RAPID PROTOTYPING USING OBJECT-ORIENTED MODELING AND TESTBED SIMULATION FOR COMPLEX REAL-TIME SYSTEMS

John G. Shea¹, Rammohan K. Ragade², Jeffrey W. Dunn³

ABSTRACT

This paper describes a new development in simulation testbed architecture which is particularly amenable to rapid-prototyping applications. This architecture comprises a "three-tier" modular hybrid simulator made up of actual system components, conventional numeric functional simulation models, and a set of "behavioral" knowledge-based representations which model hardware and software functions in terms of functional states, inputs, and outputs. The three sets of simulation modules, representing three fundamentally different functional representation paradigms, are strongly linked and embedded in an object-oriented knowledge-capture and design-representation architecture. The system thus defined provides a very flexible model-building and scenario-based simulation environment that is readily reconfigurable in support of a wide variety of investigative applications.

The paper first provides an overview of the requirements analysis process which led to the development of the concept. A discussion of the major testbed components (simulation module "core", testbed resource monitor/controller, and the intelligent knowledge capture system) follows, with emphasis placed upon the unconventional aspects of the design. The object-oriented modeling techniques and frame-based data structure which support the rapid-prototyping capabilities of the testbed are next described, together with a discussion of the capabilities of the system to interactively support the building and evaluation of rapid-prototype applications. A final descriptive section outlines a problem- and knowledge-structuring approach based on the concept of a design-oriented "Figure of Merit" and discusses the advantages and disadvantages of such an approach. The paper concludes with a short discussion of the perceived strengths and weaknesses of the concept and an outline of planned future work.

¹ Advanced Engineering Development, Code 50 (PHALANX), Naval Ordnance Station, Crane Div., Naval Surface Warfare Center, Louisville, Kentucky 40214-5001

² Professor, Engineering Mathematics And Computer Science, University of Louisville, Louisville, Kentucky, 40292.

³ Graduate Student, Engineering Mathematics And Computer Science, University of Louisville, Louisville, Kentucky, 40292.

²³ were funded in part by a contract with CACI/NAVSURFWARCCEN ORDSTA.
INTRODUCTION

In 1991 the Naval Ordnance Station at Louisville, KY set out to develop a multipurpose, multiuser hybrid simulation capability for the engineering support of the PHALANX (Close-In Weapon System MK 15) Shipboard Self-Defense System (Figure 1). The definitive PHALANX Emulation/Simulation/Maintenance Facility (ESMF), as the new testbed has come to be known, will provide the following primary services to the PHALANX engineering community:

* Preliminary design, rapid prototyping and evaluation of conceptual and developmental hardware/software functions and performance.

* Investigation of reported in-service problems and the rapid prototyping of proposed fixes.

* Development, prototyping, editing, Verification and Validation (V&V), configuration control, and production of operational software "updates" for Fleet distribution.

The primary goal of PHALANX testbed development is to provide both a rapid prototyping environment (for concept development) and a tailored, scenario-based simulation environment (for concept evaluation) within a single facility. Figure 2. presents an overview of the testbed's major elements.

Our approach to the testbed design implements a modular "three-tiered" concept, where actual system hardware, procedural software models (both existing and newly developed), and new knowledge-based software models embedded in an Object-Oriented architecture are combined into a single integrated facility. This novel simulation architecture will provide a high degree of (user-selectable) modeling fidelity, be capable of real-time operation, and will be readily reconfigurable in support of a wide variety of investigative applications, all at a relatively modest acquisition cost. The keys to the design lay in the use of an intelligent Knowledge Capture System and the adaptive knowledge-based modeling of equipment functions in terms of states, inputs and outputs; i.e., the representation of system operation by the interactions of a set of "expert systems" with selectable levels of modeling fidelity. Selective use of these modules in place of their more computationally-intensive procedural software equivalents significantly reduces the demands on the testbed computer(s), with little or no impact on overall simulation fidelity.

In the context of this paper, "rapid prototyping" is defined as the exercise of modeling and evaluating permutations of a limited set of conceptual design features which have been identified as (or are suspected to be) relevant in:

(a) resolving (or demonstrating the resolution of) certain form, fit, functional, or performance issues, and/or

(b) discovering suspected, but as yet unidentified, issues of the above types.

The modular testbed simulation concept, when combined with frame-based knowledge and data representation and an object-oriented intelligent knowledge-capture and design modeling paradigm, is particularly supportive of hardware, software, and combined hardware/software rapid-modeling applications. This paper describes the PHALANX Testbed rationale and those system features that will facilitate rapid-prototyping applications.

CONCEPTUAL APPROACH

To date, most engineering simulation models have been of two main types; (a) high-level, monolithic models built for the examination of system employment or performance issues, or (b) highly detailed models of subsystems/components focused on the evaluation of specific, limited sets of performance parameters. Monolithic models are notoriously unwieldy and difficult to maintain; moreover, such problems tend to worsen as system complexity increases. Detailed, focused models, on the other hand, are very likely to miss performance parameters that only become critical at higher levels of system indenture and/or ignore vital interfacing aspects of the design. Clearly, neither type of simulation model fully meets the requirements of flexibility and comprehensiveness called for by the rapid prototyping concept.

Several organizations have realized this and have set out to develop new simulation paradigms that will allow more effective modeling and evaluation of today's complex, multipurpose weapon
system designs. For example, ongoing work at the Engineering Design Research Center of Carnegie-Mellon University has centered on developing several "framework toolkits" to support concurrent interdisciplinary engineering functions/tasks. This work has resulted in the development of several integrated testbed concepts and designs [1]. Another example of an integrated approach to engineering design is the CAD Framework Initiative standards work currently underway at the MCC CAD Framework Laboratory [2].

In a remarkable number of instances, these organizations' investigations have converged on the "hot testbed" or "hybrid" simulator concept, where the simulation facility may include both functional computer software models and actual hardware subsystems operating as an integrated whole. Sometimes termed "hardware in the loop" simulators, these facilities are uniquely characterized by a capability for interconnecting both software models and actual hardware into a "tailored" simulator adapted to the job at hand. Perhaps the most famous example of such a hybrid simulation facility is the National Test Bed, currently under development by the Strategic Defense Initiative Office (SDIO) for use in evaluating "Star Wars" candidate designs.

Conventional hybrid simulators exhibit their own peculiar set of problems, however. Such simulators are complex and costly, are difficult to adapt to a multiuser environment, and require hardware/software substitution and/or rewriting of simulation modules to represent newly-specified system configurations. These limitations have, to date, made the hybrid simulator a poor choice for a system designed to support a rapid prototyping environment. An approach which shows great promise of mitigating these limitations, however, involves extending the concept of "hybridization" one more step, to include an adaptive architecture based on object-oriented "rapid modeling" and to undertake the development of a control and knowledge-capture architecture based on "metaknowledge" concepts, as discussed in Negoita and Ralescu [3].

