**TITLE OF PROPOSED PROJECT**

"ITR/SW: Toward a Unified Theory of Software Engineering"

---

**REQUES TED AMOUNT**

$5,192,397

---

**PROPOSED DURATION (1-60 MONTHS)**

60 months

---

**REQUESTED STARTING DATE**

09/01/00

---

**PVPD NAME**

Barry W. Boehm

Ph.D.

1964

213-740-8163

boehm@sunset.usc.edu

---

**CO-PVPD**

Ann Majchrzak

1980

---

**CO-PVPD**

Nenad Medvidovic

Ph.D.

1999

213-740-5579

neno@usc.edu

---

**CO-PVPD**

Daniel N Port

Ph.D.

1994

213-740-7525

dport@sunset.usc.edu

---

**CO-PVPD**

Rayford B. Vaughn

Ph.D.

1988

662-325-7450

vaughn@cs.msstate.edu

---
Project Summary

The recent PITAC report identified computer software as both a major enabling factor and a major limiting factor in achieving information technology (IT) potential. It cited major problems in software fragility and unpredictability, and identified software research as a top-priority area for addressing these problems. NSF has convened two workshops to identify promising software research strategies to address these problems. The workshops established the goal of "no-surprise" software as an appropriate research goal. They concluded that current IT trends toward rapid development, less-controllable COTS-intensive products, and more complex systems of systems would make the achievement of no-surprise software ever more challenging. They also concluded that research into better models of software engineering phenomenology had high payoff potential and should be strongly emphasized.

The workshops identified five classes of models that were important in addressing software fragility and unpredictability problems: domain models, process models, product models, property (quality of service) models, and success models. Most research on software engineering models addresses only one or at most two of the five models indicated. However, our research and experience has shown that the most serious sources of software fragility and unpredictability arise from clashes among all five classes of models.

We propose to develop, formalize, evaluate, and iterate a Model Integration Framework (MIF) for software domain, process, product, property, and success models. Our starting point will be USC's Model-Based (System) Architecting and Software Engineering (MBASE) approach. We have been developing, testing and iterating MBASE over the past four years, and have found it increasingly effective for model integration.

Several significant research tasks are needed to formalize and extend MBASE to meet the overall project goals. Some specific project objectives are the following:

1. Explore and develop appropriate formalisms for the current informal MBASE models. Our initial approach will be to formalize the current definition of a model clash as an inconsistency among the assumptions of a set of models. We propose to make use of the model theory of first-order languages to carefully and scientifically represent MBASE models, their assumptions, rules and invariance formulas, and mutual compatibility (i.e. model integration). Our goal is to understand the fundamental rules that models must satisfy to avoid model clashes.

2. Explore the definition and creation of a critical-mass set of models to cover the most important current and likely future project considerations, and experiment with formal modeling and analysis approaches for them. Extensions from the current MBASE model coverage will include extension of Boehm’s work on Constructive Cost Model (COSEMO) cost, schedule, and quality models; on spiral process models, and stakeholder win-win success models; extensions of Boehm’s and Port’s work on domain models and model clash characterization; integration of informal MBASE architecture models based on the Unified Modeling Language with Medvidovic’s more formal architecture models; property model extensions to include Vaughn’s security model work; and experimentation with appropriate levels of formalization of behavioral models using Majchrzak’s work.

3. Continue to evaluate and improve the MIF and model set via initial application of the results to real-client university projects in the digital library domain, and subsequent application to industrial projects in other domains via our 28 industry and government Affiliates.

4. Develop, apply, evaluate, and improve educational materials for university and continuing education.

Potential outcomes of the project include:

1. Avoiding model clashes and producing higher project success rates in achieving the PITAC objectives of more dependable and predictable software.

2. Providing strong lifelong career education for software engineering students, reinforced via real-client course project practice.

3. Providing a stronger foundation for improving many other branches of software research and education, by establishing them within a framework of unified models.
1. Motivation and Context

The recent PITAC report [PITAC, 1999] identified computer software as both a major enabling factor and a major limiting factor in achieving information technology (IT) potential. It cited major problems in software fragility and unpredictability, and identified software research as a top-priority area for addressing these problems. NSF has convened two workshops [Basili et al., 1998; Boehm et al., 1999] to identify promising software research strategies to address these problems. The workshops established the goal of “no-surprise” software as an appropriate research goal. They concluded that current IT trends toward rapid development, less-controllable COTS-intensive products, and more complex systems of systems would make the achievement of no-surprise software ever more challenging. They also concluded that research into better models of software engineering phenomenology had high payoff potential and should be strongly emphasized.

