SOFTWARE ENGINEERING ENVIRONMENT FRAMEWORK EVALUATION METHOD: THREE EXAMPLES

M. Denise Kelley
The MITRE Corporation
1120 NASA Road 1
Houston, Texas 77058
(713) 335-8566

Kathy Rogers
GHG Corporation
1300 Hercules, Suite 111
Houston, Texas 77058
(713) 488-8806

ABSTRACT

The role of a framework is to manage the products, processes, and interfaces of a software engineering environment (SEE) and seamlessly reinforce and expedite an organization's life cycle process. The framework selection process is complex due to the increasing number of available frameworks and the rapid advances in technology and applicable standards. In this paper, a method for evaluating available frameworks is applied to three frameworks: a government-sponsored prototype for developing large systems, the Software Life Cycle Support Environment (SLCSE); a commercial Computer-Aided Software Engineering (CASE) product for developing Management Information Systems (MIS), Knowledge Ware's Application Development Workbench (ADW); and a commercial product for use in an Integrated Project Support Environment (IPSE), the Atherton Software BackPlane. Conclusions and recommendations regarding the usability, power, flexibility, and potential enhancements to the framework evaluation method are provided.

Keywords: Computer-Aided Software Engineering (CASE), Computer Assisted Software Engineering Environment (CASEE), framework, life cycle, management information systems (MIS), method, software, software engineering environment (SEE).

INTRODUCTION

A framework can be described as a structure to facilitate the integration of tools used across a life cycle. A framework provides a set of interfaces that supply tool builders with access to services and resources. The interfaces should provide stability and ease of integration. The architecture of the interfaces should support the progressive acquisition of tools.

The success of software projects at the National Aeronautics and Space Administration (NASA) Johnson Space Center (JSC) is predicated on the use of capable software engineering environments (SEEs) to support the development of large, complex and long-duration projects. The risk of not using a framework for a significant system (or using an obsolete framework) is that an enterprise will inefficiently expend budget of schedule resources. A framework allows efficient and timely acquisition of a SEE in which core support can be developed (or procured) initially. Subsequently, increments (tools and capabilities) to support a long-term cycle may be added as the project continues.

BACKGROUND

The emergence of the framework concept as a basis for progressively acquiring SEEs was perceived by the JSC Software Technology Branch (STB) as an important technology area to be studied; this resulted in the decision to evaluate framework technology. The benefit of examining a wide range of available framework options is the acquisition of knowledge supporting available frameworks' risks and capabilities. The intended result of exploring multiple options is to find the most appropriate frameworks for the various NASA JSC projects. However, looking at several frameworks in depth requires expenditure of significant time and effort. If more than one person or team looks at frameworks in parallel, comparison and contrast of the results may be difficult. This is due to differences in the approach, criteria, and terminology of each evaluation. This paper describes a consistent, reusable, efficient method for evaluating frameworks that will provide a basis for comparing framework evaluations across teams of evaluators.

GOAL OF THE EVALUATION

The goal of the evaluation of framework technology is first to determine the capability of a framework to support the phases, activities, roles, and products of the software engineering life cycle, and second to learn lessons that could be beneficial to current and future programs and projects at JSC. The Evaluation Method for SEE Frameworks was created and refined during the evaluation of the Software Life Cycle Support Environment (SLCSE) framework. This evaluation method was developed to reduce the risk associated with selecting the most appropriate framework among emerging frameworks. It is intended to address the critical issues of framework selection in a flexible, efficient, and repeatable manner. Many of the products of the evaluation (such as the evaluation questions and the information gathered on the appropriate reference models) can be reused or refined across evaluations.

EVALUATION METHOD

The basis of the Evaluation Method is the evaluation of five qualities that describe the essence of a framework: cost, usability, power, flexibility, and maturity.

The cost of a framework is measured not only in terms of the dollars that are required to purchase the software that comprises the framework but also the cost of using the framework and its associated hardware and software system. These additional costs can be characterized in terms of the expense of the supporting hardware, operating system, or other supporting software, the cost (both dollars and time) for the necessary training to use the framework effectively, the cost to the organization of changing business procedures to accommodate the framework, and the cost of supporting the framework. An additional cost associated with a framework is the difficulty of integrating tools. If few tools are available, or the available tools are expensive or difficult to use, the overall expense of the framework increases. The cost of a framework to a single project can be reduced if the framework, and the associated skills developed to use the framework, can be used on several projects.

Usability of a framework can be measured in many ways. Aspects of usability include user friendliness, quality, suitability, and functionality. The general principle of user friendliness, for example, can be broken down into practicality, convenience, and helpfulness. The quality of the interface and the amount of support that is required to interact with the framework are should be noted in the criteria. In addition, the suitability of the framework to the type of development done by an enterprise must be considered. For example, the amount of effort necessary to use the framework should be weighed against the benefits of the framework; very small projects may benefit from the use of individual tools without a framework. The actual function of a framework must be considered to ensure that the framework provides sufficient (but not excessive) project support, appropriate language support, and correct project deliverables.

