

Realizing Successful System Design
Using Model-Based Design (MBD) in an
Integrated Development Environment (IDE)

Prepared by:

John F. Shaker
NovaTek Engineering, Inc.

Robert C. Wagner
NovaTek Engineering, Inc.

John E. Juhasz
TelePath & Associates

October 8, 2004

Introduction

Since the dawn of the space era a half-century ago, technology has been advancing at a pace that is both astounding and relentless. Just one example of such exponential evolution is the semiconductor industry, one of the early off-springs motivated by aerospace needs and further fueled by industry and consumer spin-off growth. Based on today's actual achievements, some now say that Moore's Law¹ is pessimistic. Similar progress has been achieved in areas of telecommunication, optics, biological and material sciences as well as other technology enablers for the modern era.

As technology has advanced, so has the evolution of the product development systems that underlie the US economy. The driving transformation in this milieu is the desire to convert "siloed" and underperforming business systems into something more responsive to stakeholder needs. In addition, there are increasing demands being placed on product development organizations due largely to resource constraints and the need to produce more, with less. The net effect of these influences in many industries has been that of **increased system complexity in terms of product structure and the organizations that produce them**. Therefore, the driving need in this changing game is to effectively manage the increasing complexity seen across the entire system development life cycle (SDLC).

Additionally, cost-effective technologies have further accelerated the demand for solutions concerning systems with ever increasing complexity. In spite of the remarkable achievements of the past, prognostications are that future developments will likely overshadow all of today's technologies. With ample headroom left in Moore's law, combined with other areas of phenomenal technical progress, the growth of system complexity for the next decade is forecast to exceed all prior developments to date. These rapid changes will exacerbate the already present and overwhelming challenge – that the complexities of systems and enterprises exceed our ability to effectively understand, design, integrate, and manage their evolution throughout the life cycle.

Significant strides have been made by engineering societies and consortia in motivating the emergence of the methodologies and toolsets required to conceive, develop, and deploy complex systems. Sophisticated modeling tools are available to support engineering analysis and the simulation of numerous technical design elements, while others focus on system design and requirements management. In spite of impressive results, the state of the art is still lagging behind actual needs. Although quite effective in solving specific technical problems, many of these tools are too heavily focused on a limited range of the system life-cycle. Most are incapable of interconnected operation and therefore fail to provide seamless capability for a fully model-based life cycle management. In the areas of business models, processes have been dominantly the province of IT departments focused mostly on operations, and not well positioned for enterprise design.

This paper is intended to provide an elaboration of the challenges confronting modern technical programs followed by a solution framework that holds the key to successful complex system development. The program context being addressed in this case deals mostly with aerospace and defense programs, although the concept and rationale advanced applies equally to any complex program undertaking.

¹ Moore's Law states that computing power in semiconductors would double each 18 months, while production costs would be reduced by half.

The Changing Game

The problems related to ineffective system design, development and operations have received wide attention and publication. For example, in the government realm of complex aerospace systems development, thought-provoking studies by NASA's Columbia Accident Investigation Board (CAIB) [1] and the Department of Defense (DoD) team led by A. Thomas Young [2] question the government's very ability to develop and manage large complex programs such as the Space Shuttle and the Space Based Infrared System (SBIRS). But the aerospace industry is not alone when it comes to the development of under-performing complex systems. Over the past two years a large volume of literature has been produced describing the mediocre performance of the enterprise-level information technology (IT) systems representing trillions of institutional investment dollars. Some executives claim these IT systems actually impede their ability to achieve badly needed business growth [3]. So what's going on here?

