NISTIR 7061

Metric Mapping Concepts for TIA

Michelle Potts Steves
Jean Scholtz

National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

NISTIR 7061

Metrics Mapping Concepts for TIA

Michelle Potts Steves
Manufacturing Systems Integration Division
Manufacturing Engineering Laboratory

Jean Scholtz
Information Access Division
Information Technology Laboratory

July 2004

U.S. DEPARTMENT OF COMMERCE
Donald L. Evans, Secretary
TECHNOLOGY ADMINISTRATION
Phillip J. Bond, Under Secretary of Commerce for Technology
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
Arden L. Bement, Jr., Director

Abstract

In this paper, we present a metrics framework and the underlying concepts that were used to help structure a series of evaluations of software tools to support intelligence workers with their data filtering and analytic processes. The deployment of the tool suite and subsequent evaluations were part of the Terrorist Information Awareness (TIA) program, a Defense Advanced Research Projects Agency (DARPA) - sponsored effort. First we summarize some of the requirements of the evaluations which drove the development of the framework. We enumerate some of the benefits of a top-down approach, particularly the GQM method from which the metrics framework is adapted. Then, we present the framework and its constituent elements. An excerpt of an instantiation of the metrics framework is shown and discussed for an example evaluation situation. Finally, we describe how the metrics framework can be used in comparative studies.

Introduction

This paper presents a metrics framework and the underlying concepts that were used in the Terrorist Information Awareness (TIA) program, a Defense Advanced Research Projects Agency (DARPA) - sponsored effort. The framework was used to help structure a series of evaluations of a changing suite of software tools designed to support intelligence workers in data filtering and analytic processes. Performance of individual system components as well as the impact of the entire software suite was of interest in the evaluations. The framework provides the possibility of comparing the results from different evaluations over time with changed tool sets. The framework calls for articulation of system goals, experiment and evaluation objectives, conceptual metrics, and measures, both conceptual and implementation-specific. We base our definition of the term 'metric' on the IEEE Standard 1061-1998 [5]. Further, we describe the concept of 'measure' as an element which contributes to the assessment of a metric. The mapping of goals down through objectives, metrics, and measures imposed a discipline on measurement capture so the evaluation team could focus on collecting the rich and pertinent measures needed for each evaluation. This up-front planning and identification of data was especially critical in the operational environment in which many of the evaluations occurred.

Requirements

There were numerous and varied requirements for the evaluations conducted under the TIA program. Several of these requirements are briefly described here to provide insight into the motivation for development of a metrics framework. Evaluation goals included assessment of individual system component performance; determination of the impact of the entire software suite on the analytic process; and evaluation of intra-organizational and inter-organizational collaboration activities. Head-to-head assessments of individual tools providing the same (or similar) capability were performed to inform the TIA system architects of the appropriateness of individual tools for a particular task or group of tasks. Assessment of a tool's performance relative to the developer's claims was also performed. Other evaluation goals included impact on the analytic process and analytic product quality.

Individual components within the tool suite changed over time. New tools replaced tools that had completed the testing process and the suite of deployed tool capabilities broadened as new tool capabilities became available that were perceived as useful. This situation required that evaluations of tools be performed in the context of required system functionalities rather than only a particular instantiation of the tool suite.

Because a series of experiments was conducted, it was desirable to have the possibility of comparing appropriate portions of various evaluations. The following is one example of such a comparison: the impact of tool *X* providing capability *M* on the analytic process at some time, $t_{n,}$ compared to the impact of tool *Y* providing capability *M* on the analytic process at t_{n+1}. In this example, the tools *X* and *Y* both provide the same capability, *M*, and they are inserted in the tool suite for different experiments, t_n and t_{n+1}, the comparison of their respective impacts on the analytic process is of interest.

Several experimental contexts or 'platforms' were devised to separate logistical concerns, most notably data characteristics. While the specifics of the platforms[1] are not relevant to this paper, the existence of different experimental contexts increased the number of evaluation environments and contexts, creating additional complexity.

