Paper

The Relationship of System Engineering to the Project Cycle

By
Kevin Forsberg and Harold Mooz — Co-Principals
Center for Systems Management
19046 Pruneridge Avenue, Cupertino, CA 95014

Presented at the joint conference sponsored by:
National Council On Systems Engineering (NCOSE) and
American Society for Engineering Management (ASEM)

Chattanooga, TN
21–23 October 1991

also presented at:
The 12th INTERNET World Congress on Project Management

Oslo, Norway
9–11 June 1994

The Relationship of System Engineering to the Project Cycle

by
Kevin Forsberg and Harold Mooz — Co-Principals

Center for Systems Management
19046 Pruneridge Avenue, Cupertino, CA 95014

Abstract. A new way of portraying the technical aspect of the project cycle clarifies the role and responsibility of system engineering to a project. This new three dimensional graphic illustrates the end-to-end involvement of system engineering in the project cycle, clarifies the relationship of system engineering and design engineering, and encourages the implementation of concurrent engineering.

Introduction

The development cycle for projects (commercial or government) usually starts with User needs, which are then translated into a feasible set of system requirements. The system requirements are progressively decomposed into baselines for segments, elements, etc., until the lowest level of detail (hardware parts or software units) are specified. The physical parts, assemblies, and/or software units are then integrated into successively higher assemblies, until the integration process is complete as evidenced by a functioning, validated system.

The traditional illustrations of this complex process present an incomplete portrait of the actual process, and tend to obscure the role of system engineering and the timely participation of concurrent engineering. As a consequence, many project team members reject the current models of the project cycle, claiming they are unrealistic and non-applicable to their situation. However, until now, nothing has been offered as an acceptable alternate and considerable confusion remains about the proper sequence of events, as well as the roles and responsibilities of the project technical team.

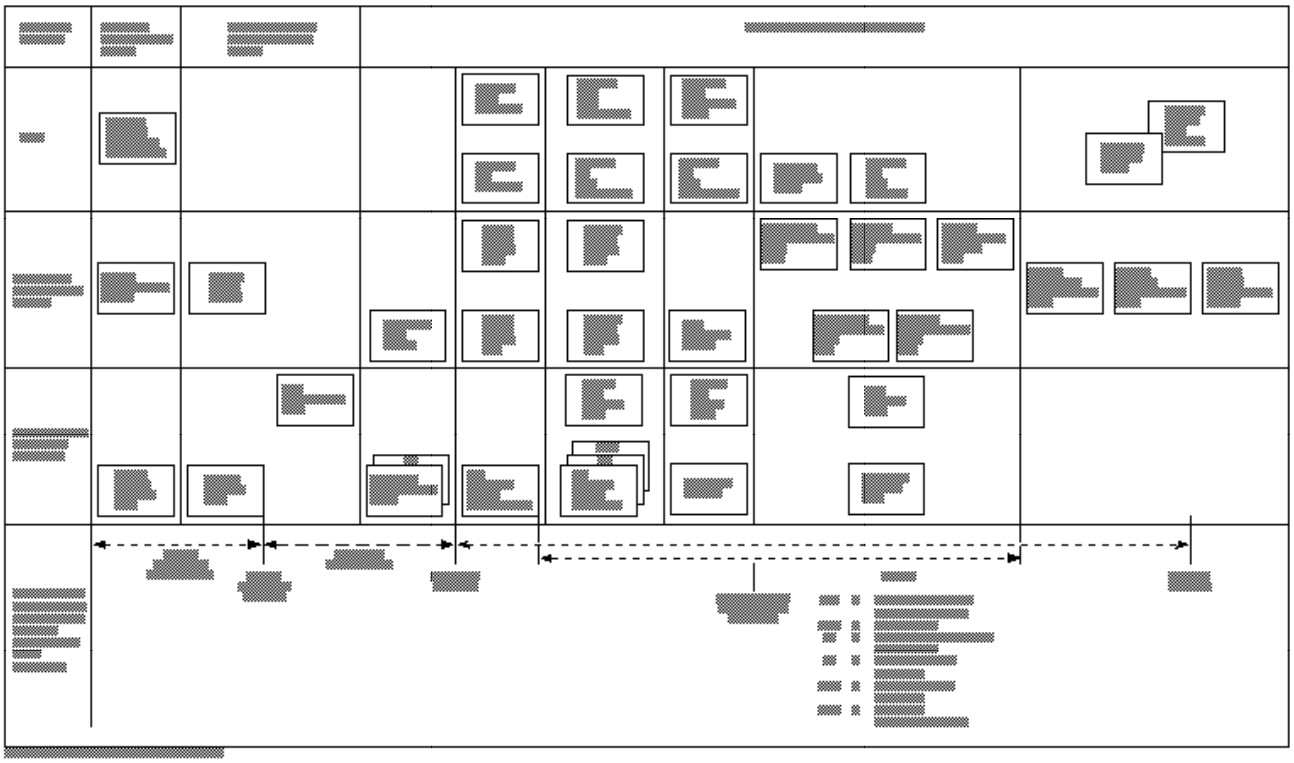
In the commercial project environment, the project cycle is often not defined, even in an informal manner, and the role of system engineering as a vital part of the project team is frequently ignored.

The Project Cycle

Current Models. The project cycle is often displayed as a linear sequence of activities moving along a horizontal line, punctuated by major reviews (System Requirements Review, Preliminary Design Review, Critical Design Review, etc.). An example of this approach is found in the DSMC Systems Engineering Management Guide (1990b) (Exhibit 1). Another graphic representation, developed by W. Royce (1970b), presents the project cycle as a series of diagonal steps vertically spaced from upper left to lower right. Project activity flows from the top to the bottom, and the process has been designated the “waterfall” model (Exhibit 2).

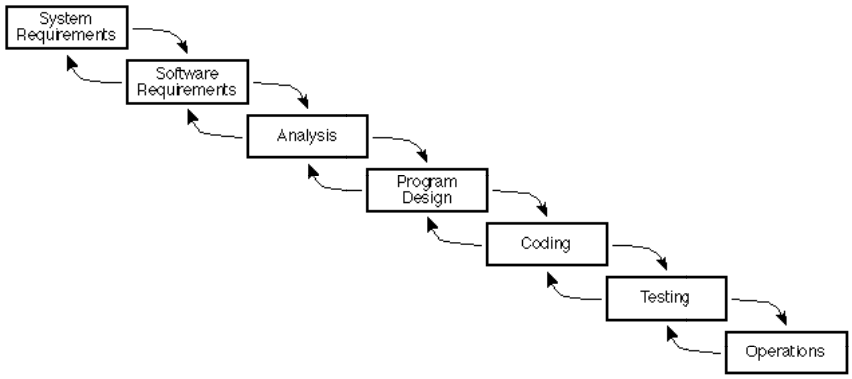
A third representation of the project cycle (Exhibit 3) is contained in DoD-STD 2167A (1988b). This exhibit illustrates hardware-related events in an upward path and software-related events in a separate downward path, conveying the false conclusion that these two vital parts of the project can be managed separately and individually, until final system integration.

