Information-Based Standards and Diagnostic Component Technology

John W. Sheppard
ARINC Incorporated
2551 Riva Road
Annapolis, MD 21401
jsheppar@arinc.com

Amanda Jane Giarla
Hamilton Software Inc.
2270 Northpoint Parkway
Santa Rosa, CA 95407
agiarla@hamsoft.com

Abstract: Software development methods have evolved over the years from structured design of procedural code, to object oriented design, to component-based design. Recent requirements by industry and government have resulted in the development of interface specifications and standards designed to facilitate acquisition of large systems based on the concepts of component technology. In this paper, we discuss the development of information-based standards for diagnostic information and diagnostic reasoning intended to provide the definition of diagnostic components within a larger test or health management environment.

Introduction

Modern systems are growing in complexity, making the problem of system development and system maintenance increasingly difficult. This trend has become apparent especially when dealing with modern software systems. Comparing the capabilities provided by "simple" word processing systems today compared to just ten years ago illustrates the magnitude of the problem. New capabilities abound.

Besides the growing complexity of the systems being developed, the means by which the systems are being utilized is complicated design, development, and maintenance. Specifically, more and more problems require access to distributed resources for them to be solved. To access these distributed resources, the means of communication between components utilizing the resources needs to be clearly defined.

In an attempt to manage the growing complexity of complex systems, design methodologies have changed from traditional top-down, hierarchical design to object-oriented design to component-based design (Szyperski, 1998). The idea behind component-based design is that a system can be subdivided into groups of interchangeable components with well-defined interfaces between those components. For a particular problem, relevant components can be "plugged" into a –"component framework" (Szyperski, 1998), and information can be shared between the components to facilitate solving the problem. With this philosophy in mind, one can envision modern systems as being defined through composition and configured "on-the-fly" based on the requirements imposed by a particular problem or class of problems to be solved.

In this paper, we discuss the definition of interfaces to a particular type of component—the diagnostic component of a test or health management system. We will focus on the role of information-based specifications and a particular information-based standard to define these interfaces. Specifically, we will examine the Artificial Intelligence Exchange and Service Tie to All Test Environment (AI-ESTATE) standards, published by the IEEE, and their role in the definition of diagnostic components (IEEE, 1995; IEEE, 1997; IEEE, 1998).

Component Technology

The vision of modern component-based systems is to be "composed" of a set of "re-usable" components that work together in providing the same capabilities that custom software solutions provide but at a lower price along with a shorter construction period and a reduction in problem solution complexity. The equipment test or health management environment is a domain that requires both software and hardware type components.

Within the context of a test system a data bus embedded within a VXI or PXI chassis along with the bus software drivers is a form of a component

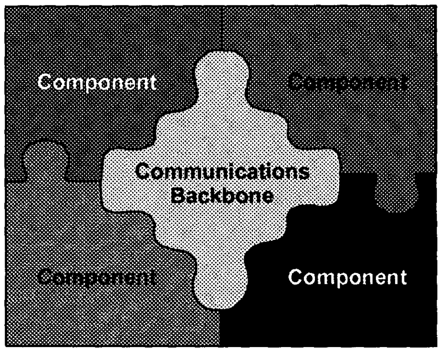

Figure 1. Component-Based Architecture

framework. This type of framework is better understood as a communications infrastructure or a backbone that contributes to overall performance of a task or solution to a problem. We can then consider components such as the unit under test (UUT), the test executive, several instruments, and the diagnostic reasoner. Figure 1 depicts such an architecture as a set of puzzle pieces.

The primary challenge for making component technology work is well-defined, unambiguous communication between the components in the architecture. For communication to be unambiguous, the entities that are communicating must agree before they attempt to communicate on the characteristics of the message. In other words, they must speak a common language.

Within the context of a computer-based system, this language must be defined such that both the syntax (i.e., the structure or format of the language) and the semantics (i.e., the meaning of the messages constructed using the language) are understood. Natural languages are wrought with ambiguity based on cultural differences, idioms, and exceptions. Formal languages eliminate the ambiguity by providing mathematical definition of both syntax and semantics.

For component-based systems, the language is defined relative to communication over the backbone, or framework. Note this backbone need not be a physical backbone but may be conceptual. The important element of this backbone is the way the components connect to it, for example COM, CORBA, DCE/RPC, Java RMI, data bus drivers. In other words, key to the definition of the communications infrastructure for a component-based system is the definition of the language used relative to the interface of the component with the backbone. Specifically, the structure of the interface defines the syntax of the language by constraining the format of the message carried across the interface. The meaning of information contained within a message, which corresponds to the semantics, is defined by agreement between the components communicating.

