

# Representing Human Performance with Human Performance Measurement Language<sup>1</sup>

Webb Stacy, Ph.D., Jeanine Ayers, Jared Freeman, Ph.D., Craig Haimson, Ph.D.

Aptima, Inc.  
Woburn, MA & Washington, DC

[wstacy@aptima.com](mailto:wstacy@aptima.com)  
[jayers@aptima.com](mailto:jayers@aptima.com)  
[freeman@aptima.com](mailto:freeman@aptima.com)  
[haimson@aptima.com](mailto:haimson@aptima.com)

**ABSTRACT.** *Effective training requires feedback about trainee performance. Doing so requires explicit specification of the ways in which trainees are to be measured during the training mission. The measurements are derived from available data for automatically computed measures and observable data for observer-based measures. Event streams on HLA- and DIS-based distributed training networks often provide suitable data for automatically computed measures, but generally do not provide the measures themselves. Similarly, observers benefit from guidance about which trainee behaviors will be most telling.*

*To specify mechanisms for computing measures from HLA and DIS data, to specify behaviorally anchored during-mission observations, and to carry resulting measurements and assessments, we have developed a language called Human Performance Measurement Language (HPML). HPML has been precisely defined using XML Schema [1] and is intended to allow training professionals, including instructor/operators and training researchers, easily to express important concepts from the training world. Aptima intends that HPML eventually be freely available to the simulation community. In this paper we outline HPML's capabilities, describe our experiences using HPML in the context of simulator-based training for Navy F/A-18 pilots and for Marine Corps forward observers, and will discuss lessons we've learned with respect to interoperability.*

## 1 Introduction

Simulations are designed to provide warfighters with realistic *practice*. Simulator-based *training*, on the other hand, must provide *deliberate practice with feedback*. 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; [3], [4]) 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

---

<sup>1</sup> This paper is an update of [2]. HPML has evolved substantially since that time both in structure and content.

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.

## 1.1 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 ([5], [6], and [3]). 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.

## 1.2 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 Schema [1], 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.

## 2 The PMO Approach

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.

### 2.1 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.**

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

### 2.2 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.

**Simulation Object** – A software object describing data in an object in a distributed simulation, typically used as a data source or to provide context for performance assessments.

**Simulation Interaction** – A software object describing data in an interaction between simulation objects in a distributed simulation, typically used directly or indirectly as a data source or to provide context for performance measurement.

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

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

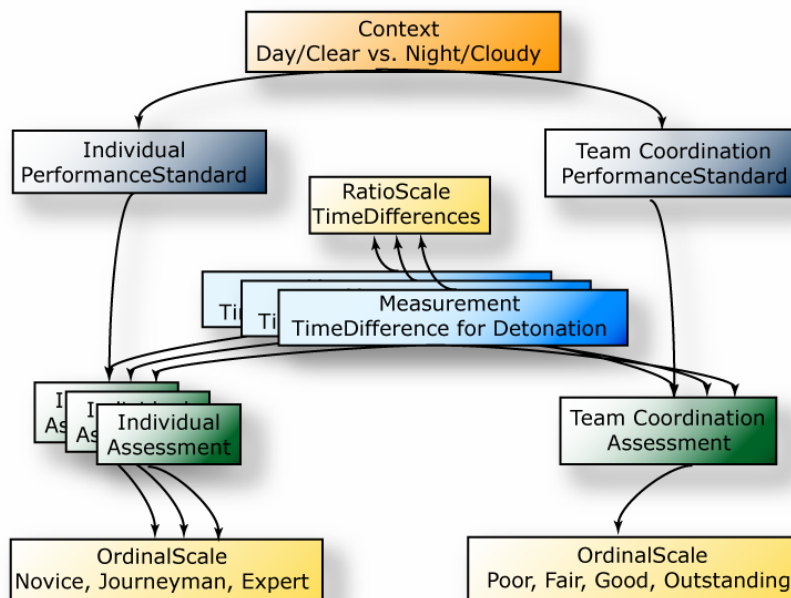
**Performance Standard** – an object that defines a mapping from one or more Measurements and/or Assessments to an Assessment.

**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.

### 2.3 Assessment Hierarchy

The final top-level class of PMOs is *Assessments*. This class complements the Measurements hierarchy,



**Figure 1. Assessment Objects for Trainee and Team Performance.**

### 3 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 ([7], [8]), 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 [3] 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.