All three of the above models of the project cycle share a similar deficiency: the graphics imply that work downstream cannot begin until the upstream major reviews (or control gates) have been satisfied. A common interpretation is that software coding or hardware fabrication should not begin until after the Critical Design Review has been completed. In real life there is a need to initiate software design and coding, and hardware modeling, earlier in the project cycle to ensure that User Requirements are understood and to prove technical feasibility. This need has led to the development and use of hardware and software feasibility models (called “prototyping” in software and breadboarding in hardware) in the earliest phases of the project cycle.



173 fhd 01

Exhibit 1—*System Engineering Management Guide* (1990b, Fig. 12-1) view of Technical Reviews and the System Development Cycle



173 fhd 02

Exhibit 2—View by W. Royce (1988b, Figs 2 and 3) of the implementation steps to develop a large computer program for delivery to a customer, with iterative interaction

The recent DoD-I-5000.2 (1991b) *requires* the use of "prototypes" from the start of the project, with specific relief from executive management required if "prototypes" are not to be used.[1] The disconnect between the earlier portrayals of the project cycle that fail to provide for modeling and the recent requirement for modeling has led to widespread belief that the waterfall model is wrong or not applicable.

A fourth view of the project cycle, developed by Boehm (1988b), attempts to resolve the above deficiency by addressing the need for early feasibility modeling ("prototyping") to identify risks and define appropriate action. While Boehm's spiral representation (Exhibit 4) achieves his objective, the system engineering role is still obscured.

[1]The dictionary defines a "prototype" as *the first thing of its kind; an original.* Traditionally, hardware prototypes are built to released drawings, under engineering surveillance, and produce an example or model for production to replicate. In aircraft design competition, built-to-print prototypes are flown to demonstrate achieved capability. Software engineering has distorted this definition by naming user requirements clarification models and technical feasibility models "prototypes." For further information see the final paragraph of this paper.

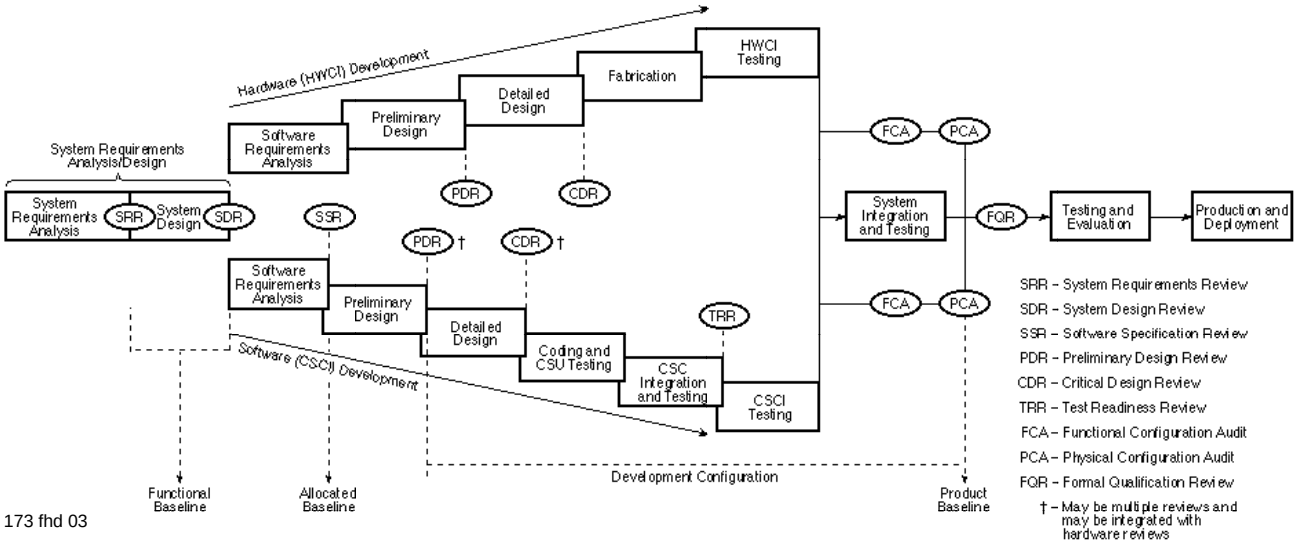
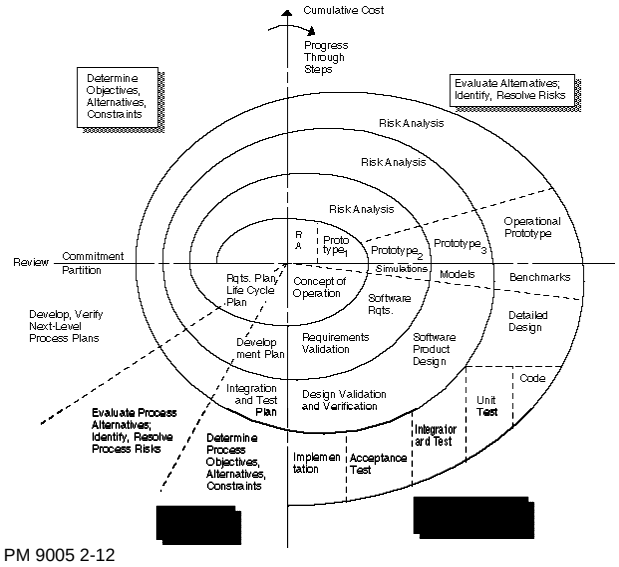


Exhibit 3—An example of system development reviews and audits, from DoD-STD-2167A (1988b), pp. 10 (Fig. 1)



PM 9005 2-12

Exhibit 4—Boehm's (1988b) spiral model of the software process

The "Vee" Model. In our approach, the technical aspect of the project cycle is envisioned as a "Vee," starting with User needs on the upper left and ending with a User-validated system on the upper right. Exhibit 5 provides a summary level overview of the cycle. On the left side of the chart, Decomposition and Definition descends as in the waterfall model. However, Integration and Verification flows up and to the right as successively higher levels of assemblies, units, components, and subsystems are verified, culminating at the system level. This summary chart follows the basic outline of the "Vee" chart developed by NASA as part of the Software Management and Assurance Program (SMAP).