The power of a framework is its ability to increase the productivity of individuals using the environment built upon the framework to do what is required to complete the project. The capacity of the framework to manage life cycle phases, activities, roles, and products must provide not only enough throughput to support the volume of traffic created by the tools within the environment, but also support the life cycle functions (or components) at an appropriate level. It may be the case that the support provided by a framework adds power to some users at the expense of others. For example, a stringent configuration control system might enhance the capabilities of the configuration management group at the expense of programmers. Information of an administrative nature might be provided at a cost to overall performance. The overall leverage provided by the framework to do complex, undesirable tasks, or repetitive tasks must be weighted against the overhead associated with using the framework. If the net result is positive, the framework provides power to the overall project.

Flexibility is defined as extensibility, tailorability, and scalability. A framework should support incremental building so that portions of the environment can be acquired as they are needed, rather than all at once. The framework should also be tailorable to accommodate the specific functions of an organization as well as to accommodate new opportunities that might arise. Scalability is the ability of a framework to scale up or down to meet the specific needs of a project. This report has already acknowledged that not all projects require an environment; but even among those that do, there is a wide range of project and team sizes to be supported.

The maturity of the framework is predicated on its experience base, trained personnel, progress over time, and stability. A framework that has a trained user base provides both available, knowledgeable personnel and a known success rate. If a framework has matured over time without significant disruptions, it is more likely to be well engineered than one that has undergone significant modification to accommodate changes. A framework that has accommodated change yet remained stable over a period of time can be classified as mature.

The evaluation method consists of five steps:

- Establish a basis for reviewing the framework Background information should be studied to establish an understanding of the current state of framework technology as well as the context in which the framework will operate to answer the questions that characterize essential framework characteristics. In order to establish a basis for understanding framework

technology at the beginning of a framework evaluation the following steps are necessary:

- Create an evaluation plan and schedule
- Research available literature
- Understand the culture of the users
- Provide a preliminary analysis of the framework
- Interview framework users
- Interview framework developers
- Select a reference model

If more than one framework is being evaluated, some of the information collected by the first evaluation team (or on the first framework evaluated) can be reused by later evaluation teams. The desired results of this step are a sound basis for conducting a thorough evaluation in a timely manner resulting in enough knowledge of the framework to determine the potential value of continuing the evaluation.

- Establish evaluation goals

Goals serve as the compass for the evaluation. To provide the evaluation customer with the appropriate amount and level of information, document the goals before the evaluation. To keep the evaluation on the right track, review the goals frequently. At each review, determine whether goals are being met and whether all the goals are still reasonable, considering what has been learned during the evaluation. Define the evaluation goals in terms of specific and bounded criteria. State questions to assess or measure the criteria in clear and concise terminology. Establish a test case that provides appropriate direction to the evaluation. Documentation of the goal(s) of the evaluation is the product of this step.

- Conduct a test case

After choosing a test case, implement that test case using the framework. Evaluate a broad range of framework functions, but explore issues of particular importance to the user organization at some depth. If a test case is used across several evaluations, do not misinterpret increased understanding of the test case as better performance of the framework. Whether similar of different test cases are being used, separate the difficulties related to the test case from the difficulties related to the framework. The evaluation may end here if the test case reveals significant problems with the framework. The test case should result in identification of the assumptions upon which the framework is based, the limitations of the framework, and the functionality the framework provides. The overall result of application of the test case determines the areas in which the framework performed well and those in which it performed poorly. If the evaluation ends based on the results of the test case, include the rationale (as it is always possible that future versions of the framework may solve the identified problems). If the framework performed well, document the results for reexamination at the end of the evaluation.

- Document recommendations and conclusions

The initial evaluation information, the framework goals and the test case information serve as the basis for making recommendations and conclusions on the framework. The recommendations discuss the changes and enhancements to be considered for future versions of the framework. The conclusions determine whether the framework is adequate for the project to be done by the organization (as well as its suitability to future projects). The recommendations and conclusions should be based upon the framework concepts in light of the implementation of the framework. The importance of documenting the findings of the evaluation cannot be overstated. If the user organization is responsible for making the final decision on the use of a framework, a clear, concise evaluation report will provide the best vehicle for communicating the information gained by the evaluation team. IF the responsibility for framework selection rests with the evaluation team, documented findings will provide a justification for the team's decision. Written findings will not fade over time or become confused over the course of multiple evaluations. Findings can also serve as a starting point for evaluating new versions or revisions of a framework. Once the evaluation is captured in writing, it can be widely disseminated to other organizations conducting evaluations or interested in the results of evaluations.

- Refine the evaluation process

Throughout the process of evaluating the framework, collect information on issues raised (not just those that were explored), the rationale for decisions and the problems that were encountered during the evaluation. The importance of documenting the evaluation itself, in addition to documentation of the evaluation findings, is important to the refinement and repetition of the process. Written findings provide a basis of comparison over the course of multiple evaluation. Documented issues provide insight to future evaluation teams without relying on the direct participation of previous evaluators. Documenting the evaluation process is the best way to communicate and preserve the lessons learned from doing an evaluation. The result of this step documents the strengths and weaknesses of the evaluation process as practiced on the test case.

