Enhancing Simulation-Based Training with Performance Measurement Objects

Webb Stacy, Ph.D., Jared Freeman, Ph.D.
Aptima, Inc.
Woburn, MA & Washington, DC
wstacy@aptima.com, freeman@aptima.com

Stephanie Lackey, Danielle Merket
NAVAIR Training Systems Division
Orlando, FL
stephanie.lackey@navy.mil, danielle.merket@navy.mil

ABSTRACT

Current simulation object models are optimized to support practice, not training. Training requires support for human performance measurement and feedback. This, in turn, requires that simulation object models give first class status to data concerning human performance. We define Performance Measurement Objects (PMO), which represent actors, behavioral data, and measurement methods. PMOs support the real-time requirements of intelligent agents, human observer/instructors, and distributed performance assessment processors.

ABOUT THE AUTHORS

Webb Stacy, Ph.D. is Vice President of Technology at Aptima. He oversees Aptima's current and future technology portfolios. His focus is the intersection of software and computer science with the science, modeling, and measurement of warfighters as individuals and as teams.

Jared Freeman, Ph.D. is a Principal Cognitive Scientist at Aptima. He specializes in the application of cognitive science techniques and technologies to analyzing the decision making of warfighters and improving their performance through training and decision aids.

Stephanie Lackey is a Computer Engineer in the Science and Technology Division of the Naval Air Systems Command, Training Systems Division (NAVAIR TSD) in Orlando, Florida. She serves as Principal Investigator (PI) for the CGF evaluation effort within the Virtual Technologies and Environments program, and as Co-PI for the Intelligent Training Support Tools (ITST) portion of the Air Warfare Training Development (AWTD) program. Stephanie earned her M.S. degree in Industrial Engineering and is pursuing her Ph.D. in the same discipline.

Danielle Merket is a Research Psychologist in the Science and Technology Division of NAVAIR TSD. She serves as Team Lead for Performance Measurement and After Action Review for the Navy Aviation Simulation Master Plan and as the Co-PI for the ITST effort within AWTD. Danielle holds a M.S. degree in Industrial and Organizational Psychology.

Enhancing Simulation-Based Training with Performance Measurement Objects

Webb Stacy, Ph.D., Jared Freeman, Ph.D.
Aptima, Inc.
Woburn, MA & Washington, DC
wstacy@aptima.com, freeman@aptima.com

Stephanie Lackey, Danielle Merket
NAVAIR Training Systems Division
Orlando, FL
stephanie.lackey@navy.mil,
danielle.market@navy.mil

INTRODUCTION

Simulator-based practice exposes warfighters to realistic events and provides them with realistic tools with which to respond. Simulator-based training, on the other hand, consists of well-structured practice that is measured and supplemented with feedback.

Current simulation object models are designed to support practice. They faithfully represent the state of the physical scenario entities – sensors, weapons, platforms – with which warfighters interact.

However, current object models do not explicitly represent human performance data that are essential for measurement and feedback. These object models are not optimized for training applications. Human observer/instructors typically compensate for this by gathering data manually or writing custom software to infer performance from changes in entity state.

This effort implemented an object model that directly supports training: Performance Measurement Objects (PMOs).

PMO are software objects in a federated High Level Architecture (HLA) simulation and training environment. They are designed to support distributed observers and performance measurement systems as well as automated intelligent agents like synthetic teammates, synthetic enemies, or artificial instructors that require data concerning trainee proficiency in order to adapt their actions and feedback.

In our current implementation for a multi-platform air warfare simulator (E2-C and F/A-18), there are two branches of the PMO class hierarchy:

- *Actors* (e.g., trainees, teams, and instructors), their assignments, and their organizational relationships;
- *Observations*, tied to one or more actors, that include a description of the data sources and computation methods as well as performance data.

In this paper, we elaborate the essential role of human performance measurement in training systems and we describe PMOs and their application in a multi-platform training simulator.