The substantial advance in visualization of the technical aspect of the project cycle, and the role of system engineering, is gained by understanding the comprehensive "Vee" chart (Exhibit 6, foldout chart, "Technical Aspect of the Project Cycle"). The shaded area on Exhibit 6 is the core of the "Vee." The activities shown on the core map directly onto the simplified display of the "Vee" (Exhibit 5).

The DoD Cycle

The major project reviews from Department of Defense (DoD) MIL-STD 973 (which has replaced MIL-STD 1521B) have been used in Exhibit 6 because they are familiar to most contractors involved in government projects. Their use is not limited to DoD. The Department of Energy, Department of Transportation, NASA, and other agencies use the same or very similar control gates. These control gates (or phase transition reviews) are also applicable to commercial projects, and have been recommended as a guideline for commercial development projects which require FDA approval. Commercial project teams often resist the DoD process, incorrectly believing that the Government method can't be correct or efficient.

Detailed discussion of the "Vee" chart

Decomposition and Definition. The "Vee" chart provides a three-dimensional view of the technical aspect of the project cycle. At each level, moving into the depth of the paper (perpendicular to the surface) there are a number of parallel boxes illustrating that there may be many Segments or Configuration Items (CIs)[2] that make up the system at that level of decomposition. Also at the System level, on the left of the

[2]A Configuration Item (CI) requires a discrete development specification; individual design reviews; individual qualification testing and reporting; individual Acceptance Reviews; and individual operator and maintenance manuals. A CI should be selected to facilitate management accountability and replacement capability.

chart, the number of parallel boxes illustrates that alternate concepts should be evaluated to determine the best solution for the User's needs. At the System Requirements Review (SRR), the choice is approved and a single concept is baselined for further definition.

As project development progresses, a series of six baselines are established to systematically manage cohesive system development. The first is the "User Requirements Baseline" established by the System Requirement Document approved and put under Configuration Management prior to the SRR. The second is the "Concept Baseline" established by the Concept Definition section of the Integrated Program Summary document at the SRR. The third is the "System Performance Baseline" (or Development Baseline) established by the System Performance Specification at the SDR. The fourth is the "'Design-To' Baseline" (or Allocated Baseline) established at the series of PDRs. The fifth is the "'Build-To' Baseline" (or preliminary Product Baseline) established at the series of CDRs. The sixth is the "'As-Built' Baseline" (or Production Baseline) established at the series of Formal Qualification Reviews (FQRs). Each of the baselines is put under formal Configuration Management at the time they are approved.

The left side of the core of the "Vee" (the shaded area in Exhibit 6) follows the well-established waterfall model for the project cycle. The Control Gates define significant decision points in the project cycle. Work should not progress beyond a decision point until the Project Manager is ready to publish and control the documents containing decisions agreed to at that point.

Unlike the commonly held view of the waterfall method, there is no prohibition against doing detailed work early in the cycle. In fact, hardware and software feasibility models may be required at the very first stage (User Requirement Analysis and Agreement) in order to clarify the User Requirements Statement, and to ensure that the User is not asking for an unachievable result such as an antigravity machine. Early application of involved technical and support disciplines is an essential part of this process; this is in fact the implementation of concurrent engineering (see IDA report, 1990b).

As the project progresses, detailed analyses, risk identification, and risk reduction modeling continues. This is shown on the chart by the vertical and descending off-core activities.

While technical feasibility decisions are made in the off-core activities only decisions at the core-level are put under Configuration Management at the various Control Gates. Off-core activities, analyses, and models are performed to substantiate the core decisions and to ensure that risks have been mitigated or determined to be acceptable. The off-core work is not formally controlled, and will be repeated at the appropriate level to prepare justification for introduction into the baseline definition.