The workshops identified five classes of models that were important in addressing software fragility and unpredictability problems: domain models, process models, product models, property (quality of service) models, and success models. Most research on software engineering models addresses only one or at most two of the five models indicated. For example, most current “model-based” methods [Fisher et al., 1988; Gargaro-Peterson, 1996; Honeywell, 1998] focus almost exclusively on product models and somewhat on product-based property models.

Our research and experience has shown that the most serious sources of software fragility and unpredictability arise from clashes among the project’s domain, process, product, and success models [Boehm-Port, 1999a; Boehm-Port, 1999b]. For example, in the National Research Council “Ada and Beyond” study [Boehm et al., 1997b] Boehm, Vaughn, and others encountered a seriously troubled software project which was unknowingly struggling with model conflicts inherited from five Department of Defense (DoD) directives:

- A DoD-STD-2167 directive to use the requirements-driven Waterfall process model.
- A Secretary of Defense directive to use a capabilities-driven COTS-intensive product model, incompatible with the Waterfall model.
- A Congressional mandate to use Ada, for which there were few bindings to COTS products.
- A Service mandate to use fourth generation languages, which did not interoperate well with Ada.
- A new DoD initiative to use Cost as Independent Variable, when it already had at least four existing incompatible independent variables.

This and similar large commercial project failures we have studied, such as Confirm and Master Net [Boehm-Port, 1999a], have had no solid framework of models and techniques for recognizing and resolving such model clashes. The primary goal of our proposed project is to remedy this shortfall.

2. Project Goals and Objectives

We propose to develop, formalize, evaluate, and iterate a Model Integration Framework (MIF) for software domain, process, product, property, and success models. Our starting point will be the Model-Based (System) Architecting and Software Engineering (MBASE) approach. We have been developing, testing and iterating MBASE over the past four years [Boehm et al., 1997a; Boehm et al., 1998a; Boehm-Port, 1998; Boehm-Port, 1999a; Boehm et al., 1999b; Mehta et al., 1999], and have found it increasingly effective for model integration.

Several significant research tasks are needed to formalize and extend MBASE to meet the overall project goals. Some specific project objectives are the following:

1. Explore and develop appropriate formalisms for the current informal MBASE models. Our initial approach will be to formalize the current definition of a model clash as an inconsistency among the assumptions of a set of models.
2. Explore the definition and creation of a critical-mass set of models to cover the most important current and likely future project considerations, and experiment with formal modeling and analysis approaches for them. Extensions from the current MBASE model coverage will include stronger domain models; integration of informal MBASE architecture models based on the Unified Modeling Language with Medvidovic’s more formal architecture models; property model extensions to include Vaughan’s security model work; and experimentation with appropriate levels of formalization of behavioral models using Majchrzak’s work.
3. Continue to evaluate and improve the MIF and model set via initial application of the results to real-client university projects in the digital library domain, and subsequent application to industrial projects in other domains via our 28 Industry and government Affiliates.
4. Develop, apply, evaluate, and improve educational materials for university and continuing education.
3. Research Approach

3.1 Model Integration Framework

The current MBASE elements are integrated via several proto-formalisms which need both further integration and more unambiguous formalization. The current top-level MBASE integration framework establishes relationships that must be satisfied among a project's process, product, property, and success models. For example, satisfactory achievement of the Life Cycle Objectives (LCO) process milestone requires demonstrating that there exists at least one product architecture, defined in the LCO package, such that a product developed to the architecture would satisfy the LCO-package requirements specification; satisfactorily support the KO-package operational concept definition and its success models; be consistent with the key properties in the LCO-package prototype; and be implementable within the processes, budgets, and schedules in the LCO-package life cycle plan [Boehm et al., 1998].

We have taken one significant step towards formalizing these relationships. This has been to express the MBASE guidelines in the framework provided by the SEI's Electronic Process Guide (EPG) tool (Metha, 1999; Metha et al., 1999; http://nmsut.usc.edu/MBASE/EPG]. The EPG tool uses a formalized definition of the static and behavioral relationships among a process's agents, activities, and artifacts [Armitage et al., 1994; Kellner, 1999]. We have now had experience in using the EPG version of MBASE on 21 real-client projects in digital library-related domains. Overall, the use was successful, but we are now in the process of refining the guidelines and developing a further EPG authoring tool to facilitate MBASE (and other) EPG updates.