The five steps are intended to ensure timely, accurate analysis of the applicable framework features. Timeliness is supported by reducing the investigation to a small but descriptive set of characteristics. Risk management and cost containment are supported by investigating the key risks and benefits in the context of the culture of the user organization. Use of the evaluation method is intended to result in the selection of a cost-effective framework.

The criteria to be measured during the selection process were refined and enhanced by the STB Configuration Control Board. The SLCSE, ADW and Atherton BackPlane were evaluated according to the criteria in

accordance with the evaluation method; the evaluation of each was included as a study of the application of the evaluation method.

RECOMMENDATIONS AND LESSONS LEARNED

The following enhancements are recommended for the Evaluation Method for SEE Frameworks:

- Continue refinement of the Evaluation Method for SEE Frameworks
- Balance evaluation team size and skills
- Schedule frequent reviews
- Document evaluations as they proceed
- Develop objective framework measurements
- Maintain awareness about planned framework enhancements

Lessons learned, based on the experiences of applying the method to three test cases, SLCSE, ADW and Atherton BackPlane include:

- Dependence of the system on a specific configuration should be eliminated.
- Performance and the user interface should be improved to meet user expectations. The preponderance of reasonably priced, powerful, graphical user workstations establishes user expectations on the level of capabilities and performance required in a system.
- Automated support for documentation and role addition should be provided. Automation is also required to support modification functions.
- The framework concept should be leveraged to provide the ability to progressively acquire a SEE. The power of the populated framework should be greater than the capability of the sum of the power of the individual tools used to populate the framework.
- The automated document-generation function should be enhanced to smooth the life cycle effort. The ability to engineer requirements, design, and implementation was found to be much greater when the user's focus did not have to shift into a documentation mode and then shift back into an engineering mode.

Current and future framework evaluations for the STB have been identified.

BIBLIOGRAPHY

ANSI/IEEE STD 1002-1987, "IEEE Standard Glossary of Software Engineering Standards," American National Standards Institute, New York, NY.

Baldwin, R. W., and D. E. Emery, 19 December 1990, "Technology Assessment of the Software Life Cycle Support Environment", Draft, The MITRE Corporation, Bedford, MA.

Begley, Zeynep, 24-27 April 1990, "Hanscom Air Force Base SLCSE Training Course", Hanscom Air Force Base, Massachusetts.

Defense System Software Development Standard, 29 February 1988, Department of Defense, DoD-STD-2167A.

Earl, Anthony, 17 August 1990[b], "A Reference Model for Computer Assisted Software Engineering Environment Frameworks", HPL-SEG-TN-90-11, Software Environments Group, Hewlett-Packard Laboratories, Bristol BS12 6QZ, England, Version 4.0 ECMA/TC33/TGRM/90/016.

Erb, D. M., 20 November 1990, "Software Engineering Methods", presented to Software Technology Branch, Information Technology Division, Information Systems Directorate.

Hogan, Martha, 23-26 October 1990a, "NASA/JSC SLCSE Training Course", NASA/JSC, Houston, Texas.

Kelley, M. D., draft, 12 February 1992, "Evaluation of KnowledgeWare's Computer-Aided Software Engineering (CASE) Application Development Workbench (ADW) Toolset", The MITRE Corporation, Houston, Texas.

Labasse, D. L., draft, 8 February 1991, "A Comparison of the Atherton Software BackPlane and the ECMA CASE Environments Frameworks Reference Model".

McDermid, J. and K. Ripkin, 1984, "Life Cycle Support in the Ada Environment", Cambridge University Press.

Rogers, K. L., M. Bishop, and C. W. McKay, March 1991[a], "An Overview of the Clear Lake Life Cycle Model", *Proceedings of the Ninth Annual National Conference on Software Technology*, Washington, D.C., pp. 206 - 222.

Rogers, K. L., April 1991[b], "Software Engineering Environment Frameworks Volume I: Evaluation Method", MTR-91W00048-01, The MITRE Corporation, Houston, Texas.

Rogers, K. L., May 1991[c], "Software Engineering Environment Frameworks Volume II: Evaluation of the Software Life Cycle Support Environment", MTR-91W00048-02, The MITRE Corporation, Houston, Texas.

SLCSE Technical Staff, August 1989, "Software Life Cycle Support Environment Software User's Manual, Volume I", CR-6-1537, General Research Corporation, SPS, and Intermetrics.

SLCSE Technical Staff, October 1989, "Software Life Cycle Support Environment 2167A Schema", General Research Corporation, SPS, and Intermetrics.

Strelich, Tom, February 1990[a], "Software Life Cycle Support Environment", RADC-TR-89-385, Final Technical Report, General Research Corporation, Santa Barbara, CA.

Strelich, Tom, 22 October 1990[b], "Software Life Cycle Support Environment A Computer-Based Framework for Developing Software Systems", paper presented at NASA/JSC, Houston, Texas.