When implemented on a hybrid simulation (testbed) architecture capable of both model-building and model-evaluation, Rapid Systems Prototyping can be said to consist of Rapid Concept Development, Rapid Modeling and Rapid Concept Evaluation. In this context, the primary testbed design question becomes one of model "fidelity" and its effects on testbed capabilities and efficiency. Accordingly, the concept of rapid modeling and the method(s) of implementation selected assume critical importance in the development of an effective rapid prototyping testbed.

Object-oriented rapid modeling of a conceptual design, by its very nature, allows investigation of both breadth of system-level impact and sufficiency of subsystem conceptual detail, something not easily achievable in a conventional modeling approach. An object-oriented modeling technique is capable of addressing both generalized inter- and intra-system issues associated with a particular conceptual subsystem design and, using the intrinsically high degree of indirection associated with such a technique, analyses of singular or exceptional behaviors at the subsystem level.

We are investigating two conceptually distinct methods of rapid modeling implementation. In one method, the spontaneous-generation approach, several models, each incorporating a slightly different perspective on the relevant design issues, are generated and exercised simultaneously. The results of the exercises are evaluated and the next generation of models built on the evaluation results. A mutation approach, on the other hand, represents design variations at the "micro" level, where each successive generation of a prototypical model varies slightly, in one or two parameters, from its predecessor. (A mutation approach to model generation suggests a genetic algorithm paradigm; this concept may be especially useful in optimizing or tuning the performance of a specific model type.)

When viewed from this perspective, the combination of a multilevel testbed architecture and an object-oriented rapid prototyping approach allows us to implement a rapid prototype of any subsystem and immediately propagate the impacts of the rapid prototype design through the other subsystems making up the total system. (Note here that modeling fidelity will be the primary determinant of the sufficiency of the system-level aggregation and therefore, the realism of simulation results.) This approach will therefore allow not only evaluation of the suitability of a particular conceptual design but also provide insight into derivative cost and resource requirements.
In summary, we are currently designing and prototyping a "three-tier" testbed architecture using actual system hardware, procedural simulation models, and a "third tier" of object-oriented software representations which; (a) model the functional behavior of the hardware and procedural software tiers and (b) facilitate the capture of new design knowledge represented by conceptual prototypes. Our work to date has indicated that the key to successful development lays in striking a workable balance between "adaptability" and "realism" in the testbed modeling paradigm. That is, we must maximize this paradigm's ability to model the functioning of physical and software systems in sufficiently generic a manner to support adaptation to a wide range of prototype design and performance characteristics and to maintain a realistic simulation environment while doing so.

TESTBED DESIGN OVERVIEW

The primary mission of the PHALANX ESMF is the support of hardware and software development through the establishment of an in-house rapid prototyping simulation capability based on the hybrid simulator concept. The major derived requirements for the testbed will include abilities to:

(a) provide realistic, dynamic, real-time simulation and emulation environments,

(b) be easily reconfigurable for specific prototyping tasks, and

(c) provide multiple levels of modeling fidelity.

The real-time requirement, coupled with the need to incorporate selectable levels of modeling fidelity for both existing and new designs, provide the primary constraints governing both cost and performance capabilities of the facility.

Modeling fidelity (here defined as the degree to which a model accurately represents the behavior of a real system in a real environment) is the central design issue in the development of a useful PHALANX testbed. When modeling a complex system, the desired fidelity usually drives both model complexity and development cost. When we add a requirement that the model be "real-time", we also place performance demands on the supporting hardware that may be very costly (or even impossible, given the state of the art) to meet. We must therefore design to "(just) worst case"; that is, we must implement just that degree of fidelity that is sufficient to accomplish the most demanding of prospective testbed applications - no more, no less.

Real-time simulation of systems as complex as PHALANX presents significant problems, which are both mitigated and exacerbated by the "hardware in the loop" feature of the testbed concept. For example, computational throughput requirements for an all-software, real-time PHALANX simulation with sufficient fidelity to support operational software development can easily exceed the capabilities of current serial processors, making a costly multiprocessor design approach necessary. (Table 1. contains estimated performance requirements for a high-fidelity real-time procedural simulation of PHALANX system operation.) Since a cost-effective PHALANX testbed must support a wide variety of current needs, as well as unknown future requirements, any technique applied to reduce throughput requirements must be examined very carefully. Attempts at "simplification" of the simulated subsystem functions, for example, will be severely restricted by the need to maintain maximum design flexibility. Similarly, reducing throughput demands by means of "pseudo real-time simulation" (where a simulation executes the function(s) of interest in real-time one frame at a time, delaying to allow updating of rest of the models before executing the next frame) requires that the PHALANX embedded computer be capable of executing in a "suspend" mode during the updating intervals.

A testbed's "hardware in the loop" feature aids in reducing throughput requirements as long as the bandwidths of the subsystem interfaces are moderate, since fewer system functions must be modeled in real-time. However, this feature also requires that the testbed models be based on a one-to-one mapping of the actual system, with very careful consideration given to the design of the mechanical and electrical interfaces. Precise control of the timing, synchronization and data content of each testbed module interface is often vital. Tight interface timing tolerances may require additional dedicated hardware, with controlling software, to isolate and buffer the simulated modules from "real" modules. This would reduce the adaptability of the testbed, especially with respect to capabilities for examining and rapid
prototyping new technologies (such as imaging electro-optical sensors) with intrinsically wide data transfer bandwidths.

Our solution to this impasse between testbed fidelity, adaptability, complexity, and cost is to include a "third layer" of simulation models in the testbed design. Figure 3. depicts a conceptual block diagram of the PHALANX testbed "core" with all three "layers" in place. The uppermost rank of testbed modules comprises actual system hardware subsystems and connecting interfaces. The second layer contains "procedural" (conventional) software simulation models representing the hardware modules' functions and salient features of the "world" external to the system. These models are "transparent"; i.e., the mathematical representations of modeled equipment functions are accessible. The lowermost layer is made up of an array of rule-based ("expert system") and rule-driven "table look-up" (interpolation) modules that represent the functional states and input/output behavior of the subsystems. These "behavioral" modules map the possible states, inputs and outputs of the first and second layers' hardware and procedural software models in terms of tabular data, "facts" and "rules" and are capable of representing, through the mediation of the object-oriented architecture, new, conceptual subsystem designs. It is the addition of this layer and its unique architecture that will enable the new testbed to achieve its mission of providing a cost-effective, real-time simulation facility capable of the rapid testbed reconfiguration and operational adaptability that the rapid prototyping environment demands.