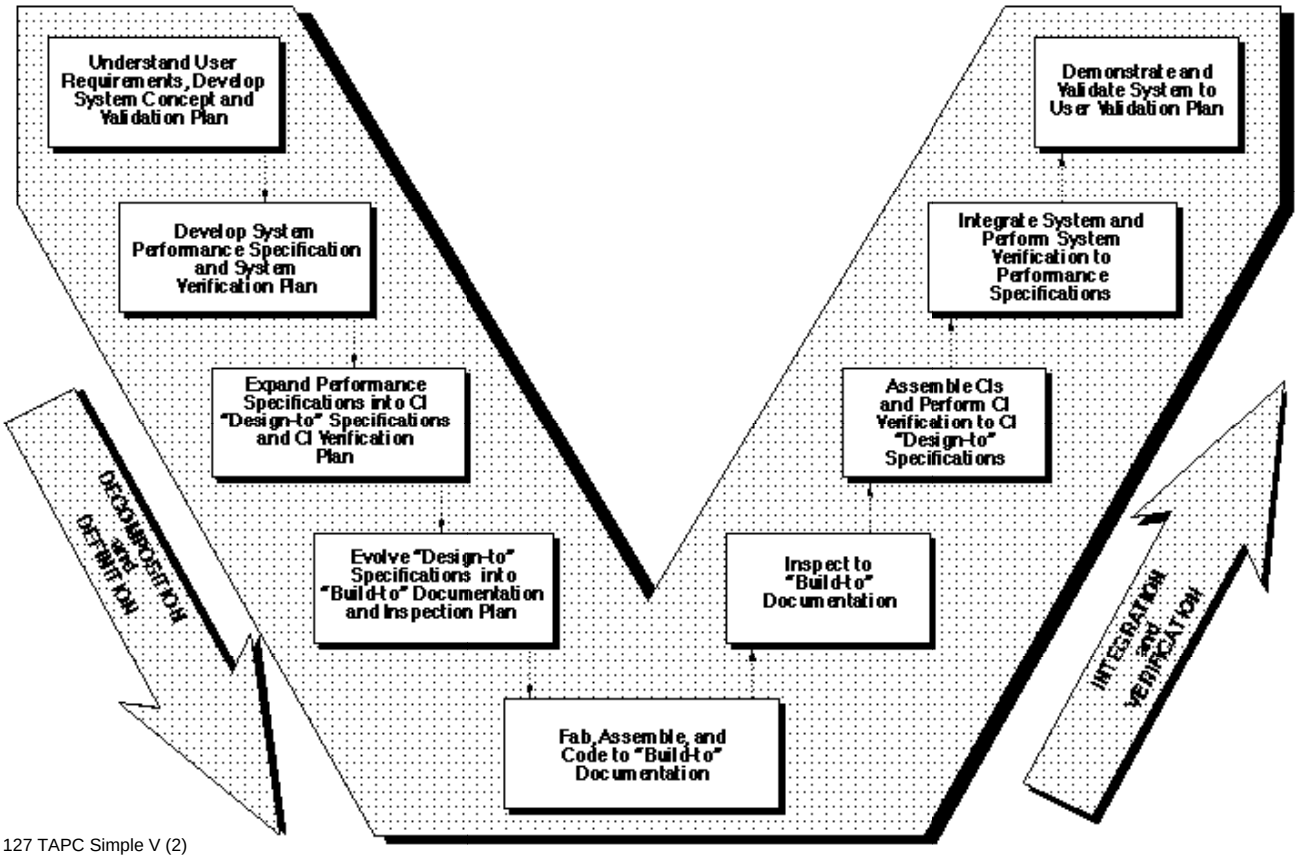


Exhibit 5—Overview of the Technical Aspect of the Project Cycle

The multiple arrows descending from the bottom of the left side of the core of the "Vee" indicate that there can, and should be, sufficient iteration downward to establish feasibility and to identify and quantify risks. Upward iteration with User Requirements (and levels leading to them) is permitted, but should be kept to a minimum unless the user is still generating requirements. The User needs to be cautioned that changes in requirements during the development process will cause positive or negative changes in the predicted cost and schedule and may cause the project to be not cost or schedule effective.

Often in software projects upward confirmation of solutions with the User is necessary because User Requirements cannot be adequately defined at the start. Even for software projects, however, iteration with User Requirements should be stopped at PDR, or the project will not converge to an acceptable solution within manageable cost or schedule bounds.

Modification of User Requirements after PDR should be held for the next model or release. If significant changes to User Requirements must absolutely be made after PDR, then the project should be stopped and restarted at the start of a new "Vee," reinitiating the entire process. The repeat of the process may be quicker because of the lessons learned, but all steps must be redone.

Time and project maturity flows from left to right on the Vee. Once a Control Gate is passed, iteration is not possible backward. Iteration with User Requirements, is possible only vertically as illustrated on the Vee.

Incremental Development. If the User Requirements are too vague to permit final definition at PDR, one approach is to develop the project in predetermined incremental releases. The first release is focused on meeting a minimum set of User Requirements, with subsequent releases providing added functionality and performance. This is a common approach in software development.

The representation of the incremental development approach is easy to illustrate using the "Vee" chart. All increments have a common heritage down to the first PDR. The balance of the project cycle has a series of displaced and overlapping "Vees," one for each release.

Concurrent Engineering. If high iteration with User Requirements is required after the System Design Review (SDR), it is probable that the project has passed early Control Gates prematurely, and it is not sufficiently defined. One cause of premature advance is that the appropriate technical experts were not involved at early stages, resulting in acceptance of requirements and design concepts which cannot be built, inspected, and/or maintained. System Engineering is responsible for involving key personnel (to address human factors, safety, producibility, inspectibility, reliability, maintainability, logistics, etc.) at each step, starting with risk analyses and feasibility studies in the Concept Definition phase. This does not require a dedicated team. However, it does require a proactive System Engineer who can ensure that appropriate expert advice and detailed assistance is applied to all areas of project risk.

The detailed evaluation of operational feasibility, identification of driving technologies, development of software and hardware feasibility models, and identification of system risks must be done by or perceptively reviewed by technical experts. These tasks are the off-core activities shown in Exhibit 6, and are key to the success of this project cycle concept.

Note that at the fourth and fifth levels on the chart (Exhibit 6), the tasks break into three parallel efforts: operations (including manual operations), hardware, and software. The System Engineer must be sufficiently competent to direct meaningful trades between these areas. Many system functions can be performed by any of the three areas, and for most projects the optimum choice is not obvious. Engineers sometimes misapply creativity to automate functions that can easily be done manually, and to create sophisticated, intellectually satisfying designs that are hard to build or expensive to operate. Early involvement by human factors and manufacturing specialists can avoid such design concept errors.

It is also essential that the system engineer address both test and facilities implications of the alternate concepts as part of the Study Period trade-off analysis process. These additional par-allel efforts could have been illustrated on the chart (Exhibit 6) to highlight their importance, but to reduce chart complexity, they were only implied.

Test philosophy and planning are part of the Verification and Validation Plans identified earlier and are developed in conjunction with the system decomposition process. The design of experiments, the design of test equipment, and the development of hardware, software, and operational test procedures, are all part of the oversight function of System Engineering that should ensure the tests will produce the tangible evidence necessary to prove System Verification.

Making sure that manufacturing producibility specialists, and facility design engineers participate early in the system development process is an example of concurrent engineering. Designing the correct solution in an orderly process is much more cost and schedule effective than retroactively correcting a defective design at a later date.

Role of System Engineering. The interface between System Engineering responsibility and Design Engineering responsibility is illustrated by the rectangle on the right side of the chart. Above the line System Engineering is responsible, and Design Engineering provides technical assistance. Below the line Design Engineering is responsible, and System Engineering performs technical audit. Note that System Engineering is influential throughout the entire project life cycle, from User Requirements development to system decommissioning.

Technology Insertion. Projects are sometimes initiated with known technology shortfalls, or with areas for which new technology will result in substantial product improvement. Technology development can be done in parallel with the project evolution, and inserted as late as Preliminary Design Review. The technology development would be represented by a horizontal bar off the core, at the Configuration Item level (or below), and would be managed and statused by the project manager and System Engineer as a critical activity (Exhibit 6).

Integration and Verification. Descending down the left side of the "Vee" represents *Decomposition and Definition*. Ascending the right side of the "Vee" is the process of *Integration and Verification*.

At each level there is a direct correspondence between activities on the left and right sides of the chart. This is deliberate. The method of verification must be determined *as the requirements are developed and documented at each level*. This minimizes the chances that requirements are specified in a way which cannot be measured or verified.

Even at the initial and highest level, as User Requirements are translated into system requirements, the system verification approach must be determined that will prove that the system does what it is required. The verification process can drive cost and schedule and may in fact be a discriminator between alternate concepts.

Note the overt distinction on the right of the core between *verification* and *validation*. Verification is the process of proving that each product meets its specification (*"Have we built the system right?"*). Validation is the process of demonstrating (as opposed to proving) that the product satisfies the User Needs, "regardless" of what the system specification requires (*"Have we built the right system?"*).

Note also the parallel pattern of Test Readiness Reviews (TRRs) and Acceptance Reviews (ARs). For some reason the prevalent impression in the industry is that Test Readiness Reviews are only done at high system levels, and then only if the customer requires them. It is the System Engineer's responsibility to ensure that Test Readiness Reviews are performed prior to *all* testing and with customer participation whenever official sell data is to be developed by the tests.

