

Objective Measures from Model-Based Testing

Mark Blackburn, Robert Busser, Aaron Nauman
(Software Productivity Consortium)

Many businesses are looking for the right project measures as they relate to project planning, scheduling, and performance. This paper gives guidance on defining, collecting, and analyzing measures derived from a model-based testing method. These measures and their use are described in terms of an information model adapted from the ISO/IEC 15939, Software Engineering - Software Measurement Process. The model-based method associated with these measures involves modeling requirements and mapping modeled requirement variables, referred to as object mappings, to interfaces of the target test system.

Keywords: Software Engineering - Software Measurement Process, Test Automation, Model-Based Testing, Requirement-based Testing

1. Introduction

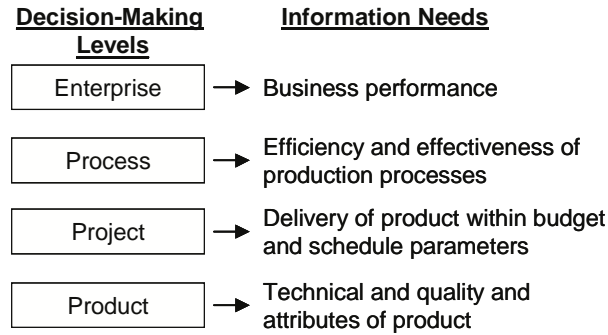
There are several approaches to model-based test generation. Each different approach relies on modeling methods that combine formalisms such as state machines, condition/action, event/action, control system, language, and hybrids. Once the tests are generated from the models, they can be transformed into test scripts that can be executed automatically. The key advantage of this technique is that the test generation can systematically derive many combinations of tests associated with the requirements represented in the model to automate both the test design and test execution processes. Robinson hosts a website that provides useful links to model-based testing information [Rob00].

The Test Automation Framework (TAF) is a method with related tool support. TAF supports several modeling approaches with automated model analysis, and test generation. Companies using TAF find that it helps reduce cost, provide early identification of requirement defects, and improve test coverage [KSSB01; BBN01; Saf00; Sta00; Sta01]. This paper describes some of the guidelines developed to help project managers better understand how measures derived from model-based analysis and testing can be used to help in project, product, and process measurement, where such measures support decision making within their organizations.

Card identified four components that establish the basis of a measurement framework [Car00]:

1. Technical or management decision-making process or individual
2. Information needs of the decision maker
3. Measurement planning and analysis process for collection and analysis of relevant data
4. Information product resulting from the analysis that satisfies the needs of decision makers

The purpose of a measurement program is to provide information that assists decision makers. The decisions that are made can be organized into four types or classes, often mapped to hierarchical levels within an organization, as shown in Figure 1. The information needs of the decision makers relate to their goals and the obstacles to achieving those goals. Information needs provide the basis for selecting the measures and analysis techniques incorporated into the information product, which is the final result of the measurement process. A measurement initiative may begin by addressing the information needs of one specific level of decision making, even though information needs of the different levels can be related.



Source: D. N. Card, *A Practical Framework for Software Measurement and Analysis*, Auerbach Systems Management Strategies, 2000

Figure 1. Decision-Making Process Relationships to Information Needs

1.1. Scope

This paper discusses ways to define, collect, and analyze measures derived from the TAF model-based testing method. These measures and their use are described in terms of an information model adapted from the ISO/IEC 15939, Software Engineering - Software Measurement Process. Although these measures can be used for product and process measurement, this paper focuses on project measures for project planning, scheduling, and performance. The paper describes the fundamental units of measure derived from the model-based artifacts. It describes how to interpret graphical representations of the base and derived measures, as well as how to use historical measures to predict project duration and usage of real-time project data to predict the completion of an ongoing project.

1.2. Organization of Paper

Section 2 provides information on the process that has been most effective for using TAF, and the briefing identifies the developed artifacts that are used in the measurement process. Section 3 defines fundamental units of measure derived from TAF artifacts. Section 4 describes how TAF measures support project decision-making information needs. Section 5 provides a summary.

2. Model-Based Test Automation with TAF

The most typical application of TAF is a process based on interface-driven requirements modeling [BBN03]. Although it is common to start the process with poorly defined requirements, inputs to the process can include requirement specifications, user documentation, interface control documents, API documents, previous designs, and old test scripts. With this approach, test engineers work in parallel with developers to stabilize component interfaces, refine requirements, and build models to support iterative testing and development. The following outlines the process, as depicted in Figure 2:

- 1) Working from whatever requirements artifacts are available, testers create a model using a tool based on the SCR method [HJL96], such as the SCRtool or T-VEC Tabular Modeler (TTM). Simplistically, each table represents an output, specifying the relationship between input values and resulting output values. Models are automatically checked for inconsistencies. The tester interacts with the requirements engineers or analysts to validate the model as a complete and correct interpretation of the requirements.
- 2) The tester maps the variables (inputs and outputs) of the model to the interfaces of the system in object mappings. The nature of these interfaces depends on the level of testing performed. At the

system level, the interfaces may include graphical user interface widgets, database APIs, or hardware interfaces. At the lowest level, they can include class interfaces or library APIs. The tester uses these object mappings with a test driver schema to support automated test driver generation. The tester works with the designers to ensure the validity of the mappings from model to implementation.