It should now be evident that the key element in communication is formal definition of the information exchanged between the parties communicating. According to Schenk and Wilson, information can be defined as "knowledge of ideas, facts, and/or processes (1994)." Key to understanding the role and importance of information is recognizing that it is information that is exchanged between parties during communication. Further, this exchange is something that can be real-time (e.g., a spoken message over a telephone) or delayed (a written message sent by email).

The Role of Information

All processes depend on the sharing of information. For information to be shared, it must be communicated. A process can be modeled as a decision cycle in which information is received and analyzed, a decision is made about what to do, and some action is taken.

This decision cycle has been represented is the command-and-control community as an "OODA-loop" (Figure 2). The OODA loop corresponds to a cycle of repeating for distinct phases:

1. **Observation:** Collecting *information* about the current state of a problem.
2. **Orientation:** Interpreting the *information* to evaluate the current state relative to some objective.
3. **Decision:** Evaluating the *information* to determine a course of action.
4. **Action:** Taking an action based on the decision made to modify the state of the problem.

In the context of test and diagnosis, the diagnostic process can be mapped to the OODA loop

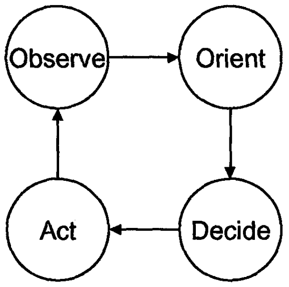

Figure 2. OODA Loop

as follows. First, the results of one or more tests are examined to capture information about the health-state of the system. This corresponds to *observing* the health-state. Second, the test results are mapped to a set of outcomes and associated inferences to refine the current understanding of the health-state relative to the goal of the test process (e.g., fault isolation). This corresponds to *orienting* the diagnostic system based on its current set of observations and its objective. Third, the test system *decides* what to do next in terms of announcing a fault has been isolated or further testing is required. If further testing is required, a test is chosen. Fourth, given a test has been chosen, the test is performed, thus defining how the system *acts*.

Information Modeling

One approach to defining the interfaces for a component of a larger system is to model, formally, the information being passed across the interface. Such a model is known as an "information model." An information model is "a formal description of types (classes) of ideas, facts, and processes that together form a model of a portion of interest of the real world (Schenk and Wilson, 1994).

The purpose of an information model is to identify clearly the objects in a domain of discourse (e.g., diagnostics) to enable precise communication about that domain. Such a model comprises objects or entities, relationships between those objects, and constrains on the objects and their relationships. When taken together, elements provide a complete, unambiguous, formal representation of the domain of discourse. In other words, they provide a formal language for communicating about the domain.

Ambiguous definition of syntax can lead to miscommunication such as the following. For example, what does the date "1-3-91" represent? In the United States, this would represent January 3, 1991; however, in Europe, it would represent March 1, 1991.

Failure to agree upon definitions (i.e., semantics) can also lead to miscommunication. For examples, what are "braces?" In the United States, one use of the word braces is to represent a device for adjusting one's teeth; however, in the United Kingdom, one use of the word braces is to represent a device for holding up one's trousers.

Key to determining semantics is an understanding of the underlying context. For example, in the following two sentences, the role of the phrase "flies like" is determined by how the phrase is used in the sentence.

- Time *flies like* an arrow.
- Fruit *flies like* a banana.

Even within a specific domain, terminology can be ambiguous. For example, what is a test? If we ignore domain, one logical response is that a test is "a set of questions used to assess the level of achievement and comprehension of a student." However, it we limit the context to the domain of equipment test and diagnosis, we are no better off. For example, a test to a digital test engineer is "a set of vectors used to determine if a digital circuit is working properly," but a test to a diagnostician is "*any* combination of stimulus, response, and basis for comparison that can be used to detect or isolate a fault."

Similarly, answering the question, "What is a diagnosis?" is equally complex. Without considering context, a diagnosis may be "a disease or condition identifiable through clinical means." On the other hand, if we limit our context once again to the domain of equipment test and diagnosis, a test engineer might define a diagnosis as "an identifiable and isolatable fault within a system." A diagnostician, on the other hand, might define a diagnosis as "any conclusion that can be drawn about the health-state of the system, including the absence of a fault."