Although the causal nature of this problem is well beyond the scope of this white paper, it is becoming abundantly clear from ongoing studies that the ***cause of system under-performance is poor system engineering execution across the SDLC. This is abetted by inadequate competence and cognizance in program management.*** Understanding user need, the development of valid requirements, the transformation of requirements into design features, and verification and validation activities all tend to be deficient in these cases. Further exacerbating these engineering and management deficiencies at the systems level are ***emergent properties*** that "emerge" from the synthesis of interactions between the components at each level of integration. Emergent properties are the property of the whole system, not the property of the parts, and cannot be deduced from the parts. They are a product of the interactions, not the sum of the actions of the parts, and therefore must be understood on their own terms. The net effect of these factors, coupled with constantly shrinking cycle times, is poor ***requirements engineering that translates into poor design quality. This manifests as an increased risk that eventually leads to unanticipated programmatic and technical failure.***

Specific system development issues can be generally categorized as those related to (1) Program / Communication Management, (2) Engineering / Technical Management, and (3) Knowledge / Information Management. Within these categories, identifiable issues and challenges are:

1. Program / Communication Management

- Large scale, long term development programs with broad scope
- The need for decision-making in the face of development uncertainty
- Early identification / mitigation of programmatic risks
- The need for constantly addressing conflicting needs across the stakeholder network
- Early identification of customer and user needs by dispersed, cross-functional teams
- Accurate extraction of valid system definition from many data sources expressed in many "languages"
- The ability to develop valid requirements from poorly articulated needs
- Ensuring accurate translation of needs by engineering specifications
- The ability to generate specifications that are understood across stakeholder groups
- The ability to manage requirements scope creep
- Managing geographically dispersed project teams
- Managing multiple development and management processes, methods, tools, and environments
- Addressing resource constraints – skills, time, and money
- Effectively managing risk throughout the SDLC

2. Engineering / Technical Management

- Increased complexity due to large numbers of components that are highly distributed
- System interdependencies requiring a holistic, iterative design environment
- Complex systems design requiring inputs from a wide range of engineering disciplines
- Constituent components with individual lifecycles (e.g. software subsystems in electro-mechanical devices)
- The need for timely decision-support based on what-if scenario analysis, trade studies, and risk analysis
- Dispersed, cross-functional engineering teams
- The effective execution of trade studies and decision support linked to requirements
- The effective management of technical risk

3. Knowledge / Information Management

- The need for timely access to relevant and accurate information to support decisions throughout the SDLC
- The proliferation of information by virtue of advancing technology – to the point of information overload
- The need for design knowledge capture (knowledge management / retention) in “aging” enterprises.
- The need to develop valid information architectures and system designs that can evolve and adapt over time
- The need for systematic methods of organizing and maintaining enterprise knowledge bases

In the end, a “total solution” must address both the management and technical challenges as a synthesis of the art and science of complex systems development.

Meeting the Challenges of Increasing System Development Complexity and Change

Successful creation of effective system designs can be achieved if the development environment properly accounts for all competing extremes across the SDLC and adapts readily to dynamic events and uncertainties. The systems development environment in its most fundamental representation is shown in Figure 1. It is a judicious integration of people, process, and enabling technology that is guided and governed by enterprise inputs that direct the transformation