- 3) Prior to test generation, T-VEC compiles the model into DCPs that represent the requirement threads of the model. The T-VEC tool generates an optimal set of test vectors by analyzing the input space for each DCP. These test vectors include test inputs and expected test outputs, as well as model-to-test traceability for each DCP.
- 4) T-VEC generates the test drivers using the object mappings and schema. A schema is created once for each test environment. The schema defines the algorithmic pattern to carry out the execution of the test cases. The test driver executes in the target or host environment. The test drivers typically are designed as an automated test script that sets up the test inputs enumerated in each test vector, invokes the element under test, and captures the results.
- 5) Finally, T-VEC analyzes the test results. It compares the actual test results to the expected results and highlights any discrepancies in a summary report.

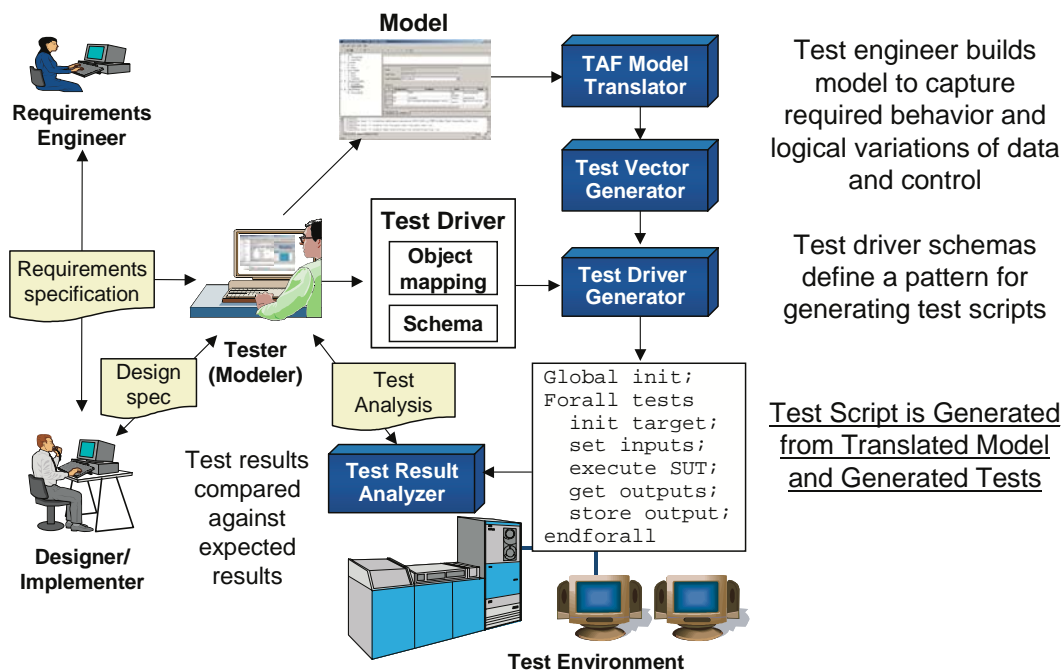
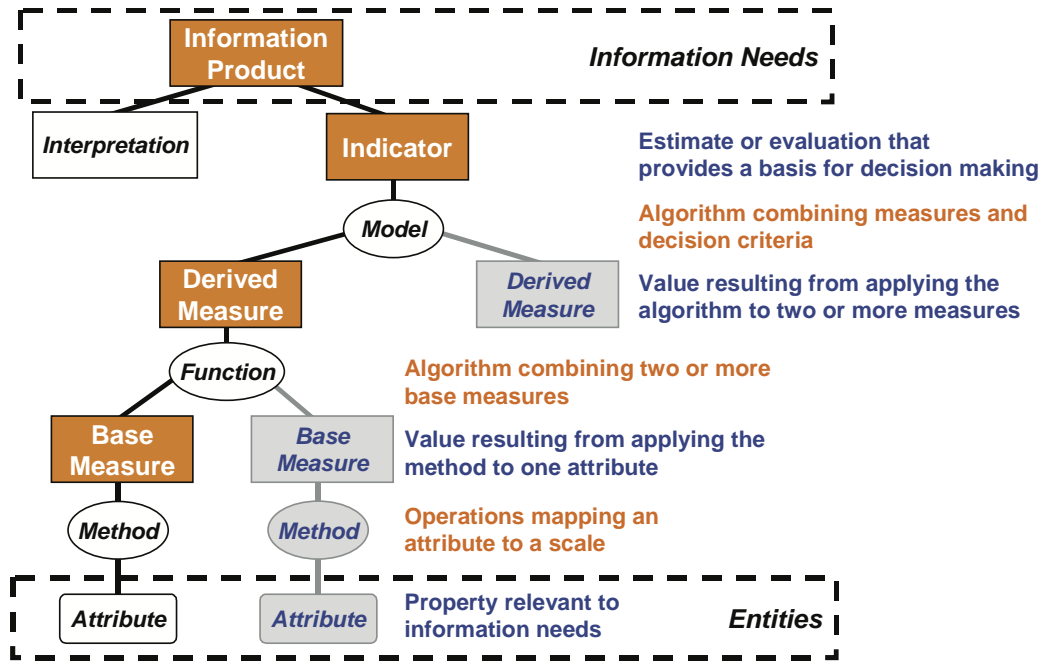


