A Systems View of Test Standardization

John W. Sheppard
ARINC
2551 Riva Road
Annapolis, MD 21401
(410) 266-2099
sheppard@arinc.com

William R. Simpson
IDA
1801 N. Beauregard St.
Alexandria, VA 22311
(703) 845-6637
rsimpson@ida.org

***Abstract*–In test engineering, "system test" is frequently considered to be testing at a particular level in a product hierarchy. With increasing complexity in systems, testing at lower levels is now faced with problems previously encountered at the system level. For this reason, it is becoming increasingly important to apply a system perspective to testing. In this paper, we present a model of a "system" to be applied to test engineering which abstracts test information above the physical level of the product. We then describe how this model supports two standardization efforts within the IEEE–P1226 and P1232.**

I. Introduction

Increasing complexity in modern electronics is forcing design and test engineers to consider even "low-level" components as systems. A system can be defined as "any aggregation of related elements that together form an entity of sufficient complexity for which it is impractical to treat all of the elements at the lowest level of detail [1]." Universities have begun to provide graduate-level curricula and courses in "systems engineering" to address the need for analysis using the systems view.

In the domain of test engineering, "system test" has been regarded as a "level" of test within the hierarchy of equipment rather than as a "view" of the test process. System test was applied to aircraft, armored vehicles, satellites, and missiles. Testing of radar subsystems, guidance subsystems, environmental control subsystems, etc. has been considered "assembly level" or "replaceable-unit level" even though they may be considered systems in their own right. In fact, down to the chip level, we may find sufficient complexity that we need to consider these "components" to be systems.

Much of the difficulty in defining system test has been in developing a definition of the word "system." In this paper, we will propose a definition of system test that approaches the definition of "system" from a functional view rather than a physical view. This definition will be based on a model developed of testing in which the subject of testing is defined by its context and test itself is defined by abstract information. We will discuss this model in detail and provide several examples illustrating multiple test problems represented as "system test" problems. Ultimately, we will argue that the "system" view of testing is applicable to all types of testing problems and, in fact, provides a medium for consistent testing that can efficiently and effectively bridge the gap across multiple levels of test.

In addition to defining the system view of testing, we will consider this definition in the context of several emerging test standards. The Department of Defense is actively pursuing using commercial standards in place of military standards and incorporating commercial solutions to its procurement needs. Two families of standards–A Broad Based Environment for Test (ABBET™) [2] and Artificial Intelligence and Expert System Tie to Automatic Test Equipment (AI-ESTATE) [3]–purport to apply a systems perspective to their families. Both families plan to enable cost-effective test solutions for military test by drawing from successful test engineering in the commercial market place. Both families of standards can benefit from the consistent view of system test as defined here, and we will describe how this view fits within the vision of these two standards initiatives.

II. The Complexity of Test and Diagnosis

In the broadest possible scope, testing is performed for various activities, including performance evaluation, mechanical/electrical integrity, periodic and unscheduled maintenance, process evaluation, operational readiness, and specification and compliance. In short, the underlying purpose of any testing process is information discovery. In this paper, we limited our view of testing to electronics test for design verification, manufacturing test, and maintenance test. In our concept, a test is a signal, indication, or other observable event that provides information about the system being tested [1]. The observation may be caused to happen (as with a stimulus/response test) or be part of a normal operational environment (as in the case of a symptom). The only purpose a test serves is to provide an outcome that can be used to infer something about the system being tested. As such, testing is driven by separate reasoning processes. The resultant reasoning on test results is context sensitive.

