A Language for Rapidly Creating Performance Measures in Simulators

Webb Stacy, Ph.D.[1], Danielle Merket[2], Jared Freeman, Ph.D.[1], Emily Wiese[1], Cullen Jackson, Ph.D.[1]

[1]Aptima, Inc.
Woburn, MA & Washington, DC
wstacy@aptima.com
freeman@aptima.com
ewiese@aptima.com
cjackson@aptima.com

[2]NAVAIR Training Systems Division
Orlando, FL
danielle.merket@navy.mil

ABSTRACT

Simulations are designed to provide warfighters with realistic *practice*. Simulator-based *training*, on the other hand must provide *measured practice plus feedback*. Doing so yields efficient and effective training. However, it is often difficult and time-consuming to construct and configure a simulation environment to provide measures and feedback. Creating performance measures may involve a lengthy cycle of measure specification, implementation of the measure in software, testing in the environment, and then repetition of the cycle to correct residual errors. Configuration of performance measures to tailor specific feedback to the trainee is no simpler; in fact, it is often skipped altogether. A simulation environment may rely on a set of more-or-less permanently configured performance measures intended to apply to everyone. Culling the output is left to overburdened trainers and their staff.

Ongoing projects for the U.S. Navy are focused on developing technology to provide capabilities to define new measures and assessments rapidly and then to configure them for specific training missions. We do this using simple visual tools that leverage a Human Performance Measurement Language. HPML is understandable to instructors and systems, and it also supports automated dialogues with other automated programs within a simulator environment. In this paper, we define HPML and associated tools, and we discuss its potential to achieve a significant increase in training productivity and, ultimately, warfighter readiness.

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. He has extensive experience in the development of mission critical software, and holds a Ph.D. in Cognitive Science from the State University of New York at Buffalo.

Danielle Merket is a Research Psychologist in the Training and Human Performance Research and Development Branch of NAVAIR Orlando TSD. She serves as Product Team Lead for the Performance Measurement and After Action Review team within the Navy Aviation Simulation Master Plan and as Principle Investigator for the Integrating Instructor Workload Reduction Tools Research & Development effort. Ms. Merket has 8 years of R&D experience in the areas of tactical aviation training, distributed training systems, aircrew coordination training and team dimensional training. Ms. Merket holds a M.S. degree in Industrial and Organizational Psychology.

Jared Freeman, Ph.D. is Vice President of Research at Aptima. He works to maintain the high quality of Aptima's research and to coordinate research efforts within the company. His research at Aptima concerns human performance measurement and assessment, training, problem solving and decision making in real-world settings, the design of organizations and their information systems. He holds a Ph.D. in Cognitive Psychology from Columbia University and a M.A. in Educational Technology from Teachers College, Columbia University.

Emily Wiese is a Human Systems Analyst and Technical Lead for Measure Development in the Organizational Research and Development Group at Aptima. Ms. Wiese's scientific work involves developing performance measures to evaluate the effectiveness of new technologies, developing innovative training programs, performing competency analyses, and using cognitive work analysis techniques to design information systems. Ms. Wiese received a B.S. and an M.S., both in Industrial Engineering, from the University of Iowa.

Cullen Jackson, Ph.D., is a Cognitive Scientist in the Cognitive Systems Group at Aptima. His research involves human performance measurement, evaluating the effectiveness of training systems, creating effective data visualizations, text classification, and knowledge modeling. Dr. Jackson holds a Ph.D. in Experimental Psychology from Brown University, and a B.S. in Computer Science and a B.A. in Psychology from Trinity University.

A Language for Rapidly Creating Performance Measures in Simulators

Webb Stacy, Ph.D.[1], Danielle Merket[2], Jared Freeman, Ph.D.[1], Emily Wiese[1], Cullen Jackson, Ph.D.[1]

[1]Aptima, Inc.
Woburn, MA & Washington, DC

wstacy@aptima.com
freeman@aptima.com
ewiese@aptima.com
cjackson@aptima.com

[2]NAVAIR Training Systems Division
Orlando, FL

danielle.merket@navy.mil

"You can't train what you can't measure."
(Adaptation of a management mantra).

INTRODUCTION

Simulations are designed to provide warfighters with realistic *practice*. Simulator-based *training*, on the other hand, must provide *deliberate practice with feedback* (Ericsson & Charness, 1994). When practice is well designed (deliberate), performance *can* be measured; to provide feedback, performance *must* be measured. However, it is often difficult and time-consuming to construct and configure a simulation environment to provide measures and feedback. Creating performance measures may involve a lengthy cycle of measure specification, implementation of the measure in software, testing in the environment, and then repetition of the cycle to correct residual errors. Configuration of performance measures to tailor specific feedback to the trainee is no simpler; in fact, it is often skipped altogether. A simulation environment may rely on a set of more-or-less permanently configured performance measures intended to apply to everyone. Culling the output is left to overburdened trainers and their staff.