We are developing an intelligent object-oriented programming (OOP) modeling environment and software architecture to link the model representations of the three tiers and the testbed user(s). This work will result in construction of a "semantic system" (Negoita and Ralescu, op. cit., pp. 6-7) which will provide:

(a) an intelligent user interface, capable of translating user specifications into a realistic, tailored simulation configuration corresponding to the desired prototype features and test environment,

(b) an intelligent testbed monitor/controller which will adapt available testbed hardware and software resources to the task at hand and inform the user if those resources are considered insufficient, and

(c) an adaptive, "bootstrapping" feature, which will continuously monitor operation of the testbed hardware and procedural software modules and use the information gained to:

(1) expand and refine the knowledge bases on which the third-tier behavioral models are based, and
(2) develop, using metaknowledge constructs, resident "domain design experts" capable of using "naive" knowledge and past applications experience to guide the users in an efficient search for an optimized system or subsystem design.

The semantic system paradigm is peculiarly supportive of a rapid prototyping capability, as it facilitates the capture of design information, allows adaptive reuse of previously defined functional entities (wholly or in part), and provides a medium for the in situ testing of prototypes. The conceptual architecture of the PHALANX testbed semantic system is illustrated in Figure 4.

Following Allen and Sikora[4], the PHALANX testbed knowledge base is an integrated semantic network of objects and rule-based functional representations which utilize those objects. The salient feature of this network is a strong linking of four primary "knowledge domains" representing, respectively, system design, hardware design, software design, and utilities design. Each domain represents a unique design dimension and is implemented through a strongly linked hierarchical network of objects. The system design domain represents all external management, control and information exchange objects associated with a particular functional entity, the hardware and software domains those physical and information processing objects internal to the entity, and the utilities domain the external services supporting the functions of the entity.

Each object is implemented using a multidimensional frame-based representation (M. L. Minsky [5]). Such a representation provides the ability to associate properties (attributes) with each object and to tailor those properties to the particular functional entity represented. Tailoring of features is accomplished through the mediation of a Knowledge
Capture System (KCS) which includes authoring methods capable of capturing both user inputs and testbed simulation exercise results and browsing mechanisms to provide access to captured knowledge. The object-oriented programming approach and the mechanisms by which knowledge is captured and browsing supported are discussed below.

**THE OBJECT-ORIENTED PROGRAMMING APPROACH**

Objects are 'things' that are meaningful in an application context. Objects can be abstract or concrete. An abstract object is usually made of other abstract objects or concrete objects. A concrete object is usually made of other concrete objects. Usually, we identify a set of 'primitive' concrete objects, that we do not seek further decomposition. Through objects we can show a direct relationship with the real world of an application. Objects promote better understanding and implementation. Some of the objects in the testbed context are: i) User-Interfaces, ii) Model libraries, iii) test-bed configurations, iv) test- configurations, v) test-bed objects, vi) abstraction libraries, vii) field knowledge, viii) scenario-descriptions. A typical PHALANX MK-15 Weapon System consists of the following major subsystems, with further subsystems being given in parentheses: i) Gun System (Gun, Projectile), ii) Radar System (Radar, Antenna), iii) Fire Control, iv) Tracking System (Servomotors, Tracking Structure), and, v) Environmental System. Each of these subsystems can be considered 'macro'- objects.

We group objects into classes and hierarchies. Classes have objects with similar attributes and methods. A method is function, procedure or operation applied to each object, its attributes and other methods. A special method is the concept of inheritance, which is the recognition of common shared attributes and methods among a group of objects, from a 'super-class'. We can define the attributes, variables and methods of an object as private, protected and public. The label private indicates that it is strictly local. The label protected indicates that it is accessible for some but not all. Public indicates that there is general accessibility.

The Object Modeling Technique (OMT) approach of Rumbaugh et al [6], and other similar methods mentioned by them in their book, take a three dimensional view of objects: i) a structural view representing the possible interaction relationships among the objects, ii) a dynamic view of the transitions in interactions for specific scenarios, and iii) a functional view, which represents the processing of input data, to yield outputs.

The main considerations for the choice of simulation environments, languages and modeling frameworks are that they support heterogenous or hybrid modeling activities. The environment should also support graphical visualization of results. A shell language or a metalanguage should enable these diverse tasks to be handled seamlessly. In addition, test-bed support activities should be the main features of the metalanguage. Design engineers should be generally familiar with the types of tasks identified by the metalanguage. The shell should use macros, where feasible, so that the user avoids unnecessary sequence of commands to reach a goal state in the analysis. There should be simple cognitive compatibility between the kinds of commands and expected outcome.

Keck [7] calls for standardized validated models to be used in a testbed environment. These models could then be made available as libraries for subsequent use in the design evaluation of newer complex systems. Goodin and Bomber[8] discuss issues of simulation modeling for operational purposes. Durham, Gilcrest and Heckman [9] are developing a test-bed for embedded multi-computing for an Autonomous Undersea Vehicle. Here real-time response is critical. Design and testing processes must examine the interdependencies between the hardware and software components critically. Their goal is to develop a fully reconfigurable multi-computer which directly interfaces with vehicle sensors and actuators. It is based on the concept of an Algorithm-Structured Computer Array/Network, whose topology is reconfigurable. The testbed therefore, is valuable in a restricted setting of special dive missions of the AUV. Each dive will determine the processor structure for the testbed. This type of testbed was feasible, due to relatively small demands on embedded computing in this case compared to the PHALANX system.

A current trend in life-cycle tools for software engineering, is the development of so-called "Knowledge-Based Design" (Lubars and Harandi [10]). In these trends, libraries of usable software
parts are maintained. Each part has an associated specification, which can be combined to meet a more global specification. These are executable specifications. Larger aggregations of modules can be designed from libraries through rule-based or frame based approaches. Rozenblit and Ziegler \[11\] have applied the framework of DEVS framework \[Discrete EVent Simulation\] for a flexible test architecture for Printed Circuit Board (PCB) systems. An entity-structure formalism is used. Activities considered are fault isolation, test strategies, and reconfiguration of the test station.

The very heterogeneity of complex systems makes it necessary, for the design process to have unit analysis along with integrated analyses done quite possibly in a 'regression mode', i.e., a back and forth mode. Changes of all types will initiate such interaction. For rapid prototyping, frameworks of user interfaces must provide for smooth interaction between diverse domains and modes of usage. A Framework can be evaluated on three dimensions:

i) **Performance:** This dimension includes prototypability, code-generation, look and feel, response, both machine and cognitive.

ii) **Implementation:** This dimension includes the architecture of the supporting platform and related hardware and software. It also includes considerations of portability, cost, maintainability and consistency.

iii) **Requirements:** Factors of this dimension are:

1. The ability to handle high level objects
2. The basic programming constructs at this (meta-) level must be simple
3. Propagation of data and constraints should be transparent in the interaction/specification process.
4. The form and function of screens should be easy to identify with.
5. The learning phase of the constructs should be relatively small.
6. Help should be easily available.
7. The interface should provide capability for seamless embedding.
8. Screen layouts should be manageable. There should be adequate navigation, archiving and history of usage facilities.