Process versus Sequence

System Analysis and the Design Process. The "Vee" chart addresses the technical aspect of the project cycle and represents the sequence of project events. The system engineering process (Exhibit 7) illustrates the activities the System Engineer must perform at each level of the project cycle during system decomposition and definition.

It is difficult to explain system engineering activities during the project life cycle without providing a clear distinction between the process and the cycle. The orthogonal model (Exhibit 8) was developed to illustrate the relationship of the two, and to emphasize that the system engineering process is repeated at every level of the cycle, and may be repeated many times within a phase.

System Verification and Integration Process. The system engineering process during integration and verification is illustrated in Exhibit 9.

The orthogonal model (Exhibit 10) illustrates the relationship of the process and cycle during each level of integration and verification.

System Engineering Definition. As a result of the foregoing, System Engineering can now be more accurately defined as the application of the *System Analysis and Design Process* and the *Integration and Verification Process* to the logical sequence of the *Technical Aspect of the Project Cycle*.

System Development Definitions. The authors would like to use this forum to start a movement for clarification of misused terminology in the Technical Development process specifically with regard to Technical Models. Common industry terms include Breadboard, Brassboard, Engineering Model, Mock-up, Simulation, Prototype, Protoqual, etc. The layman has a difficult time deciphering these terms and experienced personnel frequently misapply the referenced terms. Moreover, these terms used in contract documents can lead to confusion and even contract default. A more enlightened approach would be to always use the term "model" prefaced by the use terminology, i.e., User Requirements Model, Technical Feasibility Model, Physical Fit Model, Field Test Model, Concept Demonstration Model, Test Simulation Model, Production Demonstration Model, etc.

References

Boehm, B.W., "A Spiral Model of Software Development," in *Tutorial: Software Engineering Project Management*, edited by R.H. Thayer, IEEE Computer Society Press, Washington D.C., 1988, pp. 128–142.

Department of Defense, DoD STD 2167 A, "Defense System Software Development," 29 February 1988, pp. 10.

Department of Defense, DoD I 5000.2, "Defense Acquisition Management Policies and Procedures," February 1991.

Defense Systems Management College, *Systems Engineering Management Guide*, Jan 1990, pp. 12-2.

Institute of Defense Analysis (IDA), *Concurrent Engineering*, 1990.

Royce, Winston W., "Managing the Development of Large Software Systems," *Proceedings, IEEE WESCON*, August 1970, pp. 1–9. Reprinted in *Tutorial: Software Engineering Project Management*, edited by R.H. Thayer, IEEE Computer Society Press, Washington D.C., 1988.

Acknowledgment

The authors want to express their appreciation for the technical contributions and perceptive critique by their friend and colleague, Mr. Richard Roy. Mr. Roy was a major contributor in the development of the "Vee" chart, and to the expression of the underlying philosophy on System Engineering.

About the Authors

Kevin Forsberg. Dr. Forsberg draws on 27 years of experience in Applied Research, System Engineering, Program, and Proposal Management, followed by seven years of successful consulting to both government and industry. He received the NASA Public Service Medal for his contributions to the Space Shuttle program. He is one of the co-founders of the Center for Systems Management.

Dr. Forsberg received his B.S. in Civil Engineering at Massachusetts Institute of Technology and his Ph.D. in Engineering Mechanics at Stanford University.

Hal Mooz. Mr. Mooz draws on 22 years of experience in System Engineering and Program Management, followed by ten years of successful consulting to government and industry. Mr. Mooz has won and successfully managed highly reliable, sophisticated satellite programs from inception to operations. He is one of the co-founders of the Center for Systems Management.

Mr. Mooz received his M.E. in Mechanical Engineering from Stevens Institute of Technology.

Technical Aspect
of the
DoD Project Cycle
(MIL-STD-1521B Compliant)