MEASURING TEAM PERFORMANCE

A recent body of team research has developed measures that give considerable insight into team performance and mission outcomes. In an initial stage of the current work, we reviewed much of this literature and crafted a conceptual framework that defines factors that influence team coordination and demonstrate its effects.

Representative literature concerning team performance includes the work of Serfaty and colleagues (Entin & Serfaty, 1999; MacMillan, Entin, and Serfaty, 2004), who developed and validated measures of the *team structure, technology, and process* as well as aspects of *team coordination* such as communication and predictors of coordination such as team member awareness of team state. Smith-Jentsch, Zeisig, Acton, and McPherson (1998) developed four classes of *team coordination* measures (communication, information exchange, supporting behavior, and initiative/leadership) and validated these in studies of more than 100 Navy teams. Alberts and colleagues (Alberts, Garstka, Hayes, & Signori, 2001) have proposed a focus on *mission process* measures, such as synchronization of forces. Traditional measures of *mission effects* – such as kill and loss ratios – remain important in assessing teamwork. A wealth of additional literature also documents measures of teamwork, its entailments, and its effects (e.g., MacMillan, et al., 2002; Freeman, et al., 1997; Freeman, et al., 2003; Johnston, Serfaty, and Freeman, 2003; Freeman and Serfaty, 2002).

We developed a qualitative model (or framework) that builds on these foundations in research (see Figure 1). The function of this framework is to define the space of measures that our software product must eventually handle. From this space, we selected a representative sample of measures to exercise and demonstrate in each build of performance measurement software we

are developing. The framework describes the factors that influence mission outcomes and their major relationships – from the capabilities of individuals and their assets, through team architectures and their coordination requirements, to impacts on mission process and outcomes.

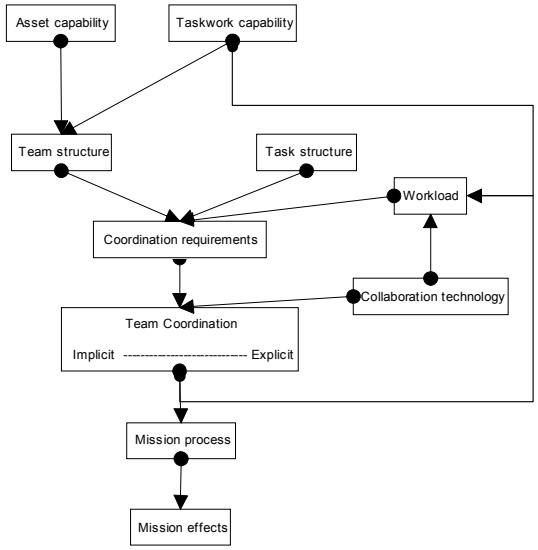

Figure 1. Team Coordination Framework.

To support performance measurement within this framework, we developed a system called Intelligent Agents for Performance. Its architecture is illustrated (see Figure 2). The constituent agents have the following functionality: (Agents marked "*" are being implemented in this project. Many of the remainder are being developed in our other efforts.)

- The *Performance Measurement Agent** is responsible for configuring the system for the measures to be collected on a mission and for arranging for them to be computed during After Action Review (AAR). It has several subagents.
- The *Data Capture Agent** listens for and captures relevant data on the HLA bus.
- The *Domain Expert Agent* allows subject matter experts (human or synthetic) to provide expert solutions and assessments of trainee performance.
- The *Survey Agent* captures scores from observer-recorded measurement instruments such as behaviorally anchored rating scales.
- The *Other Data Sources Agent** captures data from other sources, such as strike plans and scripted HLA events.
- The *Diagnosis, Coaching, and Debriefing Agents* serve the obvious pedagogical functions.
- The *Performance Data Store** stores relevant data provided by other agents in the system.
- The *Process Visualization Agent** helps visualize the internal workings of the system.