This paper concerns ongoing work for the U.S. Navy that builds on a technology called Performance Measurement Objects (PMOs; Stacy, Freeman, Lackey, and Merket, 2004; Lackey, Merket, Stacy, and Freeman, 2004) to provide capabilities to define new measures and assessments rapidly and to configure new and existing measurements and assessment for specific training missions. This is accomplished with simple yet powerful tools that leverage Human Performance Measurement Language (HPML).

Our current implementation is on a multi-platform air warfare simulator (E2-C and F/A-18). Other implementations are planned in a supporting arms trainer and close air support simulation environments.

The organization of this paper is as follows. We begin by providing an overview of the elements of the solution to the problem of defining new measurements and assessments and configuring them for a training mission. Next, we discuss HPML (and associated concepts), and briefly describe two contexts where it is or will be utilized. We then discuss some of the challenges of constructing automated performance measurement creation and configuration systems, and we conclude by speculating about the future of HPML.

The Problem

Structured practice, feedback, and evaluation of training effectiveness all rely critically on measures of team performance. The ability to implement automated performance measures has advanced significantly in recent years. In some parts of the training community, it appears that software engineering has outpaced the definition of team performance measures (Freeman, Diedrich, and Haimson, 2004; Bell, Johnston, Freeman, and Rody, 2004; Stacy, Freeman, Lackey, & Merket, 2004). However, these advances are new and unevenly applied between practitioners. Thus, it is not surprising to hear one training researcher report waiting 18 months for a small set of automated measures to be defined, designed, and implemented. The net result is that training in the military's costly, capable, and reconfigurable simulators is often attempted with a few, static libraries of automated measures and a great deal of manual effort by observers.

Elements of a Solution

In a period of tight training budgets and increasing operational demand on warfighting teams, it is essential to maximize the effects of training. This entails highly automated data capture, measurement, assessment, and feedback concerning performance. It also entails providing convenient mechanisms for creating new measurements and assessments along with selecting and configuring existing measures for training missions.

This, in turn, entails providing a precise yet comprehensible means to express and manipulate measurements, assessments, and other aspects of the simulator-based training environments. The solution we have created for doing so is called Human Performance Measurement Language (HPML). It has been precisely defined using XML[1] Schema (Fallside, 2004), but is intended to allow training professionals, including instructor/operators and training researchers, easily to express important concepts from the training world. The electronic definition of HPML and associated reference materials will be available later this year.

A major goal of HPML is to provide a bridge between the software implementation of automated measurements and assessments, on the one hand, and the thought processes of training professionals who want to define and use automated measures of performance, on the other. An interesting side-effect of this activity is that this definition has already enabled more precise thinking about performance measurement in our own work. For example, we can articulate relationships between measures of team performance and measures of individuals on the team, and have been able to discover and express some of the complexities of the relationship between measurements and assessments, especially when they are aggregated.

THE PMO APPROACH

PMOs

PMOs are software objects that represent things in the training and performance measurement world. PMOs are intended to be understood both by humans and by computers. There are three top-level classes of PMOs: Training Entities, Measurements, and Assessments.

Training Entity Hierarchy

Training Entities 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. A common source of data will be the Federation Object Model (FOM), and this fact is also reflected in the hierarchy. Sample Training Entities are shown in Table 1.

Table 1. Sample PMOs in the Training Entities Hierarchy.

Trainee – Software object describing the trainee. Subclasses describe different trainee roles for different scenarios (Pilot Trainee, Radar Officer Trainee, Forward Observer Trainee, and so on.)
Instructor – Software object describing the instructor.
Team – A defined set of Trainees and/or synthetic team members.
Task – A job to perform.
Assignment – The pairing of a Task with a Trainee or Team.

Measurement Hierarchy

The second top-level class of PMOs is Measurements. Measurements 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. Sample classes from the Measurement hierarchy may be seen in Table 2.

Table 2. Sample PMOs in the Measurement Hierarchy.

Measurement – A software object containing set of data from specified sources combined by a computation.
Computation – A software object describing a method for combining data in a specified way.
FOM Object – A software object describing data in an object in an HLA federation's object model, typically

[1] XML is a markup language that communicates the structure of documents in a manner that is both human- and machine-readable. XML is designed for extensibility, and is in fact a meta-language for defining markup languages.

used as a data source or to provide context for performance assessments.