There are several potential modes for intelligent assistance for the design engineer \[12\]. Most frameworks for User-Interfaces or test-bed simulators are not formally addressed. As a result, the functionality requirements are loosely collected. Resulting toolkits have merit in their own cohesiveness and stability, for a particular application. But, as soon as domains change, or new demands are put on the User Interface, there is no adaptive mechanism. In a multidisciplinary design environment, each engineer has a unique set of models, methods, styles and experience (the paradigmatic dimension of the engineer). Each specialist has a unique disciplinary approach, in addressing the problems of a specialty area of a discipline. Either the chief designer or the system evaluator must have a framework for integrating and evaluating all of the analyses.

At a procedural level, the analysis requires complete detailed specification of the software. At the behavioral level, only certain dominant patterns and outputs are expected. The inner workings of procedures are not of concern. Now, if the procedures are hard coded and run, we would have an emulation structure.

**RAPID MODELING AND PROTOTYPING USING AN OMT PARADIGM**

In response to a "rapid-fix" request from the field, an engineer may be required to develop and evaluate first, a simulated representation of the reported problem, and second, one or more rapid-prototypes which address the relevant issues identified by the simulation results. As outlined above, such an exercise may require highly data-intensive processing involving very high throughputs and gigabyte-size model representations. Timeliness of response is critical in a rapid-fix situation, and modeling- and model-evaluation techniques which minimize the time and resources required assume critical importance.

The concept of a modular testbed provides an environment wherein modeling and simulation may proceed expeditiously while minimizing resource demands. The rapid-prototyping paradigm, in turn, facilitates the modeling process and allows the user to aggregate "new" and existing subsystems at any level up to and including the full system. In addition, the combination of testbed evaluation and the aggregation process provides a near-real-time feedback loop that
enables the user to ascertain whether the selected level of modeling fidelity is sufficient. These features lead naturally to a "resource- and time-minimizing approach" to problem evaluation.

Rapid modeling, by its very nature, allows consideration of both breadth and detail to a degree not normally addressed by conventional modeling techniques. "Breadth" confers the capability to examine both general requirements constraining design- and system-level impacts of specific design features, while the high level of detail attainable allows analysis of singular or exceptional behaviors. In addition, the rapid-prototyping concept of "cohort-modeling" enables the examination of perturbation-driven variations in these singular behaviors. The designer/analyst is thus provided with the means of examining many different consequences of incorporating a particular model into the current system design.

In the OMT approach, rapid-prototyping takes the shape of simultaneously proceeding from requirements analysis to the lay out of three model variants - a conceptual Object model, a dynamic model, and a functional model. A portion of the original model is coded and tested and the results used to rapidly modify any or all of the three original models. In this way, the amount of design, coding, and testing is kept small. Stable prototype segments may be laid aside to proceed with other parts of the system. Stable aggregations are integrated in the rapid prototyping mode at the selected aggregation level. Eventually, a grand (and usually stable) aggregation at the system level results from repeated applications of the OMT process.

It should be noted that there are many different ways of classifying models. Models suitable for simulation analysis, in particular, depend on identifying states, inputs and outputs. Models can also represent internal processes, components, and/or subsystems and their interactions. All models have a purpose, in that they are all constructed to answer some question. How well a particular model meets this purpose determines the quality of that model. The success of the modeling process thus depends on the modeler's knowledge of the representation methodology and the relative advantages and disadvantages of alternative methodologies and the availability of the appropriate supporting technologies.

Definitions of Emulation, Procedural and Behavioral Models.

A Procedural Model is an object \(<X,Y,Q,P,q_0,>\), where \(X\) is the set of inputs, \(Y\) is the set of outputs, \(Q\) is the state if internal states, \(P\) is the set of procedures for transitions between internal states and generating the corresponding outputs if any, given the current state and the current input, \(q_0\) is the initial state.

A Behavioral Model is an object \(<X',Y',F,>\), where \(X'\) is a subset of \(X\), \(Y'\) is a subset of \(Y\) and \(F\) is the relationship that determines \(Y'\) given \(X'\).

An Emulation Model associates specific hardware/software modules and entities with the objects in a Procedural Model. It therefore assumes a target architecture for the Procedural Model.

The structural and the dynamic views in the Object Modeling Technology of Rumbaugh et al, corresponds to the procedural level of models discussed above, the functional view corresponds to the behavioral view.

In a categoric sense, models may be thought of as having classes of input types \(X\), output types \(Y\), state types \(S\), relation types \(\delta\) which, for each \(\delta\) in \(\delta\), is a relation between some \(X\) in \(X\), \(S\) in \(S\), and \(Y\) in \(Y\). It is possible to specify initial types for all of these entities. A specific M model is a realization of one such \((X,Y,S,\delta)\). Thus viewed, it allows any mixed mode or hybrid model to be considered. The model may be discrete, continuous, symbolic, rule-based, script-based, logic-programming based, image-based, etc. It can be all of these and time-varying, non-linear.

Suppose we are given a family of models \(P\), which represent the most detailed software realizations, which will be called herein Procedural Models, as their procedures or software details are available to the source code level. Corresponding to these software realizations, there is a possible mapping to a family of hardware components, for some or all of these models in real-time and space. Let this family be called \(H\). It is also possible, to have a newer derived family of models \(B\), which
represent subfamilies of $P$. This derived family, $B$, represents an abstraction or a lumping of models, caused by the user's current need during a design investigative cycle, to focus on selected performance considerations. This three tiered process and interaction may be visualized in the diagram below:

\[
\begin{array}{c}
H \\
\text{EMULATION} \\
\downarrow \\
\text{ABSTRACTION} \\
\downarrow \\
B
\end{array}
\]

We can now talk about the fidelity of emulation and abstraction. As each combinatorial possibility is explored in either direction, the degree to which a family of models in either layer can reasonably answer corresponding "what-if" questions, will make it possible to use all of those models in all the layers as the circumstance demands it.

As applied to the needs of rapid modeling activities, all possible models in the categories, $H,P,B$ may be considered to be Objects. Their interrelationships during the Modelling process determine a (meta)-Object Model.

