ICSE 2002 Doctoral Workshop: Research Abstract
J. F. Puett, Naval Postgraduate School

I. Title: Holistic Framework for Software Engineering (HFSE)

II. Research Area or Sub-Area: Software Development Tool/Model Interoperability

III. Research Overview (Technical problem to be Solved and Justification):

This research is an initial investigation into the development of a Holistic Framework for Software Engineering (HFSE) that establishes mechanisms by which existing software development tools and models can work together. The short-term goal of this research is to determine if such a framework is theoretically feasible. The long-term goals stemming from this research are to improve the efficiency of software development processes and to improve developed software's quality, safety, and reliability by establishing a mathematical framework that allows existing software engineering process models (and tools supporting those models) to seamlessly interact. A great deal of software engineering research has been conducted with the aim of developing or improving individual aspects of software development. Examples include research into software evolution models, requirements engineering, risk and cost estimation, software reuse, prototyping, testing, software integration, software maintenance, re-engineering, performance analysis, domain analysis, architecture design, etc. Typically, these individual aspects of software development revolve around the software engineer and rely on that engineer to provide any needed interface between different development models and tools (see figure 1).

![Figure 1: Typical Software Development Process Interaction](image)

However, there has been little research into holistic models of how these various threads and processes could (and should) most efficiently and effectively interact [RATI98]. Currently, there is inadequate communication of risk and requirements across disjoint tools and models. The development of such a holistic framework promises to provide seamless interoperability between these processes allowing complex software to be produced more
efficiently, reliably, and of improved quality. Additionally, the existence of such a framework enhances the discovery of dependencies among the different aspects of the software engineering process. It will help software engineers discover process improvements that provide product integrity with respect to those dependencies. The long-term goal of this research is to support all aspects of software engineering; however, the immediate goal is to demonstrate the theoretical feasibility of integrating a selected subset of models and tools using a Holistic Framework.

Central to this holistic view is software evolution. A software evolution system must provide strong version control of all artifacts produced during system development as well as tracking the dependencies of artifacts. In today's distributed development environments, the evolution control system must provide for collaboration between multiple users at multiple sites, provide mechanisms for notification when changes made by one developer affect the work of another, and when appropriate, provide blocking when on-going work of one developer would be counter-productive to attempted work by another. The artifacts to be controlled vary in both purpose and format. Examples include: organizational policy and vision documents, business case documents, development plans, evaluation criteria, release descriptions, deployment plans, status assessments, user's manuals, requirements and specifications, customer interviews, meeting minutes, code, software documentation, software architecture document, unit tests, test cases, test results. The formats vary as well: database entries, text documents, spreadsheets, images, drawings, audio files, video clips. The long-term goal of the HFSE is to establish positive control and integration over this diverse set of information.

By relating inputs and outputs of various software process models through an evolution interface that attaches and records the dependencies among evolution artifacts [HARN99]; information required by various processes can be automatically generated and obtained as needed (see figure 2). Such a model requires interaction between a GUI, an evolution control component, and an object model component. The Evolution Model and Object Model interact with subordinate software development tools and processes. Considerations in establishing this higher level holistic framework include identifying the medium for representation of information (e.g. tree structure), establishing a communications medium (e.g. net, databases, publish and subscribe with CORBA, object mechanisms using XML, etc.), accounting for process order (e.g. sequential, parallel, hybrid), providing missing data, accounting for ambiguity of inputs & outputs, accounting for conflict resolution between models, and providing for extensibility.

Figure 2: Holistic Model of Software Process Interaction
One way of developing this evolution interface is by extending an existing Software Evolution model with Quality Function Deployment (QFD) to introduce a continuum of dependencies between software artifacts [HAAG96]. Existing models rely on predefined artifacts and limited dependency tracking. A QFD continuum separates relevant dependencies/priorities from noisy data and is an improvement over current models that only provide primary and secondary dependencies with no articulation as to importance (and type) of the dependency to the rest of the design. Such an extension also improves the vertical, horizontal, and temporal dependency graph between software artifacts (e.g. horizontal: requirement 1.2 to requirement 1.2.1; vertical: specification 1.2 to test case 3.4; temporal: reuse component 4.2 of version 1.0 to reuse component 4.2 of version 1.1).