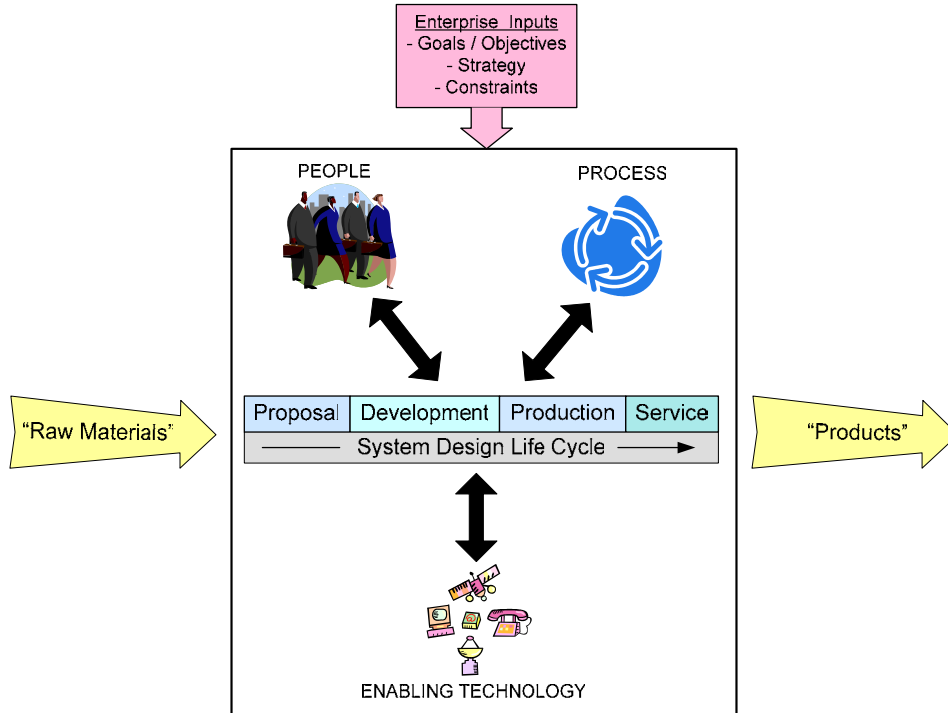


Figure 1 – The Systems Development Environment

of raw materials into value-adding products and services. Paramount to the success of this enterprise “symphony” is the linkage of downstream and upstream management and engineering activities so that holistic designs can be achieved within the constraints of the environment. The basis of the proposed approach, therefore, is *the fusion of skilled management and engineering practitioners using information and engineering technology in an integrated systems development environment whose foundation is systems engineering principles and processes*. The integration of people, process, and enabling technology in this manner supports disciplined, fact-based decision making across the SLDC that results in “real” accountability.

Although widespread use of systems engineering principles and practice has yet to achieve critical mass, methods of application have been thoroughly documented in engineering and trade journals. Furthering the adoption of formalized systems engineering practice has also been the objective of several key industry standards such as ISO/IEC 15228, IEEE 1220, and ANSI/EIA 632. There is also a significant push in the Industrial Sector to integrate systems and software engineering standards in the context of integrated product and process development (IPPD) [see Software Engineering Institute (SEI) CMMI-SE/SW/IPPD/SS/]. The merits of the structured, disciplined, and integrated systems development processes achieved by the brave few embracing it are emerging from the cutting-edge of product development. Recent examples include the Joint Strike Fighter (JSF) Program and Boeing’s current plans for 7E7 development.

Realizing an effective system-level design while addressing current development issues will require an approach that focuses on areas of the SDLC that harbor significant opportunities for improving the design process. Based on recent industry failure reviews, and our engineering experience, it is believed that the areas of systems development indicated in the callouts of Figure 2 can be exploited to achieve immediate improvements in design quality, reliability, risk aversion, and overall value. The principal benefit of this approach is that it provides focus on areas of the SDLC that provide significant leverage for increasing program success. These areas of system development include early design definition and analysis upstream, verification and validation downstream, and design insight/oversight at all points in-between. It is important to note that this approach is independent of, and adaptable to any SDLC methodology.