Evaluation results were needed to inform the next round of experiment planning. Experiments were designed and conducted by one group and evaluated by another group. Experiments were scheduled aggressively, one every three months. Evaluation results were needed almost as soon as one experiment ended so that they would influence the design of the next experiment. To produce evaluation results quickly and effectively, we needed a way of associating evaluation questions with the collected data.

[1] For more information on how platforms were used for graduated testing of software components, see the [7].

As the above requirements illustrate, the TIA program had a complex, experimental environment, aggressive schedule, and varied evaluation requirements. Well-thought-out evaluations needed to be performed, data analyzed, and the results presented in almost rapid-fire succession. Clearly, a discipline was needed to produce well-informed evaluation results.

Goal-oriented evaluation paradigms

The expectation for the TIA program was that the experiments would identify, develop, and implement technologies that aided intelligence analysts in their mission to cope with foreign asymmetric threats, such as terrorism [7]. Therefore, each tool deployment and the subsequent evaluation of its performance needed to be tied to this overarching program goal. The program planned to employ measurement to effect the evaluations, which is fortunate, since "measurement is an ideal mechanism for evaluating any software project goal." [11] Rombach further states that in order for measurement to be successful, effective 'top-down' strategies that derive metrics and associated measures from goals and interpret measurement data in the context of goals are needed.

Several approaches using this 'top-down' manner of identifying useful metrics from goals appear in the literature: the Software Quality Metrics (SQM) approach by Murine [10] based on prior work by Boehm [3] and McCall [8], the Quality Function Deployment (QFD) approach by Akao [6], and the Goal/Question/Metric (GQM) approach by Basili [host of refs]. While all three methods have the top-down measurement approach in common, which is considered a major improvement over the commonly used 'bottom-up' approach to measurement, they vary significantly in the scope of supported measurement goals and potential uses [11]. We focus on the GQM method since it is identified as one of the most widely used of such methods. Case studies of its use have shown sizable value and adaptability to specific environments [4]. Furthermore, GQM can be used for process as well as product quality, whereas SQM and QFD are limited to product quality [11]. Rombach further states that "GQM benefits include its general applicability to all kinds of measurement goals, as well as its support for identifying and tailoring of metrics for interpreting collected data in context, for validating the usefulness of the selected metrics early on, for involving all interested parties in the measurement process, and for protecting sensitive data." [11]

GQM was developed for use with software improvement projects, and indeed, the TIA program can be viewed as such a project. The GQM paradigm prescribes setting goals in operational and tractable ways. Goals are then refined into a set of quantifiable questions that specify metrics. Data is tied to specific metrics that in turn are tied to a specific goal. Because of this clear association, Basili and Rombach [1] state that use of GQM should "help in the interpretation of the results of the data collection process," and that GQM facilitates identification of the link between the actual data and the purpose for its collection. The GQM method further prescribes a process of goal, question, and metric selection via a set of templates and guidelines contained method.

Metrics framework for TIA

Although we selected the goal-oriented approach, we did not adopt the GQM process of selecting goals, questions, and metrics for the TIA evaluations. The TIA evaluation effort required more leeway in specifying framework elements than allowed by the GQM method. Additionally, we required a distinction between metrics and two types of measures to allow for the possibility of comparing results across experiments. To meet these requirements, we constructed a framework for articulating goals tied to experiment and evaluation objectives tied to associated metrics and measures that could be reused in TIA experiment after TIA experiment. The framework elements include system goals, experiment and evaluation objectives, conceptual metrics, and measures, both conceptual and implementation-specific. We felt that this would be sufficient to perform meaningful evaluations for the TIA program; fit program requirements of separate, although at times related, experiment and evaluation objectives; and, provide for the opportunity to compare evaluations over time (as appropriate), with changing tool sets and work processes. The expectation was that the metrics framework would have several major benefits, including: 1) since the measurement data were tied to system goals, data interpretation would be more efficient and effectively tied to stated goals, and 2) data collection would be orderly and help to focus on collecting just

the required data. Although we did not employ the GQM process of selecting goals, questions, and metrics in the TIA evaluations, we used its conceptual underpinnings to develop an evaluation framework to meet TIA program requirements, specifically the tying of metric and measure selection to system goals.