To illustrate these ideas, we shall consider three subsystem domains: Radar, Infra-Red, and Electro-Optics. Note, that the last two are not currently in the MK-15. The basic issue for engineering design is, say, target detection probability, by each subsystem. Procedural and Behavioral models corresponding to each should be examined. The behavioral models will be scenariospecific. Fidelity of detection for all models is of particular concern. In real-time simulation, this fidelity is determined by interactions in the scenario specifically configured test-bed. The object-oriented framework allows messages to be sent simultaneously to each model type for processing. We may also compare it to the Visual Detection Model.

In this paper the issue of fidelity is illustrated through the concept of Figure of Merit (FOM) introduced in earlier sections. For this illustration, the categories of model types are:

\{Visual Detection, Radar, Infra-Red, Electro-Optics.\}

The categoric families of Hardware Emulation, Procedural and Behavioral models are:

\[H = \{H_{\text{Visual Detection}}, H_{\text{Radar}}, H_{\text{Infra-Red}}, H_{\text{Electro-Optics}}\}\]

\[P = \{P_{\text{Visual Detection}}, P_{\text{Radar}}, P_{\text{Infra-Red}}, P_{\text{Electro-Optics}}\}\]

\[B = \{B_{\text{Visual Detection}}, B_{\text{Radar}}, B_{\text{Infra-Red}}, B_{\text{Electro-Optics}}\}\]

We thus have the following types of linkages between models belonging to two different category types:

\[MP_{\text{Radar}} <----> MP_{\text{Infra-Red}}\]

\[MB_{\text{Radar}} <----> MB_{\text{Infra-Red}}\]

where

\[MP_{\text{Radar}} \text{ is in } P_{\text{Radar}}, \quad MP_{\text{Infra-Red}} \text{ is in } P_{\text{Infra-Red}}\]

\[MB_{\text{Radar}} \text{ is in } B_{\text{Radar}}, \quad MB_{\text{Infra-Red}} \text{ is in } B_{\text{Infra-Red}}\]

Let us consider the FOM of the models:

\[
\text{FOM}(\quad MP_{\text{Radar}} <----> MP_{\text{Infra-Red}} \\
\quad MB_{\text{Radar}} <----> MB_{\text{Infra-Red}} \\
) = f(\quad \text{FOM}(MP_{\text{Radar}}) \leq \text{FOM}(MP_{\text{Infra-Red}}) \\
\quad \text{FOM}(MB_{\text{Radar}}) \leq \text{FOM}(MB_{\text{Infra-Red}})
) = \]

263
In rapid modeling all of these models are considered simultaneously. Clearly, there is a degree of difference in the models. The behavioral models may yield results sooner, than the procedural models; however, their value is only if they point to critical decisions sooner. The more detailed procedural models will meanwhile be also able to provide supporting or complementary analysis. The resulting lattice of FOM numbers is a basis for the fidelity of the testbed. It is also framework for measuring the quality of rapid modeling, hence of rapid prototyping. Note, that the design engineer must have a finite number of models in each category for rapid modeling, to handle cognitive task complexity. As Rapid Modeling is the intermediate step between rapid conceptual development and rapid concept evaluation, such a lattice of evaluation numbers based on the FOM concept, yields the design engineer a basis for Rapid Concept Evaluation. It is our intention to provide such a User Interface screen in the ESMF.

KNOWLEDGE CAPTURE SYSTEM

The Knowledge Capture System (KCS) will be designed to facilitate the capture of information from both the user and the exercise of the testbed. A graphics-oriented user interface (GUI) and an intelligent "Design Expert" are to be provided to implement a structured design process suitable for efficient conceptual development and rapid prototyping of complex systems. Design information is stored as frame-based representations in domain-specific knowledge bases based on the integrated semantic network paradigm discussed above. These knowledge bases are accessed by rule-based models ("Domain Experts") through an inferencing process to build the components of the behavioral simulation model representing the desired rapid-prototype configuration.

The use of rule-based models integrated into the KCS and operating under the "direction" of a goal-oriented "Design Expert" makes it possible to develop dynamic behavioral models of specified design configurations. These models can be used, when integrated into the testbed "core", to perturb, test and explore the workings of the captured design and implementation knowledge in terms of both the models' expected performance and impacts on other entities making up the overall system resulting from the models' incorporation.

The Design Expert provides the key to this process. This system implements the authoring mechanisms of the knowledge capture "toolkit" (hierarchical decomposition, etc.) in a goal-oriented authoring process that operates to provide a structured information-capture and storage capability. In addition, the Design Expert builds and maintains the indices of stored knowledge and the knowledge-accessing mechanisms that support "browsing".

In most cases, the goal-oriented Design Expert will operate to capture design information in terms of satisfying the parametric data requirements of a set of particular design- or domain-specific "Figures of Merit (FOMs)". While the use of this paradigm is generally suitable for specifying configuration and performance parameters for hardware and hardware/software systems, there may be certain instances where the use of a Figure of Merit approach in structuring a goal-oriented prototyping process is inappropriate. In these cases the Design Expert will default to a user-specified data structure for the design to be prototyped.

The concept of a design/domain-centered Figure of Merit is extremely useful in formulating an object-oriented representation suitable for hardware rapid prototyping applications. First, since a FOM focuses on the essentials of a design, the FOM parameters will define and constrain a minimum set of design variables which must be addressed. Second, the vector of successive FOM computations taken over concurrent and successive alternate designs can be used to define "parametric gradients" useful in defining an efficient search for an optimal design solution. This second advantage to taking an FOM approach is especially important to the implementation of the metaknowledge-based "design expert" mentioned above.

An example based on a FOM evaluation of a notional infrared sensor design may be used as an illustration of the uses of the FOM approach in structuring design-concept representations.
An Infra-Red (IR) sensor is conventionally described as a sensor operating in the infra-red spectral region (1.0-40.0 microns wavelength). Features common to all sensors and shared (inherited) by the IR Sensor include: Platform features (such as Structure, Servo System, Power System(s), Utilities, Control, and Data Interchange), Scene Environment features (including Target Reflection and Emission, Scene Steady-State Background, Scene Noise Background, and Atmospheric Transmissibility), and Platform Environment features (Imposed Motion, Inertial Forces, Vibration and Shock Forces, etc.). Features shared with some, but not all, other sensor objects include Mechanical features (Optical Design, Scanning Design, Cold-Shielding Design, etc.), Sensing Electronics features (Modulation, Circuit Noise, etc.), and Detector features (Detector Solid-State Physics, Detector Array Geometry, etc.). As may be seen from a perusal of this list, the features associated with an IR sensor comprise a wide range of system, hardware, software, and utilities domain features.

