Implementing AI-ESTATE with Prognostic Extensions in Java

Liessman Sturlaugson, Nathan Fortier, Patrick Donnelly, and John W. Sheppard
Department of Computer Science
Montana State University
Bozeman, MT 59717
{liessman.sturlaugson, nathan.fortier, patrick.donnelly2, john.sheppard}@cs.montana.edu

Abstract—**This paper is part of an ongoing effort to facilitate wider acceptance and further development of the IEEE Std 1232-2010 Standard for Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE). To that end, we describe a tool named SAPPHIRE$^{\text{TM}}$, which includes an implementation of AI-ESTATE in Java and a corresponding GUI tool that supports model creation and diagnostic inference of the standard's Bayes Network Model (BNM). In addition, we describe extensions to the BNM as well as additional reasoner services that allow for representation and inference over dynamic Bayesian networks (DBNs) for standards-based prognostics.**

I. Introduction

The IEEE 1232-2010 Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE) standard [1] incorporates concepts specific to the diagnostic domain, facilitating portability and extensibility of diagnostic knowledge, and enabling the consistent exchange and integration of diagnostic capabilities. In addition, the standard defines interfaces among reasoners and diagnostic applications, test information knowledge bases, and more conventional databases [2].

Since its publication in 2010, we have been examining ways to promote and enhance AI-ESTATE, specifically in increasing its ability to provide a standards-based approach for performing online prognostic analysis. We describe an actual implementation of an AI-ESTATE-conformant model builder, client diagnostic application, and diagnostic reasoner for use with the AI-ESTATE BNM. Furthermore, we describe an implementation of extensions to the BNM for representing first-order DBNs to enhance the standard to support prognostic applications and realize the goals of Prognostics and Health Management (PHM) in a standards-based context. The implementation of AI-ESTATE with prognostic extensions forms the core of the software created for this project, called the Standards-based Analysis Platform for Predictive Health and Intelligent Reasoning Environment (SAPPHIRE).

II. Background

A. AI-ESTATE

In February 1990, the IEEE, seeing the need for a standardized way to incorporate artificial intelligence (AI) techniques into test and diagnosis applications, approved Project Authorization Request (PAR) 1232, authorizing the IEEE Standards Coordinating Committee 20 (SCC20) to begin development of AI-ESTATE [3]. Since then, the IEEE has continued development of AI-ESTATE for supporting diagnostic model exchange and for defining a standard application programming interface (API) for interacting with a diagnostic reasoner. The current version of AI-ESTATE, IEEE Std 1232-2010 [1], defines four semantic models using the EXPRESS information modeling language [4]. These models are fault trees, D-matrices, logic models, and Bayesian networks. The Extensible Markup Language (XML) schemata have been derived from these semantic models according to the Standards for the Exchange of Product model data (STEP) from the International Organization for Standardization (ISO) [5]. Furthermore, services have been defined for standardized interactions between a test application and a diagnostic reasoner. We demonstrated AI-ESTATE's model exchange and reasoner interoperability in 2008 [6] and 2009 [7], respectively. In our previous work investigating extensions to AI-ESTATE to address requirements for prognostics and health management, we showed how AI-ESTATE can incorporate gray-scale health information. We demonstrated the extensions by extending the AI-ESTATE Fault Tree Model to encode a fuzzy fault tree [8].

A previous version of the AI-ESTATE standard (consisting of IEEE Std 1232-1995 AI-ESTATE: Overview and Architecture [9], IEEE Std 1232.1-1997 AI-ESTATE: Data and Knowledge Specification [10], and IEEE Std 1232.2-1998 AI-ESTATE: Service Specification [11]), had been implemented previously under Phase I [12] and Phase II [13], [14] of an SBIR funded by the Air Force. The intent of the project was to demonstrate the feasibility of implementing AI-ESTATE in a component-based Automatic Test System (ATS), with the results summarized in [15]. Subsequently, under contract to the US Navy, preliminary versions of the recent standard were implement to demonstrate the validity of both model exchange and service interoperability, with results reported in [6] and [7] respectively. Here, we discuss implementation of the most current version of the standard along with extensions to the BNM and services for prognostics, in order to motivate prognostic extensions such as these to be eventually incorporated into the standard.

B. Bayesian Networks

1) Static Bayesian Network: Bayesian networks are probabilistic graphical models that use nodes and arcs in a directed acyclic graph to represent a joint probability distribution over a set of variables [16]. Formally, suppose that $P(\mathbf{X})$ is a joint probability distribution over n variables $X_1, \ldots, X_n \in \mathbf{X}$, with each variable X_i being represented by a node in the graph and $\mathbf{Pa}(X_i)$ denoting the parents of node X_i in the graph. The graph representation factors the joint probability distribution as:

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|\mathbf{Pa}(X_i)).$$

2) AI-ESTATE Bayes Network Model: The AI-ESTATE's BNM is a type of Bayesian network specific to diagnostics. In particular, the BNM defines a Bayesian network in which the nodes are comprised of BayesDiagnosis entities (with subtypes BayesFault and BayesFailure) and BayesTest entities. BayesDiagnosis entities are not conditionally dependent on anything else. BayesTest entities, however, can be conditionally dependent on BayesDiagnosis entities or even other BayesTest entities.