Figure 2. Model-Based Test Automation

3. Measurement Information Product and Measurement Construct

The measurement and analysis process provides the mechanisms for identifying and addressing information needs of all types and levels. It addresses both the selection of appropriate measures to satisfy the information needs and the collection and analysis of the data. Information products make up the primary output of the measurement process. The generic structure of a measurement construct can be defined in terms of the information model, as shown in Figure 3. The information model shows how the attributes of specific software and systems entities are related to the information needs of the measurement user. This section works bottom-up from the information model of the measurement

construct to describe the relationship of the TAF attributes, base measures, derived measures, and indicators to various information products supporting project measurement.



Source: Adapted from ISO/IEC 15939, *Software Measurement Process Framework* by Joe Seppy

Figure 3. Measurement Construct

3.1. Attributes

An attribute is a property or characteristic of a process or product that is the subject of measurement. There are four attributes to support measurement analysis: **DCPs** – requirement threads, **object mappings**, **requirements**, and **variables**.

3.2. Base Measures

A base measure is a quantification of a single attribute obtained from some method or operation. There are several possible base measures that can be obtained through operations associated with the TAF attributes. Some key base measures are defined in Table 1.

Table 1. TAF Base Measures

Base Measure	Explanation
Number of DCPs	DCPs produced during a particular measurement period
Number of variables	Input and output variables produced during a particular measurement period that require a corresponding object mapping
Number of object mappings	Object mappings specified during a particular measurement period
Total number of DCPs	Sum of the DCPs measured from the start of the project
Total number of variables	Sum of input and output variables measured from the start of the project
Total number of object mappings	Sum of specified object mappings measured from the start of the project
Total number of requirements	Sum of total requirements being modeled using TAF for the project

To better understand the progress being made during project development, it is necessary to measure the DCPs, object mappings, and variables produced each week (these measures could be tracked daily, monthly, or yearly). Although it is idealistic to think that the total number of requirements will remain constant for a project, this seldom happens, so the total number of requirements for a project also should be monitored weekly. Fortunately, the total number of DCPs is generated as part of the status report for a project. Similarly, you can record the number of variables and object mappings produced each week.

3.3. Derived Measures

A derived measure combines two or more base measures using a mathematical function. For example, the **requirements per DCP** is a derived measure relating the number of requirements to modeled-derived DCPs. If a project manager can estimate the total number of requirements that must be modeled, it is possible to predict the total number of DCPs for the project. If the rate of DCP production per week were known, the scheduled end-date could be predicted. This report assumes that the measurement period is weekly. Table 2 identifies some derived measures.

Table 2. TAF Derived Measures

Derived Measure	Explanation
DCP rate	The average number of DCPs produced per week
DCP rate (current week)	The average number of DCPs produced per week at some week, where current week is a week number starting from the beginning of the project
DCP density	The number of DCPs per requirement
Object mapping rate	The average number of object mappings specified per week
Object mapping rate (current week)	The average number of object mappings specified per week at some week
Variables per DCP	The number of input and output variables per DCP

These types of derived measures can be associated with an individual, project, project team, product, product family, business unit, or an entire corporation. Project teams as well as project team members will have different DCP rates and object mapping rates. These rates can vary based on the modeling skills of the team members, and can be affected by requirement rework due to poorly documented or unknown requirements, or poorly defined interfaces.

3.4. Indicators

An indicator is a base or derived measure or a combination of such measures that are associated with decision criteria by means of a mathematical or heuristic model. Indicators often are presented in graphic or chart form. Consider the following example that illustrates the use of the measurement construct. Assume that project decision makers want to compare the DCP density of some current project to previously captured data because it is believed that a DCP density in a certain range provides optimal requirement-to-test traceability. Assume the following:

- Base measures for DCPs and requirements have been collected.
- Number of requirement headers is the base measure for number of requirements, where a requirement header is associated with a body of requirement text.

- Data collected from prior projects estimates the number of DCPs per requirement.

Starting from the bottom of Figure 4, the two attributes are DCPs and requirements. Assume some sample project developed 223 DCPs in models for 21 requirements. Dividing the number of DCPs by the number of requirements produces a derived measure of 10.6 DCPs per requirement (i.e., DCP density). Compare this derived data to historical data, where the density was 16.3 DCPs per requirement. A possible interpretation of this information product is that the requirement traceability accuracy for the current project is better than the organizational average. If the current DCP density has a variance greater than 10 from the organization average, then it may be necessary for the requirement engineers to attend a training class on techniques for developing better requirements.