#### 3.1 Benefits

PMOs expressed via HPML may be used for local communication within the performance measurement system as well as within the broader distributed simulation network. This provides several benefits:

- HPML provides a rich, flexible basis for the clear, exact expression of new measures.
- HPML provides an external and easily maintained foundation for measurements. If they are expressed in HPML, the performance measurement system infrastructure understands HPML, and the data for computing them is available on the simulation network, creating new measures or modifying existing requires no changes to any simulation or other application on the network.
- 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, naturally enough, HPML. It consists of ten 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. `MeasurementInstances` may contain `MeasurementInstanceResult` elements to report outcomes.
- **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. `AssessmentInstances` may contain `AssessmentInstanceResult` elements to report outcomes.
- **ContextInstances.** These are entity-specific descriptions of the context in which they are performing, used either as a trigger for measurement or as a modifier for assessment.
- **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.

- **Contexts.** These are aspects of context that apply to all entities, such as initial or final state.
- **SupportObjects.** Objects that support measurements and assessment, such as scales and predefined data sources.
- **ObserverQuestions.** These specify the questions to appear in, and convey the results of, an observer-based measurement application.
- **Rules.** These specify the means for updating ContextInstances.

The first four entities of HPML element, namely Entities, MeasurementInstances, AssessmentInstances, and ContextInstances, 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 other entities of HPML, namely Measurements, Assessments, Context, SupportObjects, ObserverQuestions, and Rules, work across multiple missions.

### 3.2 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.

### 3.3 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.

## 4 Context

Clearly, measurements and assessments of a trainee or team can be informed by context. But what exactly are contexts and how do they relate to measurements? Informally, measurements apply to a stream of trainee or team behavior, and the contexts of that behavior stream include events and other information surrounding it. The behavior stream may be constructed from more primitive event streams, and context may be supplied from higher-level event streams.

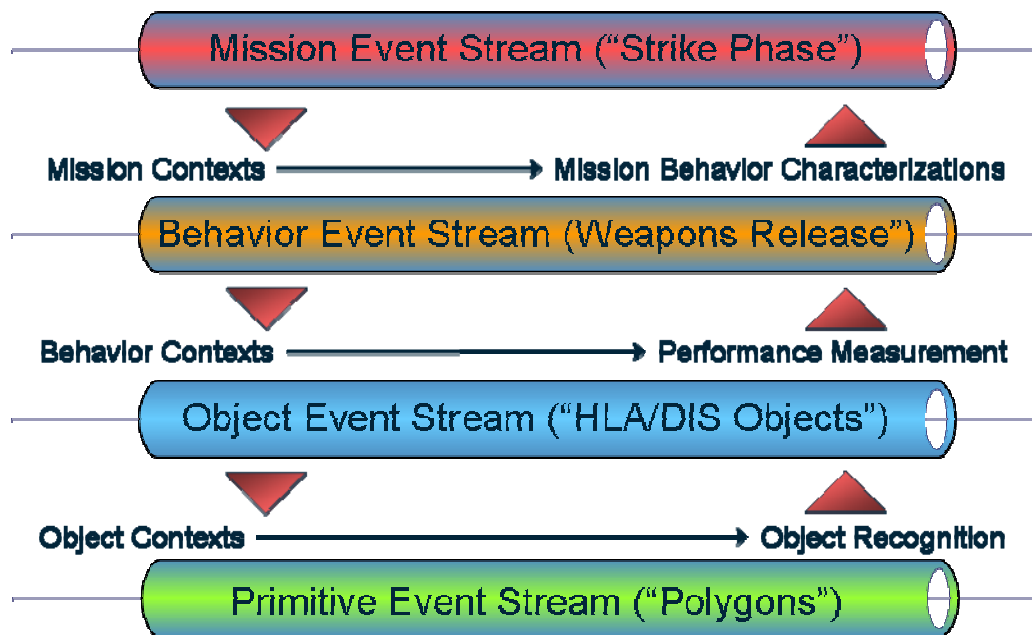


Figure 2. Event stream layers involved in performance measurement.

Figure 2 shows the situation diagrammatically. In some cases (such as some PC-based games), only primitive events such as polygons and the trainees' current 3D r level information about objects in the environment is available. Event attributes at the object layer generally serve as raw data for performance measurement, which is specified at the level of the behavior event stream. In this architecture, higher-level event streams provide context that helps interpret lower-level event streams. Importantly for the present discussion, the mission event stream provides context for interpreting the behavior event stream.

#### 4.1 Two Uses for Context

There are two distinct uses for context in a performance measurement system. These two uses depend heavily on the distinction between Measurements and Assessments detailed above.