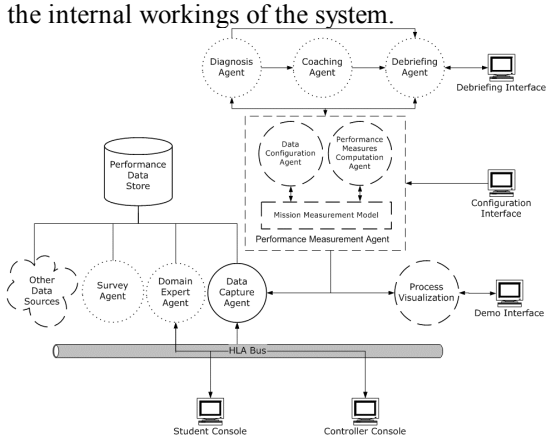

Figure 2. Architecture of the Intelligent Agents System.

PERFORMANCE MEASUREMENT OBJECTS

Communication among agents is accomplished via a set of objects called Performance Measurement Objects (PMOs). PMOs are software objects that can be used for performance measurement in a federated HLA simulation and training environment. In the HLA environment, simulation and federation objects are typically aspects of or information about the physical environment (e.g. aircraft, ships, missiles, radars, bombs, and targets). PMOs expand the world of objects in the HLA federation to include people and things in the training environment as well: students, instructors, teams, assignments, performance measurements, and so on.

For this project, PMOs were implemented as local objects, used by the Performance Measurement Agent and its Subagents as well as the Debriefing Agent, and to be used for communication with the Coaching and Diagnosis Agents when those agents are implemented. As such, PMOs provide several benefits:

- A clear, exact expression of measures.
  - The separation of performance from operationalization means that new ways of measuring the same performance can be easily substituted.
  - Clarity of individual and team assignments and measures, especially the differences between them.
- A framework for inter-agent communication about a wide spectrum of measures that is comprehensible, maintainable, and easily and gracefully extensible.
- A foundation for sophisticated mission measurement configuration by instructor/operators and automated agents, ranging from the current text

file configuration to a drag-and-drop configuration GUI to a capability of being automatically configured by other agents via a PMO-oriented Application Programming Interface (API).

When published on the HLA bus, PMOs become a Simulation Object Model (SOM), though the "simulation" might be provided by real people (e.g. trainees, instructors, domain experts) rather than an automated simulation during the mission. PMOs as a SOM are useful for the following reasons:

- Communication: They are a basis for communication among federates about synthetic team members or adversaries.
- Dynamism: They are a basis for missions where training objectives and scenarios are steered dynamically during the mission depending on trainee performance.
- Distribution: The Intelligent Agents system is designed to operate in the distributed environment of HLA federations, and therefore there is normally no reason for the Intelligent Agents system itself to be distributed. However, special circumstances such as the need for specialized measurements, load balancing, or other local or organizational convenience factors may dictate more than one Intelligent Agents system in a given federation. In these cases, PMOs-as-SOM will provide a basis for distributed performance measurement systems communicating over the HLA bus.
- Perspective: Having PMOs on the HLA bus underscores the point that performance measurement is a key aspect of effective and efficient training.

There are two top-level classes of PMOs: Training Entities and Observations. Training Entities, as the name implies, are objects from the world of simulation and training such as trainees, instructors, and observers. In addition, Training Entities include Tasks and Structure. A notable subclass of Structure is Assignment, which is a set of actor-task pairs. Figure 3 shows partial detail of this class hierarchy.

The other top-level class of PMOs is Observations. Observations describe the data and measures collected for a Training Entity or set of Training Entities, and include a description of the data source as an instance of Class DataSource, a description of the data itself, and an array of actual data points. Instances of Observation are used to configure the Mission Measurement Model and to inform the Data Capture Agent about relevant objects when the HLA bus is the data source. After the mission is run, those instances are populated from the Data Store and used to compute the requested performance measures. Figure 4 shows partial Observation and DataSource hierarchies.