From these examples, it should be evident that the definition of the language used for communication is critical. Information modeling is a tool for providing that definition. Specifically, information models define classes of ideas, facts, or hypotheses, and can be populated with instances of these ideas, facts, and hypotheses. If the model is defined using a computer processable language, e.g., EXPRESS (ISO, 1994a), then that model has the benefit of determining the syntax for information exchange, as well as the semantics of the information modeled.

Using information models, information exchange can be facilitated in two ways. The first is through a set of exchange files. Specifically, information can be stored by one party in a file and read by a second party. The file format is derived directly from the information model and defines the syntax of the message contained within it. The semantics of the message (i.e., the legal content of the file) is defined by the semantics of the model.

The second means of information exchange is through a set of services defined for a hardware component or a software component as accessed via the communications backbone. The interface definition for the component is derived from the information model and defines the syntax of the message. Once again, the legal content of the message is defined by the semantics of the model.

Standard Information Models

For component-based technology to work when a variety of organizations are developing components to work together, the nature of the communication between the components (i.e., the language) must be agreed upon beforehand, such as through a contract. Contracts contain two parts, interface definition or syntax and specification or semantics. For component markets to form and sustain commerce these contracts must be immutable (Szyperski, 1998). Such advance contracts are typically defined through standards.

Three advantages to using standard information models to defining the communications mechanism are evident. First, since standards are published documents, a large audience has access to the standard. By specifying standards in procurement documents or design documents, the designers know before detailed design begins the basis for communication.

Second, the contract defined by a standard has been validated and legitimized by the fact that a community of experts in the domain have gathered and agreed upon the content of the standard. Consequently, users of the standard can trust that a) the standard is technically correct, and b) the community of those using the standard believe the standard is useful.

Third, standards are typically endorsed and accredited by an independent accrediting body. Such endorsement certifies that the standard was developed according to an open process designed to keep the best interests of the community in mind. Examples of such accrediting bodies include IEEE, ANSI, ISO, and IEC.

Several formal languages are available for defining information models. The four most prominent languages are UML, IDL, IDEF 1x, and EXPRESS. The Unified Modeling Language (UML) was standardized by the Object Management Group and provides a set of tools for constructing object models and information models relative to object-oriented systems. The primary disadvantage to UML for defining information-based standards is that there is no defined process for creating or interpreting these models. As such, the underlying contract for communication would be incomplete.

The OMG also defined and standardized an Interface Definition Language (IDL) within the context of its Common Object Request Broker Architecture (CORBA). IDL provides formal specification of interfaces to methods within an object-oriented framework; however, once again, no process has been defined to ensure the underlying information contract is complete.

The US Air Force defined a method for specifying information to be stored in a relational database called Integrated Definition for information Modeling (IDEF). Under the auspices of the International Federation of Information Processing Standards (IFIPS), this method was standardized as IDEF 1x. The advantage of IDEF 1x over UML and IDL is that a formal implementation can be derived from the model. The primary disadvantages include limitation to relational databases an no definition of the semantics of the data.

EXPRESS, standardized by ISO, was designed to focus on the problem of formally defining information in support of communication. EXPRESS is object-oriented in flavor but focuses on formally defining the semantics of the information modeled. In addition, rules have been defined for deriving exchange files and services for information exchange directly from the EXPRESS models.

AI-ESTATE

The Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE) standards are information exchange standards for test and diagnosis. The original standards, the 1232 series, developed a means of exchange of information between diagnostic reasoners. As the information models for the 1232 standards were developed, it became apparent that these models could be used for standardizing testability and diagnosability metrics.

IEEE Std 1232-1995 defines the architecture of an AI-ESTATE-conformant system. IEEE Std 1232.1-1997 defines a knowledge and data exchange standard. In 1998, IEEE Std 1232.2-1998 was published. This standard formally defines a set of standard software services to be provided by a diagnostic reasoner in an open-architecture test environment. The standards were developed using information modeling as described above, resulting in the definition of four information models addressing static and dynamic aspects of the diagnostic domain. Further, the IEEE 1232 AI-ESTATE series of standards provide the foundation for precise and unambiguous testability and diagnosability metrics.

The vision of AI-ESTATE is to provide an integrated, formal view of diagnostic information as it exists in diagnostic knowledge bases and as it is used (or generated) in diagnostic systems. We assert that the whole purpose of testing is to perform diagnosis (Simpson and Sheppard, 1994). In justifying this assumption, we rely on a very general definition of diagnosis, derived from its Greek components (δια γιγνωσκω) meaning, "to discern apart." Given such a broad definition, all testing is done to provide information about the object being tested and to differentiate some state of that object from a set of possible states.