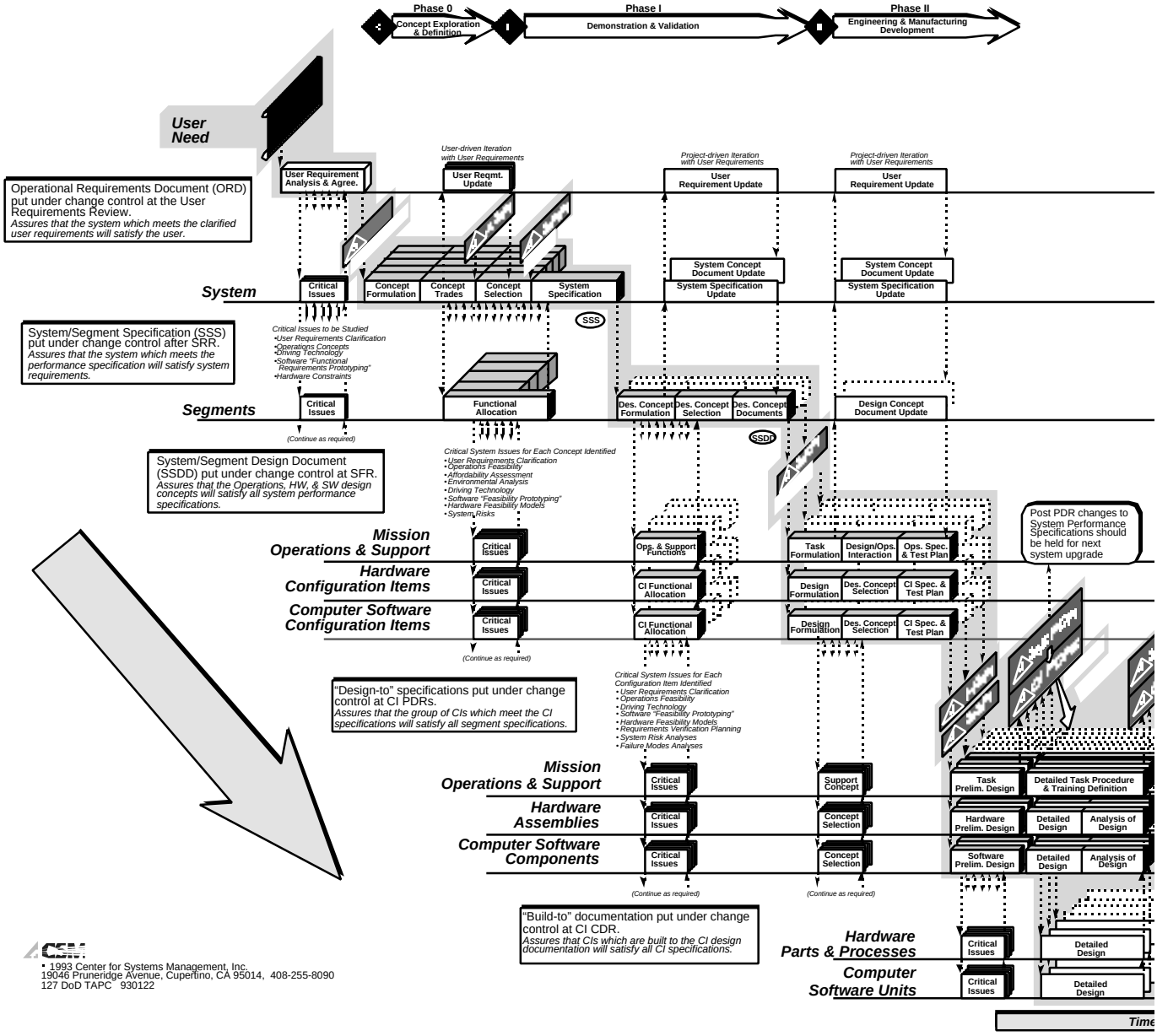


Exhibit 6A—Technical Aspect of the Project Cycle
(Part 1 of 2)

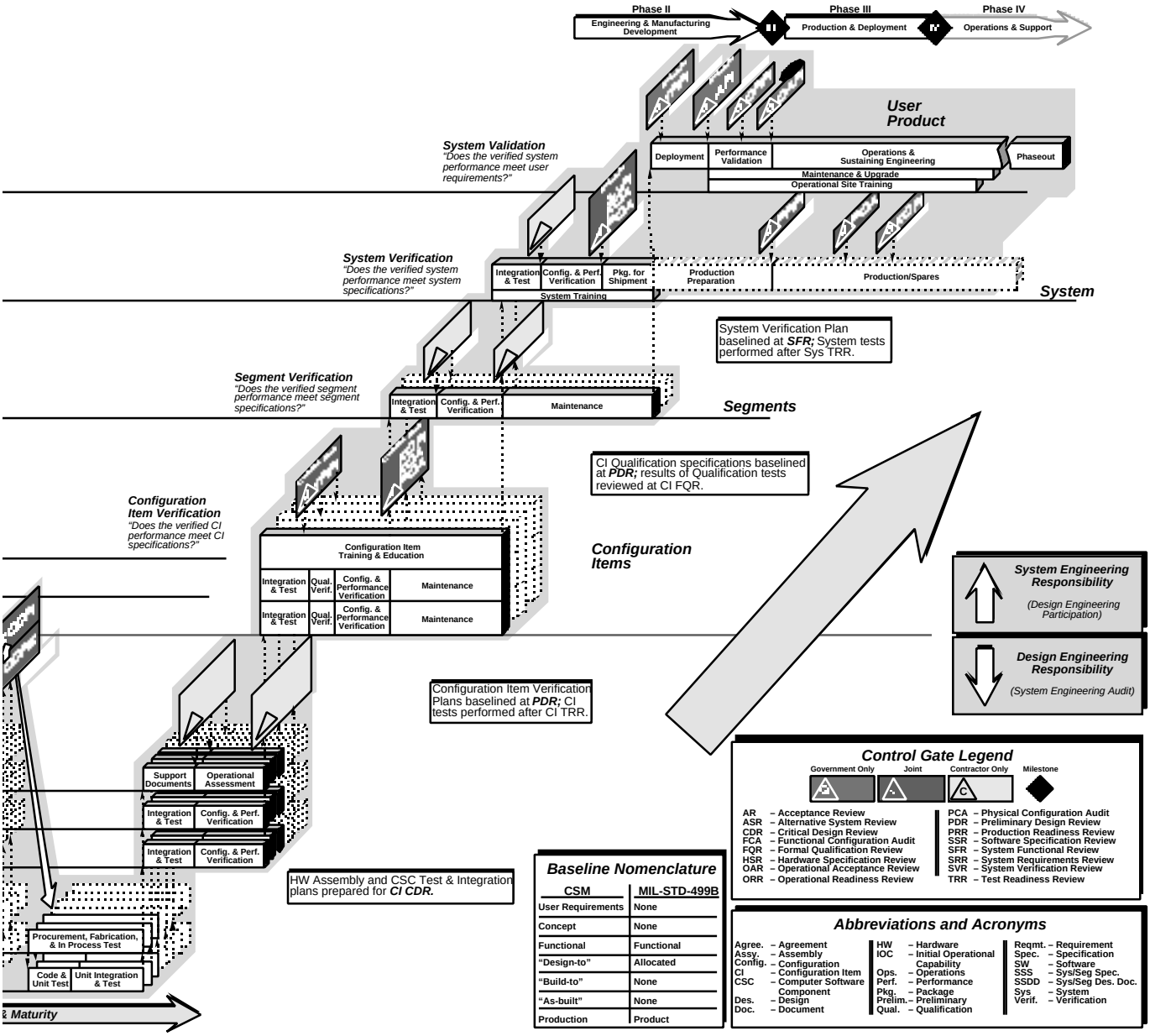


Exhibit 6B—Technical Aspect of the Project Cycle
(Part 2 of 2)

