities can be managed across a virtual enterprise that includes both intra-agency and cross-agency participants.

The Teamcenter foundation enables the PEM solution to connect resource and cost information with systems data and process information – thereby transforming today’s “planning and statusing” environments to “planning and executing” environments.

The value of Siemens’ PEM solution to government agencies includes lower cost of project management, compressed cycle

times, increased management visibility, in-process flexibility and a sustainable, scalable platform for program management and risk mitigation.

Case study – U.S. Joint Forces Command

Siemens’ systems engineering software was used to build a collaborative web-based solution for the U.S. Joint Forces Command. This “system of systems” was charged with meeting the command and control requirements of “all of the communities of interest within the joint world” – including requirements established by the joint close air support (JCAS), joint task force (JTF) headquarters core architecture and all elements of each of the U.S. functional combat commands.

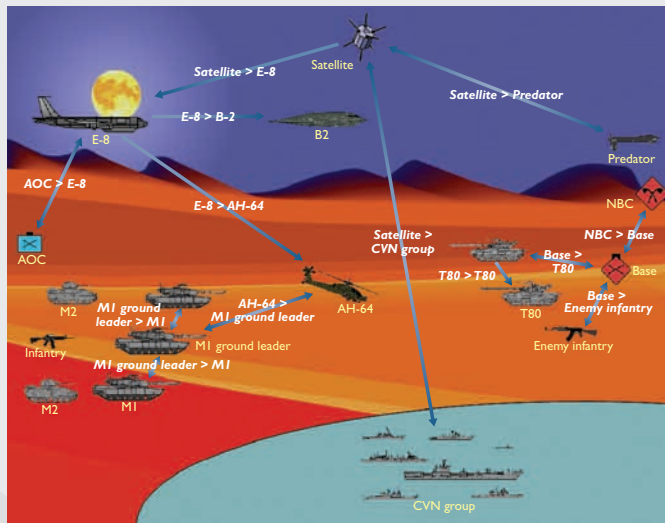
In essence, the joint services needed an easy method for commanders and their staffs to put together a joint task force. More specifically, the commands needed to be able to aggregate manpower, capabilities, functions and tactics across multiple communities of interest.

The USJFCOM program delivered these aggregated capabilities by building a single Teamcenter-based environment that provided a joint battle management command and control (JBMC2) mapping environment (JCME).

The USJFCOM program won the Excellence in Enterprise Architectures Award for Department of Defense Leadership in Government Transformation.

“By being fully integrated into a common environment, the combat commands can ensure that they are all on the same page. It’s not just doctrine, but it’s all the facets because we’re mapping all those different relationships. You can see how the command and control aspects affect all (communities of interest) and their capabilities. It gives (us) this great ability to collaborate and see each other’s work.”

Kim Frisby
Deputy Division Chief
Integrated Architectures and Systems Engineering



The next step

Successfully executing a systems development program is an endeavor fraught with communications issues and technical challenges for all involved participants. These factors contribute to today's 70 percent failure rate.

But positive steps can now be taken to enable government agencies to mitigate the risk involved in initiating programs of varying complexity. The application of systems engineering techniques has delivered real-world results for the war-fighter side of the federal government.

Using an integrated and functionally rich set of systems-based tools enables multiple disciplines in the systems development lifecycle to work in concert and deliver mission-critical systems that meet government demands for on-time, on-target and on-budget success.

Contact your Siemens representative today to learn more about Siemens' Program Execution Management solution.

Footnotes

- 1 "The 70-Percent Failure," Bob Lewis, *InfoWorld*, 2003
- 2 GAO-06-1099T, September 2006.
- 3 *Systems Engineering: Principles and Practice of Computer-Based Systems Engineering*. Bernhard Thome, 1993.
- 4 "What is systems engineering?" *An Introduction to System Engineering*. The Institute for Systems Research, 2005, www.isr.umd.edu.
- 5 A consensus of INCOSE fellows endorses The Systems Engineering Process from A.T. Bahill and B. Gissing, "Re-evaluating Systems Engineering Concepts into Systems Thinking," *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, 1998. This model defines the systems engineering process as being comprised of eight phases that account for the problem statement, alternative investigation, system modeling, subsystem/component integration, system launch (validation), performance assessment, reevaluation and variations.



About Siemens PLM Software

Siemens PLM Software, a business unit of the Siemens Industry Automation Division, is a leading global provider of product lifecycle management (PLM) software and services with nearly six million licensed seats and 56,000 customers worldwide. Headquartered in Plano, Texas, Siemens PLM Software works collaboratively with companies to deliver open solutions that help them turn more ideas into successful products. For more information on Siemens PLM Software products and services, visit www.siemens.com/plm.

Siemens PLM Software

Headquarters

Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
972 987 3000
Fax 972 987 3398

Americas

Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
800 498 5351
Fax 972 987 3398

Europe

3 Knoll Road
Camberley
Surrey GU15 3SY
United Kingdom
44 (0) 1276 702000
Fax 44 (0) 1276 702130

Asia-Pacific

Suites 6804-8, 68/F
Central Plaza
18 Harbour Road
WanChai
Hong Kong
852 2230 3333
Fax 852 2230 3210

www.siemens.com/plm

© 2009 Siemens Product Lifecycle Management Software Inc. All rights reserved. Siemens and the Siemens logo are registered trademarks of Siemens AG. Teamcenter, NX, Solid Edge, Tecnomatix, Parasolid, Femap, I-deas and Velocity Series are trademarks or registered trademarks of Siemens Product Lifecycle Management Software Inc. or its subsidiaries in the United States and in other countries. All other logos, trademarks, registered trademarks or service marks used herein are the property of their respective holders.