In support of this vision, the AI-ESTATE committee has been working on combining the existing standards into a single, cohesive standard. This "unified" standard provides formal specifications of all of the information models (both for file exchange and for diagnostic processing), from which the service specifications are then derived and specified. The architectural framework is retained at the conceptual level to emphasize that a wide variety of implementation models are possible that still support standard exchange of information as long as the definition of that information is clear and unambiguous. Thus, in a sense, the models define the architecture, and the implementation is left entirely to the implementer.

With this vision in mind, we believe AI-ESTATE plays a central role in any test environment (thus the "All Test Environments" part of the name). To date, the focus of the standards has been the development of specifications supporting diagnosis in the traditional sense of the word (i.e., fault isolation). However, the broader context within which AI-ESTATE is envisioned to participate involves tying diagnostic information to explicit product behavior descriptions, assessments of the ability of testing to satisfy its requirements, and maturation of the diagnostic process through test and maintenance information feedback.

The AI-ESTATE Architecture

According to IEEE Std 1232-1995, the AI-ESTATE architecture is "a conceptual model" in which "AI-ESTATE applications may use any combination of components and intercomponent communication (IEEE, 1995)." On the other hand, according to IEEE Std 1232.2-1998, AI-ESTATE includes explicit definitions of services to be provided by a diagnostic reasoner, where the services "can be thought of as responses to client requests from the other components of the system architecture (IEEE, 1998)." More specifically, "each of the elements that interface with the reasoner will interact through [an] application executive and will provide its own set of encapsulated services to its respective clients (IEEE, 1998)."

Although not necessarily obvious from the standards themselves, these two "views" of the AI-ESTATE architecture present an interesting apparent dichotomy. Specifically, the architecture standard provides a concept of AI-ESTATE that permits any communication mechanism to be used between components of a test environment in support of the diagnostics provided by that environment. The service specification, on the other hand, seems to cast the communication mechanism in the form of a client-server architecture.

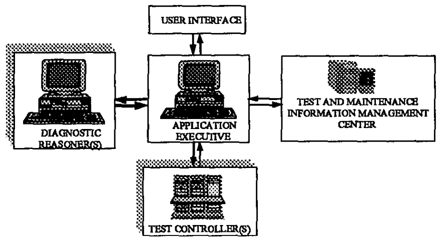

Figure 3. AI-ESTATE Embedded in Client/Server Based Architecture

In 1998 Hamilton Software, Inc., (HSI) was awarded an Air Force SBIR contract to implement AI-ESTATE within a new component based approach to ATS construction. Giarla proposed an approach that utilizes the original notions of AI-ESTATE components and inter-component communications (Giarla 1999). Giarla's approach uses both the AI-ESTATE service interface definitions and service specifications to define the 2 part component contract for the Diagnostic Engine Component.

We note that the intent of AI-ESTATE is to provide a formal, standard framework for the exchange of diagnostic information (both static and dynamic) in a test environment. This exchange occurs at two levels. At the first level, data and knowledge are exchanged through a neutral exchange format, as specified by IEEE Std 1232.1-1997 (IEEE, 1997). At the second level, specified by IEEE Std

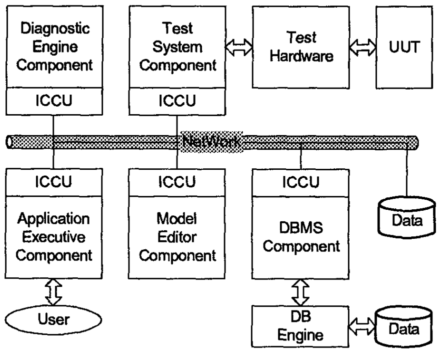

Figure 4 AI-ESTATE Embedded in Component Based Architecture

1232.2-1998 information is exchanged as needed between software applications within the test environment (IEEE, 1998). This information includes entities from a model or information on the current state of the diagnostic process.

To facilitate encapsulation of the information and the underlying mechanisms providing that encapsulation, AI-ESTATE assumes the presence of an "application executive." We emphasize that this application executive need not be a physically separate software process but can be identified as a "view" of the software process when it involves the communication activity. This view of the architecture is shown in Figure 3 & Figure 4. In the following sections, we will provide a more detailed discussion of the exchange and service elements of the architecture.