#### 4.2 Context Is a Trigger for Measurements

Context can inform us about the times it is meaningful to take a measurement, and can in fact serve as a trigger for taking that measurement. For example, expert pilot instructors who teach F/A-18 pilots how to perform the notch maneuver—a maneuver that minimizes the aircraft's visibility on enemy aircraft radar—find it helpful to know the relative altitude and angle of attack of the enemy aircraft when the trainee begins the maneuver. Though at any given point in time when enemy aircraft are present it is possible to take these two measurements, it is impractical to take them all the time in most simulation environments, since a fair amount of interpolation and other computation is often involved. But it is simple and practical to compute these measurements at the single point when context—the beginning of the trainee's notch maneuver—dictates.

#### 4.3 Context Is a Modifier of Assessments

The other main role for context is to help provide better assessments. In particular, context affects the mapping between measurements and assessments. For example, assessments of measurements of the precision of a landing on an aircraft carrier will depend on things like sea state, weather, and time of day. Assessments of the measured target location error in a Forward Observer's call for fire will depend on whether friendly forces are taking effective enemy fire. Assessments of

the efficiency of an Attack Coordinator in prosecuting a target in a Dynamic Targeting Cell in an Air Operations Center will depend on how often they have been interrupted to prosecute higher priority targets.

### 5 Observer Queries

Elements in this section describe the content of an observer-based measurement application. In addition to grouping instructions (Module and Section) and attributes like category and keywords, basic elements are YesNoQuestion, LikertScale, and InstructorComment. As with other measurements, results are reported via a MeasurementInstanceResult element. Figure XX shows a typical screenshot of SPOTLITE™.

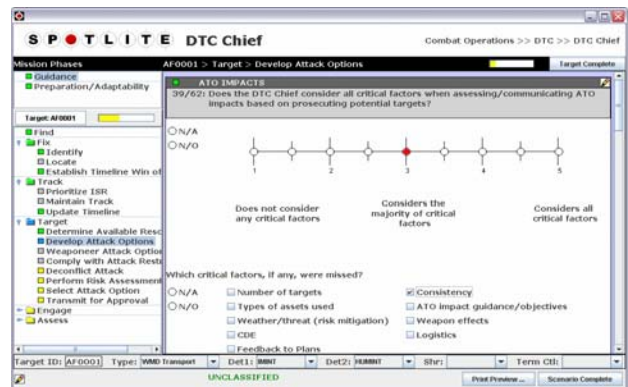


Figure 3. HPML specifies the questions in SPOTLITE.

### 6 Using HPML

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.

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 3 shows a stylesheet-based representation for the HPML in Table 5 and Table 6.

## 6.1 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 4.

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 [9].

## 6.2 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.

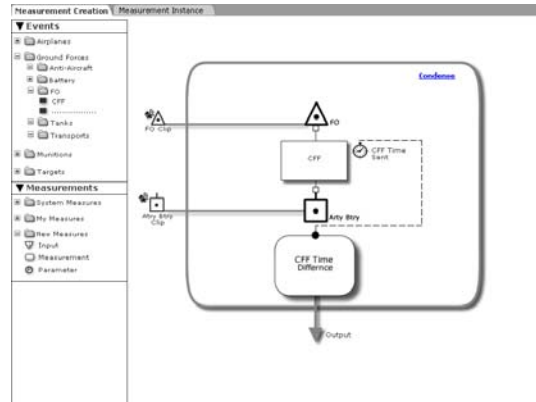


Figure 4. Visual Definition of the Time between Two Calls for Fire.

Measurement Instances				
ID	InstanceOf	SubjectRef	Parameters	
DL1	DetonationLatency	FA18Hornet2	<b>Name</b>	<b>Value</b>
			IdealDetonationSpacing	1.0
			PrecedingPilot	FA18Hornet1
DW1	DetonationWindow	FA184Ship	<b>Name</b>	<b>Value</b>
			WindowDuration	3.0

Measurements			
ID	Scale	Parameter	
DetonationWindow	TimeDiffScale	<b>Name</b>	<b>Description</b>
		WindowDuration	How long is the detonation window open?

Measurement Computations		
ID	Operator	Measurement Component
DetonationWindow	MINUS	FirstDetonationTime
		LastDetonationTime

Figure 5. Stylesheet-based Editing of HPML

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.

## 7 A Performance Measurement System

We have implemented a performance measurement engine 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. It uses PMOs for internal communication, and communicates with other applications via Web Services with an HPML payload. We used Web Services instead of the simulation network so that communication can occur even when the distributed simulation environment is not running. 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.