Another key dimension of formalization involves more rigorous definition of model clashes. We have defined a model clash as an inconsistency in the underlying assumptions of a set of models, and have begun defining the assumptions and model clashes involved in a number of common process, product, property, and success models. For example, the key assumptions underlying the successful use of the Waterfall process model are:

1. The requirements are knowable in advance of implementation.
2. The requirements have no unresolved, high-risk implications (e.g., risks due to COTS choices, cost, schedule, performance, safety, security, user interfaces, organizational impact).
3. The nature of the requirements will not change very much (during development; during evolution).
4. The requirements are compatible with all the key system stakeholders' expectations (e.g., users, customers, developers, maintainers, investors).
5. The right architecture for implementing the requirements is well understood.
6. There is enough calendar time to proceed sequentially.

However, if one is developing a user-intensive system, a key assumption for a user-friendly interface property is, 'I'll know it when I see it requirements unless there are well-established user interface metaphors for all user operations, 'which conflicts with Waterfall assumptions W1. If one is developing an electronic commerce system, some key assumptions are that the requirements will change rapidly due to competitive pressures (conflicting with Waterfall assumption W3); and that rapid development is essential (conflicting with assumption W6).

As discussed below, our proposed research involves exploration of the most appropriate formalizations for defining and integrating these and other key modeling aspects which may arise in extending and applying the models.

3.2 Model Formalization

The primary consideration within MBASE is the identification and reconciliation of model clashes. Inter and intra-model clashes arise when the assumptions that are either explicitly or implicitly stated for any models considered are incompatible, eventually leading to contradictions. Model clashes take many forms and may have subtle and catastrophic consequences. It is easy to fall prey to model clashes, as assumptions are often not carefully expressed from the start and only derivations and consequences of them are visible. Furthermore, it is difficult to directly compare assumptions between different models as they are often expressed in different languages. Even if this were not a problem, the rules by which compatibility is either ensured or invariant are not explicitly known.

We propose to make use of the model theory of first-order languages to carefully and scientifically represent MBASE models, their assumptions, rules and insurance formulas, and mutual compatibility (i.e., model integration). Our goal is to understand the fundamental rules that models must satisfy to avoid model clashes. We will use the standard first order logical symbols (e.g., implication, not, variables) and parameters (e.g., quantifiers, predicates).
functions). The primary model structure within this language is a "system domain" \( D \) which is a function on the set of parameters whose universe is the (non-empty) set of all possible property values of the system domain. For example "Library Information Service" could be an example domain. The particular predicate "\( Rx \)" will be defined as "\( x \) is resolvable" meaning that there exists a recursive function that assigns values to \( x \). Model elements will be abstractions that are recursive sets where \( \forall x \in A (Rx) \) in the domain \( D \). Given a collection of first-order formulas \( \Sigma \) chosen to be \( \text{TRUE} \), an abstraction \( A \) satisfies \( \Sigma \) when any formula involving \( A \) and formulas in \( \Sigma \) are entailed by \( \Sigma \). In such a case, \( A \) is a model for \( \Sigma \). A model clash is when \( \Sigma \) entails \( \text{FALSE} \) for two models of an abstraction \( A \).

We are primarily interested in discovering tautologies (formulas that are vacuously \( \text{TRUE} \) for all models) and invariance formulas that are \( \text{TRUE} \) on sets of models. Invariance formulas may hold within a particular class of models such as Product-Product, or between different classes such as Process-Property-Product. Once these are established we may begin identifying necessary and/or sufficient conditions on models that avoid or reduce model clash. As a simple example of \( A \) (a very common) model clash consider the Waterfall Process model with the use of an IKWISI Success model. A requirement is a specialized abstraction that contains qualities such as pre-conditions, outputs, scenarios, and so forth. There are many equally valid and useful ways to model requirements. At present we concern ourselves only with the fact that however we choose to model a requirement, it is ultimately resolvable (as an abstraction) within the domain. The Waterfall model assumption \( W1 \) may be expressed as \( \forall v \exists v_2 (Iv \land Py_2) \) where \( I \) is the unary predicate "is implemented" and \( P \) the unary predicate "is a requirement." The IKWISI Success model has a requirements specification assumption \( \forall v \exists v_3 (Pv_3 \lor Iv_3) \). The non-equality of the abstractions is used to avoid degenerate cases such as where the requirements are modeled precisely by the implementation. Given a finite non-empty set of abstractions modeled with Waterfall and IKWISI, there will be abstractions \( v \neq v' \) for which \( I(v) \neq I(v') \) by the pigeonhole principle.

3.3 Extending and Integrating the Model Set

Domain Models

Based on our experience to date, we are establishing domain models as another class of models to integrate rather than as a subset of product models. We have an extensive set of domain models for a number of digital library subdomains (multimedia archive, selective dissemination of information, automated reference services, etc.). We have characterized those subdomains in terms of primary stakeholder roles and responsibilities; mission workflows; and operating environment elements; domain-system simplifiers and complicators which have been useful in stakeholder expectations management [Boehm et al., 1999].