3) Dynamic Bayesian Network: The dynamic Bayesian network (DBN) expands on the Bayesian network formulation defined above by using a series of connected time-slices, each of which contains a copy of a regular Bayesian network $\mathbf{X}_t$ indexed by time t. The probability distribution of a variable at a given time-slice can be conditionally dependent on states of that variable (or even other variables) throughout any number of previous timesteps. In first-order DBNs, the nodes in each time-slice are not conditionally dependent on any nodes further back than the immediately previous time-slice. Therefore, the joint probability distribution for a first-order DBN factors as:

$$P(\mathbf{X}_0, \ldots, \mathbf{X}_k) = P(\mathbf{X}_0) \prod_{t=0}^{k} P(\mathbf{X}_{t+1}|\mathbf{X}_t).$$

Spanning multiple time-slices, the DBN can include any evidence gathered throughout that time and use it to help reason about state probability distributions across different time-slices. Often, the conditional probability tables of the DBN can be defined compactly by defining a prior network $\mathbf{X}_0$ and a single temporal network $\mathbf{X}_t$. The temporal network $\mathbf{X}_t$ is then "unrolled" into $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_k$ for k time-slices.

The DBN model has been used in many application areas that have sequences of observations. Often these problems are concerned with performing classification only in the current timestep based on prior evidence. However, DBNs can also be unrolled further and forecast future states based on the evolving marginal probabilities. Some predictive tasks that the DBN has been used for include clinical prognostics, such as [17] and [18], and mechanical prognostics, such as [19], [20], [21], and [22]. We show extensions to the AI-ESTATE BNM that allow for DBNs that are specific to diagnostics and prognostics.

III. The SAPPHIRE Software

In this section, we review the SAPPHIRE software package. This project was created over Phase II of a Navy-funded STTR whose objectives included development and demonstration of a concrete implementation of AI-ESTATE, as well as exploring extensions for standards-based prognostics.

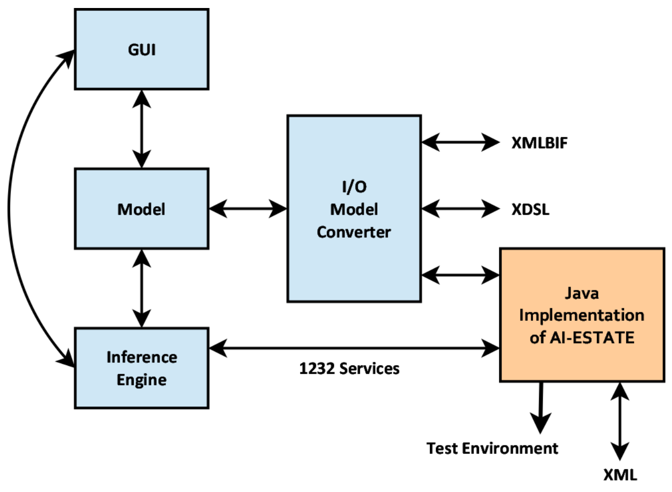


Fig. 1. Block diagram of SAPPHIRE software system.

A. Architecture

Generally, the architecture designed for this project is based on standard methodologies and widely used software libraries where possible. There are several advantages to this approach:

1) Standardization: Since many projects rely upon these popular libraries, a *de facto* standard is realized among the communities that use them.
2) Documentation: The popularity of the libraries provides a powerful impetus for the composers of such libraries to document usage and case scenarios in a detailed, accurate manner.
3) Performance: The libraries have a proven record among many entities in the software engineering field.

The software libraries in use include the Commons Math library [23] from the Apache Jakarta project and the Colt library from CERN [24], which are two math libraries that both enjoy a broad user base.

The architecture of our system is shown in Figure 1. The Model block represents the BNM diagnostic model. The Graphical User Interface (GUI) allows a user to create and modify the BNMs and interact with the Inference Engine. The Inference Engine runs Bayesian inference and reasoning on the Model as an AI-ESTATE-conformant diagnostic reasoner.

This system provides the framework to allow import and export (I/O) from many possible model formats by extending the codebase of the I/O Model Converter. This design permitted future extension to handle import and export of other common model file formats, such as the Interchange Format for Bayesian Networks (XMLBIF) [25] or the XDSL format used by the Structural Modeling, Inference, and Learning Engine [26]. The GUI tool uses the AI-ESTATE standard as the primary I/O format.