FOM Interaction – A software object describing data in an interaction between federates in an HLA federation's object model, typically used directly or indirectly as a data source**.**

Observer Score – A value provided by an observer watching the mission, usually on a behaviorally anchored rating scale. Used as a data source.

Assessment Hierarchy

The final top-level class of PMOs is *Assessments*. This class complements the Measurements hierarchy, providing an interpretation of the raw numbers that are Measurements. For example, a pilot trainee might cause his bomb to detonate 11 sec. later than ideal (a Measurement) which might be judged to be in the Novice category (an Assessment). Table 3 shows some of the objects in the Assessment class hierarchy, and Figure 1 shows a somewhat complex arrangement of PMOs that provides assessments of pilot trainee performance, where three individual Measurements feed three individual Assessments and one team Assessment, with Performance Standards providing the method by which they are mapped.

Table 3. Sample PMOs in the Assessment Hierarchy.

Measurement – a specific location on a referenced Scale
Performance Standard – an object that defines a mapping from one or more Measurements and/or Assessments to an Assessment.
Context – an object that will be quite elaborate in the future, but for now can simply be queried and will return one of a small set of states (for example, clear or cloudy, day or night)
Assessment – a specific location on a referenced Scale, computed with reference to a Performance Standard, with one or more Measurements and/or Assessments as input.

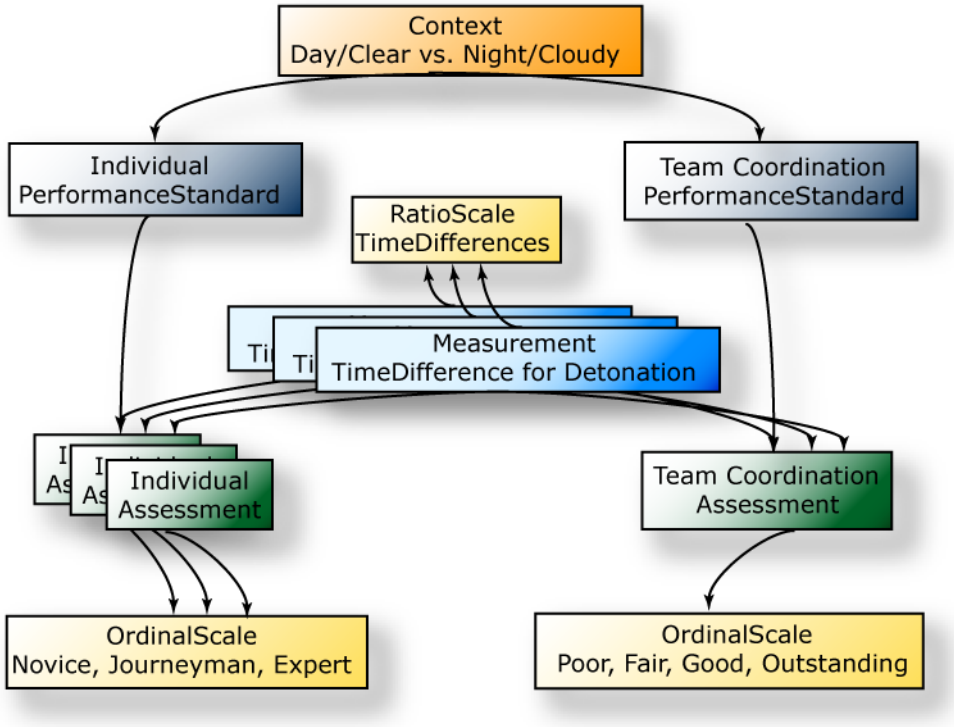

Figure 1. Assessment Objects for Trainee and Team Performance.

HPML-BASED MEASUREMENT

HPML was designed as a language to express PMOs. One way to make the automated measurement software structure more comprehensible to people who are not software specialists is to create meaningful structures in software that are close to those used by people who work in the training domain. Although object-oriented approaches are not a perfect match for human categorization (Lakoff, 1987; Rosch & Mervis, 1975), they provide a much more accessible set of concepts than the usual litany of database schemas, network protocols, and other computer jargon. The object-oriented approach presented here is that of PMOs (see Stacy, Freeman, Lackey, and Merket, 2004 for more on PMOs), and the means we use to express the PMOs is HPML.

The hierarchy in HPML is derived from the PMO hierarchies described above, but differs somewhat in order to meet the specific requirements of representing both generic concepts (e.g., measurements and assessments) and mission-specific concepts (e.g., instances of measurements and instances of assessments). By making these distinctions, HPML is able both to describe available resources and to express the tailoring of those resources for a given training mission.