Data and Knowledge Exchange

ISO 10303–11 (EXPRESS) and ISO 10303–12 (EXPRESS-I) are used to define information models and exchange formats for diagnostic knowledge (ISO, 1994a; ISO, 1994b). The STEP (Standard for the Exchange of Product model data) community is maintaining these international standards. The current approach to static information exchange within AI-ESTATE is to derive the exchange format from the formal information models as specified in the ISO standards.

When IEEE 1232.1 was published, it was published as a "trial-use" standard to provide a period for people to study it, attempt to implement it, and provide feedback to the AI-ESTATE committee on the ability of the standard to satisfy the stated requirements. Since publication, comments have been received to indicate that ambiguity still exists in the information models.

Because of the concern that the information models are still ambiguous, the models are undergoing close examination and modification. It is interesting to note that much of the ambiguity has been identified in connection with a related standard being developed by the AI-ESTATE committee—P1522 Standard for Testability and Diagnosability Metrics and Characteristics. AI-ESTATE's approach to developing this new standard involved defining the metrics based on the information models within the P1232 standard. As we were identifying metrics to be

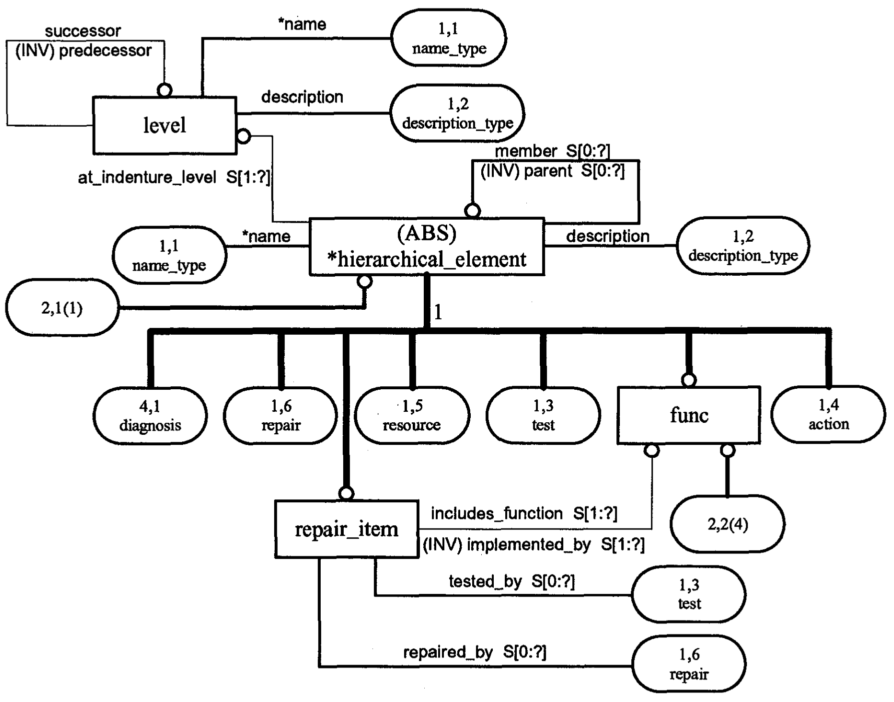

Figure 5. Revised Common Element Model

standardized, we discovered that the current models were incapable of supporting their definition.

A conceptual view of the revised common element model is shown in Figure 5. Of note in the revised model is the addition of a context entity and the differentiation between fault and function. Many diagnostic tools are highly context dependent (e.g., different procedures are suggested based on the environmental conditions of the test or the skill levels of the maintenance technicians). In addition, several tools focus on modeling function rather than physical faults to support modeling at the system level. Since the distinctions among context and type of analysis were not previously made explicit, new entities were defined to eliminate ambiguity that may arise from different approaches and contexts for modeling.

Diagnostic Services

The approach taken to defining services in AI-ESTATE has been based on the traversal (i.e., the following of the relationships defined between model entities to access specific pieces of information in the models) of the information models. The "simplest" services involve traversing the models defined in IEEE 1232.1 (i.e., the exchange models); however, these models provide little functionality in terms of actual diagnosis.

In IEEE 1232.2, a novel use of information modeling was applied in that a dynamic information model was specified to support dynamic services. This model, called the "dynamic context model," relied on dynamically creating entities that populate the model during a diagnostic session. In fact, as suggested by "dcm.session" and "dcm.step" in the model shown in Figure 5, a diagnostic session is modeled as a sequence of steps instantiated from the set of possible values specified in the static model. Details of how the service specification is expected to be implemented can be found in (Sheppard and Maguire, 1996; Sheppard and Orlidge, 1997).