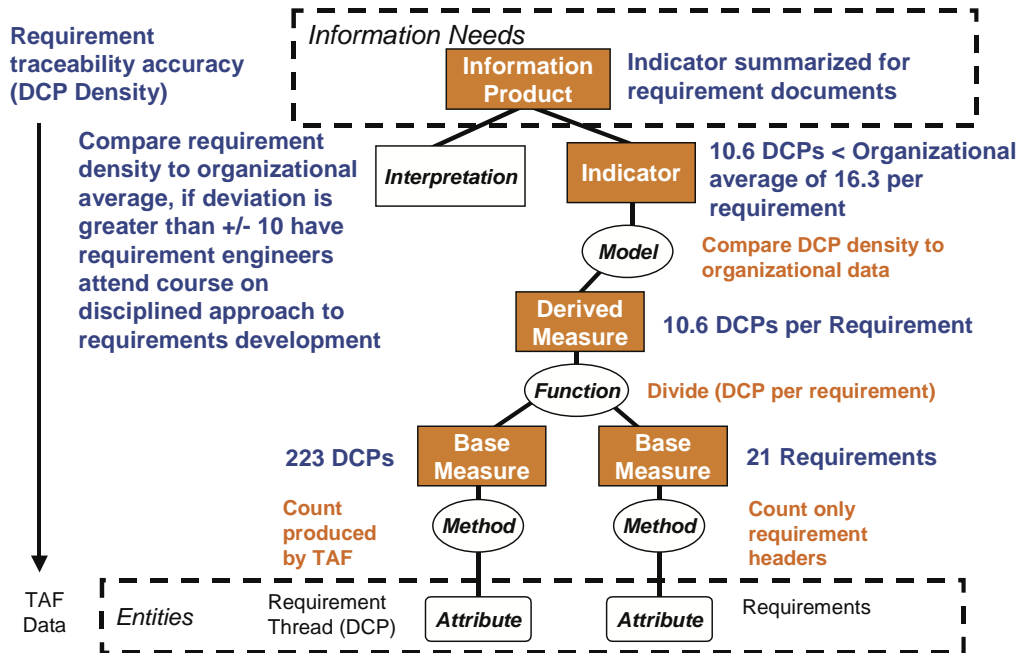


Figure 4. Measurement Construct Example

4. TAF Measurement for Project Information Needs

This section illustrates the use of TAF measurement to address project measurement information needs. The estimated project duration and estimated project completion date are usually key project information needs. This information can be affected by requirements creep and rework, but this can be factored into the measurement process. For purposes of this paper, the examples assume that the number of requirements for a project are known and remain constant throughout the project.

To estimate the project duration requires accounting for the time to model the requirements and specify the object mappings to all modeled variables. Once this is done, tests are automatically generated and can be executed automatically. All tests must pass. It is also desirable to have requirement and model reviews, but during a project, these typically happen continuously as the models are being developed. Therefore, this time is factored into the recorded data. In addition, the recommended process of interface-driven model-based testing also forces the object mapping development to proceed in parallel as discussed in Section 2. When all requirements are modeled, and all object mappings are created, the test execution can be performed. This requires a complete implementation also. Therefore, the actual end of these tasks follows very closely in time with the completion of the project. This is common for actual projects that have used TAF.

4.1. Fundamental TAF Measures

This section presents several graphs of base or derived measures that are used in the indicators described at the end of this section. It provides some example interpretations of the information presented. The line graphs plot the project week along the x-axis. The y-axis varies depending on the interpretation of the data or indicator. The graph shown in Figure 5 plots the weekly number of DCPs. This type of chart accentuates the variances in progress for each week. For example, the dips in progress (or DCPs) for weeks 3, 6, and 11 should trigger a project lead's interest. This information should be analyzed to better understand why little progress was being made during these weeks.