First, we present a diagram of the framework and then discuss each of the levels in more detail. Each successive level in the framework is a refinement in scope. This refinement can also be thought of as a level of abstraction or sphere of concern. For example, a system goal may have one or more experiment goals. An experiment goal, in turn, may have one or more evaluation goals. Each evaluation goal will have one or more conceptual metrics, which when assessed with its associated measures, will contribute to an assessment of if, and potentially how well, a particular system goal was met. See figure 1.

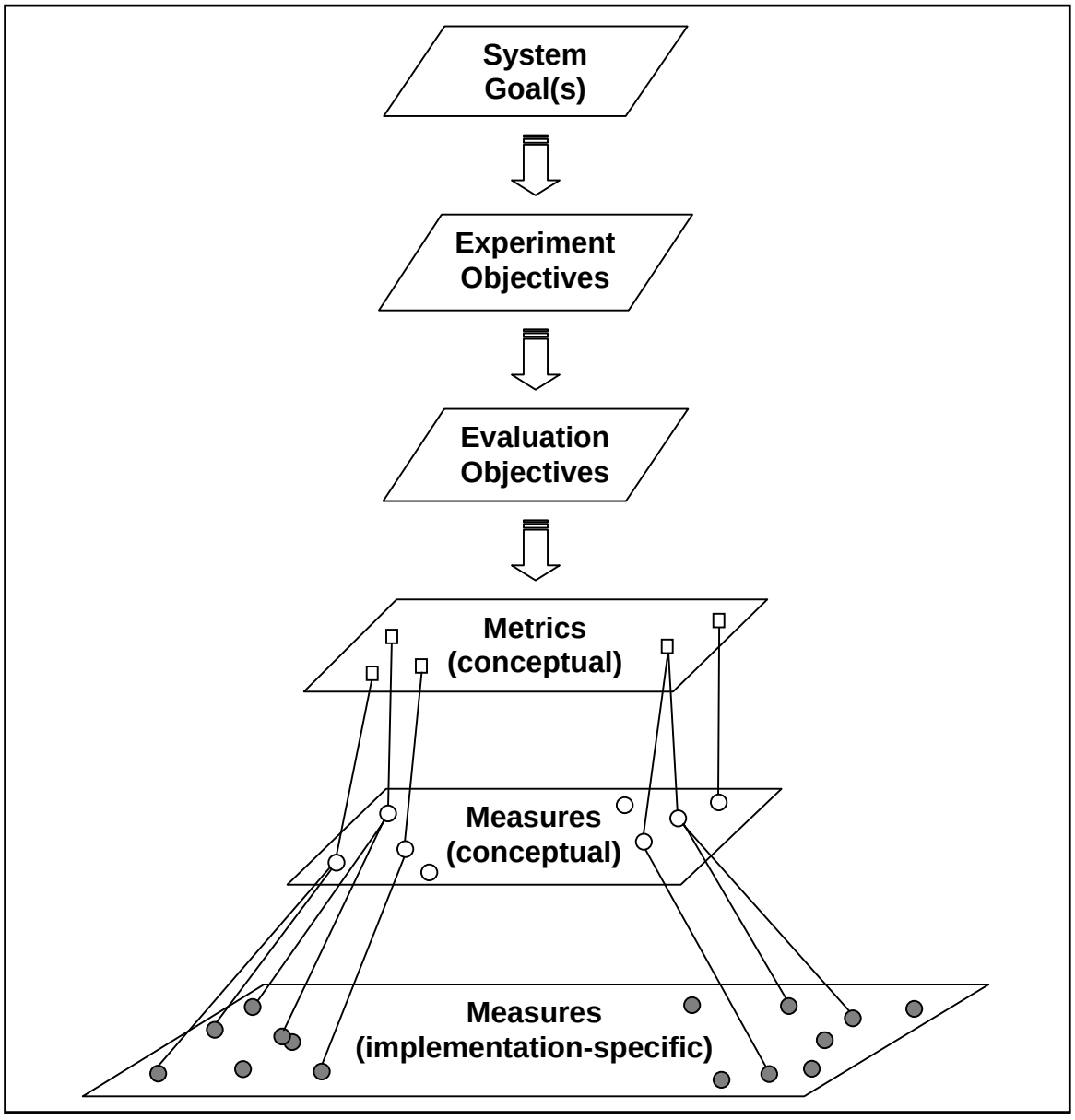

Figure 1: Framework for metrics mapping

1. System goal

The framework element ***system goal*** is the intended benefit or functionality the software system will provide. The system goal element provides two critical ingredients of system evaluation design: those aspect(s) of the system of primary importance in the work domain and the high-level question the evaluation must answer – whether the stated goal was met. At this uppermost level, the system goal constitutes the beginning of the evaluation design and the end of the analysis process. Therefore, thoughtful and careful construction of system goal statements is particularly important so that relevant evaluation questions can be formulated in the evaluation design and subsequently answered in the evaluation analysis.

While it was not our intent to utilize the GQM method for goal definition per se, we believe that Basili provides some useful considerations for goal statement construction. The GQM template for goal definition includes concepts such as analysis purpose and intent, point of view, and environmental context. See the template below and [1, 2, 9] for additional detail on the GQM guidelines for goal definition.

GQM template for goal definition [11]

- Purpose:
  Analyze some
  (objects: processes, products, other experience models, …)
  for the purpose of
  (why: characterization, evaluation, prediction, motivation, engineering, control, …)
- Perspective:
  With respect to:
  (focus: cost, correctness, changes, defect removal, user friendliness, maintainability, reliability, …)
  from the point of view of the:
  (who: user, customer, manager, developer, corporation, …)
- Environment:
  In the following context:
  (environment factors: problem, people, resources, processes, …)

2. Experiment objectives