An example may clarify this abstract description. One measure of taskwork in an E2-C training scenario is the time taken by a trainee Radar Officer (RO) to identify a track on the radar screen. In the PMO class hierarchy, the trainee is an instance of class RO, the job of identifying a radar track is an instance of class Task, and one case of track identification by the trainee is an instance of class Assignment, which pairs the RO instance with the Task instance.

```
Training Entity
  Actor
    Trainee
      E2C Trainee
        Combat Information
             Center Officer (CICO)
        Air Combat Officer (ACO)
        Radar Officer (RO)
      F/A-18 Trainee
        Pilot
        Navigator
      Instructor
      Team
    Task
    Structure
      Assignment
      Team Structure
      Task Dependency
```

Figure 3. Partial Training Entity Class Hierarchy.

```
Observation
  ValueObservation
    SyncRelativeTimelineObservation
    SyncTimelineWindowObservation
    TargetDestroyedObservation
  PairObservation
    IDLatencyObservation
    SyncTimelineObservation
DataSource
  FOMObject
    MunitionDetonationSource
    Link16TrackIdentificationSource
    TrackDestroyedSource
  OtherDataSourceObject
    TrackAppearanceSource
    StrikePlanSource
```

Figure 4. Partial Observation and DataSource Class Hierarchies.

The Observation instance associated with measuring latency has two instances of class DataSource. One is the mission script, which determines the timing of the appearance of the track. The other is an HLA Track object that announces that the trainee has identified the track. The difference between the timestamps of

these two DataSource instances determines a latency representing the time to identify the track. The Observation instance for this particular RO will in general contain many such differences.

PMOs IN ACTION

Our target testbed involved an E2-C crew working with a fourship of F/A-18s. To demonstrate the prototype, and particularly PMOs, in action, we created a scenario involving several plausible performance measures for this team: The main purpose of the demonstration was to show the variety of measures that can be addressed by the architecture and PMOs. The scenario was therefore reasonably brief and simple.

1. *Individual tasking*. The selected measure was for E2-C crew members, namely, the time to ID a track. This is the example measure just described.
2. *Team Coordination*. Two measures were selected, both of them concerning the fourship of F/A-18s. For the first one, individual and team temporal accuracy was measured in a strike plan that required each plane in a fourship to strike a target exactly one minute apart, measured by computing the difference between the time of bomb impact between aircraft. For the second measure, team temporal accuracy was measured in a strike plan that required each plane in a fourship to strike a target within a four minute window. TrainingEntity PMOs involved were an instance of RO for the E2-C, four instances of Pilot for the F/A-18, and one instance of Team whose members were the F/A-18 pilots. All had instances of Assignment. Instances of Observation included as DataSources both the simple strike plan and timestamps on the HLA Mission Detonation Objects.
3. *Mission Effects*. Here, the measure was simply the damage done to targeted objects. TrainingEntities were four instances of Pilot, and Observation instances were included as DataSources HLA Entity Objects with state 'destroyed.'

There are three phases for using the prototype: Pre-mission, Mission, and After-Action Review (AAR). Specific functionality is available during each phase.

During the Pre-mission phase, the user configures the system for specific performance measures. In the system as implemented, this configuration is based on the contents of a text file and the Instructor/Operator's request in a simple prototype interface. Based on this information, the system is configured to listen for the relevant data on the HLA bus. Also in the Pre-mission phase, other available and relevant data such as strike plans and preplanned HLA events are entered. For the Pre-mission phase of this scenario, the Data Capture Agent was configured to listen for the relevant HLA objects and the simple strike plan was entered into the Other Data Sources Agent.

During the Mission phase, relevant HLA data is captured from the HLA bus and stored for later use. HLA data not relevant to the configured performance measures are ignored.

During the AAR phase, the Instructor/Operator requests that some or all of the configured measures be computed and displayed, and the system uses the data entered during the Pre-mission phase and captured during the Mission phase to compute and display those measures. Figure 5 shows the prototype measures request screen, and Figure 6 shows the result of the request.