0-7803-3379-9/96 $5.00 © 1996 IEEE

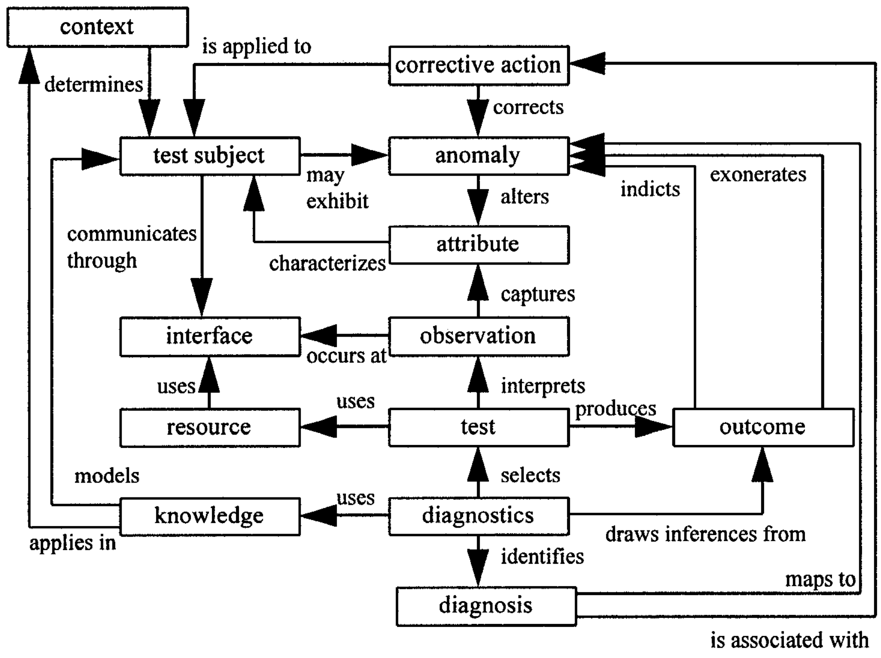

Figure 1. Model of System for Test.

The process of testing systems (regardless of the product hierarchy or the test objective) is highly complex. In [1], we presented several complexity results for the general problem of optimal fault diagnosis and for isolating multiple faults. Specifically, we proved that the number of potential test sequences that can be generated is $O(2^t t!)$ for binary tests where t is the number of tests available. Further, the number of possible decision trees (i.e., fault trees) that can be generated is also $O(2^t t!)$. This demonstrates that it is not possible to enumerate all possible diagnostic strategies and "pick the best one." Further, it has been proven that constructing an optimal binary decision tree is *NP*-complete [4].

Many believe that incorporating domain-specific information will reduce the complexity of the problem by focusing on characteristics unique to the domain. Unfortunately, this actually tends to increase complexity. Domain specific information can be used in one of two ways: either through identifying constraints and reasoning with those constraints or through capitalizing on specific information provided by technology-specific testing (e.g., *I*DDQ or digital fault dictionaries). For the case where one is reasoning about constraints, the general task of solving constraint satisfaction problems is also *NP*-complete [5].

For domain-specific testing, obviously the actual tests must be tied to the domain of the system. However, one frequently finds that attempting to use one specific approach to testing can be very expensive. The digital fault dictionary provides an excellent case in point. The cost of generating comprehensive digital fault dictionaries for detection is very expensive [6]. In fact, this has led to research attempting to reduce the number of vectors in the dictionary and simplify the matching process in the dictionary. When using fault dictionaries for isolation, the problem becomes significantly worse since now the need arises for tracking state changes, especially after a fault is detected. It should be clear that an approach for managing this level of complexity is required.

III. A New Model of System Test

To begin to address the issue of complexity described in the previous section, we define a new model for a system within the context of test and diagnosis (Figure 1). Our definition of system [1] supports a hierarchical view where a test strategy is applied within a level of the hierarchy. The objective of the model is to provide a consistent framework for testing across hierarchical levels to manage the complexity of testing within a particular hierarchical level. This model is based on an earlier model developed for the upper layers of the ABBET architecture [7].

In interpreting this model, note that we have not applied any of the standard object [8][9], activity [10], or information modeling [11] techniques commonly in use today. To read this model, each of the blocks represent entities or objects within the test environment. The relationships between the entities provide "processes" or "constraints" that one entity performs or imposes on another. In some cases, the processes are highly dynamic (e.g., `diagnostics selects` test), and in others they simply define a relationship (e.g., knowledge applies_in context). In all cases, the entity to entity relationships can be read as a simple sentence (subject verb object). In fact, we can now read the model much like a story or narrative description of test.

The primary focus of the model is test_subject which we can consider to be the "system under test." Thus, to define a system view of test that is applicable in all test environments, we focus on the test_subject and define the test_subject in relation to the other entities of the test environment. In all cases, these entities should be sufficiently concrete to be identifiable by a test engineer yet sufficiently abstract to be applicable in multiple contexts.

In general, we test to draw conclusions about some test_subject. This test_subject is defined by the context in which we are testing, meaning that elements such as the purpose of test, the operating conditions under which testing occurs, the accessibility of parts of the system, determine the types of tests available to us and the types of conclusions we can infer from testing. Thus the portion of the system that is actually being evaluated and the way in which it is being evaluated is defined by these elements of context and becomes the test_subject.