It is possible to derive a Figure of Merit representing system characteristics and performance at the topmost level of the design hierarchy, but such a FOM is too complex to readily serve as an example for our purposes. A less generalized (and more tractable) Figure of Merit expression which addresses system suitability for a particular sensing environment is "Noise-Equivalent Delta Temperature", given by:

\[
NE\Delta T = \left[ \frac{L}{\tau_A \left( \frac{S}{N} \right) \left( \frac{dL}{dT} \right) \lambda} \right] (\circ K)
\]

where:
- \( L \) is the scene radiance (Watts/sec-cm²-sr)
- \( \tau_A \) is atmospheric transmittance (nd)
- \( S \) is signal strength (Watts)
- \( N \) is noise power (Watts)
- \( \frac{dL}{dT} \) is the differential change in scene surface radiance with respect to surface temperature (Watts/sec-cm²-sr-°K)
- \( \lambda \) is the wavelength of interest (microns)

\( NE\Delta T \) takes into account the suitability of a design, not only in terms of "design efficiency but also in terms of the "fit" of a particular design in its intended spectral operating environment. While a FOM such as this provides a good general measure of the suitability of a given design, we must go into greater detail (focusing on deeper levels of features and corresponding Figures of Merit) in order to adequately support a rapid prototyping application.

A typical detail-design level FOM is "Noise-Equivalent Irradiance (NEI)", functionally defined as:

\[
NEI = f \left[ \text{Design Efficiency, Optical Design, Coverage/Rate, Detector Quality} \right]
\]

If NEI is defined as the ratio of target irradiance (at the plane of the detector(s)) to the minimum SNR required for detection:
where \( \Lambda_D \) is detector bandwidth (Hz.)
\( t_D \) is detector dwell time (sec.)
\( k_o \) is optical efficiency (nd)
\( k_e \) is electrical efficiency (nd)
\( k_s \) is scanning efficiency (nd)
\( k_{os} \) is overscanning efficiency (nd)
\( F \) is the system "F-Number"
\( D_o \) is (equivalent) optical diameter (cm.)
\( N_D \) is the number of detectors
\( \Omega \) is the Field of View solid angle (sr.)
\( t_f \) is frame time (sec.) for a particular detector
\( D^* \) is specific detectivity (cm.-Hz./Watt)

NEI evaluates parameters which can be directly derived from conceptual design features. Specific detectivity, for example, is a solid-state physical measure based on the detector chemical makeup (and certain properties of the electronic circuitry in which the detector is embedded), evaluated at a particular wavelength, while dwell time is a function of the design's optical geometry and scanning methodology. NEI is normally evaluated at several wavelengths within the detector passband and the results combined into an empirical curve representing "design worth". (In NEI calculations, "smaller is better"; i.e., a design with an average NEI of 10' would be considered a better design than one with an average NEI of 10'.)

We will conduct a "toy" NEI computation to: (a) illustrate the computation process, (b) gain some insight into how the PHALANX testbed KCS will work to capture and utilize design and contextual information, and (c) illustrate some of the pitfalls of the design-capture paradigm. Our sensor will consist of a mechanically-scanned system where the platform servos provide gross Field of View (FOV) coverage and a spinning mirror provides a raster scan within the specified scanning FOV of 20 degrees. The instantaneous FOV will be focused upon a single detector. The notional target will be an unpowered, low-speed airborne vehicle.

Optical and Electrical Efficiencies

Optical and electrical efficiencies are dependent upon many detail mechanical, optical and circuit design parameters; here we will assume efficiencies of 90% for each.

Detector Dwell Time, Scanning and Overscanning Efficiencies, and Optical Focal Length

We have assumed a spinning 45-degree, 6-facet mirror scanning configuration. For this type of setup the mirror axis of rotation is parallel to the sensor axis (Figure 5) and a change in rotation angle \( \Theta \) will produce a change in \( \Theta \) in the Line of Sight (LOS) angle. For this configuration we can express the dwell-time of the spinning mirror as:

\[
\tau_D = \frac{\tau_A}{n_e \times n_f} \quad \text{(sec.)}
\]

where
\( \tau_A \) is the active scan time (sec.)
\( n_e \) is the number of angular resolution elements
\( n_f \) is the number of mirror facets

We desire a linear resolution of 1.0 meter at 2000 meters range. This equals an angular resolution of:
\[ \alpha = \arctan\left(\frac{1.0}{2000.0}\right) = 2.87 \times 10^{-2} \text{ degrees} = 5.0 \times 10^{-4} \text{ radians} \]

For a scan angle coverage of 20 degrees per raster scan, this resolution requirement equals 700 pixels per frame dimension. Substituting back into the \( t_p \) relation:

\[ t_D = \frac{t_A}{700 \pi f} \text{ (sec.)} \]

\( t_A \) is found by:

\[ t_A = k_{sc} \tau_M \text{ (sec.)} \]

where \( \tau_M \) is mirror rotation period (sec.)

Again by definition:

\[ k_{sc} = \frac{n_s \theta}{2\pi F_M} \]

where \( \theta \) is the scanned FOV (20 degrees or 2.98 \times 10^{-1} \text{ radians})

\( F_M \) is the ratio of mirror rotation angular change to LOS angular change (1 for a 45° mirror)

Substituting:

\[ t_D = \frac{t_A}{700 \pi n_s} = \frac{\tau_A}{1400\pi \tau_M} \]

For a mirror rotation rate of 3600 rpm (60 rps):

\[ t_D = \frac{(2.98 \times 10^{-1})}{1400\pi (1)} = 6.78 \times 10^{-5} \tau_M \]

\[ t_D = 6.78 \times 10^{-5} \times \frac{1}{60} = 1.13 \times 10^{-6} \text{ sec.} \]

We can also find the required optical focal length by means of the relations:

\[ \alpha = \frac{\Theta}{n_s} = \frac{ds}{f} \text{ (rad.)} \]

where \( \alpha \) is the instantaneous FOV (rad.)

\( ds \) is the detector active sensing width (height) (in mm.)

\( f \) is the optical focal length (approximately equal to the geometric focal length for 10 degrees.) (in mm.)

Rearranging:

\[ f = \frac{n_s \times ds}{\Theta} = \frac{700 \times ds}{2.98 \times 10^{-1}} \text{ mm} \]

A typical detector \( ds \) is 1.0 mm. Using this value:

\[ f = \frac{700 \times (1.0)}{2.98 \times 10^{-1}} = 2.35 \times 10^3 \text{ mm.} \]

This is a relatively long focal length. It is likely that space and vibration considerations would force us to utilize a "folded-optics" (Cassegrainian, etc.) sensor design, with consequent slight loss in optical efficiency.

The scanning efficiency is:
Overscanning efficiency is a measure of overlap coverage between successive scans. For continuous coverage, \( k_w = 1 \), \( k_w > 1 \) for overlap, and \( k_w < 1 \) for incomplete coverage. In terms of percentage of coverage, \( P_o \), we may write:

\[
k_{osc} = 1 + \frac{P_o}{100} \quad (nd)
\]

We will assume a 10% overlap and a \( k_w \) value of 1.1.