Next, it is necessary to develop an interaction framework between the subordinate process models and the extended evolution model. One promising approach is to use an Object-Oriented Model for Interoperability (OOMI) for resolving representational differences between heterogeneous systems [YOUN01]. This approach establishes a high level Federation Interoperability Object Model (FIOM) that allows interaction between the objects of existing heterogeneous systems. By establishing such an object federation between existing process models (or their tools) and then integrating that federation with the extended evolution model, inputs and outputs between the subordinate models (or tools) will be available to each other while at the same time reporting that interaction to the extended evolution model. The success of this research will help clarify the tradeoff between interoperability via conformance to a single global data standard versus the use of multiple representations, ontologies, and translations as supported by the FIOM approach.

Once the evolution model has been extended and an interaction framework established, it would then be possible to improve the efficiency and effectiveness of software development in a number of ways. First, the entire process of software development will become more automatic. As long as model/tool inputs and outputs can be supplied through the holistic model, different tools will be able to interact automatically, with less involvement by the software engineer. Second, because all artifacts within the holistic model are tracked together as a large dependency graph, it is possible to extract select "slices" of the dependency graph for particular purposes, allowing more "focused" development. For example, since the holistic model interacts with existing process models for software risk, reuse, testing, etc; it will then be possible to extract a "slice" of the entire dependency graph (a slice that represents the greatest risk) so that prototyping and analysis effort is not wasted on developing artifacts that are already well defined, understood, and/or successfully implemented in previous versions.

IV. Hypothesis. The following is a statement of the Research Hypothesis:

It is theoretically feasible to integrate a selected set of software development tools and/or models through application of a Holistic Framework for Software Engineering (HFSE). Where:
• The HFSE consists of an extended Software Evolution model integrated with a Federation Interoperability Object Model of the subordinate software development tools/models.

• The integrated tool/model set provides additional interoperability (i.e. additional data exchange and joint task execution) beyond that interoperability available prior to the application of the HFSE to the set.

V. Expected Contributions. The most important original contribution to the field of Software Engineering that this Dissertation proposes is to establish a Holistic Framework for Software Engineering. Embedded within this contribution are several other contributions:

• Identify and characterize the aspects of individual software development process models and tools within a holistic framework.

• Embed QFD within the Relational Hypergraph Software Evolution Model.

• Apply Young’s Object-Oriented Model for Interoperability for heterogeneous systems to an entirely different domain by establishing a Federation Interoperability Object Model (FIOM) between software development process models and tools.

• Integrate the extended Evolution model and the FIOM.

VI. Research Plan, Methods & Results. Conducting this research consists of executing the following major tasks

1. Complete the literature review.

2. Identify characteristics of individual software development process models and tools that must be accounted for within the holistic framework.

3. Embed QFD in the Relational Hypergraph Software Evolution Model [HARN99].

4. Apply the Object-Oriented Model for Interoperability for heterogeneous systems [YOUN01] to establish an interoperability federation between software development process models.

5. Integrate the extended Evolution model and the object federation.

6. Prototype the HFSE. During tasks 3 through 5 above, develop a working prototype of the HFSE.

7. Apply the HFSE to a selected set of tools (requirements engineering, project risk, prototyping, reuse, and testing). Establish evidence that the interoperability of the integrated tool set is improved (see below).
Experimental Design: In essence, I will choose a small representative subset of tools/models to show that the HFSE can be used to unify them, provide evidence that the interoperability of the sub-set of tools is improved, then attempt by theoretical arguments to characterize the class of tools and models that could also be unified with additional effort.

I will undertake a static group comparison test to provide confirming evidence of the dissertation hypothesis. This comparison can be characterized as follows [CAMP63]:

\[
\begin{array}{c|c}
X & O \\
\hline
O & O
\end{array}
\]

[experiment 1]

Where:

\[O \subseteq \{ \text{all software development tools and models} \}\]

\[X = \text{Application of the HFSE to } O\]

This is an experimental design in which a group which has experienced X is compared with one that has not, for the purpose of establishing how X effects the observation group O. In this case, the HFSE is applied to a selected sub-set of tools/models. The integrated sub-set of tools/models (after the application of the HFSE) are then compared to the same tools/models in a "stand alone" mode (i.e. without the benefit of integration by the HFSE). The comparison in this case will be to determine if there are any improvements in interoperability between the tools (i.e. improvements to data exchange and/or joint task execution). Specifically, I will be accumulating evidence of additional data exchange and additional joint task execution enabled by the application of the HFSE to the sub-set of tools/models. I will also be seeking counter-evidence that the HFSE reduces (or inhibits) data exchange and/or joint task execution.

VII. References:


