Evaluation of the Functional Testing of Control Programs

by

W. R. Elmendorf

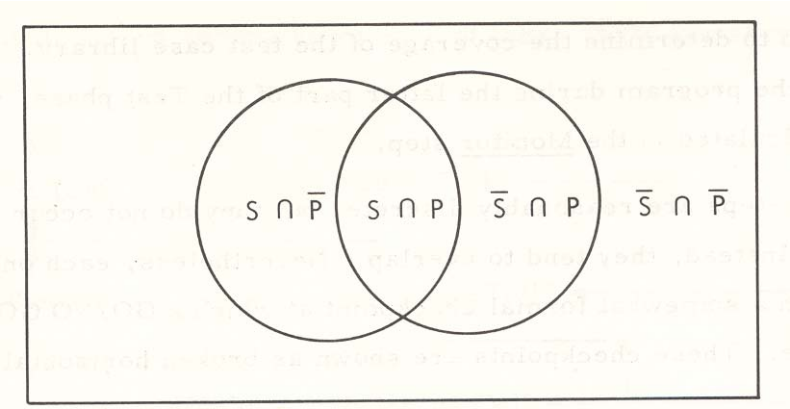
International Business Machines Corporation
Systems Development Division, Poughkeepsie, New York

Editor's Note:

In this paper Bill addresses eight issues which at the time were revolutionary for software testing:

1. "… a more scientific approach [to testing] is needed – one which overcomes the deficiencies of the 'testing-is-an-art' approach."

2. We need to test the software from a requirements perspective and from an internal perspective – i.e., you cannot test 100% of a system from the requirements given the nature of technology. His classic Venn diagram first appears in this paper.



Specification Testing Distinguished From Program Testing.

S stands for specified functions
P stands for programmed functions

Ideally the specified functions match the programmed functions. This is verified by specification and program testing. You look for specified functions not matched by programmed functions via specification based testing and programmed functions not matched by specified functions via program testing. You must also look for functions neither specified nor programmed but needed.

3. Test coverage should be weighted by such factors as "vulnerability to programming error, vulnerability to system failure, and market impact." – i.e., he is describing Risk Based Testing.

4. Functional variations should be defined by identifying the variables and the states to test for each variable. The states to test should include acceptable values, values at the extreme upper and lower acceptable values, and values beyond the extreme upper and lower acceptable values – i.e., equivalence class testing with boundary analysis.

5. Functional variations are then mapped into test cases resulting in a coverage matrix. This is used to track the test status.

6. Test Plans should be reviewed to ensure they are adequate – i.e., test the Test Plan.

7. Test progress should be tracked by the number of successful variations executed, not just the percentage of tests run successfully.

8. Regression testing must be blended with new function testing.

Today we take this all for granted. How else would you test? However, in 1967 this was a major advance on the state of the art.

Richard Bender
rbender@BenderRBT.com
October 2006

ABSTRACT

Functional testing of control programs is undergoing a transition from a predominantly imprecise art to an increasingly precise science. The evaluation of the test effort is maturing correspondingly. Systematic, comprehensive and precise planning of test objectives now precedes test case development. Furthermore, objective and quantitative measurement of test coverage and exposure now occurs at multiple points in the testing cycle. These measures increase the efficiency of the test effort and the reliability of the tested product.

There are four steps in the evaluation process:

(1) The survey of the product establishes the general scope of testing. The output is a "coverage matrix" that establishes the extent to which each part of the product should be tested at each level of specification.

(2) The identification of the variations in the specifications which fall within the scope established in the survey pinpoints the test objectives. A list of eligible 'variations is generated and then subset to include only those variations to be tested.

(3) The appraisal of the test plan determines its coverage of the objectives identified in the identification step.

(4) The review of testing progress verifies the implementation of the test plan.

INTRODUCTION

For the product manager of a control program development effort to objectively evaluate the effectiveness of the functional testing and the resultant reliability of the tested product, four kinds of test information are needed:

1. A precise description of the scope of testing.

2. Quantitative measurements of test coverage and exposure within this scope.

3. Evidence of the efficient use of testing resources.

4. Informative documentation of the test cases.

This paper describes guidelines designed to yield this information.

Note that the subject is functional testing. This means the logical testing of a product; it includes unit, component, system, regression, compatibility and configuration testing. Other kinds of testing, such as performance testing and application-oriented testing, are not considered.

The subject is treated from a strictly technical viewpoint. Administrative considerations such as funding, scheduling and subcontracting are not discussed. Neither are other aspects such as test case construction, test case selection from existing libraries, test case execution or standards compliance.

The technical and administrative contexts in which different pieces of a control program are developed are themselves different. Accordingly, the guidelines are not restrictions, but rather suggestions. It is expected that the guidelines will be "customized" to fit various situations.

PHILOSOPHY

A software product could be tested by assembling a group of knowledgeable programmers, defining their objectives in broad terms, and then letting them ramble toward these objectives as their individual backgrounds, interests and biases dictate. Underlying this laissez-faire approach is the view that testing is an art. There is no preliminary mapping of the territory to be explored and no precise assignment of territorial responsibilities. Consequently, there is no way of precluding redundant exploration, no means of limiting exploration to those territories with the higher probable payoffs and no definitive measure of the territory explored. In short, there is little control over the testing process and few objective measures of test coverage.

In certain areas it is appropriate that testing continue to be an art. However, in the hard-core areas of program testing a more scientific approach is needed -- one which overcomes the deficiencies of the “testing-is-an-art” approach. The guidelines described in this report emphasize both systematic, comprehensive and precise preplanning of the testing process and quantitative measurement of the plan's implementation.

The product manager's involvement in the testing process can be divided into four steps:

1. Survey of the product to establish the desired breadth and depth of testing.

2. Identification of all variations in the external and internal specifications which fall within the scope established in the survey.

3. Appraisal of the test plan to determine its coverage of those variations identified in the identification step.

4. Review of the testing progress to verify the implementation of the test plan.

GUIDELINES

SURVEY

The initial step - - the survey - - establishes the scope of the test effort; this includes both the parts of the product to be tested -- the breadth -- and the level of specification to which each part is to be tested - - the depth. The product is first analyzed to determine the most natural functional delineation of its parts. This analysis establishes the potential breadth of testing.

Each part can be tested at one or more levels: Verification that the program behaves as stated in the external specifications. This is external interaction testing. Verification that the program behaves as stated in the internal specifications. This is internal interaction testing. Verification that the specifications fully describe the program's behavior. This is interference testing. These three levels constitute the potential depth of testing.

Interaction testing verifies that the specifications have been correctly programmed, whereas interference testing verifies that the program has been correctly specified. See Figure 1. The primary concern in interference testing is to find destructive program actions which must be either corrected or protected against through restrictive specifications. Of secondary interest are additional constructive program actions that can be added to the specifications. Interaction testing, determined wholly from the stated specifications, therefore is bounded. Interference testing, on the other hand, is a search for unstated specifications and is unbounded.

Those combinations and permutations of specifications which are not explicit specifications in their own right but which are implicitly permitted are the subject of

interference testing, not of interaction testing. A simple test can be used to judge whether a particular combination or permutation is an interaction or interference variation. Does the specification express a logical relationship of the form: "if... and..., then..."? If the answer is "yes", then the variation is an interaction type; if "no", then it is an interference type.

The output of the product survey is a coverage matrix whose dimensions are the breadth and depth of testing. This matrix indicates the extent of testing of each part at each level of specification. Where external interaction testing is involved, the extent of testing is expressed as a percentage of the total number of specification variations. Where the subject is internal interaction or interference testing, this measure is not feasible since the total number of variations cannot reasonably be determined. Some alternate measures of coverage are: first, a confidence level expressed as a percentage of variations tested without functional failure; second, a relative extent based on the extent of external interaction testing and; third, an absolute extent such as the number of man-months to be invested in the testing.

Weighting criteria used in creating the coverage matrix include:

Vulnerability to programming error

* Is the design conventional or novel?

* Is the logic simple or complex?

* Are the programmers experienced or inexperienced?

* Is the development process progressing smoothly or irregularly?

Vulnerability to system failure

* Is the part widely used by other parts of the operating system or by problem programs?

* Is the part in a critical path of the operating system?

Market impact

* How much revenue is dependent on the part?

* What is the anticipated life of the part?

The coverage objectives are also based on two assumptions: first, that external interaction testing will generally have a higher yield than internal interaction testing and; second, that interaction testing as a whole will generally have a higher yield than interference

testing. (Here, yield is expressed in terms of programming and/or specification errors detected per dollar invested in testing.)

Note also that interaction testing, particularly at the external level, supports an IBM commitment to the customer. It should, therefore, have priority over interference testing. This does not imply that interference testing is of no value; undetected interferences can impact the use of a control program just as destructively as can faulty interactions.

IDENTIFICATION

Identification is the precise identification of every specification variation that should be verified. This involves two steps: the identification of all eligible variations and; the selection of a subset of these variations for testing.

There are instances where a list of eligible variations already exists. Here the analyst can proceed directly with the subsetting. Such is the case where the product being tested is only partially new, requiring regression testing of the unchanged parts to make sure they have not been impacted by the new ones. A list of eligible variations will generally also exist for a product which is itself unchanged but which must be proven compatible with another product.

External Interaction Variations

Since exhaustive testing is a possibility in external interaction testing, identification of all variations in the external specifications is crucial. To achieve this, the following categories of eligible variations can be used, singly and in combination:

* Variations in the notation used to express individual fields of macros, statements and commands.

* Variations in the interaction of fields within a given macro, statement or command.

* Variations in the interaction of different macros, statements and commands.

* Variations in the attributes of modules, devices, data sets, volumes and output classes referenceable by macros, statements and commands.

* Variations in the attributes of tasks, steps and jobs referenceable by macros, statements and commands.

* Variations in the interaction of tasks, steps and jobs, both sequential and concurrent.

* Variations in the interaction of devices, data sets, volumes and output classes.

* Variations in the software and hardware configurations.

* Variations in the interaction between software and hardware.

* Variations in the interaction between software and the operator.

Included under notational variations are syntactical and macro-expansion variations. If the actions taken by the control program are predictable, acceptable ranges of field values, field sizes, field repetitions, etc., will be identified as variations at, and just beyond, the extremes of each range. There are external limits which, if exceeded, cause unpredictable actions. If the specifications are enlarged to cover these cases, then external interaction variations can be identified for them; otherwise, they are considered interference variations.

The preceding categories of variations can occur in the following three contexts which should be considered singly and in combination:

* Control Flow. This deals with the creation, deletion, starting, stopping, skipping and routing of tasks, steps and jobs.

* Information Flow. This refers to the passing of parameters, subpools, return codes, data sets, etc., between tasks, steps and jobs. Also included is the passing of messages, replies, completion codes and storage dumps between the control program and the environment.

* Resource Contention. This involves the scheduling and allocation of resources; this includes information resources such as program modules and data sets, physical resources such as core storage, direct access storage, I/O devices and the CPU, and abstract resources specified by ENQ and DEQ macros.

In short, every external specification of the control program, whether mandatory, optional or restrictive is identified as a specification variation.

Internal Interaction Variations

Because of the volatility of internal specifications and the differing interpretations of what constitutes an internal specification, internal analysis is not expected to be as complete or as up-to-date as the one based on external specifications, Nevertheless, a reasonable effort toward the exhaustive identification of variations is desirable. A subset of these variations is then included in the test plan.

To aid in identifying variations in the internal specifications a list of categories, similar to the one given for external specification analysis, can be used. Its key elements are control blocks, queue elements, table entries, system routines and user routines. In the interest of brevity, the list of categories is not included here.

Interference Variations

In interference testing we are concerned with verifying the absence of unspecified control program action. Here, exhaustive testing is out of the question for it would involve an unmanageable number of combinations and permutations of the specifications. Therefore, we must rely on skilled practitioners of the testing art to identify the eligible variations. A subset of these is then selected for testing.

Variation Subsetting

The objective of subsetting is to select from the lists of eligible variations generated in the study of external, internal and interference variations, those variations which should be tested. The size of each subset is governed by the coverage percentage decided on in the survey step. The following criteria enter into the selection of a subset:

* The relative yield of the eligible variations.

* The relative significance of the eligible variations.

* The availability of the necessary test tools.

* The availability of the appropriate system configurations.

The relative significance and yield of contending variations can be judged using both the weighting criteria given in the survey step and the following:

* Does the variation represent usage which is consistent with the philosophy of the system?

* If the variation were to fail, would the cause of failure be hard to localize and correct?

* Would failure of the variation have a major impact on continued system operation?

* Would failure of the variation be difficult to neutralize in ways other than by correction (e. g., by masking the error or by introducing restrictions in the specifications)?

* Would the cost of testing the variation be minimal relevant to the exposure?

Affirmative answers to these questions will tend to increase the significance or yield subject variations.

APPRAISAL

One purpose of the identification step is to create an objective yardstick for measuring the testing effort. In the appraisal step, the test plan is the object of measurement. This plan should list, in detail, the functional variations it encompasses. This list of variations is compared with those of the yardstick: the variations appearing on both lists represent coverage; those on the yardstick but not in the test plan represent exposure; those in the test plan but not on the yardstick may represent extraneous testing.

Both the exposure and extraneous testing may or may not be justified. New information on the yield and significance of eligible variations can lead to revised subsetting. The subsetting criteria given under the discussion of variation subsetting can also be used here to judge the wisdom of the revisions.

Exposure can be measured objectively for external interaction testing since all of the variations can be identified. Measurement of exposure for internal interaction and interference testing, however, is more difficult and requires the application of subjective criteria.

REVIEW

In the review step, the actual test progress is measured against the test plan agreed upon in the appraisal step. There are two required inputs: a map relating test variations to test cases and; periodic information on the progress of test case writing, debugging and formal testing. From the map we can deduce two things about the test cases: their coverage and exposure and; their efficiency.

The exposure of the test cases represents variations included in the test plan but not in the actual cases. Here again, we apply the criteria used in the appraisal step to judge the yield and significance of the omissions.

The efficiency of the test cases is a reflection of the redundant or extraneous testing revealed by the map. To a certain extent these characteristics are unavoidable. First, there are practical reasons for starting with small, simple test cases and then building up to large, complex ones. The result is a hierarchy of test cases in which the simple cases are run initially and, when successful, are followed by more complex ones. In this situation, redundancy is more imaginary than real. Second, it is necessary to use simple variations to create the desired environment for testing complex variations. If these simple variations are not in the assigned test domain, then a misleading indication of extraneous testing will result.

In the identification step, both new and unchanged parts of the system are analyzed. The analysis of new specifications yields new variations to be tested. This generally means writing new test cases. On the other hand, the analysis of unchanged specifications yields unchanged variations which are tested using existing test cases from a regression library.

This establishes a need for selecting an optimal subset of test cases from the library. The map of test variations versus test cases can assist in this task.

It is not advisable to create the map ex post facto; memories fade and programmers leave. Instead, the mapping process should be a continuing responsibility of the test case programmers and should become an integral part of their documentation. This also permits reporting of test progress in terms of variations rather than test cases. The variations express functional coverage better than do the test cases.

Using the status information, we can plot curves showing the testing progress to date; we may also be able to extrapolate these curves to predict testing progress in the future. The review is a continuous responsibility throughout the entire process of test plan implementation from appraisal to release of the product to the field.

SUMMARY

The set of guidelines described in this paper are a means of evaluating the functional testing of control programs. They define and contribute to the test information needs of the product manager. In summary, the guidelines:

* Describe criteria for establishing both the breadth and depth of a test effort.

* Aid in the exhaustive identification of specification variations. This provides an objective basis for quantitatively measuring test coverage and exposure.

* Suggest criteria for weighting the significance and yield of contending variations. This aids in the selection of the variations to be tested, thus improving the efficiency of the testing effort.

* Establish the map of test variations versus test cases as a new vehicle for documentation of test case scope. This map can be used to calculate test case coverage and exposure and also to subset test cases for regression testing.

ACKNOWLEDGMENTS

Significant technical contributions have been made by Stanley Graham, Frederick J. Hennard, Arthur W. Pellman and Thomas R. Turner. Indispensable editorial assistance has been given by Mrs. Gertrude S. Elmendorf.

