Proceedings of the 2004 Winter Simulation Conference
R .G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, eds.

USING XML AND BOMS TO RAPIDLY COMPOSE SIMULATIONS AND SIMULATION ENVIRONMENTS

Paul Gustavson
Tram Chase

SimVentions, Inc.
11903 Bowman Drive, Suite 102
Fredericksburg, VA 22408, U.S.A

ABSTRACT

This paper explores the application of Base Object Models (BOMs), an emerging XML standard, for rapidly composing simulations and simulation environments. We examine how pattern components, supported by the Interface (IF) BOM and how behavior components, supported by the Encapsulated (ECAP) BOM, can be used to enable simulation composability at design time and dynamically at run-time. We also explore the potential capabilities BOMs and other XML standards provide for the future of web-based simulation and training.

1 INTRODUCTION

The ability to compose simulations and simulation environments rapidly and efficiently is a key need within the Modeling and Simulation (M&S) arena. What is required are meaningful components that can be coupled together to represent and model simulations and interoperable simulations environments useful for testing, analysis, training, mission rehearsal, and the prototyping and acquisition of new systems. The need for such a component standard is recognized in the following statement.

> *“To allow maximum utility and flexibility, ... modeling and simulation environments [need to] be constructed from affordable, reusable components interoperating through an open systems architecture.” (Zimmerman 2002)*

One such proposed standard, which leverages the eXtensible Modeling Language (XML), is the Base Object Model (BOM) concept. A BOM can be thought of as a reusable package of information representing a pattern of simulation interplay. This pattern reflects a set of activities among conceptual entities used to accomplish a common objective, capability, or purpose. By capturing patterns and defining components that model the needed capabilities to support these patterns, it is possible to provide an essential building block capability needed within the M&S community as illustrated in Figure 1.

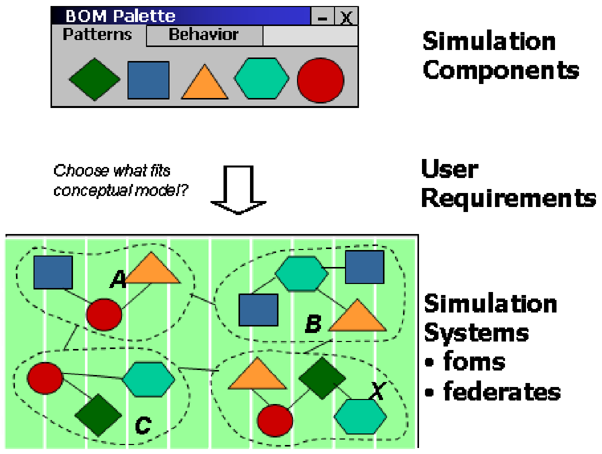


Figure 1: BOM Tool Palette to Compositions

Through the auspices of the Simulation Interoperability Standards Organization (SISO) and the support of the Defense Modeling and Simulation Office (DMSO), specifications for BOMs are being openly developed and refined with the goal to submit as a potential standard and a composability enabler for the M&S community. The results for this effort are beginning to yield the necessary formal definitions, standards and formalisms for facilitating composability. The first specification for BOMs titled, the “BOM Template Specification Volume I – Interface BOM” (SISO-STD-003.1-Trial-Use-V0.9) is now available as a trial-use specification. Additionally a draft release titled the “Guide for BOM Use and Implementation” is also available. Currently under development is the “BOM Template Specification Volume II – Encapsulated BOM”.

The BOM component architecture based on these specifications and documents provide the needed formalisms to influence the following capabilities within the M&S community:

1. Interoperability – The application of the XML and XML Schemas prescribed for BOMs provides a mechanism for defining and validating context, and facilitates understanding of the data being exchanged. Furthermore, the flexibility offered by BOMs allows for greater application of simulation interoperability within other domains.
2. Reusability – The meta-data cataloged within a BOM such as intent-of-use, historical use, behavioral information, Verification and Validation (V&V) history, and potential visual information will facilitate greater reuse of components.
3. Composability – BOMs will facilitate the ability to rapidly compose simulations and simulation environments both statically (design time) and dynamically (run-time).
4. Adaptability – Mega-BOMs, which are produced by BOM compositions, can be used to represent the standard data exchange interface for systems and simulations. Unlike the current High Level Architecture (HLA) approach in which all federates must comply with a common Federation Object Model (FOM), federates can continue to use their specific Mega-BOM interface to play within environments comprised of other simulations and systems represented by their own unique Mega-BOM interface. Adaptability is accomplished by the receiving federate deploying and applying the appropriate XML-based transformations, which represent mappings between common BOMs within a Mega-BOM. This minimizes the effort typically spent in re-tooling federates associated to complying with a specific FOM.
5. Tools, Repositories and Web Services – It is envisioned that the next generation of tools and web services (such as collaborative development environments and repositories) could come about supporting the creation, deployment and use of BOMs for simulation development, maintenance, and run-time support.