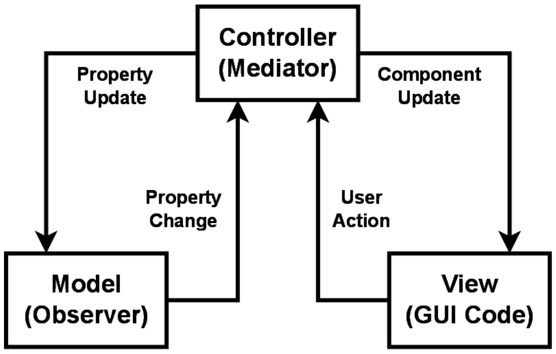
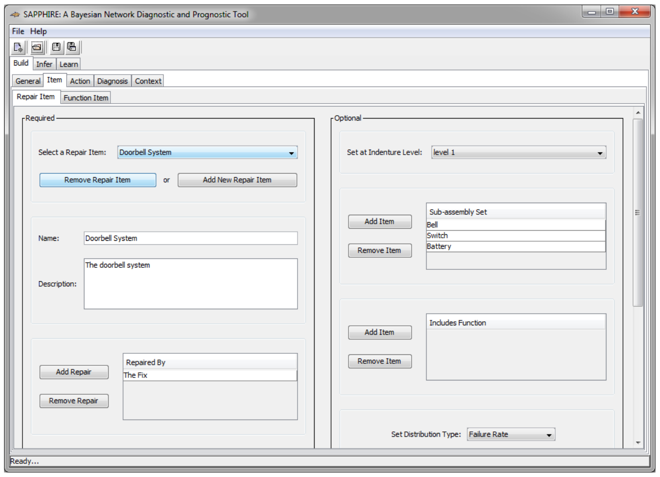


Fig. 2. Model-View-Controller (MVC) diagram.


SAPPHIRE: A Bayesian Network Diagnostic and Prognostic Tool
File Help
Build Infer Learn
General Item Action Diagnosis Context
Repair Item Function Item
Required
Remove Repair Item or Add New Repair Item
Add Repair
Remove Repair
Optional
Add Item
Remove Item
Add Item
Remove Item

Select a Repair Item: Doorbell System

Name: Doorbell System

Description: The doorbell system

Repaired By
The Fix

Set at Indenture Level: level 1

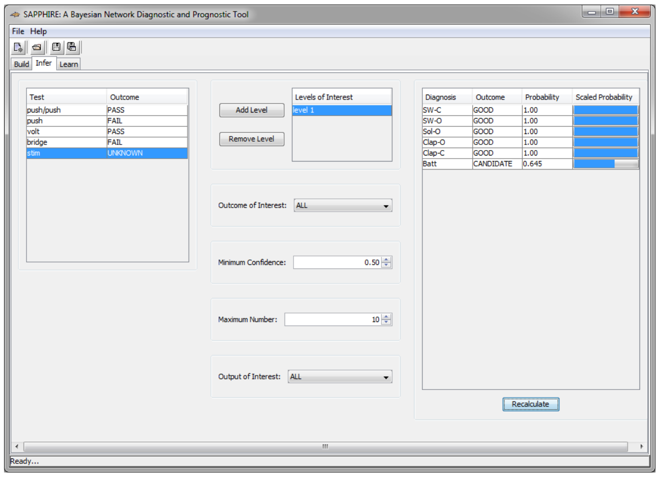
Sub-assembly Set
Bell
Switch
Battery

Includes Function

Set Distribution Type: Failure Rate

Ready...

Fig. 3. Screenshot of the Build tab in SAPPHIRE



Test	Outcome
push/push	PASS
push	FAIL
volt	PASS
bridge	FAIL
stim	UNKNOWN

Levels of Interest
level 1

Outcome of Interest: ALL

Minimum Confidence: 0.50

Maximum Number: 10

Output of Interest: ALL

Diagnosis	Outcome	Probability	Scaled Probability
SW-C	GOOD	1.00	
SW-O	GOOD	1.00	
Sol-O	GOOD	1.00	
Clap-O	GOOD	1.00	
Clap-C	GOOD	1.00	
Batt	CANDIDATE	0.645	

Ready...

Fig. 4. Screenshot of the Infer tab in SAPPHIRE

The architecture features the typical Model-View-Controller architectural pattern, shown in Figure 2. This architecture allows a decoupling between the data being modified (Model) and the interface which facilitates the modification (View), mediated by the Controller. Loosening the coupling between the model and view, which in this case represent our system's internal functionality and the front end to this software, respectively, facilitates concurrent development of both systems independently. This minimizes representation mismatches between the model and view, and as a result allows a smoother development process relatively free of the complexities associated with model and view integration.

B. Graphical User Interface

The GUI allows a complete BNM model to be created, with support for all the entities defined in the standard. This is a significant advancement over [6], which was a scaled-down version of the BNM to support basic diagnostics, but the GUI did not offer a way to define the complete set of features found in AI-ESTATE. A screenshot of the Build tab is shown in Figure 3. The subtabs organize the entities into their respective categories, and the split pane on each tab delineate between required and optional entities.

The GUI also features an interface to a diagnostic reasoner for performing inference over the loaded BNM. A screenshot of the Infer tab is shown in Figure 4.

C. Implementation of CEM and BNM

The SAPPHIRE software package, a hierarchical polymorphic design of the IEEE STD 1232-2010 AI-ESTATE standard, has been designed explicitly to mirror the inheritance of the EXPRESS model. The SAPPHIRE software was developed entirely in Java and utilizes a highly modular architecture to facilitate adaptation and expansion as the models are enhanced to support prognostics. System items (repair items and function items), actions (tests and repair actions), and diagnoses (faults and failures) can be specified in the model. The tool also supports entity (test and diagnosis) dependencies and their associated probability tables.