Benefits

PMOs expressed via HPML may be used for local communication within the performance measurement system as well as the federates' communication bus. This provides several benefits:

- HPML provides a rich, flexible basis for the clear, exact expression of new measures.
- HPML provides a foundation for sophisticated mission measurement configuration by instructor/operators and automated agents.
- HPML provides a framework for automated portions of the performance measurement system to communicate about a wide spectrum of measures that is comprehensible to humans, maintainable, and easily and gracefully extensible.

The top-level element in HPML is called the `MeasurementModel`. The `MeasurementModel` consists of six sets of entities that are inspired by the PMO hierarchy, as follows: (note: HPML names are shown in `Courier New` font):

- **`Entities`**. Objects from the training world: trainees, teams, instructors, observers, tasks, assignments, and so on. These are all mission-specific entities; they will be different from mission to mission if the trainees, teams, or assignments differ, for instance.
- **`MeasurementInstances`**. Measurements to be taken with respect to specific entities for specific missions. They are generally based on `Measurements` that have already been defined.
- **`AssessmentInstances`**. Assessments to be performed on specific measurement instances with respect to specific trainees, teams, assignments, performance standards, and mission contexts. They will generally be based on `Assessments` in the library.
- **`Measurements`**. Templates for measuring trainee or team performance, generally available in a library. Data sources and computations are specified. May be parameterized.
- **`Assessments`**. templates for assessing measurements of trainee or team performance, generally available in a library. Inputs—one or more measurements or assessments—are specified, as are performance standards and mission context.
- **`SupportObjects`**. objects that support measurements and assessment, such as scales and predefined data sources.

The first three entities of `MeasurementModel`: `Entities`, `MeasurementInstances`, and `AssessmentInstances`, are specific to a given mission. That is, each mission will specify its own set of entities and specific instances of measurements and assessments of them.

The second three entities of `MeasurementModel`: `Measurements`, `Assessments`, and `SupportObjects`, work across multiple missions.

An Example

In a simplified mission used to illustrate our performance measurement prototypes, a four-ship of FA-18 trainees flies to a predetermined target location and each drops a bomb on a target. Trainees are assigned to attack so that their bombs detonate exactly one minute apart, the team of pilots also is assigned to drop all their bombs within a three-minute window. The success of both assignments is measured, as is the bomb damage on the target.

This situation is easily described with HPML. The examples that follow illustrate HPML snippets in full technical detail; later we will describe ways to create and edit it that are more user-friendly. Even so, it is possible for people who are not software specialists to read HPML directly for "gist" without great difficulty.

Table 4 shows a snippet of HPML describing the four-ship team, and Table 5 shows snippets of HPML describing the measurement of an individual interdetonation latency measurement instance and of a team detonation window measurement instance.

Table 4. HPML Definition of a Team.

```
<Team ID="FA184Ship">
  <Description>
    Four-ship of F/A 18 pilot
trainees
  </Description>
  <Member Who="FA18Hornet1"/>
  <Member Who="FA18Hornet2"/>
  <Member Who="FA18Hornet3"/>
  <Member Who="FA18Hornet4"/>
</Team>
```

Table 5. HPML Definition of an Individual and a Team Measurement Instance.

```
<MeasurementInstance

InstanceOf="DetonationLatency"
        ID="DL1">
  <SubjectRef Ref="FA18Hornet2"/>
  <ParameterValue
        Name="IdealDetonationSpacing"
        Value="1.0"/>
  <ParameterValue
       Name="PrecedingPilot"
        Value="FA18Hornet1"/>
</MeasurementInstance>

<MeasurementInstance
        InstanceOf="DetonationWindow"
        ID="DW1">
  <SubjectRef Ref="FA184Ship"/>
  <ParameterValue
        Name="WindowDuration"
        Value="3.0"/>
</MeasurementInstance>
```

The preceding examples are appropriate for configuring a simulator-based training mission. The measurements themselves are effectively templates for measurement instances such as those shown in Table 5. The definition of the `DetonationWindow` measure for a team is shown in Table 6.

Table 6. HPML Definition of DetonationWindow.

```
<Measurement Scale="TimeDiffScale"
             Dimension="TIME"
             ID="DetonationWindow">
  <Description>
        Did all the bombs explode
        within the detonation
        window?
  </Description>
  <Parameter Name="WindowDuration">
    <Description>
        How long is the detonation
        window open?
    </Description>
  </Parameter>
  <MeasurementComponent
             ID="FirstTime">
      Time first team member's
      bomb detonates
  </MeasurementComponent>
  <MeasurementComponent
             ID="LastTime">
      Time last team member's
      bomb detonates
  </MeasurementComponent>
    <MeasurementComputation
             Operator="MINUS">
      <MeasurementComponentRef
             Ref="LastTime"/>
      <MeasurementComponentRef
             Ref="FirstTime"/>
  </MeasurementComputation>
</Measurement>
```

It is clear that the definition of a measurement is more involved than the use of a measurement instance. Fortunately, the measurement creation and configuration environments described below encapsulate much of the complexity, freeing up training professionals to think about the measures and the mission rather than the mechanics of specifying them.

