Measurement Information Specification
Depth of Test
Version 2.1

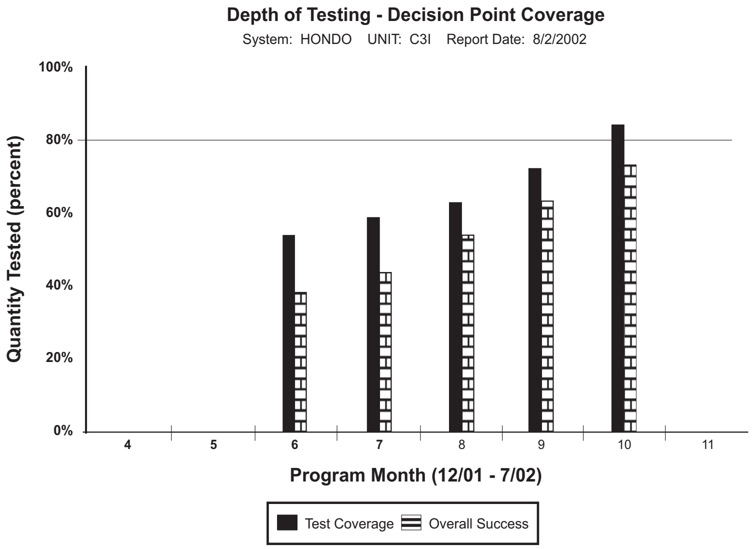
Information Need Description	
Information Need	Is the software design too complex? Is the software design physically correct?
Information Category	Schedule and Progress

Measurable Concept	
Measurable Concept	Work Unit Progress

Entities and Attributes	
Relevant Entities	Software design architecture
Attributes	• Paths, • Statements, • Inputs, and/or • Decision points

Base Measure Specification	
Base Measures	1. Attribute occurrences 2. Attribute occurrences tested 3. Attribute occurrences successfully tested
Measurement Methods	1. Count the number of the attribute occurrences in the software component being tested (path, statement, input, or decision point). 2. Count the number of the attribute occurrences that are exercised in test at least once. 3. Count the number of the attribute occurrences that are successfully exercised in test at least once.
Type of Method	Objective
Scale	Integers from zero to infinity
Type of Scale	Ratio
Unit of Measurement	Attribute occurrences

Derived Measure Specification	
Derived Measure	1. Test coverage 2. Test success
Measurement Function	1. Number of attribute occurrences tested divided by attribute occurrences times 100 (percentage) 2. Number of attribute occurrences successfully tested divided by attribute occurrences times 100 (percentage)

Indicator Specification	
Indicator Description and Sample	Depth of Testing: This indicator displays the derived measure of the percent of attribute occurrences tested (test coverage) in the software component under test, and the derived measure of the percent of attribute occurrences successfully tested (test success).  Sample Depth of Testing Indicator
Analysis Model	• Test coverage reported the percent of total software attributes that were tested. The percent of coverage that had to be achieved depended on the attribute that was measured. All decision points and statements were tested prior to delivery, and the “highest possible” percent of all possible paths and inputs were tested prior to delivery. • Test success reported the percent of total software attributes that were tested and verified to be correct. The criterion for test success was that all attributes that failed the test must be fixed and tested as correct prior to delivery.
Decision Criteria	All decision points and statements had to be successfully tested at least once (100%), and the “highest possible” percent of all possible paths and inputs had to be successfully tested.
Indicator Interpretation	This indicator told the project manager that the percent of decision points that was tested and the percent of decision points that was proven correct by successful test had been steadily increasing during the planned 14-month integration-test period. The high percent of test coverage (percent tested) and test success (percent tested correct) at month 10 and the increasing trends of these derived measures indicated that there was little risk in achieving software integration on the original schedule.

Data Collection Procedure (for each Base Measure) ***Complete this section for each base measure listed on the previous page.***	
Frequency of Data Collection	Daily beginning at unit test (through development test until the time that a configuration-controlled code baseline had completed testing). Monthly during post deployment software support (PDSS).
Responsible Individual	• The test team collected the test data on a daily basis. • The test team reported the test data to a CM representative each week for aggregation in the CM repository test records.
Phase or Activity in which Collected	During unit test, integration test, developmental test, and post deployment software support
Tools Used in Data Collection	McCabe's tools were used to provide a physical mapping of software code structure and to identify which attribute occurrences were exercised in testing.
Verification and Validation	During unit test, the individual programmers performed "white-box" testing to determine the correctness of the code in the software units they had designed. Correctness was based on the test-process criteria that had been established. The CM representative reviewed the output of the "white-box" test tool and certified that the test met the organization's unit test criteria to enter the unit in the approved software baseline. During integration or other developmental test, the test team used the "white-box" test tool to determine the correctness of each attribute in the integrated software baseline. Each week, the CM representative reviewed the output of the "white-box" test reports to ensure the integration test met the organization's test process criteria.
Repository for Collected Data	• Test records contained data collected daily • CM repository contained weekly aggregated records

Data Analysis Procedure (for each Indicator)	
Frequency of Data Reporting	• Reporting was monthly during most of the testing and weekly at the end. • This measure also was reported monthly as a record of regression test on software changes during development test and PDSS.
Responsible Individual	CM representative
Phase or Activity in which Analyzed	Beginning at unit test, continuing throughout developmental and integration test, and as required during post-deployment software support
Source of Data for Analysis	Test records in the CM repository
Tools Used in Analysis	PSM Insight
Review, Report, or User	• CM used "white-box" test results to certify all software units for entry into the approved software baseline. • Monthly progress reviews reported "white-box" test results as an indicator of software complexity and design progress. • Design reviews used "white-box" test results to verify completion of integration test.

Additional Information	
Additional Analysis Guidance	The depth-of-test indicators do not assess the “correctness” of the design or code. It is expected that unit tests and unit integration and test will use test cases that demonstrate proper code design. These cases should be supplemented by other cases to yield coverage and success measures that provide satisfactory confidence that unexpected control or data conditions will not occur. Software test programs usually require that software structure be successfully demonstrated only after passing some “realistic” number of test cases, under both representative and maximum stress loads. It is understood that exhaustive testing of all control and data combinations is prohibitive. Because illegal inputs are used, the depth of test measure provides an indication of the robustness to the software design. Some judgment is required to interpret the input measure, because it is unlikely that the program will be subjected to all possible input streams.
Implementation Considerations	The recommended minimum elements to track for this measure are paths, statements, and inputs. The recommended data definitions for this measure are collected for each element; however, data may also be collected at the component or system level if adequate test tools are available. Depth of testing data collection should be tailored to consider the data collection effort. The following guidelines are suggested: • Always compute the coverage of the input domain that has been achieved. • Always compute the path and statement measures over the set of basis paths, on units that implement high-priority requirements, or if a unit’s complexity values exceed established thresholds. • Compute more comprehensive path and statement measures if automated tools are available.