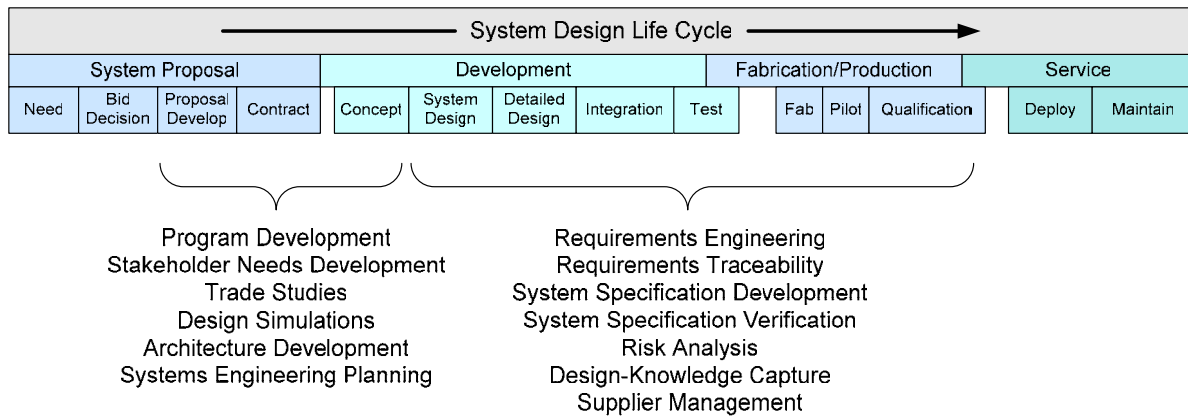


Figure 2 – Valuing-Adding Systems Development Activities

The implementation of this advanced systems development approach requires three principal components: (1) a systems engineering process, (2) a well structured “system model” of the target system, and (3) an integrated development environment (IDE). The implementation and justification of a formal systems engineering process can be found in a number of references (e.g. [4], [5], [6], and [7]) and will not be expounded upon here. Successful systems design in this case appears to be contingent on an organization’s ability to deploy an IDE that supports a highly responsive systems engineering process. The key, therefore, to successful complex system design resides within the second and third elements of the proposed approach. However, creating a well-structured target system using a traditional systems engineering “paper-based” approach is extremely difficult to achieve by virtue of design interdependencies and the distributed nature of the development teams. Also, creating an effective IDE requires integration of engineering and management processes with information technology, which has also proven difficult to do.

Realizing State-of-the Art System Design

A significant critical success factor for exploiting the valuing-adding areas presented in Figure 2 is in the implementation of *model-based design (MBD)*. The MBD concept as applied to the problem of complex system design was introduced in the mid-90’s ([8], [9] and [10]). More recently this paradigm has been expanded by others ([11] and [12]) as it continues to gain wider acceptance in the aerospace and defense, automotive, and electronics industries. The principles of MBD are (1) the embodiment of system design in a modeling environment and (2) the utilization of models to simulate system behavior. Utilizing a model-based context for system design transforms the traditional paper-based approach to one that supports extensive evaluation of design trade spaces early in the SDLC.

The model-based simulation of operational scenarios makes it possible to characterize system performance with respect to user requirements. Executing simulations in an IDE using a rules-based application of design requirements and performance “states” enables rapid design tradeoffs, a quick evaluation of operations scenarios, and a thorough examination of design trade spaces. In the advent of system design changes, an MBD system can be updated quickly and reliably with no sacrifice to development continuity. With the proper system architecture the MBD systems models can be integrated with design artifacts of detailed design and with testing phases of the SDLC. System design implemented in this way supports balance across the extremes of cost, schedule, and performance.

The vehicle for delivering the promise of MBD is the IDE shown in Figure 3. The envisioned IDE for MBD will be a high-performing, collaborative system design environment made up of advanced computing, networking, and information management systems. The central purpose of the IDE will be the cross-functional development of a comprehensive “system model” to capture the functional behavior, performance, cost, and risk factors in the emerging design and link it to user need (represented by scenarios).

Beginning with user needs and constraints, system architecture and operational scenarios are developed and used to create high-level system requirements and design. System design and requirements information is input to the system simulation environment, which is comprised of modeling and analysis software. Depending on the fidelity of the design and simulation, it may be necessary to assemble and analyze subsystem models prior to full system assessments. Once the system models have been created and linked with operational parameters, scenarios can be executed and the resulting system performance compared to user requirements. The system parameters can be systematically varied until the desired performance is achieved within operational constraints.

The ultimate objective of this systems development paradigm is to integrate systems management, knowledge management, and engineering information across the SDLC in a manner that results in receiving the fact-based information required for “right-time” decision support. The essential features to this approach include:

- The effective translation of user needs into system requirements
- Early problem discovery and resolutions that minimize cost, schedule, and technical impacts
- Timely emphasis on risk management
- Opportunity for user-specific views of systems development information
- The ability to be integrated into lower level design activities (detailed design and testing)
- Greater design team accountability resulting from inherent progress visibility

This represents the *next-generation* in complex systems development platforms.

Another important feature of this development paradigm is that it can be instituted in a piecewise-continuous fashion starting with improvements in high-level system design and management processes. Once the initial process modifications in these areas have been stabilized, the transformation of other portions of the SDLC such as detailed design, testing, and system qualification can be updated as well. In this way, overall process transformation risks can be better managed because improvements are evolutionary rather than revolutionary. Furthermore, proceeding in this fashion will also allay the fears of project managers and engineers alike who are wary of the changes to existing processes that may add risk to their programs.