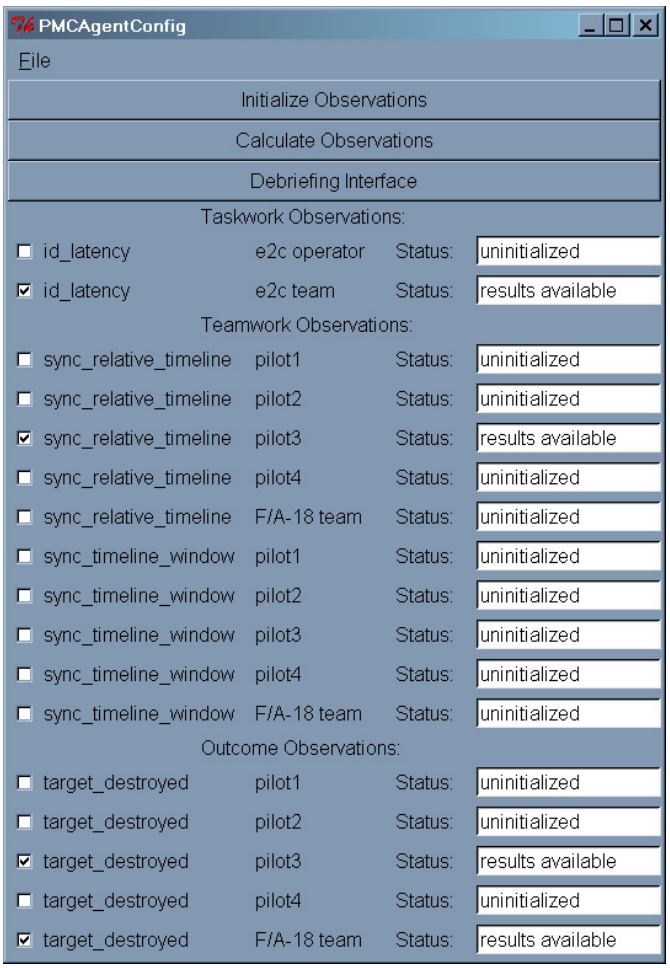

Figure 5. Screenshot of Typical Request for Measures.

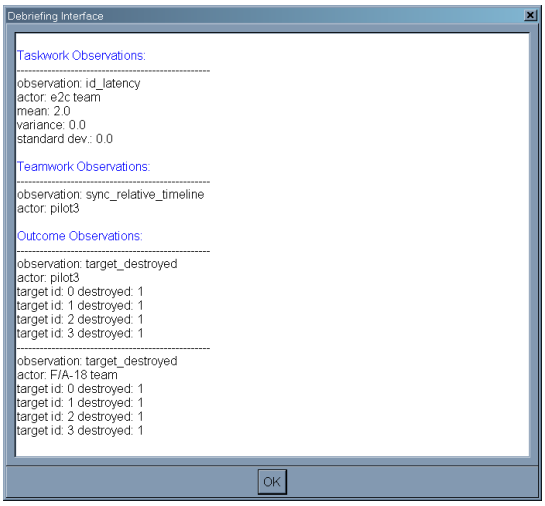

Figure 6. Screenshot of Resulting Measures.

For purposes of illustrating the internal mechanisms of the system during all three phases, the activities of the Intelligent Agents are displayed on a process visualization screen. This functionality is important to understand how the system works, but will be a minor part of the eventual system, since once the system is understood the real-time and historical interaction of the agents will mainly be useful only for debugging system problems. Figure 7 shows a screenshot of a typical visualization during a request for measurements in the AAR phase. In this diagram, the components displayed in blue are active (Raw Data Store, Data Configuration, Measurement Agent, and Debriefing Interface), those in green have been initialized but are inactive (Mission Measurement Model, Data Capture, and the HLA Bus). Note the similarity between this screen and the architecture diagram (see Figure 2).