**F-Number, Optical Diameter, Angular FOV, and Frame Time**

The F-number is defined as the ratio of focal length to optical diameter:

\[
F = \frac{f}{D_o} \quad (nd)
\]

If we select a relatively "slow" system of F20.0, the optical diameter of the sensor is:

\[
D_o = \frac{2.35 \times 10^3}{20} = 11.75 \, cm.
\]

Since we are describing a line-scanning sensor, where each line scanned constitutes a frame, the frame coverage solid angle, \( \Omega \), may be found by:

\[
\Omega = (2.98 \times 10^{-1}) (5.0 \times 10^{-4}) = 1.49 \times 10^{-4} \quad sr.
\]

Frame time in this case is equal to active time, \( t_a \):

\[
t_F = t_A = k_{osc} t_W = 0.285 \times (1/60) = 4.75 \times 10^{-3} \, sec.
\]

### Specific Detectivity and Bandwidth

Selection of a detector constrains the selection of a passband bandwidth and vice-versa; the two design parameters must be considered jointly and the process must take account of other, system-level considerations. For purposes of our "toy" application, we will ignore these caveats and arbitrarily select a GeHg detector (refer to the "Detector Response Curves" block of Figure 6) configured with a narrow (1.0 \( \times \) 10^4 Hz.) passband centered at the detector's 11 micron response peak. (It is presumed here that the target radiance is likely to lay in this spectral region.) These selections correspond to a peak \( D' \) of 2.0 \( \times \) 10^10 cm\(^{-2}\) Hz\(^{-0.5}\) W\(^{-1}\) with a bandwidth of 1.0 \( \times \) 10^4 Hz.

### NEI Computation

The NEI for our notional detector at 11.0 \( \mu m \) wavelength is then found by:

\[
\text{Design Efficiency} = \frac{(1\times 10^{14})^{0.5} (1.13 \times 10^{-6})^{0.5}}{(0.9)(0.9) [((0.285)(1.1))]^{0.5}} = 2.3 \times 10^4
\]

### Optical Design Factors

\[
= (20)^{0.5} = 11.75 = 0.381
\]

### Coverage/Rate Factors

\[
= \left[ \frac{1.49 \times 10^{-4}^{0.5}}{4.75 \times 10^{-3}} \right] = 1.77 \times 10^{-1}
\]

### Detector Quality

\[
= \frac{1}{D_h = 11 \, \mu m} = \frac{1}{2.0 \times 10^{10}} = 5.0 \times 10^{-11}
\]
\[
\text{NEI} = \frac{4}{\pi} \left[ (2.3 \times 10^4) (0.381) (0.177) (5.0 \times 10^{-11}) \right] \\
= 9.87 \times 10^{-8}
\]

Note that, in the NEI computation process, there are several instances where the interactions between design parameters must be considered explicitly. In some cases all relevant parameters are defined within the FOM. In other cases, parameters or requirements not explicitly addressed within the FOM framework are critical in determining the appropriate FOM parameteric values. As indicated above, the FOM concept will operate to provide a minimum set of relevant design parameters: this set will not necessarily be sufficient. The example set-out in the following paragraphs will serve to illustrate this situation.

**SOME NOTES ON THE REQUIREMENTS FOR A "DESIGN EXPERT"**

In infrared sensor design, \(\lambda_p\) must be matched to the response bandwidth of the particular detector selected. The detector, in turn, must be matched to the expected radiated spectrum of the specified target set, the predicted atmospheric transmissibility in the spectral region that can be expected to contain the most information concerning the target set, and the type of information we wish to gain. For example, simple detection of a target is enhanced if the detector is selected so that its specific detectivity response curve matches as closely as possible the spectral region where the greatest amount of target set radiant energy will be transmitted to the detector. The bandwidth of detection can then be narrowed to a small increment of wavelength centered on the peak response wavelength of the detector selected, lowering detected energy but also lowering in-band noise and the minimum required SNR. Alternatively, classification of targets is facilitated if the maximum amount of spectral information is available at the detector. This requires that we widen the passband, within the constraints of the detector response curve and atmospheric transmissibility, so that a spectral "signature" rather than a simple signal is available for processing.

IR detector and passband bandwidth selection is therefore as much art as science. Informed judgements must be made over a complex, interrelated set of parameters, some of which (target spectral signature, for instance) are by nature uncertain. Figure 7 graphically depicts this process. Clearly, this is a prototypical application area for the use of one or more "domain experts" so long as the operations of these experts are carefully supervised by a "Design Expert" cognizant of relevant requirements and constraints not reflected in the particular FOM being evaluated over the prototype candidate designs.

As an illustration of the need for such a system, consider the case where the target set of interest consists of several airborne vehicles, some powered and high-speed and others unpowered and low-speed (gliders). Each of these vehicles would possess a particular time-varying spectral signature in the infrared, depending on the propulsion system used, the flight velocity, and certain optical and geometric properties of the vehicle airframe. Since atmospheric transmissibility in the infrared region is concentrated in two "windows", at 3-4 microns (Near IR) and 8-12 microns (Far IR) wavelengths, the targets’ radiance spectra in these regions are the signatures of interest. Referring to Figure 7, we note that, with other variables held equal, an InSb detector would be expected to provide peak response in the Near IR and a GeHg detector the peak response in the Far IR region. If we are constrained to the use of just one type of detector (not an unusual situation, for various design-related reasons), a GeHg detector provides the best performance over both spectral regions.

Unfortunately for our hypothetical designer, the specified target set exhibits a bimodal distribution of signatures. Most of the radiated energy in the Near IR region is produced by chemical reactions in a vehicle propulsion system’s hot exhaust, while the Far IR signature is dominated by thermal (black- and graybody) radiation caused by aerodynamic heating.
Since we have specified that the target set contains targets with propulsion and targets without propulsion, the detection spectral region is forced to cover both atmospheric windows. This forces the designer to use either two types of detector or to use the GeHg detector with consequent less than optimal performance in the Near IR band. Both of these solutions will add design complications not reflected in the NEI computation, but still extremely important to the viability of the conceptual design.

To further complicate the situation, if "target classification", rather than simple "detection" is required, the passband(s) must likely again be widened to collect sufficient signature data to discriminate between most or all members of the target set. This requirement, which is not reflected in the NEI relation, will result in a higher NEI value for a target-classifying system than for a mechanically and optically identical system that is expected to produce detection signals only.