Libraries

The specification of measurements is more demanding than the HPML shown in Table 6: it is also necessary

to specify how the measure is computed. In some cases, this is a simple computation such as a difference or an average, but others, such as comparing a trainee flight path with an ideal, are considerably more complicated.

HPML was deliberately designed *not* to express such complex computations. To do so fully would mean inventing a programming language, and this would satisfy neither software engineers nor training professionals. Instead, HPML was designed to specify only simple arithmetic expressions and to provide the ability to invoke measurements or measurement components from a library. More complex calculations themselves are relegated to those libraries, which will be implemented in the software engineers' language of choice.

Usage

We envision that HPML will be used in three kinds of environments: text-based editors, visual editors, and automatic program-to-program communications. Each has advantages and disadvantages.

Text-based Usage

Since HPML is simply XML which in turn is simply text, it is quite possible to use a text editor to create and edit HPML. Of course, doing so exposes the user to the full technical complexity of HPML and XML. Though there are many XML editors that in theory make it easier to create and edit XML, most training professionals will find it almost as tedious to use them as to use simple unassisted text editors.

A step up is to use a stylesheet to present and edit the XML in table form. One tool for doing so is Altova, Inc.'s *Authentic* (available for download at no charge from http://www.altova.com). The advantage of editing HPML this way is that it is no longer necessary to deal with the mechanics of XML and HPML syntax; only content need be specified. Figure 2 shows a stylesheet-based representation for the HPML in Table 5 and Table 6.

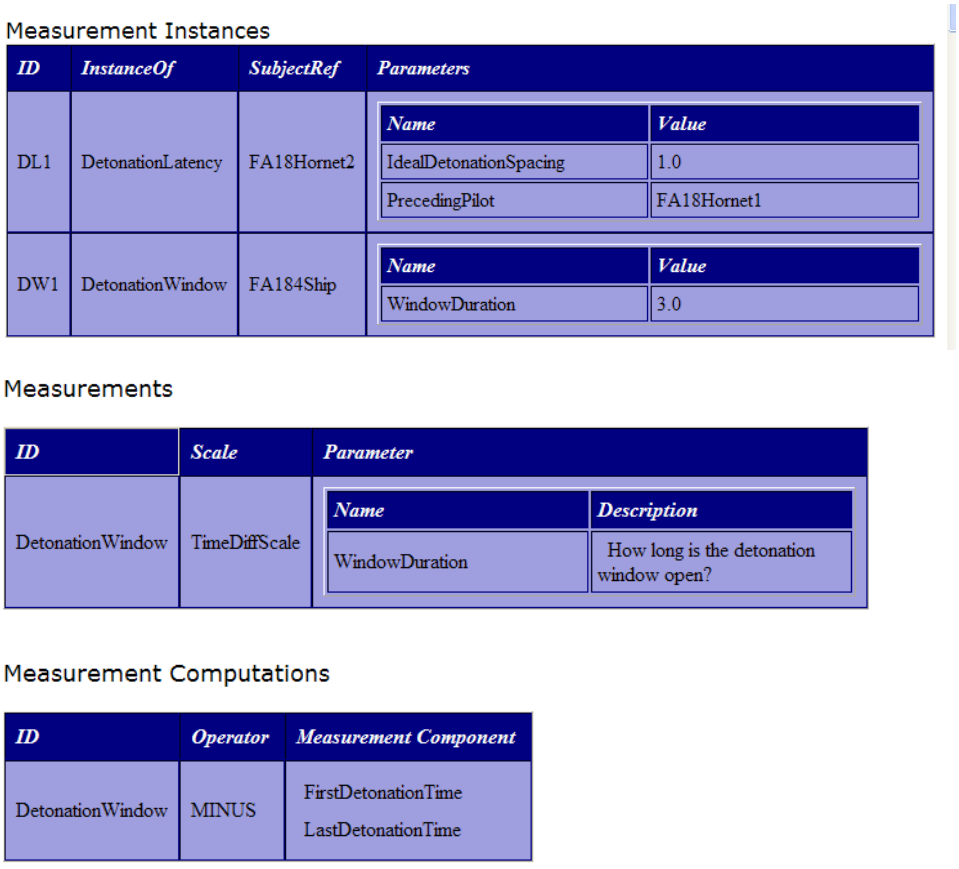

Figure 2. Stylesheet-based Editing of HPML.