What we know about a test_subject is derived from observing characteristics of the system in context. Reading the model, we see that a test interprets observation which captures attributes which in turn characterize test_subject. This illustrates that testing is an inherently uncertain process since it is limited to interpreting the way specific characteristics of the system are manifest. There is no way to *directly* observe the presence or absence of all faults that may exist in a system; therefore, it is possible that error can be introduced into the process. The error may result from having too little information within an observation to draw a reliable conclusion, or we may be focusing on an inappropriate set of attributes.

The specific attributes of the system may be dynamic (behavioral) or static (physical). This is a change from the model presented in Sheppard and Simpson [7] where the focus of testing was on behavior. It was argued that static properties could be cast in terms of system behavior, but for some tests this seems to be unnatural. The natural way of representing the observable characteristics of a system is by considering a "supertype" of the dynamic and static properties. This is captured in the entity, attribute. Of course, if a system is not functioning according to some set of requirements, then we say the test_subject may_exhibit anomaly. Clearly, whether or not an anomaly is in fact exhibit depends on all of the facts that ultimately define the context for testing. Further, these observable anomalies alter attributes of the nominal system. The purpose of testing (in this context) is to determine whether or not a system is nominal, and if not, to determine what the anomalies are. Further, when the anomaly is identified, we want to determine the proper corrective_action to restore the system to its nominal state (thus corrective_action corrects anomaly).

Testing does not take place in a vacuum. Generally, test uses resource to make its observations. Further, the resource uses interface of the test_subject (thus test_subject communicates_through interface to the resource) to make its observation. For this reason, we say that an observation occurs_at interface. Note that the notion of an interface is very broad but still cast in the terminology of systems (as distinct from the port–cell–net model being used by EDIF [12]). As such, an interface may correspond to a physical mating between the test_subject and the resource, or it may correspond to a functional or logical mating (e.g., in software).

The final element of the model that we need to discuss drives the test process. As we said before, the purpose of testing is information discovery and drawing conclusions about the test_subject. The process of drawing conclusions from test information is referred to as "diagnostics.". Thus we consider the diagnostic process as centering on an entity called diagnostics. Here diagnostics uses knowledge to draw_inferences_from outcome to then identify diagnosis. As one might expected, diagnostics selects test which in turn produces outcome. Then outcome either indicts or exonerates anomaly. Once the diagnosis has been determined, one can take appropriate action since diagnosis maps_to anomaly and is_associated_with corrective_action. The whole process of performing a diagnosis is fully dependent on what diagnostics knows about the test_subject. This is captured by knowledge which models test_subject and applies_in context.

IV. Mapping Products Into Systems

The primary advantage of providing an abstract model for test and diagnosis and associating that model with a "system view" of the test subject is that it provides a means for abstracting details out of the test problem until needed. This in turn permits the structure of the test problem to be optimized, thus managing the complexities inherent in manipulating large, heterogeneous systems. For this to work, we need to be able to see how to map real systems into the model. This in turn enables us to identify the critical elements of the test problem and to optimize the solution. In this section, we map two "systems" into the model to illustrate.

The first system we will consider would not be considered by many to be a system since it is very simple. Consider an integrated circuit contain four D flip-flops (e.g., an SN5475). The `test_subject` would correspond to the this IC, and we would identify the `attributes` of the IC as the nominal behavior of the chip (as we might find in the truth table for each of the flip-flops). In addition, we might add characteristics such as voltage, temperature range, orientation of the pins, size and color of the packaging, etc. The actual set of attributes of interest to us would depend on the `context` under which we are testing. If we only care about failure modes associated with the pins and the logic of the chip, we may restrict our focus to the logic values observed at the pins. If we are interested in manufacturing issues, we might include characteristics of packaging.

For the sake of discussion, we will limit the context to be assessing the logical performance of the chip at the pins. Thus we can restrict our attention to the logic specification and to the set of stuck-at faults. This set of faults would define the `anomalies` that might be exhibited by the chip. If we use a digital tester with a fault dictionary, then `knowledge` would correspond to the fault dictionary itself and `diagnostics` would include the test controller and the matching algorithm in the dictionary. The `tests` would be defined by the vectors in the fault dictionary (actually, each of the output values for each vector would correspond to a different `test`). Whenever a vector is processed, the associated tests would either pass or fail (thus we know the `outcomes`), and these outcomes would point to the possible anomalies of the chip. Possible `corrective_actions` in the event the chip is faulty include replacing the chip or re-setting (or re-soldering) the chip in the socket.