The next level of the framework is the ***experiment objective*** level. It is the set of objectives that drives the overall experiment, within the context of the system goal(s). The objectives at this level influence the formation of the more specific evaluation objectives for a particular experiment. This set of objectives may be gathered from multiple input sources. Clear articulation, at an appropriate level of detail, of these objectives provides further refinement in the 'top-down' design of the evaluation.

In the case of TIA, there were two main groups providing input on the experiment design process: the Experiment Planning TIA Project Team (TPT) and the Metrics TPT. Each group had its own concerns and the Metrics TPT-spawned experiment objectives were not necessarily entirely a subset of the Experiment Planning TPT's objectives. For this reason, the framework distinguishes between experiment and evaluation objectives, since the Metrics TPT had responsibility for all the evaluation objectives being met, but not necessarily the responsibility for all the experiment objectives. In cases were experiment and evaluation objectives do not diverge, the experiment and evaluation objectives framework elements could be collapsed into one objectives element. In any case, good communication between these groups was critical to the success of experiment and evaluation design. See figure 2.

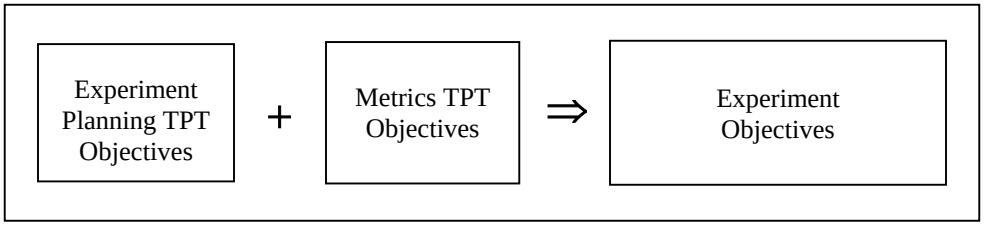

Figure 2: Formation of TIA experiment objectives with input from multiple TPTs

At this high level, the Metrics TPT had the opportunity to specify objectives that spanned experiments. Doing so opened the possibility of designing multiple evaluations whose results could be compared. For example, the Metrics TPT specified overarching objectives as follows:

1. Provide feedback on how to improve the tools
2. Assess how the tools support the current process
3. Assess how and why the analytic process changes
4. Identify areas about which more must be learned

Each successive refinement further characterizes the environment or context in which the evaluation will be executed. For example, the Experiment Planning TPT often defined experiments to exercise particular aspects of the tool suite. This meant that some tools would have little or no use during a particular experimental period. If, for example, a measure concerning spontaneous tool use was collected and a particular tool's use statistics dropped drastically from one experimental period to another, outside the context of this experiment objective, one might interpret the data to mean there was something undesirable about that tool, which might very well be a completely false conclusion.