THE FUTURE OF PMOs

The objects described in the hierarchies above are notional in that they are noncommittal with respect to any particular environment's approach to representing objects, provided that objects have (at least) single data inheritance and a few other basic characteristics. Remaining noncommittal like this has at least two advantages. First, systems involving them can be implemented in many programming languages. The prototype described in this paper was implemented in Python, for instance, but it could have been implemented in Java, C++, PHP, C#, Visual Basic, or many others. Second, with a small amount of implementation work, this approach is compatible with a number of distributed/federated on-the-wire formats, most notably HLA, but also including Distributed Interactive Simulation (DIS) and most local inter-process communication schemes.

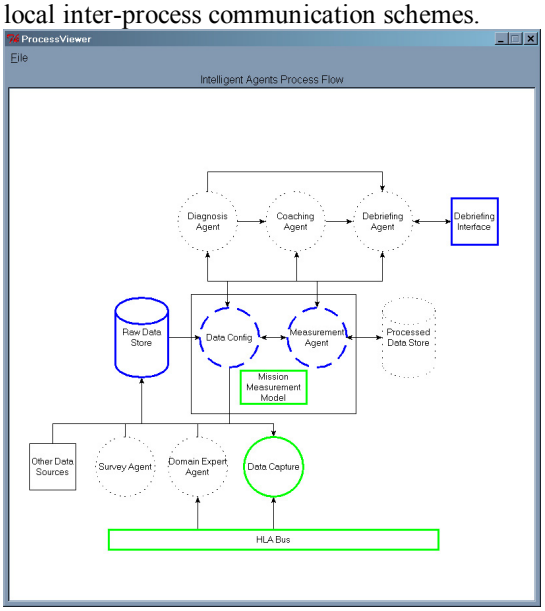

Figure 7. Screenshot of Process Visualization During Request for Measures.

The other side of this flexibility is that some important aspects of the measurement system, namely the techniques for actually doing the computation, need to be expressed outside of the PMO framework in an implementation language. In practice, this should present no problem because a system using PMOs will generally be built using programming language that can be used to complete the PMOs.

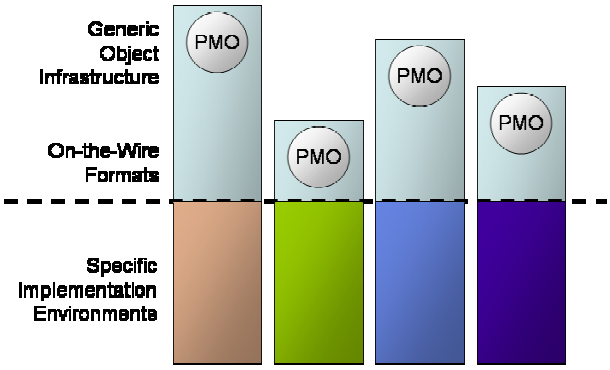

Figure 8. The Role of PMOs in Specific Implementations.

Figure 8 shows this dichotomy. What is clear is that any system implemented using PMOs will need to be adapted to the specific implementation environment. In the future it is likely that the dashed line in Figure 8 will be pushed downwards as we discover more useful, but implementation-free, features to represent. Despite

the possible inclusion of computation semantics into PMO representations,, the dividing line will always exist. Given that this is so, we have begun developing a standard way to represent the implementation-free aspects of PMOs in XML by creating an XML Schema based language called Human Performance Measurement Language (HPML). We believe such a language can be useful in a variety of settings. In many cases, there are even automated utilities to turn HPML entities into a set of classes in a particular programming language, further easing the implementation burden. Examples include Castor (Castor, 2004) for Java, generateDS for Python (Kuhlman, 2004), and the Enterprise Edition of XMLSpy (Altova, 2004) for Java, C++, and C#.

The hierarchy in HPML we are about to describe is derived from the PMO hierarchies described above, but differs in those areas where our thinking has advanced as we have gained more experience with the concepts.