The abstract object Entity serves as the superclass of most objects in the hierarchy. An entity contains the name and description of the object (if relevant) as well as enforces methods for validation and exportation to XML as illustrated in Figure 5. Each object has appropriate access methods (e.g., get() and set()) for all object components. Each relevant object component has appropriate list management operations (e.g., add(), remove(), get()). The inverse relationships defined in the EXPRESS model are faithfully represented, but handled automatically by the parent object.

Multiple inheritance is a concept of object-oriented software engineering in which an object can have multiple parents. While EXPRESS supports multiple inheritance, Java does not. To support this functionality, multiple inheritance has been implemented as a series of interfaces in Java. For example, the entity BayesFault has parents BayesDiagnosis and Fault. Both BayesDiagnosis and Fault have the same parent class, Diagnosis. In SAPPHIRE, BayesFault has been implemented as a subclass of Fault and therefore will inherit all public and protected methods and variables from the Diagnosis entity. BayesFault also implements the interface BayesianDiagnosis, which forces implementation of all methods in the EXPRESS entity BayesDiagnosis.

Entity	
name:	NameType
description:	DescriptionType
id:	UUID
+validate()	
+toXML()	
+toXMLref()	

Fig. 5. Object diagram of Entity class.

The SAPPHIRE software package includes a complete implementation of the Common Element Model (CEM) and the Bayes Network Model (BNM). All objects have exhaustive Javadoc comments in the code so that, when this tool is released, users can easily follow the design through the API and readily understand the correspondence to each class's counterpart in the EXPRESS model. This Javadoc documentation includes the commentary given in the EXPRESS model but has also been expanded and revised as it relates to SAPPHIRE's particular implementation of the standard.

D. Implementation of DCM and Services

We have also encoded the EXPRESS entities specified by the Dynamic Context Model (DCM) and the Reasoner Service Model (RSM) into SAPPHIRE. The XML import/export functionality, including validation, has been implemented for the DCM so that session histories can be saved.

The service API makes use of the RSM and DCM entities to allow the reasoning functions and procedures defined by AI-ESTATE to be executed on many different systems. These functions provide feedback to the client and make changes to DCM entities as specified in the standard. Currently, the Bayesian network inference engine developed concurrently with SAPPHIRE supports the following AI-ESTATE services as part of its API:

- initializeDiagnosticProcess: This function starts a diagnostic session for a given system under test and returns the new session.
- loadDiagnosticModelFromLocation: This procedure is used to load a Diagnostic Model that can then be used by the Reasoner.
- setActiveModel: This procedure activates the given Diagnostic Model so that it can be used to obtain diagnostic results.
- applyActions: This procedure informs the Reasoner of any actions that have been performed along with any outcomes that have been observed or costs that have been incurred.
- getDiagnosticResults: This function obtains a list of diagnostic conclusions from the inference engine and returns a subset of these conclusions to the client based on a set of input parameters.

These are sufficient for performing a basic diagnostic session with a BNM. All other functions required for claiming AI-ESTATE conformance will be implemented as part of the service API in the near future.

E. EXPRESS and XML Validation

Through inheritance from the Entity superclass, each object implements a validate() method. This method validates the particular entity against the standard, the UNIQUE and WHERE clauses of the EXPRESS specification. For example, the validate() method is overridden in the BayesTest and BayesDiagnosis entities to ensure that their probability distributions sum to one. For enforcing unique name constraints, the constructor of the Entity superclass registers the name of each object with a UniquenessConstraint class, which handles the uniqueness checking across all entities.

The tool imports valid XML files according to the Bayesian Network XML schema specified by AI-ESTATE. The validation portion of this functionality has been tested to ensure that it finds errors in non-valid XML files. This feature is also used to test the correctness of the XML export functionality.

The EXPRESS constraint validation for the BNM, CEM, and DCM entities ensures that exported models and diagnostic session traces do not violate any of the formal propositions specified by the AI-ESTATE standard. Each of these propositions describes a specific constraint, such as the requirement that the structure of Actions and their subActions corresponds to a directed acyclic graph or the requirement that the sum of all probabilities given its dependent configuration results in a value of one. This functionality is crucial because XML validation alone cannot ensure that these constraints are met.

F. Import and Export Functionality

Whereas the reasoner side of the GUI tool maintains an internal Bayesian Network representation for efficient inference, the functionality of the tool to import and export to XML that conforms to the AI-ESTATE schemas is essential for claiming AI-ESTATE conformance. The tool must be able to load and validate BNM XML originating from AI-ESTATE-conformant applications as well as generate valid BNM XML that can be loaded into these other applications.

1) Import: Figure 6 shows the sequence for importing and validating an AI-ESTATE BNM file into the BNM GUI tool.

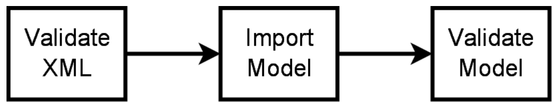


Fig. 6. Model validation sequence for import.