2 BOM ARCHITECTURE OVERVIEW

Before we explore how XML is being used to represent the various BOMs and enable composability, let us examine the overall BOM architecture.

2.1 Conceptual View

By leveraging the conceptual model of what's intended to be modeled, exercised or analyzed among a set of conceptual entities, we can determine the activity relationship among conceptual entities, and the anticipated model behavior for any one specific conceptual entity. The examination of the activity relationship among conceptual entities results in the identification of the patterns of interplay, whereas examining the anticipated conceptual entity behavior resulting from the events associated to a pattern of interplay results in the ability to identify the necessary model behavior. As illustrated in Figure 2, there are two essential BOM building blocks that can be produced from this conceptual analysis: pattern components, which are codified as Interface BOMs, and behavior components, which are codified as Encapsulated BOMs.

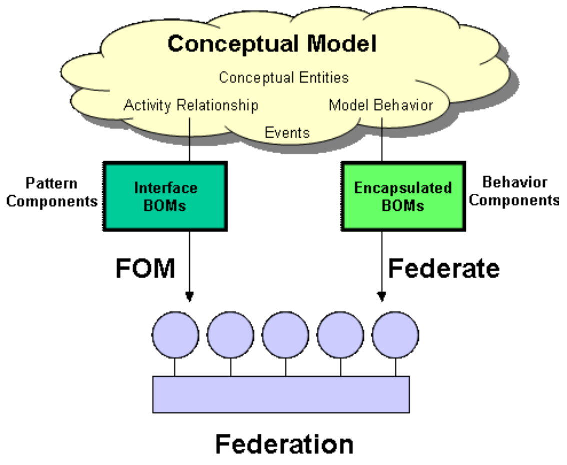


Figure 2: BOM Conceptual View

IF BOMs represent the relationship of activities among conceptual entities (FOM level), whereas ECAP BOMs represent the needed behavior required of a conceptual entity to support one or more patterns of interplay (Federate level).

2.2 Architecture Layers

Figure 3 illustrates the four basic areas of the simulation problem space as it relates to the HLA with BOMs providing an enabler in two ways:

1. supporting the creation of an agreed upon Federation Object Model, which serves as the interface contract among all participating federates (see IF BOMs), and
2. providing the necessary behavior model for a federate to fulfill the "interface contract" (see ECAP BOMs).

From the bottom up, we have the Communication Layer which includes the runtime infrastructure (RTI) or any other communication mechanism used for the distribu tion of data messages among federates at play.

Next, we have the Interface Layer, which is typically represented by an HLA FOM for identifying what type of

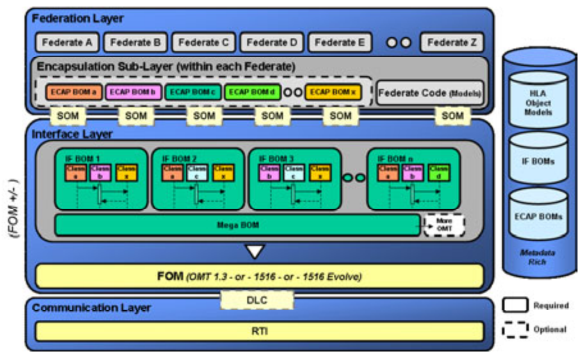


Figure 3: HLA-Based BOM Architecture View

data will be conveyed through the Communication Layer. From a composability standpoint, the Interface Layer for a federate can be supported by Interface (IF) BOMs and a Mega-BOM, which represents the collection of coupled IF BOMs. An IF BOM is a reusable representation of a pattern of interplay characterized by one or more events.

At the top, we have the Federation Layer, which includes the collection of federates supporting the Interface Layer (the FOM). Within the Federation Layer, we have the Encapsulation Layer, which reflects the behavioral code necessary for a federate to carry out and model the conceptual entities of the federation that were identified in the FOM using Object Model Template (OMT) constructs. At this layer, Encapsulated (ECAP) BOMs can be used to describe the necessary behavior for one or more federates.

At the right of Figure 3, a repository is represented to show that it can be used to identify and select appropriate object models, including IF BOMs and ECAP BOMs, for fulfilling the goals and objectives of a federation.