All the above discussion serves to illustrate the necessity for a "Design Expert" supervisor capable of guiding and optimizing the operations of the various "FOM-optimizing Domain Experts" in terms of overall (global) system and environmental considerations. While a FOM is a useful measure of design worth in one or more particular domains, indiscriminate use of the approach may result in a suboptimal design in terms of the system-level requirements. It must also be kept in mind that, while a FOM appears to be an objective quantitative measure, often certain of the parametric values making up a FOM are derived or selected in a partially quantified/partially subjective/partially "fuzzy" recursive process. The FOM is a useful paradigm for defining a minimum domain-specific data structure. It is less useful as a framework for optimization unless it is realized by the user that it is often based, at least partially, on non-quantitative concerns and/or addresses only a very limited subset of the design "problem domain" and the necessary compensations made for these facts.

CONCLUSIONS

The foremost conclusion the authors have reached is that the modular testbed, combined with an object-oriented knowledge capture and representation capability, provides the robust environment necessary for rapid prototyping. In particular, the concept of an integrated, comprehensive model-building and evaluation environment shows great promise in minimizing the time and resources required for comprehensive evaluation of alternate conceptual designs.

A powerful feature of this concept is that it facilitates the partitioning of a problem and the proposed design solutions into logical components and enables the examination of these components at any desired level of aggregation. Once a solution to any individual problem component has been reached, that solution can be shared with other subproblem representations and stored for future reference and use. This feature, when combined with the interactive, visually-oriented nature of the user interface, provides many of the advantages of a heuristic problem solving technique with the underpinning of a rigorous quantitative data representation.

The concepts described here do not purport to solve all problems associated with rapid prototyping and simulation-based conceptual design evaluation. Of the three traditional problems associated with these and other simulation-based techniques (performance, closure, and transition), we believe that we have made substantial progress with the first. The use of a third testbed tier of behavioral models will greatly reduce throughput and data representation demands on the simulator hardware and software resources. We believe that, by implementing the three-tier concept, significant speedups and memory requirements reductions can be achieved for most, if not all, testbed simulation applications.

"Closure" is a measure of the extent to which a particular analytic or other investigative technique is capable of providing solutions to every (similar) problem. Rapid prototyping and testbed simulation is basically a method for synthesizing a set of unique problem-solving representations based on a defined set of assumptions concerning the feasible "states of the world". It is inevitable that there will be unusual or "pathological" situations arising that had not been previously considered and that the described techniques will prove less than adequate in dealing with such occurrences. This risk may be mitigated by limiting the problem domain or by making provision for extensive trial and error analysis, but there can be no totally satisfactory resolution of the closure issue under the described paradigm.
"Transition" describes the extent to which a particular conceptual development and evaluation technique supports the transformation of a prototype into a deliverable design. The modular testbed/object-oriented knowledge capturing and model-building paradigm we describe does not claim to address this problem specifically. (It should be noted, however, that testbed simulation of conceptual prototypes will inevitably produce detail data that will be extremely useful in the product engineering of a selected prototype design.)

There are several areas wherein we are planning further investigations of, development work involving, and/or extensions to this concepts described in this paper. One area, that of using the described methodologies to conduct model-based diagnostic investigations (failure analysis by the automated "reverse engineering" of reported incidents), is felt to be particularly well-supported by the described paradigm. Another applications area that we feel deserves further investigation is the possibility of extending the concept to the design of system support equipment, including the conceptual prototyping of test and measurement instrumentation. Finally, a third area where the unique testbed configuration and problem-representation approaches described may find productive use is in the use of the intrinsic "polymorphism" of the system design to support multiple system-level simulations simultaneously on the same testbed. This would allow simulation and investigation of command and control problems exhibited by multiple-system installations without requiring the use of multiple simulation platforms.

REFERENCES


### TABLE I

ESTIMATED COMPUTATIONAL REQUIREMENTS FOR A REAL-TIME PHALANX SYSTEM SIMULATION (BASED ON CURRENT SYSTEM CONFIGURATION AND CYCLE-TIME)

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>THROUGHPUT (VAX MIPS)</th>
<th>THROUGHPUT (VAX MIPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(MODERATE FIDELITY)</td>
<td>(HIGH FIDELITY)</td>
</tr>
<tr>
<td>SCENE GENERATION</td>
<td>10.000</td>
<td>30.000</td>
</tr>
<tr>
<td>PHYSICAL ENVIRONMENT</td>
<td>0.010</td>
<td>0.015</td>
</tr>
<tr>
<td>ENVIRONMENTAL SENSING</td>
<td>0.010</td>
<td>0.015</td>
</tr>
<tr>
<td>SHIP MOTION</td>
<td>0.75</td>
<td>0.750</td>
</tr>
<tr>
<td>DECKPLATE FORCES</td>
<td>0.020</td>
<td>0.030</td>
</tr>
<tr>
<td>SYSTEM DYNAMICS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- STABILIZATION &amp; SERVOS</td>
<td>1.700</td>
<td>4.000</td>
</tr>
<tr>
<td>- RECOIL LOADS</td>
<td>0.800</td>
<td>2.000</td>
</tr>
<tr>
<td>SYSTEM SENSORS</td>
<td>0.500</td>
<td>15.000</td>
</tr>
<tr>
<td>SIGNAL PROCESSING</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- SEARCH</td>
<td>1.000</td>
<td>10.000</td>
</tr>
<tr>
<td>- TARGET TRACK</td>
<td>2.500</td>
<td>5.000</td>
</tr>
<tr>
<td>- PROJECTILE TRACK (CLOSED LOOP FIRE CONTROL)</td>
<td>0.100 (Per Track)</td>
<td>0.120 (Per Track)</td>
</tr>
<tr>
<td>MULTISENSOR DATA FUSION</td>
<td>2.000</td>
<td>3.000</td>
</tr>
<tr>
<td>OTHER SYSTEM COMPUTER FUNCTIONS</td>
<td>1.000</td>
<td>1.500</td>
</tr>
<tr>
<td>SYSTEM ENVIRONMENTAL SUPPORT</td>
<td>0.200</td>
<td>1.200</td>
</tr>
<tr>
<td>TARGET STATUS</td>
<td>0.010</td>
<td>0.800</td>
</tr>
<tr>
<td>SIMULATOR &quot;HOUSEKEEPING&quot;</td>
<td>0.500</td>
<td></td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>21.000</strong></td>
<td><strong>76.310</strong></td>
</tr>
<tr>
<td></td>
<td>+ 0.10 per Projectile Track</td>
<td>+ 0.12 per Projectile Track</td>
</tr>
</tbody>
</table>

Note: This tabulation presumes a "worst-case" application, with all testbed capabilities fully employed.
PHALANX ESMF TESTBED
MAJOR HARDWARE AND SOFTWARE COMPONENTS