3. Evaluation objectives

The third level in the framework is ***evaluation objectives***. The set of evaluation objectives is a synthesis of the experiment objectives at a lower, more refined level of abstraction. Evaluation objectives provide guidance for selecting appropriate metrics and measures. This is the step in the process of defining objectives that scopes evaluation activities.

4. Metrics and measures

The lowest three levels in the framework constitute ***metrics*** and two types of ***measures***. Metrics and measures are somewhat interwoven; however, differences in levels of abstraction can be discerned. We start by defining the terms.

The Institute of Electrical and Electronics Engineers (IEEE) standard for Software Quality Metrics Methodology defines the term 'software quality metric' as "a function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which software possesses a given attribute that affects its quality." [5] We define *metric* as the interpretation of one or more contributing elements, e.g., measures or other metrics, corresponding to the degree to which the set of attribute elements affects its quality. Interpretation can be human assessment of the contributing elements or a computation. The computation can be the summed, weighted values, of the contributing elements. When weights are used, they are assigned to elements according to their importance to the respective attribute with respect to the particular assessment scenario. Our definition agrees with the IEEE definition, except that we also recognize that an attribute may have multiple elements, referred to in this paper as *measures*, which contribute to an attribute's assessment, and the assessment of a metric may be computed or interpreted. For example, a complicated attribute of a software component like efficiency, is partially derived from the interpretation of elements, i.e., measures, like time, user ratings, and tool usage.

We define a *measure*, a noun, as a performance indicator that can be observed singly or collectively. This concept corresponds to the term 'element' used to assess an attribute. Measures are countable events, can be directly observed, computed, calculated, and may, at times, be automatically collected. A simple way to distinguish between metrics and measures is by the following statement: a measure is an observable value, while a metric associates meaning to that value by applying human judgment, often through a formula based on weighted values from the contributing elements or measures.

4. a. Metrics

Using the top-down design approach advocated in this framework, a metric is scoped by its parent evaluation objective. Likewise, assessments for each of the metrics within the set of metrics for a particular evaluation objective inform the assessment of that evaluation objective. Each metric scopes and is informed by its associated measures.

4. b. Measures – conceptual and implementation-specific

The lowest two levels in the framework constitute 'measures.' These levels represent measures that are required to substantiate the formulation of answers to evaluation questions. There are two levels of abstraction for measures: conceptual and implementation-specific. Conceptual measures identify the type of data to be collected. Implementation-specific measures identify the specifics about a particular collection instance, e.g., data element(s), associated tool, and collection method or protocol. For TIA, there are two additional attributes for implementation-specific measures: measure type (see below) and platform relevance. Once a value is obtained for an implementation-specific measure, that value is analogous to the term 'measured value' [9]; specifically, it is "the numerical result obtained from the application of a measurement method to an object, possessing a quantity." [9]

The following example shows the difference between conceptual and implementation-specific measures: task completion time (a conceptual measure) can be calculated in different ways with different implementation-specific measures. The calculation can be based on start and end times or based on start time and duration.

Implementation-specific measure specifications should be identified and documented for each evaluation as part of the TIA experiment documentation. Because they are contextually-specific, and not conceptual, they are not captured in the evaluation template that spans experiments. Selection of implementation-specific measures uses an essentially a 'bottom-up' approach, since it is rooted in pragmatic issues such as which data elements are available and cost to collect. However, since conceptual measures are rooted in top-down requirements and place constraints on the domain of possible implementation-specific measures, we feel that the use of the bottom-up approach is quite appropriate, if not required, here.