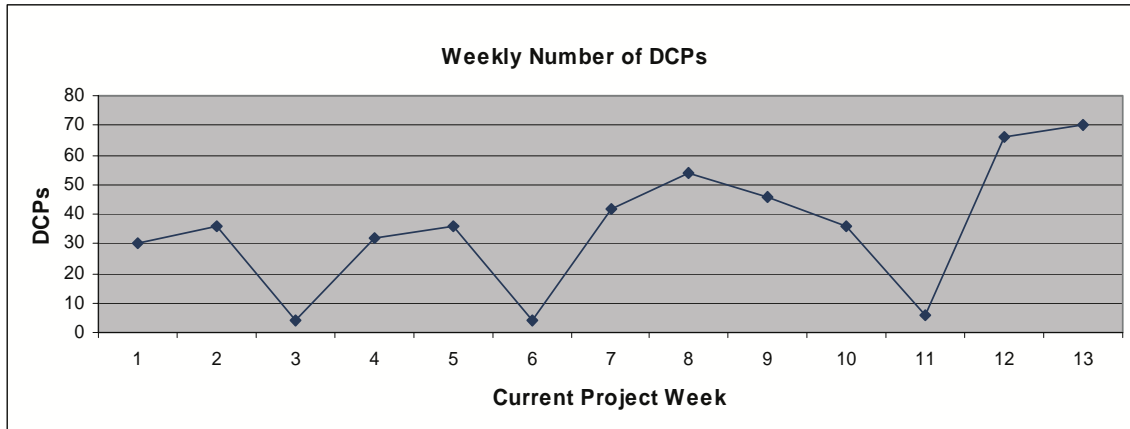


Figure 5. Weekly Number of DCPs

The project DCP rate is the total number of DCPs divided by the current project week. Figure 6 plots the average DCP rate against the actual DCP count per week. Even with 3 weeks of low productivity, an interpretation of this graph is that the number of DCPs per week is increasing. This is a reasonable expectation as people on the project become more familiar with the project requirements, and the modeling skills improve. At the end of the project, the average DCP rate establishes the baseline for a product DCP rate, which is the average DCP rate for producing one or more releases of a product.

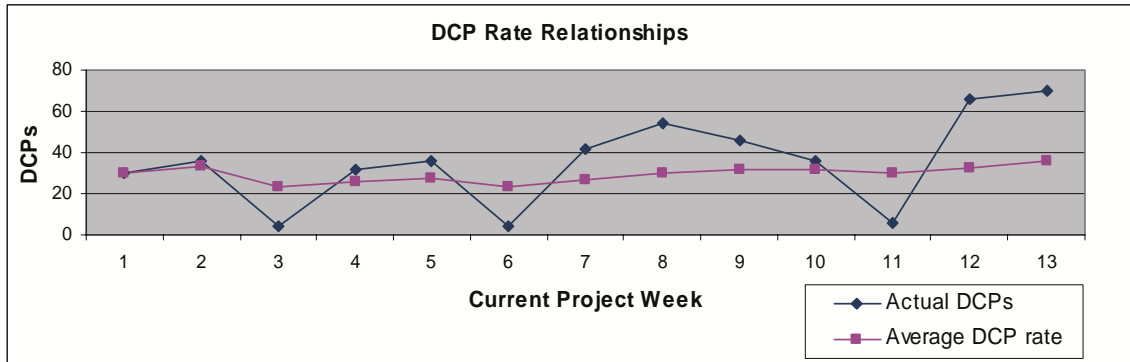


Figure 6. DCP Rate Relationships

4.2. Estimated Completion of Requirement Modeling Indicator

The following example is simplified to focus on the completion of the requirements modeling without adding analysis for the object mapping data. If historical data is available, then the estimated completion of requirement modeling can be predicated using Formula 1. For example, assume the

total number of requirements is 50, and the DCP density is 16. This means there will be an estimated 800 DCPs. If the DCP rate is 40 DCPs per week, then the estimated project will require approximately 20 weeks to complete. The accuracy of this prediction is only as good as the historical information.

Formula 1	
Estimated Weeks to Complete Requirements	$= \left[\frac{(\text{Total Requirements} * \text{DCP Density})}{\text{Average DCP rate}} \right]$

Organizations attempting to predict the estimated completion date of the first TAF project will not have existing information; however TAF measures can be used if an actual or predicted number of requirements are known. If the DCP density and DCP rates are not known ahead of time, they can be derived as the project progresses. By using a real-time calculation of the DCP rate and DCP density during the project, a prediction can be made about the estimated completion date for modeling the requirements. A variant of Formula 1 is used to continuously predict the remaining number of weeks to complete the requirement modeling. The calculation is shown in Formula 2.

Formula 2	
Real-time Estimated Weeks to Complete Requirements	$= \left[\frac{(\text{Total Requirements} * \text{DCP Density}[\text{Current week}])}{\text{Average DCP rate}[\text{Current week}]} \right] - \text{Current week}$

The hybrid graph in Figure 7 shows three different information components. The x-axis represents the current week, and the y-axis shows information for the following:

- Average DCP rate (circle data points)
- DCP density (square data points)
- Estimated weeks remaining before project completion (triangle data points)