For the second system, we will consider a case at the other end of the spectrum. Consider a network of satellites in orbit that provide the backbone for a communications network. One test `context` of interest would be to verify that all of the defined links in the network are present (including satellite-to-satellite cross links and ground links). Thus we have defined our `test_subject` to be the connectivity of the network. `Attributes` of this system might include transmission of messages between two points in the network in a reasonable period of time, and we could define `tests` to be a set of messages that traverse the network in some predictable way. The `anomalies` would correspond to links being down. (This is actually analogous to a kind of stuck-at fault in the digital model.)

For this system, the `knowledge` required for diagnosis and the associated `diagnostics` might be significantly different from the fault dictionary of the first system. For example, it is possible we would be able to rely on a set of SNMP (simple network management protocol) traps to signify when a link drops in the network (assuming the endpoints of the links are treated as SNMP managed objects). In this case, the `knowledge` would correspond to the MIB (managed-object information base), and the `diagnostics` would be a passive network monitoring system or alarm correlator.

V. Applying the Model to Test Standards

The advantage to an abstract model such as the one we describe in this paper is that we might be able to develop an approach to test and diagnosis that is consistent with all systems mapped onto the model. Our previous two examples, in fact, outline very different approaches to test and diagnosis; however, this is not necessary. If we can standardize on 1) the process for controlling test resources, 2) the process of making a diagnosis, and 3) the knowledge used by these two controllers, then we can devise a generic approach to test and diagnosis that is independent of the underlying technology.

This is exactly the intent of the ABBET and AI-ESTATE standardization efforts. ABBET is attempting to define a set of service specifications for managing test resources in a test environment, and as the name implies, the intent is for this test environment to be "broad based." Recently, AI-ESTATE modified the meaning of the acronym to emphasize broad applicability as well. The new definition of AI-ESTATE is "Artificial Intelligence Exchange and Service Tie to All Test Environments." AI-ESTATE is concerned with defining the knowledge and services to be provided for a diagnostic reasoner in a test environment.

As described in [7], the model of the ABBET upper layers is very close to the model presented in this paper. Currently, ABBET is structured around a "test foundation framework" and a set of application frameworks specific to test. One view of the ABBET architecture is presented as a set of five layers: the product description layer, the test strategy/requirements layer, the test procedure layer, the test resource layer, and the instrument layer. Accordingly, the following projects have been authorized by the IEEE for developing ABBET standards:

- P1226—ABBET overview and architecture
- P1226.3—Resource management
- P1226.4—Software interface for instrument drivers
- P1226.5—Software interface for instrument buses
- P1226.6—Guide to the understanding of ABBET
- P1226.7—Software interface for product data
- P1226.8—Software interface for test strategies
- P1226.10—Software interface for runtime services

Current work performed in these projects are devoted to standardizing the interfaces and services which focus on the resources used for automatic testing. The current set of standards deal almost exclusively with testing on Automatic Test Systems (ATS), but the ultimate intent is to apply the standards in a much broader context. The model described above could be used to help focus this initiative.

The AI-ESTATE initiative is much more tightly focused in terms of the systems view. The AI-ESTATE standard P1232 is being developed to standardize on the interfaces between test systems and artificial intelligence based systems. In addition, AI-ESTATE is including standard representations for several types of knowledge bases and databases. Currently [13], the standard specifies representations for fault tree models (FTMs) and enhanced diagnostic inference models (EDIMs).

Currently, AI-ESTATE is being developed using a cooperative processing model. Under this model, all processes communicate across a communications pathway or bus and access other parts of the system through a set of services. Thus each functional block of an AI-ESTATE system consists of an object in the sense of object-oriented analysis. Specifically, each functional block is defined by the operations that can be performed on or by that block. The attributes of the block are specified in a class lattice in order to maximize reuse of components. If one functional block needs to interact with another, it does so by using the services provided by that block. Using this architectural concept, the objects communicating in an AI-ESTATE systems include the test system, the reasoner, the human presentation system, a maintenance data collection system, the unit under test, and the operating system. Any data that is used by the objects on the pathway will be specified in some standard representation. Even though we currently have specifications for only the FTM and the EDIM, AI-ESTATE is working on a specification for a constraint model [14]. AI-ESTATE anticipates defining or referencing standards for other data and knowledge bases.

Recently, considerable progress has been made in defining P1232.2 [15] which is the service specification for AI-ESTATE. This document provides a definition of the services to be provided by a diagnostic reasoner according to the cooperative processing model described above. Details describing the application of this standard in a test environment are discussed in [16].