The top-level element in HPML is called MeasurementModel, and it has three optional subelements: Entities, Measurements, and MeasurementInstances. Entities correspond directly to the PMO class TrainingEntity. Subelements of Entity can be seen in Figure 9.

Similarly, MeasurementInstances correspond directly to the PMO class Observation. However, there is nothing in the PMO hierarchy that corresponds to Measurements. The distinction between Measurements and MeasurementInstances deserves comment. Measures are relatively context-free descriptions and can express the fact that some measurements are built from others. This is true in a simple sense when measurements involve differences between pairs of other measurements, but can also be true when complicated measures with complex interdependencies are required. Measurements have a dimension (e.g. time, space, count) attached and have attributes that describe triggering pre- and post-conditions for the measurement. MeasurementInstances, on the other hand, combine a context-free Measurement with a context consisting of one or more Entities, one or more DataSources, one or more data types, and the actual values of the observation when available.

For example, measuring the time to identify a new radar track for Radar Operator trainee RO1 in a given scenario will require a MeasurementInstance that links to a Measure (that describes time to identify a track in general), to two DataSources (a preplanned script action and the timestamp of the appearance of a Track ID object on the bus), a data type of integer, and one or more actual measurements of such differences. Taking the same measure for Radar Operator trainee RO2 might involve a different MeasurementInstance that links to the same Measurement element but that

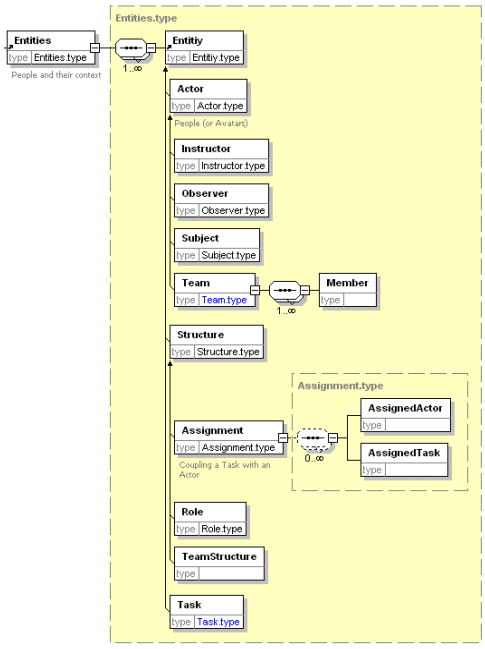

Figure 9. HPML Entities and its Subelements.

has different DataSources (perhaps a synthetic training agent that launches a threat dynamically in the scenario to challenge the trainee). Figure 10 shows the subelements for Measurement and MeasurementInstance elements.

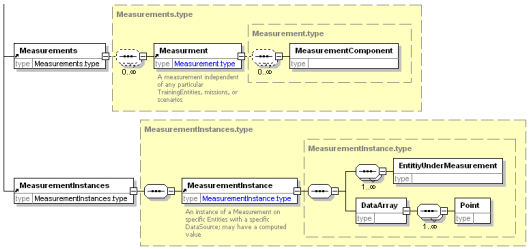

Figure 10. HPML Measurements, MeasurementInstances, and their Subelements.

CONCLUSION

Representing humans in the training system, their tasks and assignments, their roles and structures, and especially their performance, is valuable for several reasons. Locally, it provides both flexibility and precision. In a distributed system, it opens up possibilities ranging from dynamic steering of the scenarios based on human performance to load-

balancing to describing the behavior and effects of synthetic teammates. Perhaps as importantly, it serves to remind us of the difference between simple practice and carefully planned training in simulator-based environments.

PMOs further these goals, and a proof of concept prototype has been implemented and tested. It is believed PMOs and their younger sibling HPML offer the potential to improve performance measurement in simulator based training while making it considerably more convenient to implement.

ACKNOWLEDGEMENTS

The Navy Program Manager for Aviation 205 funded this work. The opinions expressed here are those of the author and do not necessarily reflect the views of the sponsor or the Department of Defense.