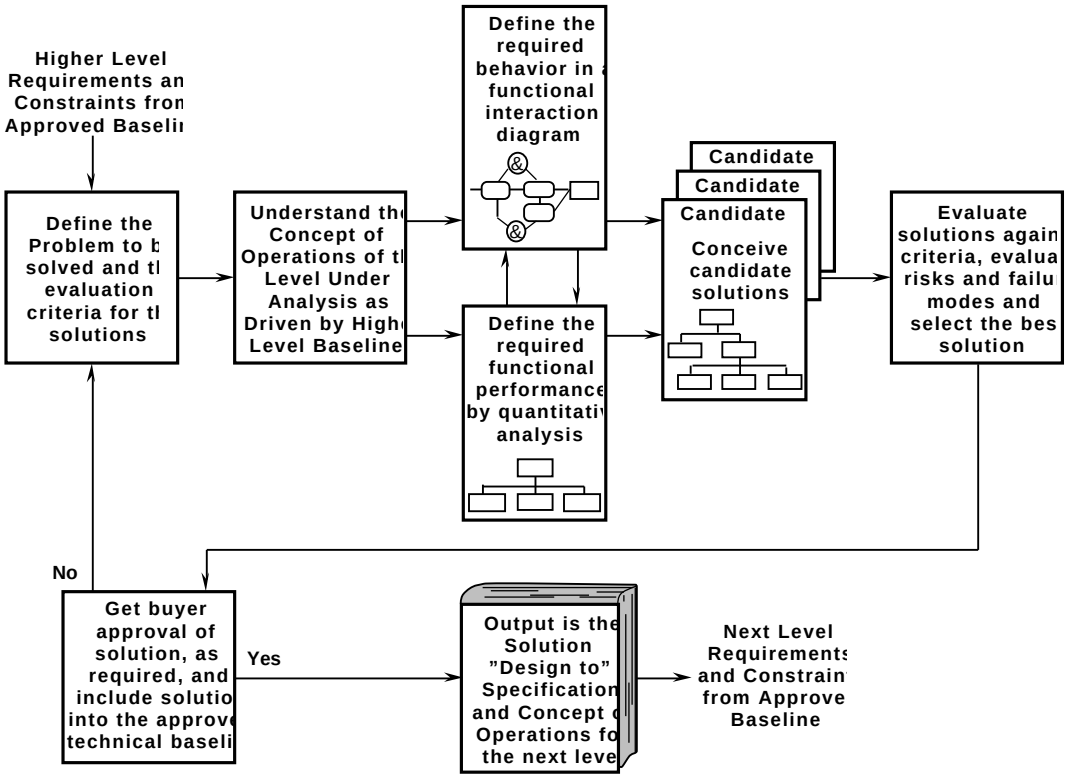


Exhibit 7—System Analysis and Design Process

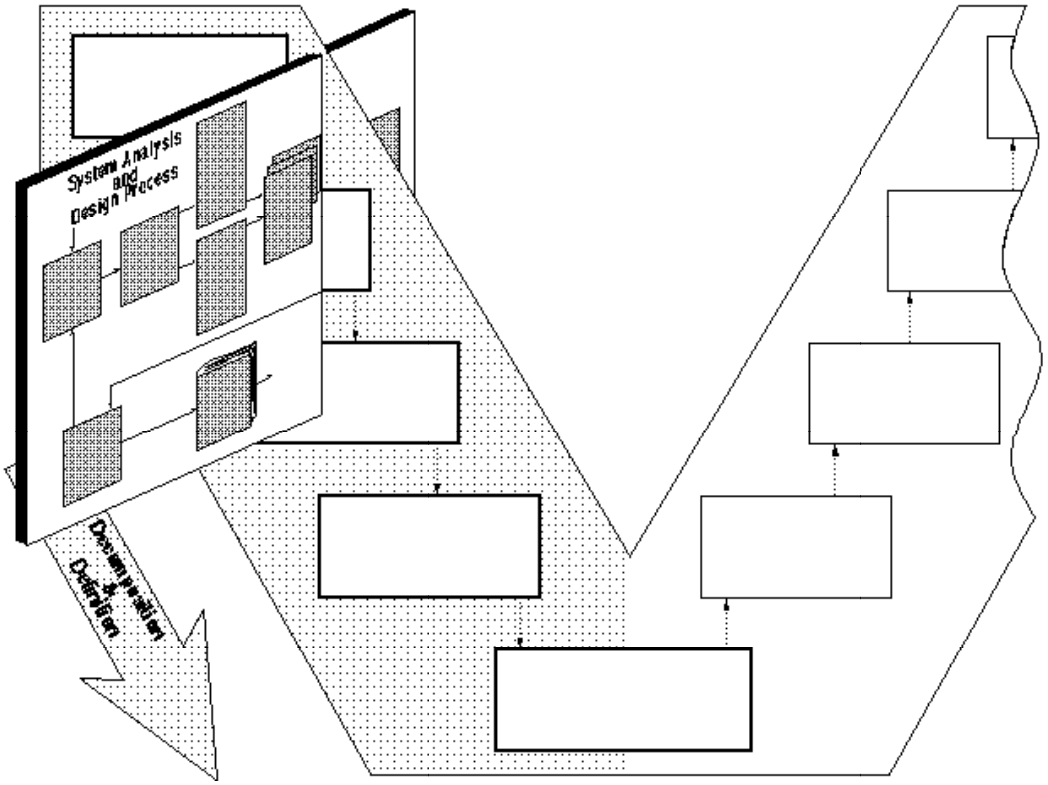


Exhibit 8—Application of the System Analysis and Design Process to the Technical Aspect of the Project Cycle