Logical types of measures for TIA

The Metrics TPT defined three logical categories of implementation-specific measures that represent different perspectives. They are as follows:

- Cognitive – focus on the behaviors and cognition of the individual actors in TIA
- Operations – focus on the processes (i.e., "behaviors") of an organization in an operational context
- Technical – focus on actor interactions as well as system performance

The Metrics TPT used this logical division as an aid in checking whether a particular metric was informed from multiple perspectives. For example, assessments of metrics that had only one measurement data source were more suspect than assessments with multiple, correlating data from multiple perspectives and sources. Additionally, good representation of measures from all three perspectives provided the sense that a holistic view of system performance was being determined.

Example

To illustrate the concepts we provide the following example. A project manager wants to assess the impact of instituting a messaging system in a particular work process. Not only are different types of messaging systems being considered, e.g., instant messaging and computer-based audio tools, but various tools within a particular capability category have been suggested for trial, e.g., different instant messaging tools.

In this example, both individual tool performance and impact on process are of evaluation interest. An evaluation template can be employed to compare results on individual tools. Below is an excerpt from an example evaluation framework which includes the conceptual elements: goal statements, evaluation objectives and metrics mapped through to conceptual measures (CM) using the framework introduced in

this paper. Additionally, sample implementation-specific measures (IM) are shown with their associated conceptual measures.

Goal statement 1: The generic messaging tool provides for synchronous and asynchronous communication.

Evaluation objective 1-1: Assess synchronous communication capability.

Metric 1-1a: Effectiveness
CM: composition time
IM (text): start of typing until 'send' function activated
IM (audio): start of record until 'send' function activated

CM: editing capability
IM (text): number of editing features
IM (audio): absence/presence of re-record option

CM: expressive capability
IM (text): number of formatting features
IM (audio): comfort level of people to use voice inflection in recording messages

Metric 1-1b: User satisfaction of interface
CM: configurable message notification controls
IM (text & audio): user rating

CM: frustration levels
IM (text & audio): user rating

Evaluation objective 1-2: Assess asynchronous communication capability. (The metrics and measures for this evaluation objective will be quite similar to Evaluation objective 1-1, and are not given here in the interest of brevity.)

Goal statement 2: The messaging tool will positively impact the work process.

Evaluation objective 2-1: Assess changes in communication response times.

Metric 2-1a: Efficiency
CM: response times
IM (text & audio): time to listen, time to respond

Metric 2-1b: Effectiveness
CM: usage statistics
IM (text & audio): number of round-trip communications

CM: number of misunderstandings
IM (text & audio): count of clarification only messages

In the excerpt, two system goal statements are given. The first is concerned with the technical performance of the tool and its interface for individual users with regard to synchronous and asynchronous communication. The second system goal is concerned with assessing the impact on the work process. These two system goal statements are given as examples, in a real evaluation clearly more system goal statements would be required to give a fuller understanding of how well any particular tool might perform. Additionally, sample evaluation objectives with associated conceptual and implementation-specific measures are given. Some implementation-specific measures might apply to both text-based and audio-based messaging tools as noted, although the specific collection details may vary and would be noted in a fully-specified evaluation plan.

Using the framework for comparative evaluations

Capitalizing on the design of the TIA program as a series of experiments, we took the concept of the metrics framework one step beyond its use in designing evaluations for individual experiments. We also proposed that the TIA Metrics Team develop an ***evaluation template*** using the framework that could be used in subsequent experiments. The evaluation template specifies the high-level, conceptual elements of the metrics framework – goals, objectives, metrics, and conceptual measures. These elements require careful selection to be applicable for multiple TIA experiments. For any individual experiment, high-level elements would need review for applicability and low-level measures would need specification. While specification of conceptual elements that are applicable to multiple experiments requires more up-front work, it offers the possibility of being able to compare results of like-structured experiments.

There are situations where it is desirable to affect comparative studies of systems and for determining how organizational processes are affected when technologies are inserted into a particular process. To structure such a comparative study, an evaluation template specifying the conceptual elements of the framework is designed for a series of like-structured experiments. For each individual experiment, the low-level, implementation-specific measures are specified, as they will differ by varying degrees for each experiment. Thoughtful specification of the evaluation template elements is critical so that the template is applicable to each experiment in the envisioned series.