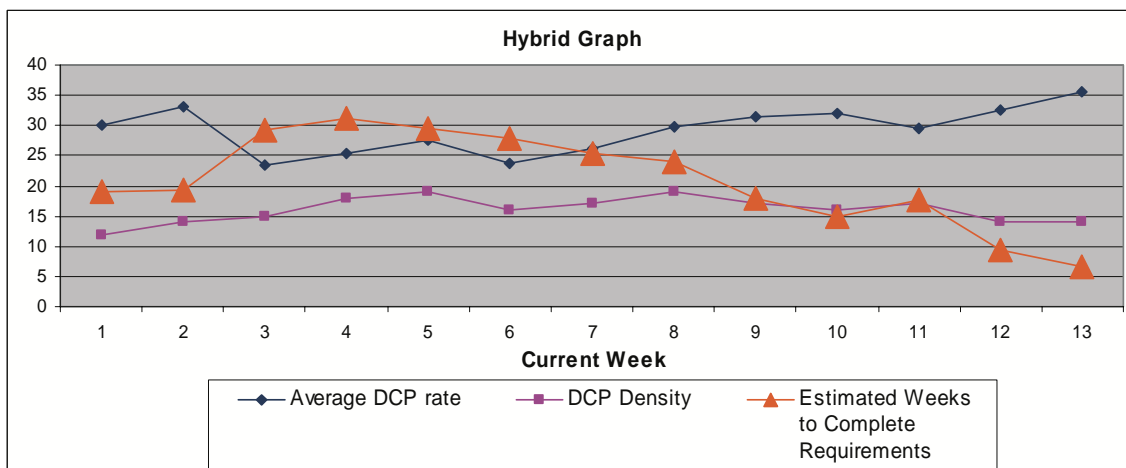


Figure 7. Hybrid Graph Estimating Weeks Remaining in Project

4.3. Indicators for Object Mappings

From a development and measurement point-of-view, object mappings have two states: introduced and specified. Once a model variable is introduced into the model, an object mapping is introduced,

but it is not specified; therefore object mappings' data can be tracked according to these states. The number of variables is known in advance of the object mapping specification. This is reflected by Figure 8, which shows the actual number of variables, actual object mappings (specified object mappings), and the average rate of completing the object mappings.

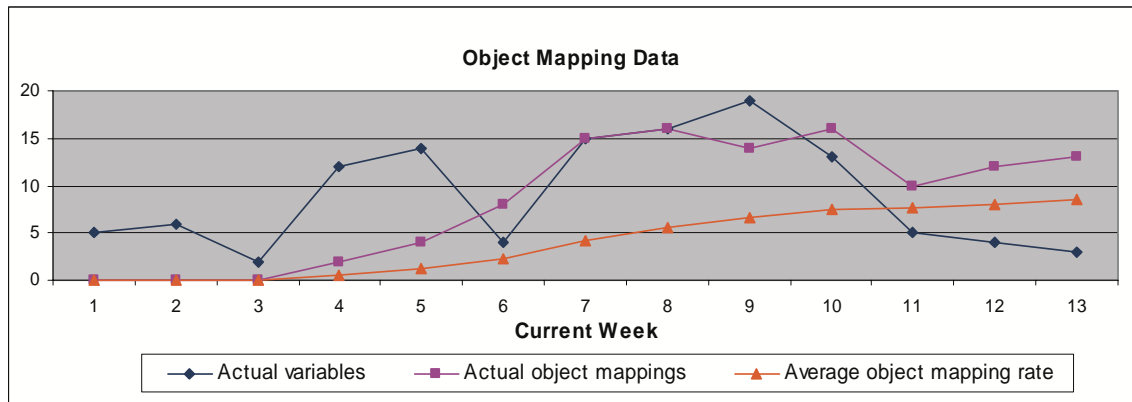


Figure 8. Object Mapping Data

4.4. Estimating the Number of Object Mappings

When a project begins, there may be an estimate of the total number of requirements. With this estimate, it is possible to predict the estimated completion date of the requirement modeling when there is a DCP density and a DCP rate. For object mappings, it is usually more difficult because the number of variables per requirement can be application-specific. However, once modeling begins, the number of variables per DCP can be calculated as reflected in Figure 9. This sample data indicates that there is approximately one variable for every 4 DCPs at week 13. This means that the total number of variables that will be introduced in the model can be estimated.

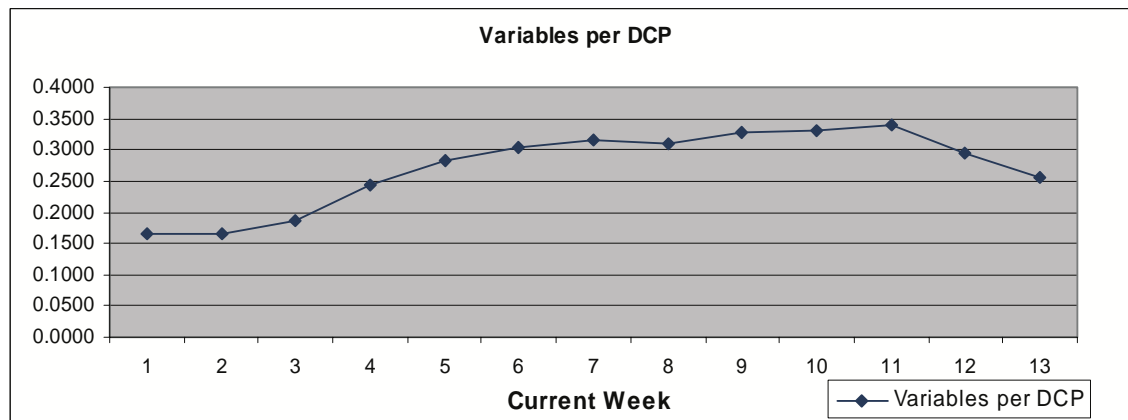


Figure 9. Variables per DCP

4.5. Combined Estimate-to-Complete

There are two possible ways to look at the estimate-to-complete:

- Using historical data to predict the project duration
- Using project data to predict the number of weeks until completion

4.5.1 Estimating Project Duration

One goal of project planning is to predict the duration of the project given some set of resources that have predictable modeling and object mapping development rates. Given a predictable number of requirements and historically derived measures for DCP density, object mapping rate, and variables per DCP, the duration of a project can be calculated (Formula 3).

Formula 3	
Estimated Project Duration	$= \frac{(\text{Total Requirements} * \text{DCP Density} * \text{Variables per DCP})}{\text{Object mapping rate}}$

4.5.2 Estimated Weeks to Project Completion

The estimated weeks to project completion uses information gathered during the execution of the project to predict the number of remaining weeks before the modeling and object mappings will be completed. This measure is useful when there is no historical information, but it is also useful as a way to track project progress. There also are some other benefits of this measure:

- DCP density is more accurate because it reflects the project-specific density.
- DCP and object mapping rates are more accurate because they are directly related to the current staff performing the work.
- Variables per DCP measure is more accurate because the number of variables per DCP can vary significantly from one application to another.

The formula to continuously predict the remaining number of weeks to complete the object mappings (see Formula 4) is similar to the formula for calculating the real-time estimate for completing requirement modeling (see Formula 2).

Formula 4	
Real-time Estimated Weeks to Complete Object Mappings	$= \left[\frac{(\text{Total Requirements} * \text{DCP Density}[\text{Current week}] * \text{Variables per DCP}[\text{Current week}])}{\text{Average object mapping rate}[\text{Current week}]} \right] - \text{Current week}$

The accuracy of the formula is typically poor early in the project when no object mappings have been specified because the object mapping rate is zero; and this causes a divide-by-zero computation. Even if the number of object mappings is very small, the computation is still grossly inaccurate because the denominator is very small compared to the numerator. Figure 10 shows two data lines: estimated weeks to complete requirements (diamonds – Formula 2), and estimated weeks to complete object mappings (squares – Formula 4). The project is typically not complete until the object mappings are complete, but during the first part of the project, there are usually no object mappings specified; therefore, the best early estimate is provided by the estimated weeks to complete requirements, which is described in Section 4.2. Early during the project, Formula 2 is a better predictor, but at some point during the project, when there are a substantial number of object mappings completed, the estimate-to-complete based on the object mapping rate (Formula 4) becomes a better predictor than the estimated weeks to complete requirements.

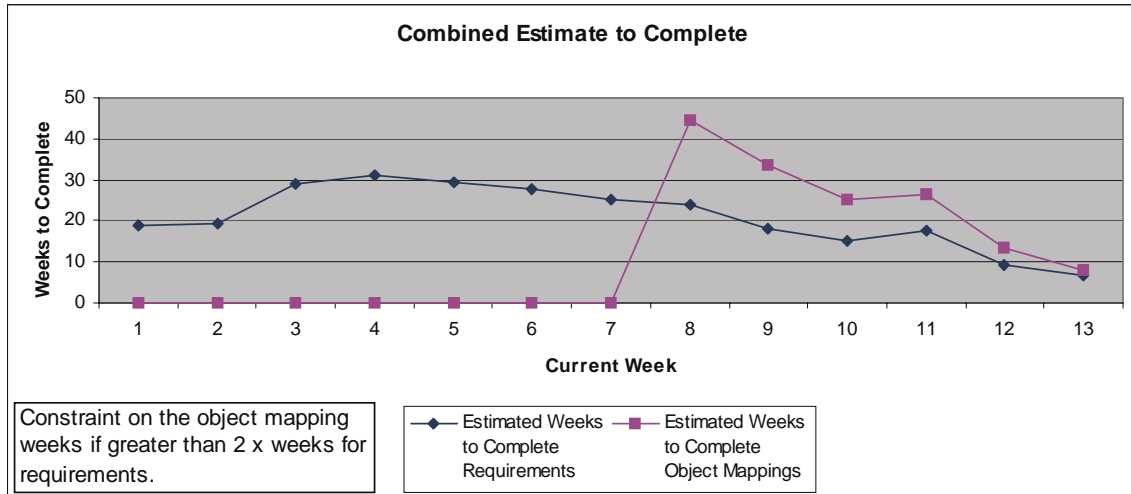


Figure 10. Combined Estimate-to-Complete

5. Summary

This document provides introductory guidance for project measurement based on using measures derived through the TAF model-based testing method. Figure 1 provides a perspective on the key measurement information and how it relates to the typical TAF method of interface-driven requirements modeling.