Visual Usage

Though the stylesheet-based usage of HPML is an improvement over direct text editing, it still requires that users master the semantic structure of HPML. This may detract from their ability to focus on creating and configuring performance measures. To solve this problem, we are creating a visual drag-and-drop editing environment that will allow direct manipulation of performance measurement components, as may be seen in Figure 3.

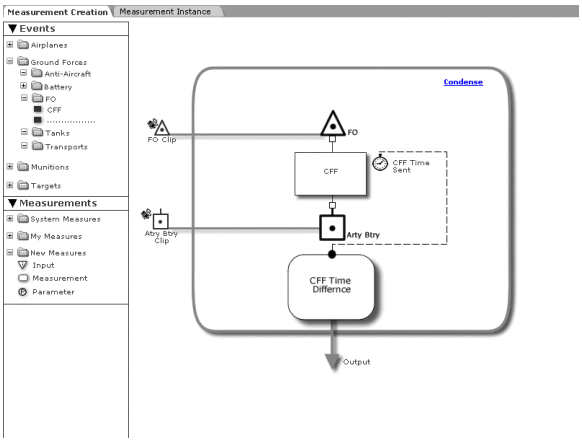

Figure 3. Visual Definition of the Time between Two Calls for Fire.

Here, the timestamps on calls for fire (CFFs) between a forward observer and an artillery battery are subtracted to measure the time between them. The measure is being defined for the HLA environment of the Forward Observer Personal Computer Simulator (FOPCSIM) forward observer training environment (Brannon & Villandre, 2002).

Programmatic Usage

A performance measurement system needs to interface with other systems in the training environment. In particular, it must communicate with a subsystem that allows for specification of performance measures in line with mission training objectives, and it must communicate with a subsystem that allows for after-action review in order to provide feedback on the performance measures.

Fortunately, there is a great deal of third-party infrastructure that enables the use of XML (and therefore HPML) as a medium for automated communication between programs. It is therefore straightforward and convenient to use the very same HPML created and configured by humans to communicate between programs.

A PERFORMANCE MEASUREMENT SYSTEM

We have been implementing an intelligent agent based performance measurement system that uses PMOs and HPML. It is integrated with the testbed at the Manned Flight Simulator (MFS) at Patuxent River Naval Air Station, and we have begun integrating it with a Forward Observer/Forward Air Controller testbed in the Office of Naval Research's Virtual Environments and Technologies (VIRTE) program.

The performance measurement system's architecture is illustrated in Figure 3. The constituent agents have the following functionality:

- 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 **Assessment Agent** maps raw performance numbers to an assessment (such as pass/fail or novice/expert/journeyman or 93rd percentile.)
- The **Diagnosis** and **Coaching Agents** identify the root causes of performance failures and remediate them.
- The **Debriefing Agent** communicates measures to an external after action review system.
- The **Data Capture Agent** listens for and captures relevant data on the HLA bus.
- The **Domain Expert Agent** allows human experts (e.g., observers) or synthetic entities to provide expert solutions and assessments of trainee performance.
- The **Survey Agent** captures scores made by observers using measurement instruments such as behaviorally anchored rating scales.

- The **Other Data Sources Agent** captures data from other sources, such as those that represent foreknowledge of training events (e.g., strike plans and scripted HLA events).

- The **Performance Data Store** holds relevant data provided by other agents in the system.

- The **Scenario Mapping Agent** is responsible for associating measurements and assessments with specific training objectives and scenario events provided by an external Instructor/Operator interface. In both of the testbeds in which this system is or will be integrated, this Instructor/Operator interface will be the Common Distributed Mission Training Station (CDMTS) under development by NAVAIR Training Systems Division.