The Interface Layer and the Encapsulation Layer are the two architectural layers related to BOMs that can be applied to support composability. The BOM types used to support these two layers are further examined in Sections 4 and 5.

3 APPLICATION OF XML-BASED TECHNOLOGIES

Rather than re-inventing technologies and standards, the application and integration of commercial technologies and standards based on the Extensible Markup Language (XML) is widely regarded as an appropriate direction for enabling composabilty. Significant cost benefit and acceptance of XML technologies and standards can be applied and used to support composability. The critical technologies needed for BOM standardization (and the proliferation of BOM components) are predominately centered upon XML based standards and approaches.

The BOM architecture utilizes the XML for the following means:

- To support the codification of pattern and behavior components into Interface BOMs and Encapsulated BOMs respectfully.
- For supporting the essential meta-data to be captured, cataloged, and carried forward within a BOM (examples include purpose, integration history, relevant references, etc…).
- For representing platform independent models, called PIMs.
- For promoting adaptability via the Extensible Language Transformation (XSLT) between similar but different BOMs.
- For addressing breadth of community interests.

BOMs should be viewed as a flexible component-based standard for simulation interoperability that embraces outside XML standards and initiatives such as UML (XMI), SRML, and X3D. Additionally, as we further explore, the BOM component architecture is very much in synch with the concepts and principles centered upon the Object Model Group's (OMG) Model Driven Architecture (MDA).

The sections that follow explore the various applications of XML and XML technologies for enabling BOMs.

4 COMPOSABILITY THROUGH IF BOMS

As illustrated in Figure 4, the coupling of IF BOMs can be used to form a Mega-BOM, which serves as a higher order pattern used to represent a logical collection or assembly of IF BOMs.

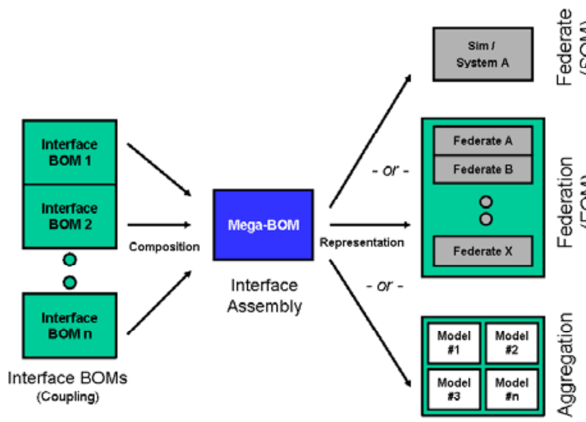


Figure 4: IF BOM Coupling

A Mega-BOM is essentially an interface assembly much like an HLA Simulation Object Model (SOM) or a FOM. It can be used to represent a federate, or federation or even an aggregation of entities within the simulation space, typically identified as an entity group.

As illustrated in Figure 5 the template components that define an IF BOM provide the following capabilities:

- Offers a common meta-data level summary identified as the Model Identification,
- Describes the activities (steps) of a pattern within the Pattern Description, and

- Defines events and key HLA OMT 1516 elements within the Model Definition.

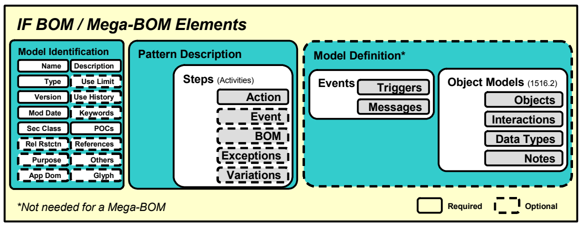


Figure 5: IF BOM / Mega-BOM Template Elements

The *Model Identification* provides what's needed for discovering likely candidate models, whereas the *Pattern Description* provides critical metadata necessary for ensuring proper integration and reapplication of an IF BOM. Figure 6 illustrates the *Pattern Description* relationship. Each step, identified as an activity, has on either an Event, which is defined in the Model Definition, or with another IF BOM defined independently.

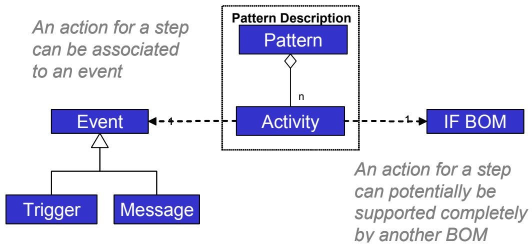

An action for a step can be associated to an event
Pattern Description
Pattern
n
Event
Activity
1
IF BOM
Trigger
Message
An action for a step can potentially be supported completely by another BOM

Figure 6: Pattern Description

The identification and creation of XML schemas was necessary to support the IF BOM ontology depicted previously in Figure 5. The SISO lead BOM Product Development Group (PDG), under the efforts of an internal Drafting Group (DG), generated an IEEE 1516.2-based OMT schema to support the OMT elements used for the IF BOM *Model Definition* table, and defined and refined a suitable schema for the IF BOM *Model Identification, Pattern Description* and the Events aspect found in the *Model Definition,* which was not supported by the OMT.