The tool uses a third-party library for validating XML against the AI-ESTATE schema. This is done by comparing the schema against an XML file. Therefore, on import, the tool first checks the XML file against the schema. The XML is then imported into the SAPPHIRE representation. Then the EXPRESS constraints are validated against the SAPPHIRE model.

2) Export: Each object implements two methods: toXML() and toXMLref(). These methods convert the content of the object into an XML Element or Attribute, depending on the nature of the entity, and depending on if the full XML specification is required or merely the reference to the XML element. The components of any object will themselves also have these XML methods, and therefore an object can be converted to XML format through bottom-up recursion. This tool exports valid models to XML files according to the standards based XML schema.

Figure 7 shows the sequence for exporting and validating an AI-ESTATE BNM file from the BNM GUI tool.

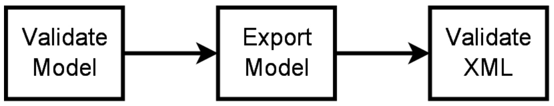


Fig. 7. Model validation sequence for export.

The order is the opposite from that of import. In this case, the EXPRESS constraints are checked first. After this, the XML is generated and written to a file. The third-party library is then used to check whether this XML validates against the schema.

G. Diagnostic Bayesian Reasoner

The SAPPHIRE software interfaces with a diagnostic Bayesian reasoner. While the reasoner supports fully general Bayesian network inference, it has been specialized in SAPPHIRE to work with the BNM to perform diagnostics and prognostics (as discussed in the next section). Specifically, the reasoner offers the following inference algorithms.

- Variable Elimination [27]: an exact inference algorithm based on successively marginalizing out variables.
- Likelihood Weighting [28]: a particle-based approximate inference algorithm that weights particles based on their likelihood.
- Gibbs Sampling [29]: a particle-based approximate inference algorithm as a Markov chain Monte Carlo (MCMC) method.
- Bootstrap Filter [30]: a particle-based approximate inference algorithm designed specifically for performing inference over dynamic Bayesian networks without having to unroll the network before performing inference.

IV. Extensions to AI-ESTATE for Prognostics

In this section we describe specific changes that were made to the standard to allow for DBNs. These changes required the modification of several EXPRESS information models, which resulted in new XML schemata. We incorporated them into SAPPHIRE under the assumption they will eventually be included in the standard. This will also enable us to provide the results of our implementation as a demonstration and validation of the proposed changes to support future adoption, which is one of the objectives of this project.

Specifically, several modifications were made to the BNM, CEM, DCM, and RSM. We have created revised information models and the corresponding XML schemata incorporating proposed extensions to AI-ESTATE in support of prognostics.

The EXPRESS-G for the updated BNM is shown in Figures 8 and 9. We introduced temporal links from diagnosis to diagnosis (of the various types) and from test to test. These temporal links differ from previously existing links between DependentElements in that self-referencing links may occur. Currently, we do not permit temporal links between entities of different types; however, this is being consider as a future enhancement. Two attributes, timeInterval and intervalUnit, were introduced to the BayesNetworkModel entity as shown by the EXPRESS in Figure 10. These represent the time elapsed between adjacent timesteps in the DBN. The additional constraint consistentTime ensures that these are attributes are set together or not at all, in the case of static Bayesian networks.

In the CEM, we extended the Outcome entity to allow for multiple user-defined OutcomeValues. This is shown by the new attribute userName in the Outcome entity as shown by the EXPRESS in Figure 11. The additional manyNames constraint forces each user-defined OutcomeValue to be given a custom name.

In the DCM, the ActualOutcome entity has an additional optional TimeStamp attribute named timePredicted, referencing the timestep of the particular Diagnosis, Test, or Action outcome. This allows Outcomes to be tied to specific timesteps in the unrolled DBN.

Several changes to the RSM were also required to allow for DBN support. We extended the ActualAction entity in the RSM to include a time interval over which the specified outcomes are valid, as shown below:

```
timePerformed : OPTIONAL TimeValue;
```

It is assumed that, if no time interval is specified, the values are always valid. If no upper limit is specified, the validity persists unless changed.

For the services, the applyActions service was modified to include a sequence of time-series test results with associated validity intervals. A new setTime service enables the baseline time for analysis to be established. This would be done through the ActualUsage entity in the DCM. However, it may be possible to specify the baseline time for analysis through the existing setUsage service.

To support inference, we defined a getFutureDiagnosticResults service. This service returns a hypothesis for a diagnostic state at a specified number of time steps from the current time associated with the current Step in the DCM. In the future, a getTimeToFail service will also be added. This service will be used to estimate the time to failure for a specified diagnosis in the model. If no diagnosis is specified, the service will return the time to fail estimate of the failure expected to occur first. This returned time could be used with getFutureDiagnosticResults to find the corresponding diagnosis.