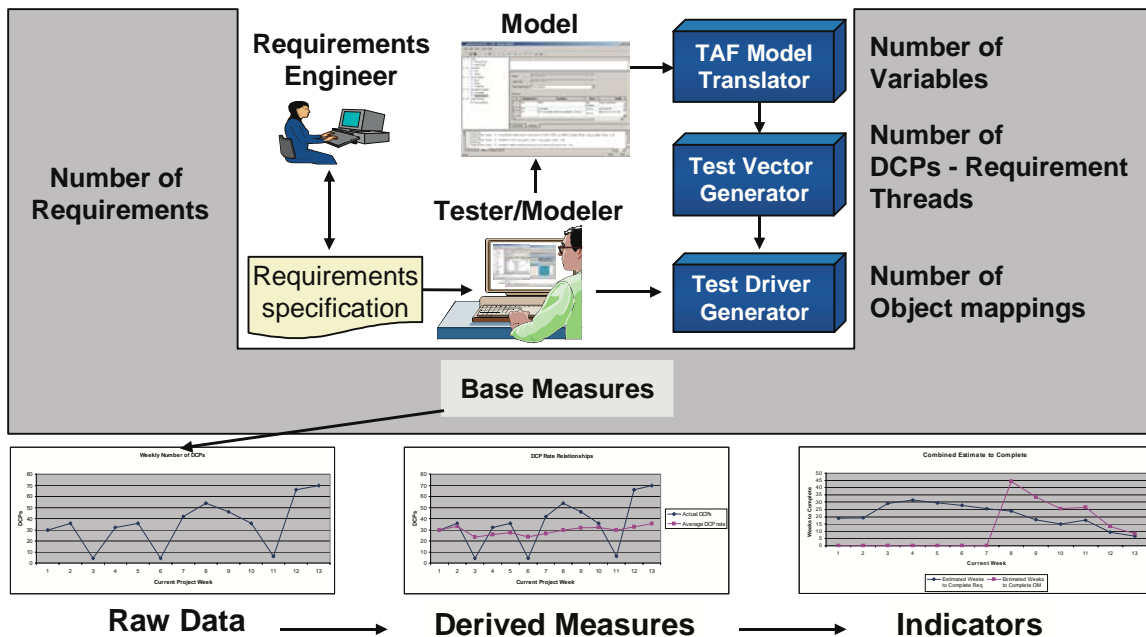


Figure 11. Process View of TAF Measurement

The TAF measurement is based on four key base measures. Requirement engineers produce requirements, which results in the base measure number of requirements. A tester/modeler works in parallel with developers to refine requirements and build models to support iterative testing and development. Modeling introduces model variables, and this results in the base measure number of variables. After model translation and processing, the model requirements are converted into DCPs,

which is a base measure related to requirements. Finally, to support test driver generation, test execution and results analysis, the base measure number of object mappings is used. Object mappings relate model variables to the implementation interfaces.

Section 4 discussed how these various base measures can be plotted as graphs to provide raw data information related to project progress in producing the TAF artifacts. Section 4 also discussed ways to transform these base measures into derived measures. Finally, these base and derived measures are used to produce indicators that provide key information related to project duration and the estimated project completion.

This measurement-related information should help managers and project leads with predicting schedule duration and estimating project completion dates. Historical measurement information can be used prior to the start of a project, but it also is important to use data derived during the project. This paper provided a suggested set of information needs for project managers and presents guidance for developing related indicators and their supportive measures.

6. References

- [BBN03] Blackburn, M. R., R. D. Busser, A. M. Nauman, Interface-Driven Model-based Test Automation, CrossTalk, May 2003, <http://www.stsc.hill.af.mil/crosstalk/2003/05/blackburn.pdf>.
- [BBN01] Busser, R. D., M. R. Blackburn, A. M. Nauman, Automated Model Analysis and Test Generation for Flight Guidance Mode Logic, Digital Avionics System Conference, 2001.
- [Car00] Card, D. N., A Practical Framework for Software Measurement and Analysis, Auerbach Systems Management Strategies, 2000.
- [HJL96] Heitmeyer, C., R. Jeffords, B. Labaw, Automated Consistency Checking of Requirements Specifications. *ACM TOSEM*, 5(3):231-261, 1996.
- [KSSB01] Kelly, V. E.L.Safford, M. Siok, M. Blackburn, Requirements Testability and Test Automation, Lockheed Martin Joint Symposium, June 2001.
- [Rob00] Robinson, H., <http://www.model-based-testing.org/>.
- [Sta00] Statezni, David. Test Automation Framework, State-based and Signal Flow Examples, *Twelfth Annual Software Technology Conference*, April 30 - May 5, 2000.
- [Sta01] Statezni, David. T-VEC's Test Vector Generation System, *Software Testing & Quality Engineering*, May/June 2001.
- [Saf00] Safford, Ed L. Test Automation Framework, State-based and Signal Flow Examples, *Twelfth Annual Software Technology Conference*, April 30 - May 5, 2000.