To illustrate, let us consider the following example. A comparative study from a series of evaluations could be constructed using an evaluation template for the example evaluation project described in the previous section regarding the evaluation of different messaging systems. To accomplish this, the most important capability and impact goals are identified. From there the evaluation objectives, and associated metrics and conceptual measures are specified for each goal. For each tool selected for evaluation, implementation-specific measures are identified for associated conceptual measures. After the series of evaluations is performed, the results of individual experiments can be compared since the series of experiments presumes the same work groups and processes were used throughout. Additionally, because the most important system functionalities are articulated in the conceptual elements of the framework, the results can be 'rolled up' from the measures to assess how well tools meet operational objectives.

Use of evaluation templates requires more up-front work in the design of the entire series of evaluations. However, we feel its use provides several important benefits including: (1) the overhead of determining metrics and measures for each evaluation is greatly reduced after the initial work to develop the template is performed, (2) data requirements are well-understood prior to the analysis phase of the evaluation – therefore, data collection can be planned in an orderly manner and the impact of not being able to collect particular measures will be understood in advance of the analysis phase, and finally, (3) because the high-level, conceptual elements of the evaluation are reused in subsequent experiments, it might conceivably be possible to compare evaluation results over time – as long as other factors are controlled appropriately.

For TIA, the most important advantages were: providing the opportunity to effect comparative studies and with the reduction of evaluation design time for individual experiments, execution of evaluations could keep pace with the demanding experiment schedule.

Summary

This paper provides a description of a framework for developing evaluations using a top-down approach for the TIA program. The rationale for choosing the approach is given in the context of evaluation requirements and constraints for the program. Additionally, capitalizing on the nature of the program, reuse of an evaluation design template is postulated for a series of similar experiments, with the possibility of comparing evaluation results across experiments.

References

[1] B. Basili and H. Rombach. “Tailoring the software process to project goals and environments”. In Proceedings of the 9th International Conference on Software Engineering, pp 345-357, 1987.

[2] B. Basili and H. Rombach. “The TAME project: towards improvement-oriented software environments”. In IEEE Transactions on Software Engineering, SE-14(6): 758-773, June 1988.

[3] B. W. Boehm, J. R. Brown and M. Lipow, “Quantitative Evaluation of Software Quality,” Proceedings of the 2nd International Conference on Software Engineering, 1976, pp. 592-605.

[4] K. E. Emam, N. Moukheiber and N. H. Madhavji. “An Empirical Evaluation of the G/Q/M Method”. In Proceedings of the 1993 Conference of the Centre for Advanced Studies on Collaborative Research: software engineering, Vol. 1, Toronto, Ontario, Canada, Pages: 265 - 289

[5] IEEE Standard for a Software Quality Metrics Methodology, IEEE Std 1061-1998, 1998, pp. 2-3.

[6] M. Kogure and Y. Akao, “Quality Function Deployment and CWQC in Japan,” Quality Progress, October 1983, pp. 25-29.

[7] G. Mack, K. Longeran, J. Scholtz, M. Steves and C. Hale. “A Framework for Metrics in Large, Complex Systems”. In Proceedings of the IEEE Aerospace Conference, Big Sky, Montana, US, March 2004.

[8] J. A. McCall, P. K. Richards, G. F. Walters, “Factors in Software Quality,” RADC TR-77-369, 1977.

[9] MEL/ITL Task Group on Metrology for Information Technology (IT), “Metrology for Information Technology”, NISTIR 6025, National Institute of Standards and Technology, Gaithersburg, MD, 1997, pp 8.

[10] G. E. Murine, “Applying software Quality Metrics in the Requirements Analysis Phase of a Distributive System, “Proceedings Minnowbrook Workshop Blue Mountain Lake, New York, 1980.

[11] H. D. Rombach, “Practical Benefits of Goal-Oriented Measurement,” Software Reliability and Metrics, eds. N. Fenton and B. Littlewood, Elsevier Science Publishing Co, London, 1991, pp. 217-235.