V. Conclusion

The rising complexity of systems is forcing test engineers to apply novel approaches test and diagnosis. Engineers have recognized that complexity is continuing to rise and no clever algorithm or process will eliminate that complexity. Consequently, approaches are required to manage the complexity and control the impact of the rate of growth on current or new test processes. Perhaps the best way to approach a difficult problem is to cast that problem in a new light. By attempting to represent the requirements and constraints of a problem from a new perspective, one can gain added insight into the source of complexity. This in turn can shed light on how one might best approach the problem.

The purpose of this paper has been to provide a new perspective of the test problem to facilitate developing a structure for managing the complexities associated with efficient and effective test and diagnosis. The approach involved modeling the test problem from an abstract point-of-view in which the context defines the subject of testing and the requirements to be achieved by testing. The approach abstracts low-level details of the problem out of the model until a structure is in place for testing, at which time these details can be reapplied to the problem.

The focus of the paper has been on applying this model in the context of standards for test and diagnosis. The primary purpose of standardization is to facilitate the development of tools and systems that are predictable and widely applicable, thereby managing the complexity associated with their target problems. By defining the new model fot system test and mapping this model into the ABBET and AI-ESTATE test standards, we have provided direction for developing and applying these standards in such a way that the stated objectives of standardization can be achieved.

References

[1] Simpson, W. R. and J. W. Sheppard. 1994. *System Test and Diagnosis*, Norwell, Massachusetts: Kluwer Academic Publishers.

[2] IEEE Std 1226-1993. *Trial Use Standard for A Broad Based Environment for Test (ABBET): Overview and Architecture*, Piscataway, New Jersey: IEEE Standards Press.

[3] IEEE Std 1232-1995. *Trial Use Standard for Artificial Intelligence and Expert System Tie to Automatic Test Equipment (AI-ESTATE): Overview and Architecture*, Piscataway, New Jersey: IEEE Standards Press.

[4] Hyafil, L. and R. Rivest. 1976. "Constructing Optimal Binary Decision Trees is *NP*-Complete," *Information Processing Letters*, Vol. 5, No. 1, May, pp. 15–17.

[5] Dechter, R. 1992. "Constraint Networks: A Survey," *Encyclopedia of Artificial Intelligence*, Stuart C. Shapiro (ed.), New York: Wiley.

[6] Sheppard, J. W.. and W. R. Simpson. 1996. "Improving the Accuracy of Diagnostics Using Fault Dictionaries," *Proceedings of the 14th IEEE VLSI Test Symposium*, Piscataway, New Jersey: IEEE Press.

[7] Sheppard, J. W., and Simpson, W. R. 1995. "A View of the ABBET™ Upper Layers," *AUTOTESTCON '95 Proceedings*, Piscataway, New Jersey: IEEE, pp. 51–56.

[8] Booch, G. 1994. *Object-Oriented Analysis And Design With Applications*, 2nd Ed. Benjamin Cummings.

[9] Schlaer, S., and Mellor, S. L. 1992. *Object Lifecycles: Modeling the World in States*, Englewood Cliffs, New Jersey: Yourdon Press.

[10] FIPS-183. 1993. *Integrated Definition for Function Modeling (IDEF0)*. National Institute of Standards and Technology.

[11] ISO 10303-11. 1992. *Industrial Automatic Systems—Product Data Representation and Exchange–Part 11: EXPRESS Language Reference Manual*, International Organization on Standardization.

[12] ANSI/EIA Std 618. 1994. *EDIF 3 0 0*, American National Standards Institute.

[13] IEEE P1232.1. 1996. *Trial Use Standard for Artificial Intelligence and Expert System Tie to Automatic Test Equipment (AI-ESTATE): Data and Knowledge Specification*, Draft 4.7.

[14] Sheppard, J. W. and J. Åstrand. 1995. "Modeling Diagnostic Constraints with AI-ESTATE," *Proceedings of AUTOTESTCON '95*, Piscataway, New Jersey: IEEE Press.

[15] IEEE P1232.2. 1996. *Trial Use Standard for Artificial Intelligence and Expert System Tie to Automatic Test Equipment (AI-ESTATE): Service Specification*, Draft 2.0.

[16] Maguire, R., and J. W. Sheppard. 1996. "Application Scenarios for AI-ESTATE Services," *Proceedings of AUTOTESTCON '96*, Piscataway, New Jersey: IEEE Press.