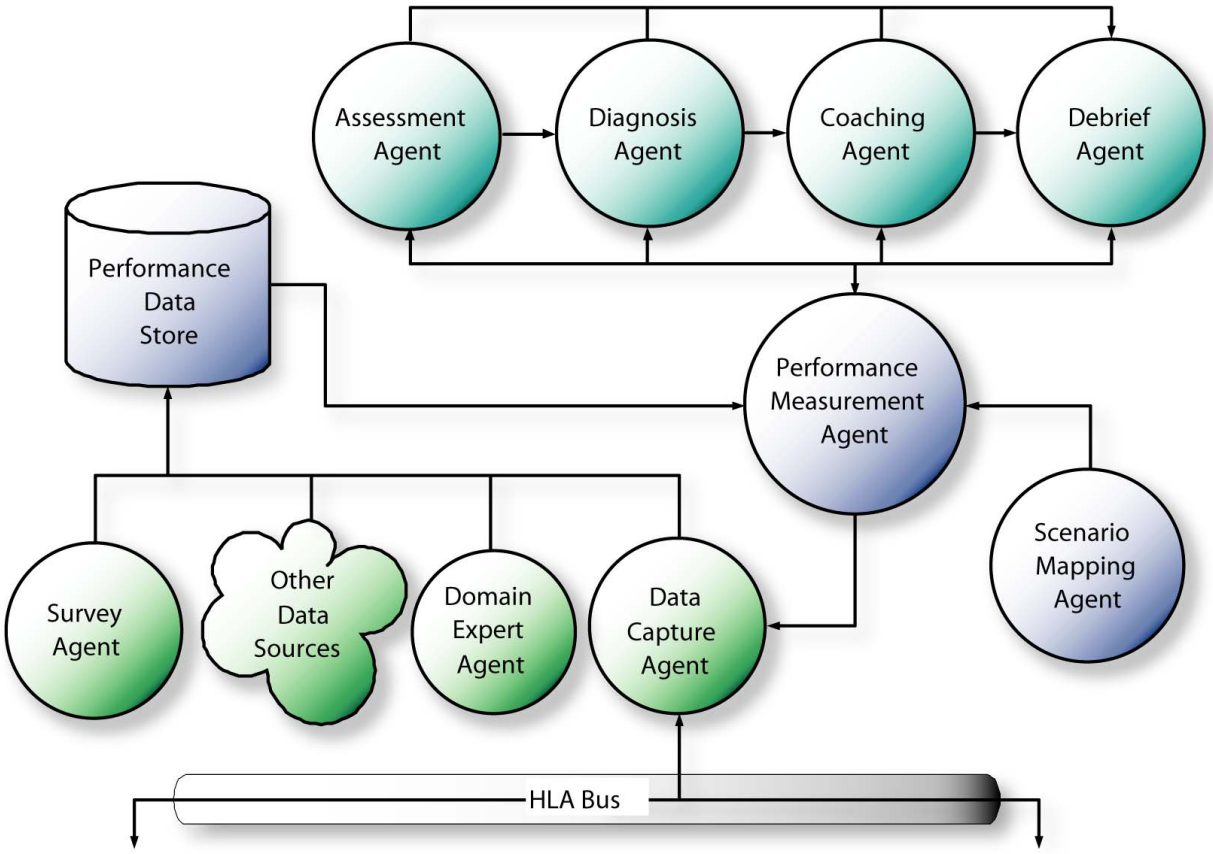

Figure 4. Architecture of the Intelligent Agents Performance Measurement System.

Communication among agents in this prototype is accomplished via PMOs expressed in HPML. We envision that this system will eventually become a full-fledged HLA federate with its own Simulation Object Model (SOM) that will be populated by PMOs likely to be of interest to other federates.

CHALLENGES

The HPML-based system just described is very promising, but must address several challenges. We discuss three of them here: the possibility of installing the system in multiple environments, issues in the construction of performance measurement libraries, and the need to use object histories to establish object attributions.

Working in Multiple Environments

The performance measurement system just described is ultimately intended to work in a wide variety of HLA and non-HLA contexts. Adapting the Data Capture

Agent to new contexts is potentially a significant challenge.

Of course, a new context often means new training objectives and data sources, and this in turn probably means that new measures will need to be specified and configured. The good news is that once the performance measurement system is running in a new context, all the performance measurement creation and configuration machinery—including the visual HPML editor described above—will apply wholesale, greatly accelerating the process of inserting performance measurements into a new environment. Should there already be a set of performance measurements in use in the new environment, the HPML tools will be able to take full advantage of them, too, so that there will be no need to discard prior performance measurement work.

For other HLA contexts, the system already has an HLA service layer (not shown) intended to make it easy to integrate into new HLA federations that involve new FOMs, This is the layer that enables the integration of the system into both the MFS and VIRTE testbeds.

One easy, non-HLA expansion will be to DIS-based systems. Because HLA is in part historically based on DIS, the protocols have strong similarities. We expect the system will run without difficulty once the appropriate adaptations are made to the Data Capture Agent and new measures and assessments are defined.

Beyond distributed simulation buses, however, all that is really required for the system to work is some kind of meaningful event stream. We are currently exploring the possibility of leveraging event streams in multiplayer, online role-playing games to create training with integrated performance measurement.