Our primary research agenda items for domain models involve extensions to other domains; characterizing stakeholders and their cultural aspects [e.g., Majchrzak-Beath, 1999]; formalizing domain elements and their relationships; and extending existing domain modeling approaches such as FODA [Kang et al., 1990] and SPC's guidelines [SPC, 1992].

Process Models

A primary agenda item for process models is to extend the characterization of Waterfall process model assumptions shown above to cover other major process models and sub-process models. For example, the [Madachy, 1994] model of the software inspection process showed that the cost-effectiveness of inspecting an artifact rested on the assumption that the artifact had a nontrivial number of defects. If the inspection is preceded by such defect elimination sub-processes as Cleanroom and the Personal Software Process, this assumption may not be valid. Particulary for subprocess interactions, this work will draw on formal process representations such as the ILL [Santina-Osterweil, 1997] and Oz [Ben-Shaul-Kaiser, 1995] formalisms, along with its initial SEI-EPG starting point.

Another major agenda item partly addressed in the Section 3.1 Waterfall model example involves the analysis of compatibilities and incompatibilities between process models and other classes of models. Another useful starting point for this analysis is our Process Model Decision Table [Boehm, 1989, pp. 436-37], which suggests the best process model for various product characteristics (available componentry, understanding of system/software requirements or architecture) and property characteristics (required robustness, scalability, cost or schedule constraints). A further agenda item is to extend software process models further toward models of business value realization such as [Theob, 1998].
approaches discussed above. Other success models derive from organizational goals, such as business-case or mission-success models. The most effective modeling approach we have found for such success models is the LMR Benefits Realization Approach (Thorp, 1998). The initiative-contribution-outcome-assumption structures in its Results Chain are a good fit both to MBASE and to the use of assumption analysis for model clashes.

The primary challenge in dealing with success models is that most projects have a considerable number of them to be reconciled. The best success model (or more properly, metamodel) we have found for addressing this challenge is the stakeholder-win-win approach, including the Theory W management approach (Boehm-Bose, 1989), the Win-Win Spiral process model (Boehm-Bose, 1994), and the Win-Win tool's negotiation model (Booth et al., 1995) which has been formally analyzed (Lee, 1996). We propose to explore the integration of this approach with Majchrzak's recent research on learning and negotiation during systems development (Majchrzak-Beath, 1989).

3.4 Experimental Application, Evaluation, and Iteration

We have been informally refining and integrating the current set of MBASE models for over three years, on an annual set of 15-20 real-client student-team digital library projects at USC (Boehm et al., 1988b). In 1999, we successfully transitioned the course to Columbia U. from a 2-semester graduate course series to a 1-semester graduate and undergraduate courses (Port et al., 1999). The projects have been extensively instrumented and analyzed (Boehm-Egyed, 1998a; Boehm-Egyed, 1998b; Egyed-Boehm, 1999; Boehm et al., 1999c).

For the first two years of this proposed project, we plan to extend and integrate a set of domain models to the evolving process, product, property, and success models in the digital library domain. This will involve a closed-loop feedback process in which insights on the unified theory will strengthen the individual models, and vice versa. We would also selectively experiment with projects in other domains and off-campus projects at USC and Mississippi State. For years 3-5, we would apply and refine the evolving unified theory and models on projects involving our 28 industry and government Affiliates. Several Affiliates have expressed a desire to participate in such experiments.

3.5 Education and Curriculum Development

As indicated above, we have developed, iteratively applied, and successfully exported a good deal of course material for projects of this nature, including lectures, videos, readings, tools, tutorial homework, workshop exercises, case studies, Web pages, product and process guidelines, and course project instrumentation. In collaboration with Mississippi State, we propose to develop more extensive, integrated, and modular versions of these materials, to experiment with distributed delivery of course content, and to package and transition the material for broad use.

4. Conclusions: Eight Good Reasons to Support the Project

1. Nothing like it has ever been done before. Most software research addresses one or at most two classes of models.
2. It is good science. It is focused on real software phenomenology, and attempts to put its results on a sound formal basis.
3. It is sorely needed. Avoiding model clashes has produced a high project success rate in achieving the ETAC objectives of more dependable and predictable software.
4. It builds on strengths, both on the track record of the people and on the MBASE approach to date.
5. It is interdisciplinary. It draws on computer science, mathematics, behavioral sciences, economics, and domain sciences.
6. It will provide strong lifelong education for software engineering students, reinforced via real-client course project practice.
7. It has a rapid conversion path to practice via USC's industry and government Affiliates.
8. It will provide a stronger foundation for improving many other branches of software research and education, by establishing them within a framework of unified models.
D. REFERENCES CITED