In the testbeds in which this system is or will be integrated in the near future, this Instructor/Operator interface is the Common Distributed Mission Training Station (CDMTS) under development by NAVAIR Training Systems Division.

Communication among agents in this prototype is accomplished via PMOs expressed in HPML.

### 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.

#### 7.1 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 environment 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.

#### 7.2 User Specification of Context

An important challenge is to find a way to make the specification of context available to users such as instructor/operators whose areas of expertise typically do not include rule-based knowledge representation, human performance measurement, or software engineering. We saw in the last section one way that a user interface might help gather the required information, especially as it is put into use in reporting assessments. This is only a first step towards solving the general problem of helping users specify the rules that define context, especially as the rules begin to

require sophisticated features to develop more accurate representations of context.

Our plan of attack has several facets. First, we intend to develop a library of reusable context rules and rule modules that have been parameterized to encourage adaptation to new situations. Our belief is that it is often easier to recognize and configure existing rule-based specifications than it is to develop new ones from scratch. Such a library can be populated as users develop from-scratch solutions, and, on a slower cycle, can also be populated offline by personnel with more expertise in rule-based systems. Second, as the artifacts of training mission planning become electronic and standardized, we will investigate ways to translate them automatically into a rule-based representation of context. Finally, we believe there are simplifications of the authoring tasks that would allow instructor/operators to easily construct many common contextual representations.

### 7.3 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.

## 8 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.

## 9 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.

## 10 References

- [1] Fallside, D.C. XML Schema Part 0: Primer Second Edition. W3C Recommendation 28 October, 2004. <http://www.w3.org/TR/xmlschema-0/>.
- [2] Stacy, E. W., Merket, D., Freeman, J., Wiese, E., & Jackson, C. A Language for Rapidly Creating Performance Measures in Simulators. *Proceedings of the 2005 Interservice/Industry Training, Simulation & Education Conference*, Orlando, FL, 207-218, 2005.
- [3] Stacy, E.W., Freeman, J., Lackey, S., & Merket, D. Enhancing Simulation-Based Training with Performance Measurement Objects. *Proceedings of the 2004 Interservice/Industry Training, Simulation & Education Conference*, Orlando, FL, 2004.
- [4] Lackey, S., Merket, D., Stacy, E.W., & Freeman, J. Intelligent Training Support Tools: Technology for the Future. *Proceedings of the 2004 American Institute of Aeronautics and Astronautics Modeling and Simulation Technologies Conference*, 2004.

- [5] Freeman, J., Diedrich, F.J., Haimson, C., Diller, D.E. & Roberts, B. Behavioral representations for training tactical communication skills. *Proceedings of the 12th Conference on Behavior Representation in Modeling and Simulation*. Scottsdale, AZ, 2003.
- [6] Bell, B., Johnston, J., Freeman, J., and Rody, F. STRATA: DARWARS for Deployable, On-Demand Aircrew Training. *Proceedings of the Interservice/Industry Training, Simulation & Education Conference*. Orlando, FL, 2004.
- [7] Lakoff, G. *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*. Chicago: Chicago Press, 1987.
- [8] Rosch, E. and C. Mervis. Family Resemblances: Studies in the internal structure of categories. *Cognitive Psychology* 7, 573-605, 1975.
- [9] Brannon, D., and Villandre, M.. *The Forward Observer Personal Computer Simulator*. Master's Thesis, Naval Postgraduate School, 2002.

## Author Biographies

**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.

**JEANINE AYERS** specializes in leading teams of software engineers in development efforts focusing on designing and building data-driven software applications in support of computer-based training

systems, and performance measurement and assessment. In addition, Ms. Ayers develops after action review capabilities for Close Air Support and Military Operations in Urban Terrains simulation systems. She is also researching the area of dynamic workflow. Ms. Ayers received an M.B.A. from Boston University and a B.S. in Industrial Management, with an additional major in Information and Decision Systems, from Carnegie Mellon University.

**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.

**CRAIG HAIMSON, Ph.D.**, specializes in the development and delivery of automated training and assessment systems. His primary interests are in cognitive modeling, critical thinking, and decision making. Dr. Haimson applies his expertise to simulation-based training and assessment systems for domains such as intelligence analysis, information operations, and dismounted infantry operations. His work includes training teamwork skills using synthetic agents, expert system models, and multiplayer online gaming environments. Dr. Haimson received a Ph.D. in Cognitive Psychology from Carnegie Mellon University and a B.A. in Psychology from Harvard University.