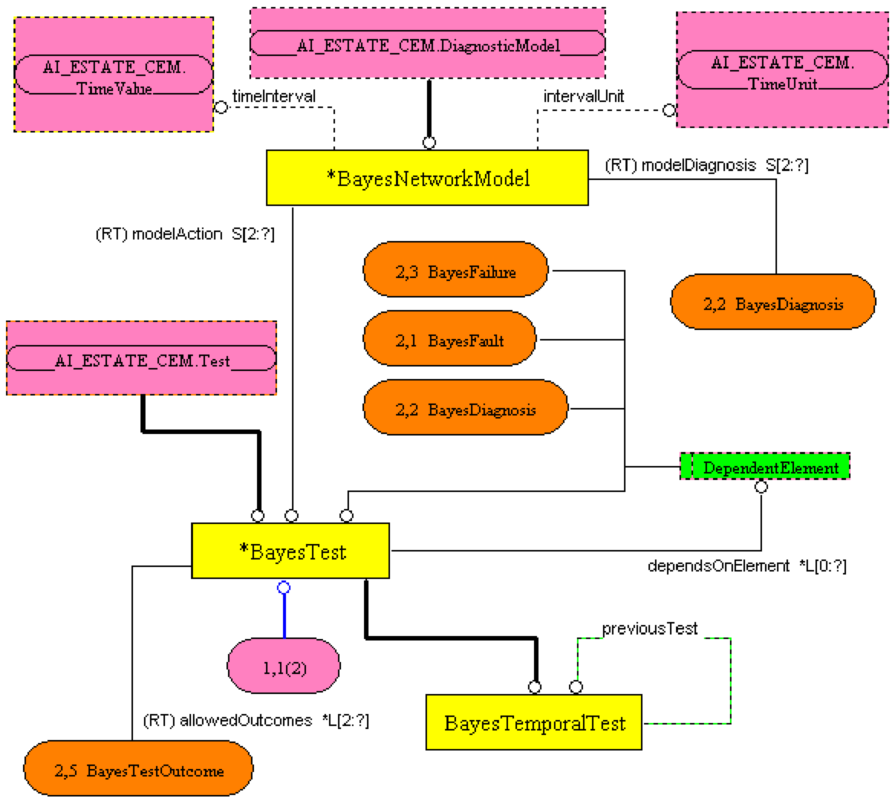


Fig. 8. EXPRESS-G with new BayesTemporalTest entity and BayesNetworkModel entity incorporating time.

V. Future Work

While SAPPHIRE provides a complete representation of the CEM, BNM, DCM, and RSM, the accompanying BNM diagnostic reasoner has yet to implement some of the services defined in the standard. These services include those involving sending or requesting information beyond test or diagnosis outcomes (such as setDiscrepancies and recommendActions) and services for diagnostic session control (pause and resume, disaster recovery, etc.). All of the services defined in the standard will be implemented in the near future.

The client diagnostic application and diagnostic inference engine presented here allow test results to be updated and inference to be drawn about probable diagnoses. However, the responsibility of performing tests and reporting the test results to the client application is left entirely up to the user. Thus, an important direction for future work is to deploy the AI-ESTATE-conformant client diagnostic application and the diagnostic reasoner, integrating them with actual Automated Test Equipment (ATE) in a real-life and real-time setting.

Another direction for future work is to implement and integrate other standards for test and diagnosis into SAPPHIRE, such as IEEE Std 1636.2 Software Interface for Maintenance Information Collection and Analysis (SIMICA): Maintenance Action Information (MAI) [31] and IEEE Std 1636.1 Software Interface for Maintenance Information Collection and Analysis (SIMICA): Exchanging Test Results and Session Information via the Extensible Markup Language (XML) (Test Results) [32]. This would allow for tighter, standards-based integration across the complete process for testing and maintenance. To that end, we could also look into how SAPPHIRE and Montana State University's Bobcat knowledge discovery toolset [33] could be integrated together to allow for greater coupling between the diagnostic and maintenance processes, such as is needed for model maturation.

VI. Conclusion

In this paper, we described an actual implementation in Java of the most current version of the AI-ESTATE standard. Standards are not intended to stand alone, and their purpose is realized when concrete applications are created that conform to the standard. Therefore, this paper seeks to facilitate wider acceptance and conformance to AI-ESTATE for the exchange of standardized diagnostic models and their use with standardized interfaces with diagnostic reasoners.

Furthermore, we presented extensions for BNMs to represent first-order DBNs, further enhancing the standard to support prognostic applications. In standardization efforts, existing applications should drive the requirements, rather than standards requirements drive the applications. Considerable work is being performed in developing new and better applications for PHM. Ideally, standardization efforts should accompany advances in PHM technology. Thus, these applications are demonstrating the need for diagnostic standards to further incorporate prognostic capabilities such as the one presented here.