Library Specification and Construction

Because HPML's computational power was deliberately limited, much of the complex computation will occur in the libraries used by HPML. This means that the contents of a library will directly affect the usefulness, flexibility, and breadth of performance measures. The libraries are therefore of key importance, and the challenges are to identify the right number of the right capabilities: too few, and it will be difficult to express new measures; too many, and it will be confusing to express new measures; too specific, and it won't be possible to express new measures in new domains; too generic, and it won't be possible to express new measures at all.

Some of these capabilities cut across many domains. For instance, depending on the way that the simulation environment treats time, it will sometimes be necessary to remember to supply real timestamps to events in order to measure actual human reaction time. The library will contain routines for doing so. Similarly, it may be important to filter messages about an object's state for relevance, and again the library will contain flexible routines for doing so. And for performance measurements made by observers, the library will contain a rich set of routines for specifying behaviorally-anchored and other kinds of observations.

Many other capabilities in the library will be domain-specific. For instance, in aviation-based scenarios, we expect there will be a need for manipulating and comparing flight paths so that trainee solutions can be compared to expert solutions; for dismounted tactical scenarios, we expect there will be a need to manipulate and compare geographic locations and troop maneuvers; and other domains will surely require other capabilities.

Attribution and Event History

It often occurs that an object in a simulation needs to be attributed to a trainee or a team. For example, if we would like to measure whether the bombs dropped by a four-ship of pilots detonated within an appropriate time window, we will need to be able to identify all the bombs dropped by that four-ship of pilots. However, when bomb objects appear on the HLA bus, they do not identify anything about their history; they simply appear. In a complex scenario with multiple four-ships dropping multiple bombs, it will not be possible to measure the performance of a given four-ship without a solution to this problem. How would we know which bomb detonations to assess?

It is not reasonable to ask instructor/operators to solve this problem. For one thing, in dynamic scenarios it is not always possible to know ahead of time how many objects with what history will exist at any point in the scenario; for another, it is simply confusing to ask instructors to specify a predetermined event history for all HLA objects of potential interest.

The solution is to track the history of objects in such a way that attributions like this can be made automatically. Instructors will need to specify simple attributions like "the first bomb from the four-ship" and "the last bomb from the four-ship." We are currently conducting research to understand how best to track event history without having to remember the entire HLA event stream. Based on preliminary

analysis, we are optimistic that efficient and implementable solutions are possible.

CONCLUSION

Transforming simulators from platforms for practice into environments for objective-based training will improve training efficiency and productivity, improve warfighter readiness, and ultimately save instructor/operator time and costs. This transformation requires performance measurement and feedback. HPML can facilitate both.

Using HPML to represent humans in the training system, their tasks and assignments, their roles and structures, and especially their performance, is valuable for several reasons. It provides flexible and precise performance measures and assessments. Just as important, it allows the construction of intuitive and flexible technologies to advance the science and practice of performance measurement in training.

ACKNOWLEDGEMENTS

The Navy Program Manager for Aviation 205 and the Office of Naval Research funded this work. It is managed by NAVAIR Training Systems Division, Orlando, FL. The opinions expressed here are those of the authors and do not necessarily reflect the views of the sponsors or the Department of Defense.

REFERENCES

Bell, B., Johnston, J., Freeman, J., and Rody, F. (2004). STRATA: DARWARS for Deployable, On-Demand Aircrew Training. *Proceedings of the Interservice/Industry Training, Simulation & Education Conference*. Orlando, FL.

Brannon, D., and Villandre, M. (2002). *The Forward Observer Personal Computer Simulator*. Master's Thesis, Naval Postgraduate School.

Ericsson, K. Anders and Charness, N. (1994). Expert performance: Its structure and acquisition. *American Psychologist*, 49(8), 725-47.

Fallside, D.C. XML Schema Part 0: Primer Second Edition. W3C Recommendation 28 October, 2004. http://www.w3.org/TR/xmlschema-0/.

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.

Lackey, S., Merket, D., Stacy, E.W., & Freeman, J. (2004) Intelligent Training Support Tools: Technology for the Future. *Proceedings of the 2004 American Institute of Aeronautics and Astronautics Modeling and Simulation Technologies Conference.*

Lakoff, G. (1987) *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind.* Chicago: Chicago Press.

Rosch, E. and C. Mervis (1975). Family Resemblances: Studies in the internal structure of categories. *Cognitive Psychology* **7**, 573-605.

Stacy, E.W., Freeman, J., Lackey, S., & Merket, D. (2004) Enhancing Simulation-Based Training with Performance Measurement Objects. *Proceedings of the 2004 Interservice/Industry Training, Simulation & Education Conference,* Orlando, FL.