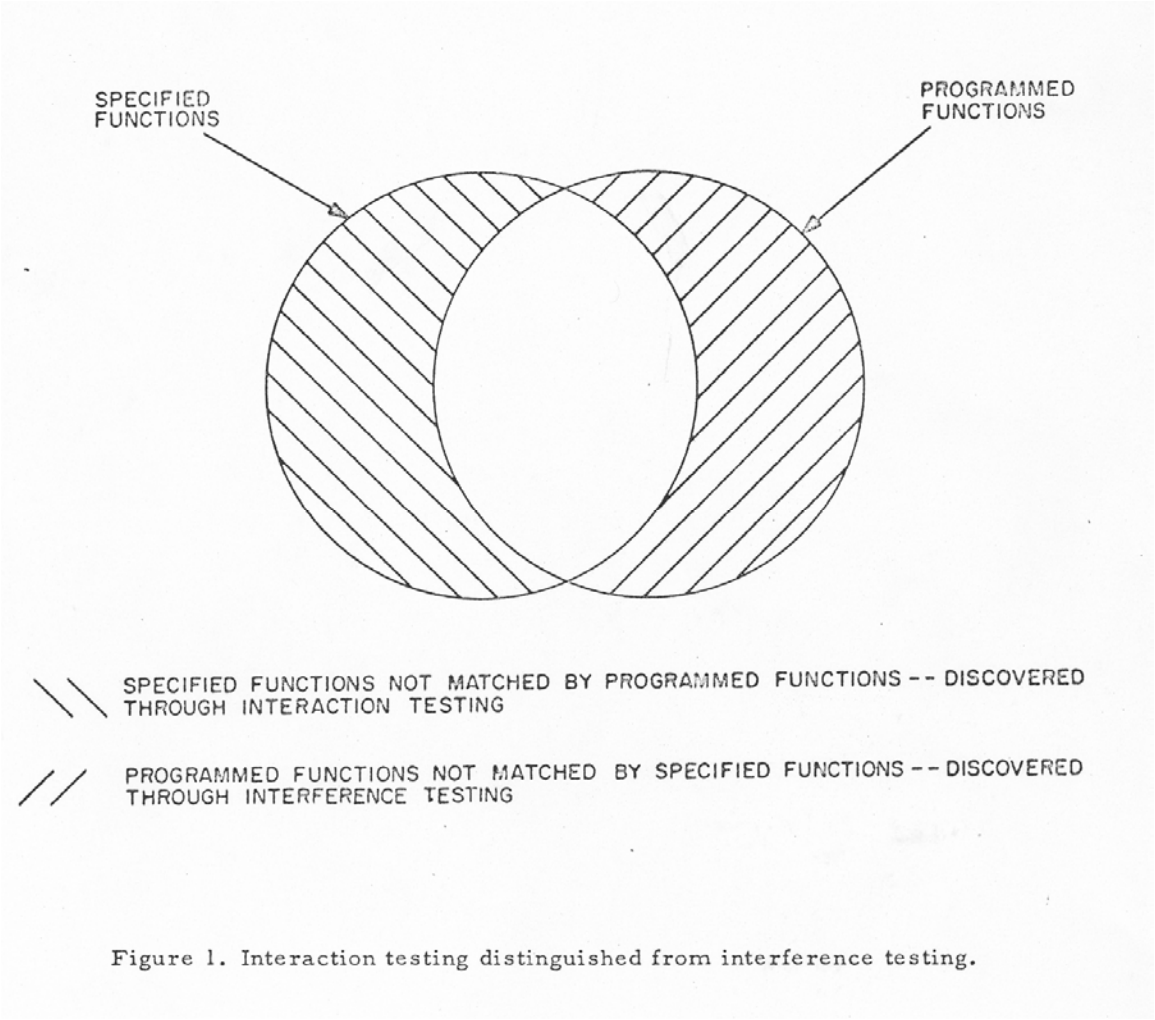
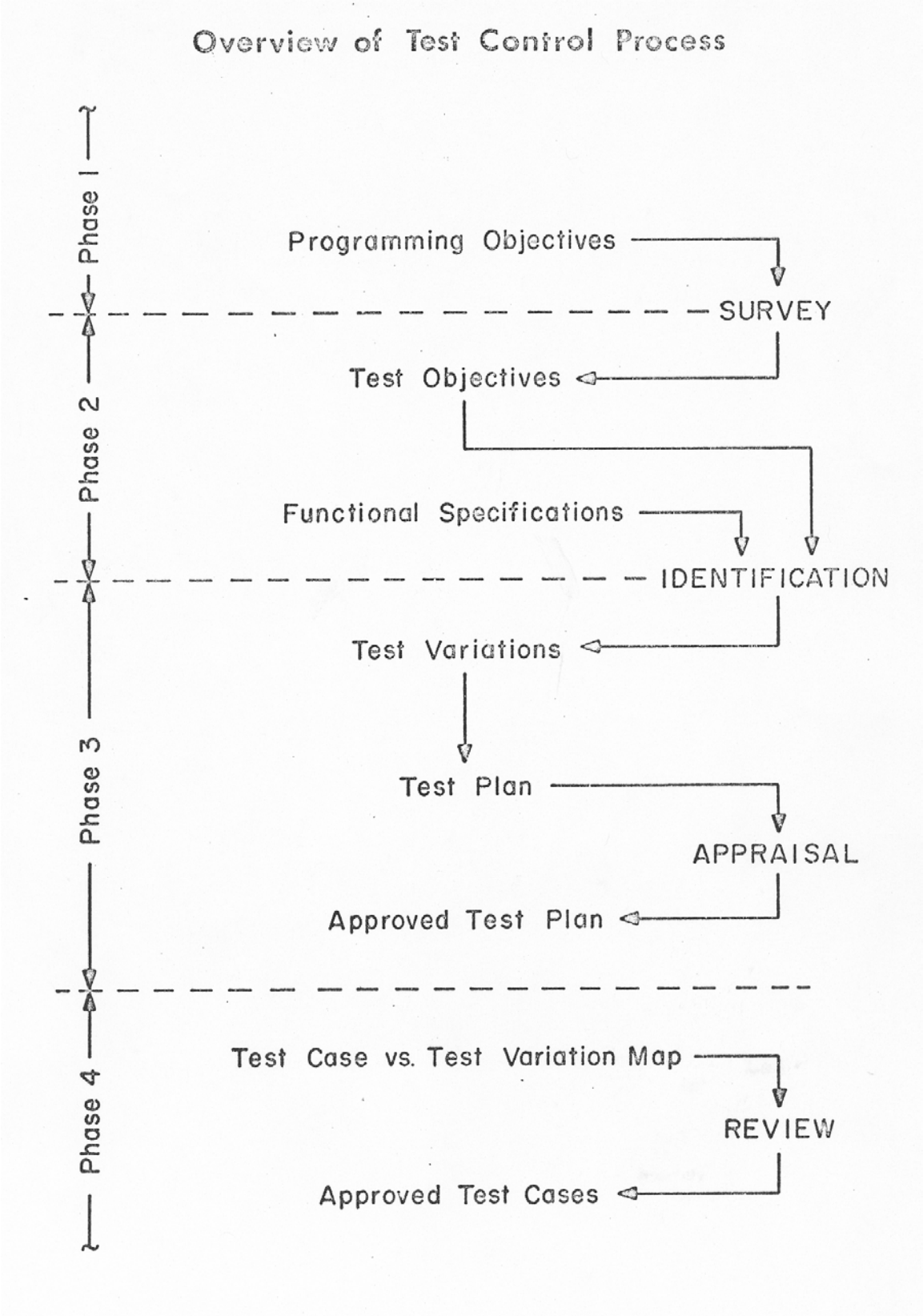


Figure 1. Interaction testing distinguished from interference testing.


Overview of Test Control Process
Phase 1
Phase 2
Phase 3
Phase 4
Programming Objectives
SURVEY
Test Objectives
Functional Specifications
IDENTIFICATION
Test Variations
Test Plan
APPRAISAL
Approved Test Plan
Test Case vs. Test Variation Map
REVIEW
Approved Test Cases

Sample of Coverage Matrix

Specifications			Variations		
			External Interaction	Internal Interaction	Interference
System Generation	New		95%	5% *	10% *
	Old		10%	0%	
System Initialization	New		95%	5% *	
	Old		10%	0%	
System Operation	New	Lookaside	95%	5% *	
		Interfaces	95%	95%	
	Old		20%	0%	

Sample of Test Variation List

12. Channel Program Read/Write Specifies Buffered DASD for which Specified Track is:

    a. Already in Buffer Memory

    b. Not Currently in Buffer Memory, and
        (1) Buffer must be Cleared
        (2) Buffer Available