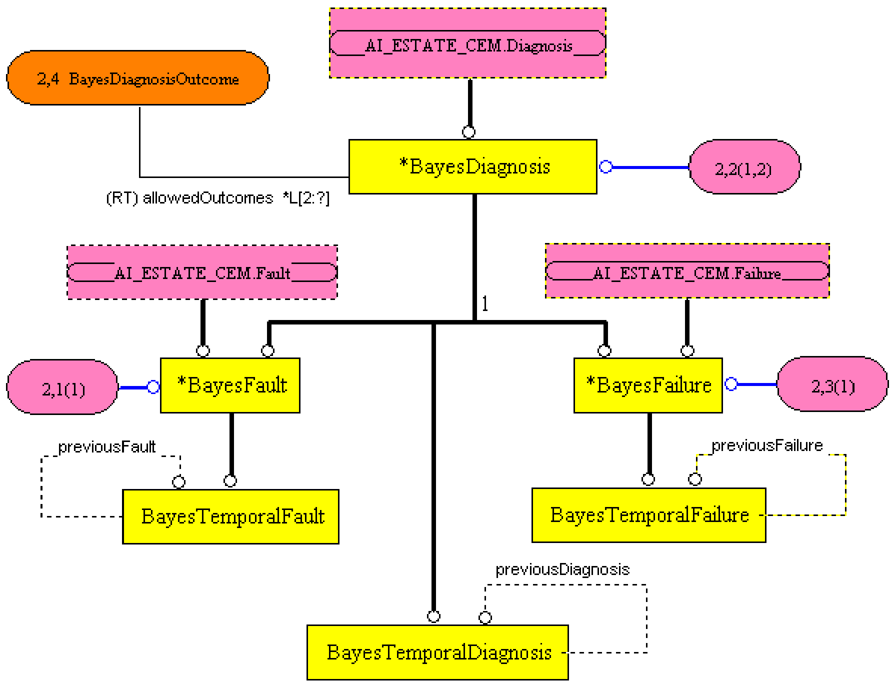


Fig. 9. EXPRESS-G with new BayesTemporalFault, BayesTemporalFailure, and BayesTemporalDiagnosis entities.

```
ENTITY BayesNetworkModel
  SUBTYPE OF (DiagnosticModel);
    SELF\DiagnosticModel.modelAction :
               SET [2:?] OF BayesTest;
    SELF\DiagnosticModel.modelDiagnosis :
               SET [2:?] OF BayesDiagnosis;
    timeInterval : OPTIONAL TimeValue;
    intervalUnit : OPTIONAL TimeUnit;
  WHERE
    consistentTime :
     (EXISTS(SELF.timeInterval)
     AND EXISTS(SELF.intervalUnit))
     XOR (NOT(EXISTS(SELF.timeInterval))
     AND NOT(EXISTS(SELF.intervalUnit)));
END_ENTITY;
```

Fig. 10. EXPRESS for BayesNetworkModel entity incorporating time.

```
ENTITY Outcome
  ABSTRACT SUPERTYPE OF(ONEOF(Diagnosis-
   Outcome, TestOutcome, ActionOutcome));
    maxConfidence :
            OPTIONAL ConfidenceValue;
    allowedValue : OutcomeValues;
    userName : OPTIONAL NameType;
  WHERE
    manyNames : ((SELF.allowedValue =
                             UserDefined)
    AND EXISTS(SELF.userName))
    XOR (NOT(SELF.allowedValue =
                             UserDefined)
    AND NOT (EXISTS(SELF.userName)));
END_ENTITY;
```

Fig. 11. EXPRESS for Outcome entity with multiple user-defined outcomes.

Acknowledgments

This project was supported as an STTR under US Navy contract N68335-11-C-0506. We thank Mike Malesich and Jennifer Fetherman for the discussions leading to developing the proposed extensions to this standard. We also thank Patrick Kalgren and John Gorton of Impact Technologies/Sikorsky Innovations for their collaboration with this STTR.

References

[1] IEEE Std 1232-2010, *IEEE Standard for Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE)*. Piscataway, NJ: IEEE Standards Association Press, 2010.

[2] J. Luo and Z.-Z. Su, "Design and implementation of intelligent diagnostic system based on AI-ESTATE," in *Information and Computing (ICIC), 2011 Fourth International Conference on*. IEEE, 2011, pp. 237–240.

[3] L. A. Orlidge, "An overview of IEEE P1232 AI-ESTATE. The standard for intelligent reasoning based systems test and diagnosis arrives," in *AUTOTESTCON'96, Test Technology and Commercialization. Conference Record*. IEEE, 1996, pp. 61–67.

[4] ISO 10303-11:1994, *Industrial Automation Systems–Product Data Representation and Exchange–Part 11: The EXPRESS Language Reference Manual*. Geneva, Switzerland: The International Organization for Standardization, 1994.

[5] ISO 10303-28:2007, *Industrial Automation Systems–Product Data Representation and Exchange–Part 28: XML Representation of EXPRESS Schemas and Data Using XML Schemas*. Geneva, Switzerland: The

International Organization for Standardization, 2007.

[6] J. Sheppard, S. Butcher, P. Donnelly, and B. Mitchell, "Demonstrating semantic interoperability of diagnostic models via AI-ESTATE," in *Proceedings of the IEEE Aerospace Conference*, 2009, pp. 1–13.

[7] J. Sheppard, S. Butcher, and P. Donnelly, "Demonstrating semantic interoperability of diagnostic reasoners via AI-ESTATE," in *Proceedings of the IEEE Aerospace Conference*, 2010, pp. 1–10.