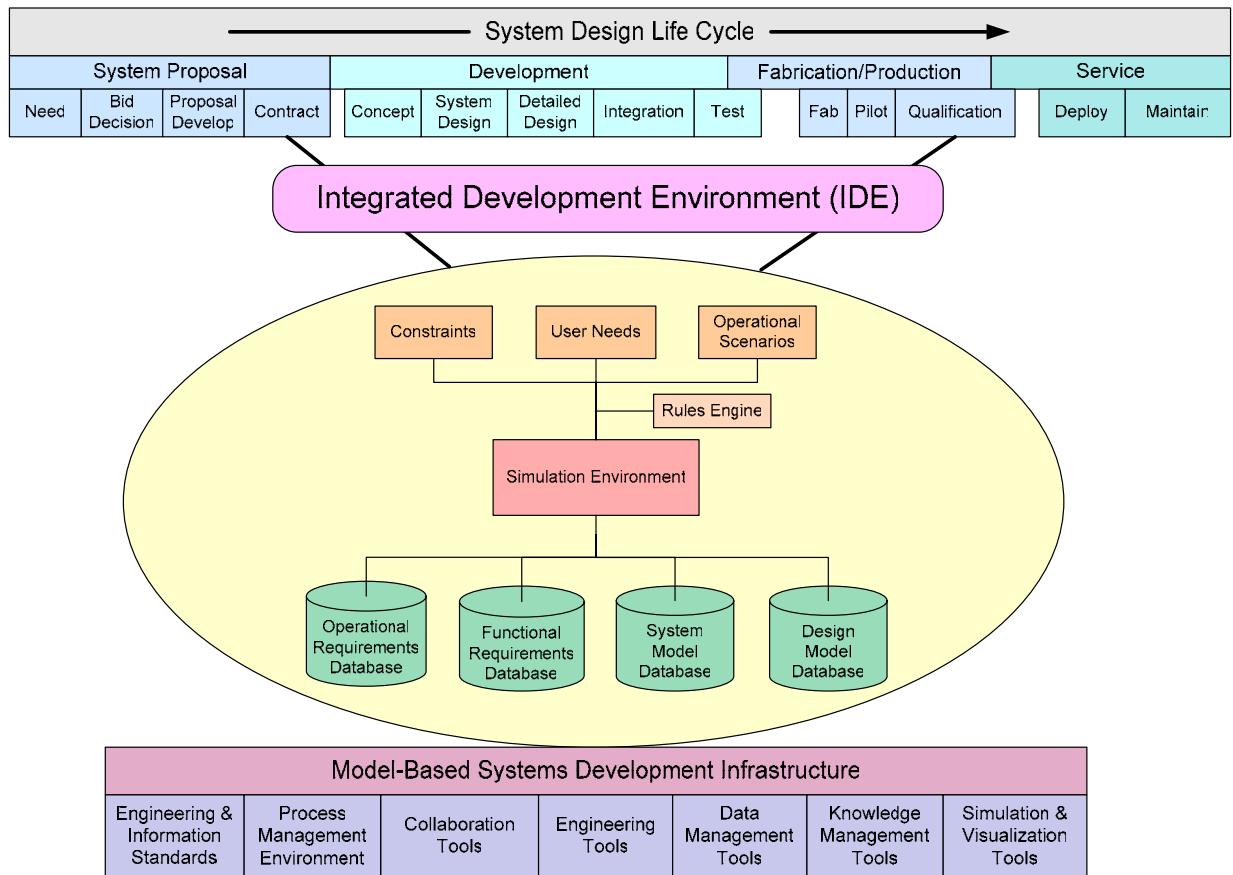


Figure 3 – Integrated Development Environment (IDE)

The specific work products that can be systematically generated using the combined power of skilled professionals, an MBD system engineering process, and an IDE include, but are not limited to:

- Early phase program development plans
- A validated and documented needs statement
- Multiple views of dependent requirements
- Integrated system concept, architecture and design documents
- Trade study and scenario analysis reports
- Systems Engineering Plans
- A Risk Management Plan
- Web-based system design configuration and document management
- Requirements traceability matrices

Moreover, all of these work products are based on integrated, accurate, and valid system development data.