These schemas provide a key element in describing how the composition of individual BOMs for defining a simulation or simulation environment can be used to form IF BOMs and Mega-BOMs and, in support of HLA, provide the basis for generating FOMs and SOMs from these BOM compositions. Both these schemas are available at `<http://www.boms.info/Schemas>`.

The result of defining and applying XML schemas identifies the essential meta-data, ontology, and model definition to be captured, cataloged and carried forward within a BOM in order to provide for shared understanding and community reuse. These schemas are being incorporated within proposed Simulation Interoperability Standards Organization (SISO) standards and are intended to be registered within the DoD XML Repository upon SISO consensus.

5 COMPOSABILITY THROUGH ECAP BOMS

The ECAP BOM can be used to prescribe the necessary model behavior required of a conceptual entity to support one or more patterns of interplay as illustrated in Figure 7. It should be noted that federates and simulation spaces derived from IF BOMs are not required to use ECAP BOMs. If the capability to be modeled is already an intrinsic element of the federate's behavior, then the usage of an ECAP BOM may not make sense. However, if a federate lacks a specific behavior and that behavior model can be found within an ECAP BOM implementation (EBI), then this becomes an enabler for a more complete simulation.

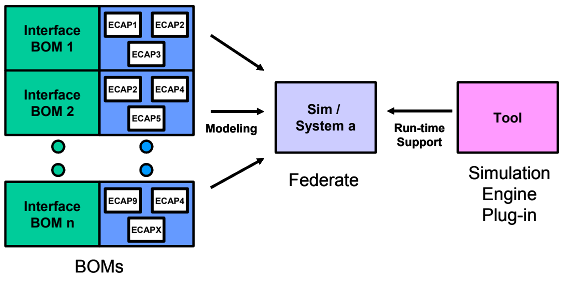


Figure 7: Application of Component Models by a Federate

As illustrated in Figure 8 the template components that define an ECAP BOM provide the following capabilities:

- Offers a common meta-data level summary identified as the Model Identification,
- Describes the states (actions) of a conceptual entity to fulfill one or more patterns of interplay within the Behavior Description, and
- Leverages other markups including VV&A, Digital Rights for protecting intellectual property, and, the rendering markup for visual representation.

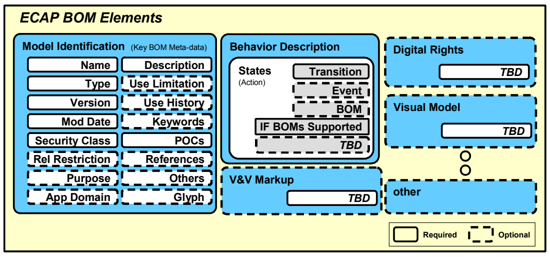


Figure 8: ECAP BOM Template Elements

The *Model Identification* used to provide what's needed for discovering likely candidate IF BOMs is also used to identify ECAP BOMs. The *Behavior Description* providescritical metadata necessary for ensuring proper integration and reapplication of ECAP BOM. Specifically, the *Behavior Description* details the behavior for a conceptual entity described as a set of states associated to the events of a pattern or another ECAP BOM.

5.1 Platform Specific (PS) ECAP BOM Implementations (EBIs)

The current draft standard for ECAP BOMs is centered upon Platform Independent Models (PIMs) providing a language and platform independent mechanism for codifying the behavior capability needed of a conceptual entity. However, for ECAP BOMs to be effective, the draft standard supports and encourages the generation and usage of ECAP BOM Implementations or EBIs.

Manifestations of EBIs might include a source code module – such as a C++ class or Java class - or it might include a binary or byte-code assembly - such as a Windows DLL, ActiveX, Unix DSO, .NET assembly, or a Java Bean. There are benefits for building and using either type of EBI. For instance, it's possible for an EBI that is manifested as a binary or byte-code assembly EBI to be fetched and loaded dynamically during an exercise by a federate – during execution run. No matter which type, an EBI ultimately provides the necessary processing code to perform the operation described by the ECAP BOM.