[8] P. J. Donnelly, L. E. Sturlaugson, and J. W. Sheppard, "A standards-based approach to gray-scale health assessment using fuzzy fault trees," in *AUTOTESTCON, 2012 IEEE*. IEEE, 2012, pp. 174–181.

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

[10] 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, 1997.

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

[12] A. Giarla, "Implementing AI-ESTATE in a component based architecture. Phase-I," in *AUTOTESTCON '99. IEEE Systems Readiness Technology Conference, 1999. IEEE*, 1999, pp. 27–33.

[13] A. J. Giarla and W. L. Simerly, "Implementing AI-ESTATE in a component based architecture, Phase-II," in *AUTOTESTCON Proceedings, 2000 IEEE*. IEEE, 2000, pp. 438–450.

[14] ——, "Implementing an AI-ESTATE based diagnostic engine component," in *Digital Avionics Systems Conference, 2000. Proceedings. DASC. The 19th*, vol. 2. IEEE, 2000, pp. 6B3–1.

[15] J. W. Sheppard and A. J. Giarla, "Information-based standards and diagnostic component technology," in *AUTOTESTCON Proceedings, 2000 IEEE*. IEEE, 2000, pp. 425–433.

[16] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[17] M. A. van Gerven, B. G. Taal, and P. J. Lucas, "Dynamic Bayesian networks as prognostic models for clinical patient management," *Journal of Biomedical Informatics*, vol. 41, no. 4, pp. 515–529, 2008.

[18] K. Exarchos, G. Rigas, Y. Goletsis, and D. Fotiadis, "Towards building a Dynamic Bayesian Network for monitoring oral cancer progression using time-course gene expression data," in *Information Technology and Applications in Biomedicine (ITAB), 2010 10th IEEE International Conference on*, nov. 2010, pp. 1 –4.

[19] F. Camci and R. Chinnam, "Dynamic Bayesian networks for machine diagnostics: hierarchical hidden Markov models vs. competitive learning," in *Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on*, vol. 3, July-4 Aug. 2005, pp. 1752–1757 vol. 3.

[20] A. Muller, M.-C. Suhner, and B. Iung, "Formalisation of a new prognosis model for supporting proactive maintenance implementation on industrial system," *Reliability Engineering & System Safety*, vol. 93, no. 2, pp. 234 – 253, 2008.

[21] M. Dong and Z.-b. Yang, "Dynamic Bayesian network based prognosis in machining processes," *Journal of Shanghai Jiaotong University (Science)*, vol. 13, pp. 318–322, 2008.

[22] K. Medjaher, J.-Y. Moya, and N. Zerhouni, "Failure prognostic by using dynamic Bayesian networks," *Dependable Control of Discrete Systems.*, vol. 1, pp. 291–296, 2009.

[23] Commons Math: The Apache Commons Mathematics Library. [Online]. Available: http://commons.apache.org/math/

[24] Colt. [Online]. Available: http://acs.lbl.gov/software/colt/

[25] F. G. Cozman, "The interchange format for Bayesian networks," *http://www.cs.cmu.edu/afs/cs/user/fgozman/www/Research/Interchange Format*, 1998.

[26] M. J. Druzdzel, "SMILE: Structural Modeling, Inference, and Learning Engine and GeNIe: a development environment for graphical decision-theoretic models," in *Proceedings of the 16th National Conference on Artificial Intelligence and the 11th Innovative Applications of Artificial Intelligence Conference*, ser. AAAI '99/IAAI '99, 1999, pp. 902–903.

[27] B. D'Ambrosio, "Inference in Bayesian networks," *AI magazine*, vol. 20, no. 2, p. 21, 1999.

[28] R. D. Shachter and M. A. Peot, "Simulation approaches to general probabilistic inference on belief networks," in *Uncertainty in artificial intelligence*, vol. 5, 1989, pp. 221–231.

[29] R. M. Neal, "Probabilistic inference using Markov chain Monte Carlo methods," Technical Report CRG-TR-93-1, University of Toronto, 1993.

[30] N. J. Gordon, D. J. Salmond, and A. F. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," in *IEE Proceedings F (Radar and Signal Processing)*, vol. 140, no. 2. IET, 1993, pp. 107–113.

[31] IEEE Std 1636.2-2010, *IEEE Trial-Use Standard for Software Interface for Maintenance Information Collection and Analysis (SIMICA): Exchanging Maintenance Action Information via the Extensible Markup Language (XML)*. Piscataway, NJ: IEEE Standards Association Press, 2010.

[32] IEEE P1636.1-2007, *IEEE Trial Use Standard for Software Interface for Maintenance Information Collection and Analysis (SIMICA): Exchanging Test Results and Session Information via the eXtensible Markup Language (XML)*. Piscataway, NJ: IEEE Standards Association Press, 2007.

[33] S. Strasser, E. Howard, and J. Sheppard, "An integrated toolset for ontology-guided diagnostic knowledge discovery," in *AUTOTESTCON, 2012 IEEE*. IEEE, 2012, pp. 280–290.