Utilizing this approach can result in tangible benefits for both management and engineering teams such as:

- Mission assurance for program management by requirements-driven, systematic evolution and verification
- Improving visibility into program performance through continuous system simulations
- Reducing programmatic risk, and increased probability of development success
- Increasing accountability due to increased information transparency
- Providing a mechanism for concrete design and organization performance measurements
- Providing a rapid trade study and scenario analysis capability (holistic design inquiry)
- Providing a “real” foundation for unobtrusive collaboration across the SDLC
- Providing a basis for “roundtrip engineering”
- Delivering a platform that can support integrated software and hardware development

Each of these benefits, and their combinations, has great potential for increasing the overall value of the systems development process.

Solution Applications

An application of the proposed systems development paradigm will for the most part benefit complex development projects. This approach was developed primarily for highly distributed systems with many physical and organizational interfaces. Large aerospace programs like those in NASA's Exploration Initiative such as CEV, Project Prometheus, and Mars/Lunar would benefit greatly from the proposed systems development solution. The proposed solution emphasizes performance during up-front system definition and downstream validation, which has proven to be the "Achilles-heel" of these types of programs.

In addition to large aerospace programs, the proposed solution schema harbors great potential for application in other industrial sector applications such as shipbuilding, automotive, and electronics industries, as well as in information technology system development. In each of these cases there is a need to manage large numbers of interdependent hardware, software, and human systems possessing long-duration life cycles and global development teams.

Summary

The problem of developing complex systems is one that requires a complete understanding of the technical and management components across the SDLC. Achieving this end necessitates the modeling, analyzing, and verifying of system behavior in the context of its mission environment to ensure stakeholder needs are met. The success of this endeavor will require timely development of the design information necessary for "right-time" decision-making. The proposed system development schema can achieve this end by tightly coupling MBD techniques and procedures with an IDE that together support rapid and repeatable systems simulation. This approach provides valuable technical and management decision-making support early in the development process when the cost of correcting design deficiencies is a minimum. By addressing issues early in the SDLC, it is also possible to build robust systems that are more likely to evolve within program constraints. Finally, the proposed system development methodology represents a basis for transforming inadequate text-based system design processes into something that can support holistic design inquiry.

References

- [1] Columbia Accident Investigation Report, Volume 1, August 2003.
- [2] Defense Science Board (DSB) and Air Force Scientific Advisory Board (AFSAB), "Report of the Defense Science Board/Air Force Scientific Advisory Board Joint Task Force on Acquisition of National Security Space Programs," May 2003.
- [3] "Top Execs Say IT Can Inhibit Growth," Information Week, Eric Chabrow, August 26, 2004.
- [4] B. Blanchard, *Systems Engineering Management*, Second Edition, John Wiley & Sons, 1998.
- [5] B. Blanchard and W. Fabrycky, *Systems Engineering and Analysis*, Third Edition, Prentice Hall, 1998.
- [6] J. Martin, *Systems Engineering Guidebook*, CRC Press, 2000.
- [7] A. Sage, W. Rouse (Editors), *Handbook of Systems Engineering and Management*, John Wiley and Sons, 1999.
- [8] J. Juhasz, *Enterprise Engineering in the Systems Age*, Handbook of Technology Management, Gerald H. Gaynor (Editor), McGraw Hill, 1996.
- [9] L. Baker, P. Clemente, R. Cohen, L. Permenter, B. Purves, and P. Salmon, "Foundational Concepts for Model-Driven Systems Design," *Proceedings of the INCOSE Model Driven Systems Design Working Group*, 1997.

- [10] R. Wagner, "Applications of Modern Structured Analysis Paradigms to the Development of Complex Systems," *Second Annual Technical Symposium, National Council of Systems Engineers*, 1992.
- [11] I. Ogren, On Principles for Model-Based Systems Engineering, *Systems Engineering* 2 (1999).
- [12] S. Wall, "Model-Based Engineering Design for Space Missions," *IEEE Aerospace Conference*, 2003.