One of the concerns raised by a member of the AI-ESTATE committee was whether the standard specifies a set of services or simply an Application Programming Interface. The claim was that the service specification must include a behavior specification as well and that this can only be accomplished by defining a set of baseline behaviors, perhaps through some sort of test bed.

The committee observed that people have different opinions over the difference between a service specification and an API specification. Further, it was determined that including test cases to specify standard behavior was not desirable in this context due to the wide variety of diagnostic approaches using common diagnostic knowledge. Rather, it was believed that it was more important for the information itself to be standardized and the specific behavior to be left to the implementation.

Summary

In this paper, we argued that ensuring unambiguous communication within a component-based architecture requires formal definition of the information communication. This formal definition is accomplished through the creation of information models. From these models, standard information exchange can be accomplished via exchange files and software services.

The AI-ESTATE family of standards was presented as an example of an information-based standard used to define the way a diagnostic component interacts with a test system or health-management system. The benefits afforded diagnostic components built using these standards include:

- Communication of the information between parties is reliable because the syntax and semantics of the information is agreed upon beforehand.
- Diagnostic components constructed according to the standard facilitate competition in the marketplace, this reducing cost and driving advancement in capability.
- The availability of reusable components provides flexibility to those building complex system, thereby reducing the overall complexity and cost of those systems.
- Providing information according to standard models increases the pool of information resources available, for example in terms of available models.
- Standards indicate a level of maturity in the underlying technology, thus increasing confidence and reducing risk.

Information is the key to communication in any process, including computer-based processes and the development of commercial component contracts. Complex systems being built today require the interaction of a large number of processes, often distributed both logically and geographically. For these distributed processes to interact properly, the processes must be able to connect and communicate in such a way that the parties of the communication understand the shared information.

Acknowledgments

In many ways, it is unfortunate that a paper such as this includes only the names of two authors. The work reported here is the result of efforts of a committee of devoted volunteers who have supplied their expertise in system test and diagnosis to develop strong, sound standards supporting the diagnostics community. We would like to thank Les Orlidge, Randy Simpson, Tim Bearse, Tim Wilmering, Greg Bowman, Mark Kaufman, Bill Simerly, Dave Kleinman, Lee Shombert, Sharon Goodall, Len Haynes, Jack Taylor, and Helmut Scheibenzuber for their efforts in developing the standards and promoting their use in industry.

References

IEEE Std 1232-1995. *IEEE Standard for Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE): Overview and Architecture*, Piscataway, NJ: IEEE Standards Press.

IEEE Std 1232.1-1997. *IEEE Standard for Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE): Data and Knowledge Specification*, Piscataway, NJ: IEEE Standards Press.

IEEE Std 1232.2-1998. *IEEE Trial-Use Standard for Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE): Service Specification*, Piscataway, NJ: IEEE Standards Press.

ISO 10303-11:1994. *Industrial automation systems and integration – Product data representation and Exchange – Part 11: Description methods: The EXPRESS language reference manual,* Geneva, Switzerland: International Organization for Standardization.

ISO 10303-12:1997. *Industrial Automation Systems and Integration—Product Data Representation and Exchange—Part 12: Description Methods: The EXPRESS-I Language Reference Manual,* Geneva, Switzerland: International Organization for Standardization.

Giarla, A., 1999. "Implementing AI-ESTATE in a Component Based Architecture, Phase-I," Proceedings of *Systems Readiness Technology Conference, Autotestcon 1999, IEEE 1999.*

Press.Schenk, D., and P. Wilson. 1994. *Information Modeling: The EXPRESS Way*, New York: Oxford University Press.

Sheppard, J., and R. Maguire. 1996. "Application Scenarios for AI-ESTATE Services," *AUTOTESTCON '96 Conference Record*, New York: IEEE Press.

Sheppard, J., and L. Orlidge. 1997. Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE)—A New Standard for System Diagnostics," *Proceedings of the International Test Conference*, Los Alamitos, CA: IEEE Computer Society Press.

Simpson, W., and J. Sheppard. 1994. *System Test and Diagnosis*, Boston: Kluwer Academic Publishers.

Szyperski, C. 1998. *Component Software : Beyond Object-Oriented Programming,* New York: Addison-Wesley Publishing Company.