REFERENCES

Altova (2004). *XMLSpy Enterprise Edition*. http://www.altova.com/products_ide.html.

Aptima, Inc., & BBN Technologies. (2002). *Enhancing the Usability of Computer-Generated Forces: Air Weapons Officer Training System Effectiveness Study*. Aptima Report AP-R-1188 to the Air Force Research Laboratory. Woburn, MA: Aptima.

Cohen, M.S., Freeman, J.T., & Thompson, B.T. (1998). Critical Thinking Skills in Tactical Decision Making: A Model and A Training Method. In J. Canon-Bowers, & E. Salas (Eds.), *Decision-Making Under Stress: Implications for Training & Simulation*. Washington, DC: American Psychological Association Publications.

Castor (2004). The Castor Project. http://www.castor.org.

Diedrich, F.J., Roberts, B., Diller, D.E., MacMillan, J., & Deutsch, S. (2002). Hybrid team training testbed for AWACS aircraft controllers. *Proceedings of the 46th Annual Meeting of the Human Factors and Ergonomics Society*. Baltimore, MD.

Entin, E.E and Serfaty, D.(1999). Adaptive team coordination. *Human Factors*, *41*, 312-325.

Freeman, J., Diedrich, F.J., Haimson, C., Diller, D.E.& Roberts, B. (2003). Behavioral representations for training tactical communication skills. *Proceedings of the 12th Conference on Behavior Representation in Modeling and Simulation*. Scottsdale, AZ.

Freeman, J. (2002). I've got synthers. Who could ask for anything more? *Proceedings of the 46th Annual Meeting of the Human Factors and Ergonomics Society*. Baltimore, MD.

Freeman, J., & Serfaty, D. (2002). Team critical thinking. *Proceedings of the 7th International Command and Control Research and Technology Symposium*. Monterey, CA.

Freeman, J., Thompson, B., Littleton, E.B., Craig, P., Rubineau, B., Bailin, S., Serfaty, D., & Cohen, M.S. (2000). *Metrics for Evaluation of Cognitive Architecture-Based Collaboration Tools*. Aptima Technical Report AP-R-1119. Woburn, MA: Aptima.

Freeman, J.T., Cohen, M.S., Serfaty, D., & Thompson, B. (1997). Beyond pattern recognition: Improving situation assessment by training critical thinking skills. *Proceedings of the 2nd Annual Symposium on Situational Awareness in the Tactical Air Environment.* Patuxent River, MD.

Johnston, J.H., Serfaty, D., & Freeman, J. (2003). Performance measurement for diagnosing and debriefing distributed command and control teams. *Proceedings of the 8th International Command and Control Research and Technology Symposium.* Washington, DC.

Kuhlman, D. (2004). *generateDS for Python*. http://www.rexx.com/~dkuhlman/#generateds-py-generate-python-data-bindings-from-xml-schema.

MacMillan, J., Entin, E.E., & Serfaty, D. (2004). A Framework for understanding the relationship between team structure and the communication necessary for effective team cognition. In E. Salas, S.M. Fiore, & J. Cannon-Bowers, (Eds.), *Team Cognition: Process and Performance at the Inter- and Intra-Individual Level*. Washington, DC: American Psychological Association.

MacMillan, J., Paley, M.J., Levchuk, Y.N., Entin, E.E., Serfaty, D., & Freeman, J.T. (2002). Designing the Best Team for the Task: Optimal Organizational Structures for Military Missions. In M. McNeese, E. Salas, and M. Endsley (Eds.), *New Trends in Cooperative Activities: System Dynamics in Complex Settings*. San Diego, CA: Human Factors and Ergonomics Society Press.

Smith-Jentsch, K.A., Zeisig, R.L., Acton, B., & McPherson, J.A. (1998). Team dimensional training: A strategy for guided team self-correction. In Cannon-Bowers & Salas (Eds.), Making decisions under stress: Implications for individual and team training. Washington, DC: American Psychological Association.