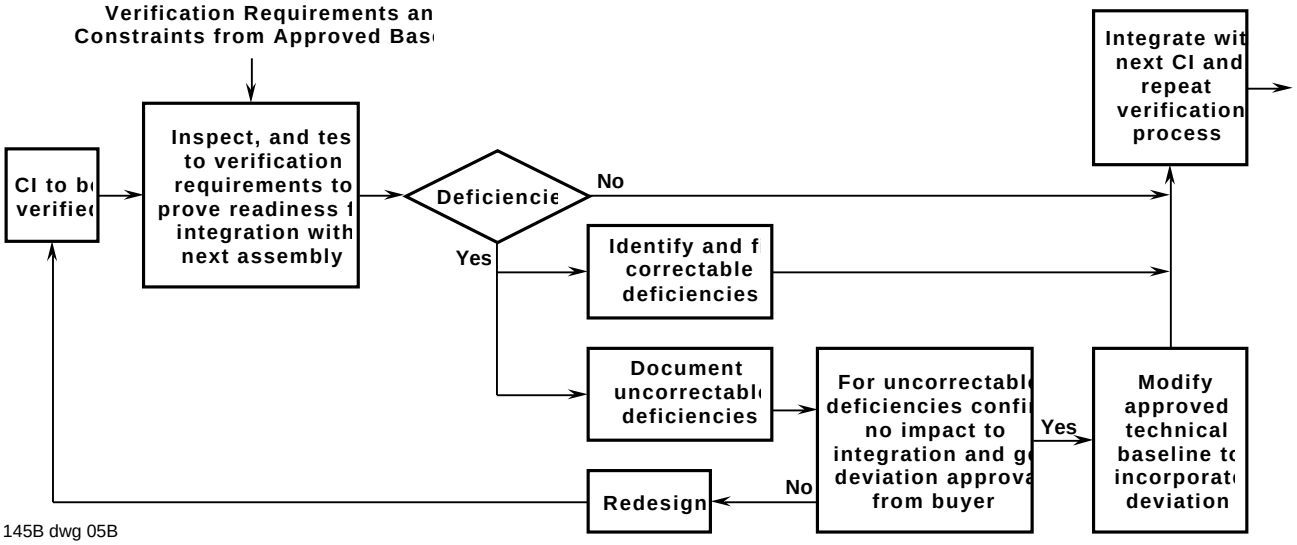


Exhibit 9—System Verification and Integration Process

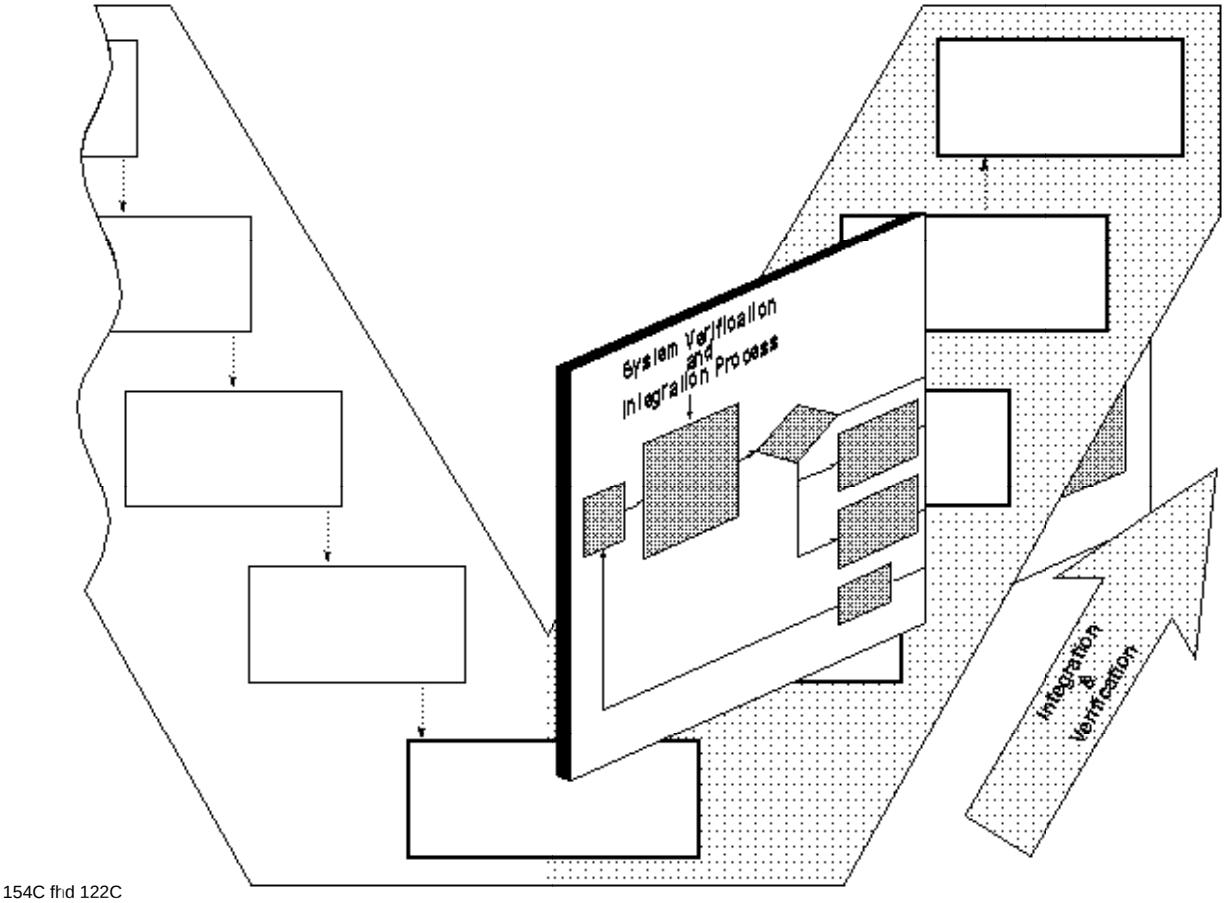


Exhibit 10—Application of the System Verification and Integration Process to the Technical Aspect of the Project Cycle

Addendum #1 to

The Relationship of System Engineering to the Project Cycle

Barry Boehm's Spiral Process Model, as referenced in our NCOSE technical paper, *The Relationship of System Engineering to the Project Cycle*, is becoming very well known and is being used as a guide in system development. This explanation is provided to relate the Spiral Process Model to CSM's Vee Process Model which is applicable to all commonly understood development processes.

The Spiral Process Model was conceived to illustrate a risk driven software management process that is based on successive risk analysis model development until sufficient confidence exits that the standard Waterfall Process Model can be used with confidence.

The Spiral Process offers the two options: one to retain the risk analysis model if it is sufficiently valuable to be built upon into a more comprehensive analysis model as in the Evolutionary Development Process; or second, to discard the risk analysis model as too impractical to build upon, and proceed with the efficient development of a more practical, more advanced risk analysis model based on all the lessons learned to date and incorporating the risk reduction decisions decided upon. In either case, the most significant development risk has the highest priority and should be addressed next.

One possible point of confusion with the Spiral Process Model is that it is not based on a typical horizontal linear time base as are most project cycles, and as a result may incorrectly convey the impression that the process doesn't take any appreciable project time. The influence of time can be shown by unraveling the spiral and illustrating it against a horizontal time base where it will appear similar to a sidewinder snake. The sidewinder form will descend when pursuing risk analysis activities and models, and will ascend when demonstrating the results with recommended approaches to the customer/user(s) to obtain feedback and approval necessary for the next descent in pursuit of the next risk analysis.

This same process is illustrated in the upper left portion of the Vee Process Model. The risk addressing, User Requirements Clarification Models, Operations Feasibility Models, and Technical Feasibility Models are all illustrated in the downward, off the Vee core, risk management activities. Customer/user evaluation of the technical solutions is shown in the upward off-core activities. The progression from on-core risk identification, down to off-core risk analysis model development, and then up to off-core user evaluation of recommended approaches, and back down to on-core baseline approval, is the sidewinder representation of the Spiral Process Model. A low risk project would pursue little or no off-core risk analysis activity and the project would follow the traditional Waterfall Process Model approach through the Decomposition and Definition Phases of the Vee.

Although the Spiral Process Model incorporates technical review at each upward pass of the left horizontal axis, the illustration fails to emphasize baseline management and configuration control that is an essential discipline to good system management. The Spiral Process Model should illustrate the establishment of the technical baseline and the subsequent maturation of the baseline with each turn of the spiral and crossing of the left horizontal axis.

The Vee Process Model illustrates baseline management in that each time the customer approves the incorporation of risk reduction results, then those results become baselined on the core of the Vee at the associated control gate. Subsequent changes to the approved baseline require formal approval of those affected.

The Vee Process Model can also be used to understand and illustrate the Incremental Development Process. With Incremental Development, the Vee Process Model fans into a set of displaced Vees that are offset starting at the level of decomposition affected by the incorporation of the approved enhancements.