So far, all of these examples of EBIs are recognized as platform specific implementations because of their language or platform dependence. However, there is also the potential to define an EBI that is platform dependent through the application and use of the Simulation Reference Markup Language (SRML).

5.2 SRML-Based EBIs

Figure 6 previously illustrated the relationship of a pattern with the usage of matching behavior components at runtime. What is also being shown is that the execution of Encapsulated BOM Implementations (EBI) can be supported by a simulation engine plug-in used by the federate.

For a moment consider how a Web Browser has provided a common interface mechanism for presenting HTML to a user. Well, in a similar way a simulation engine plug-In can provide a common execution interface mechanism for processing behavior to a simulation. As HTML is to a browser, the Simulation Reference Markup language (SRML) can be to a simulation engine. SRML is like HTML in that it provides for executable content using the same kinds of mechanisms such as object models, scripting, plug-ins, and the ability to dynamically download and assemble content. The two languages differ in that while HTML operates on electronic documents for the purpose of display, SRML operates on electronic models for the purpose of modeling and simulation. SRML provides a general-purpose schema for embedding simulation behavior into arbitrary XML content so that the resulting content may be executed directly by an engine.

What's important to note here is that SRML can be used to provide a PIM construct for an ECAP BOM Implementation. Specifically, the behavior associated to a design pattern represented in XML can be loaded and executed with an SRML engine, given that the implicit and explicit behaviors of the pattern are also included. BOMs include an XML schema component for describing design patterns, as described earlier. That schema component defines elements that allow the representation of explicit sequences of activities among objects at various levels of generalization. With SRML, a particular ECAP BOM Implementation can be loaded directly into the simulator matching with a pattern behavior element (i.e. class/actor) defined by the Interface BOM.

5.3 Generating Platform Specific EBIs

As a potential capability, the creation of a platform specific ECAP BOM could be achieved by taking a platform independent BOM, such as one defined using SRML, and passing it through an XML Transformation layer (via XSLT), which takes the specified XML meta-data into the specific language of choice (or MDA through their meta object facility). The resulting code could then be compiled into a platform specific ECAP BOM that is platform / language specific.

Such PSMs could be contained in the form of a Dynamic Link Library (DLL), Dynamic Shared Object (DSO), Library (LIB) file, ActiveX component, Java Bean or Java byte code, or .NET assemblies. An engineer needs only to implement these Platform Specific EBI within their software; the functionality contained in the original Platform Independent EBI is optimized for a specific platform and may very well contain binary data allowing for more responsive, non-interpreted components and libraries. This capability provides a key mechanism necessary for supporting Dynamic Composability, which is described later.

6 IMPLEMENTATION PATTERN PROCESS

From a developer's standpoint, the steps associated to creating IF BOMs and supporting ECAP BOMs include:

1. Identifying Patterns based on understanding and analysis of the requirements.
2. Representing the Patterns as IF BOMs
3. Identifying the necessary classes to support an IF BOM, which is captured in the Model Definition element
4. If candidate federates or ECAP BOMs do not exist that support the identified classes, codification of the classes as behavior component models (or ECAP BOMs) os [referred
5. Create necessary ECAP BOM Implementations as needed for specific federate platform/language.

In addition to these steps, it's encouraged that BOM Developers and Users adhere to the Federation Development and Execution Process (FEDEP), which is an IEEE Standard providing a seven step process for creating, maintaining a federation.

7 AUTO BOM EXAMPLE

Now that we've explored the BOM architecture and the application of XML for supporting IF BOMs and ECAP BOMs, let us look at an example of a race car simulation built using BOMs and SRML.

The conceptual entities for this example consist of a race manager, a competitor, a track, and the environment. The roles of each participant can be laid out in a UML sequence diagram as shown in Figure 9.

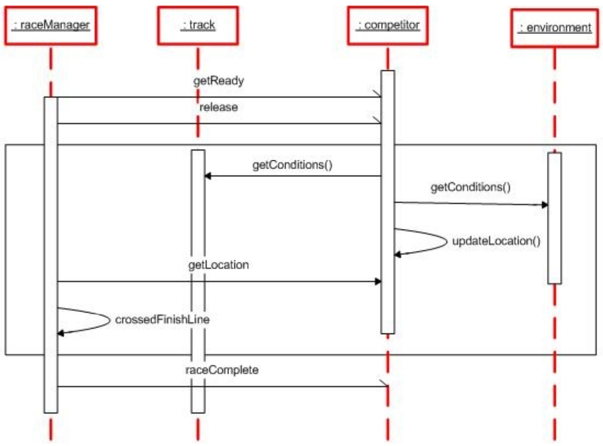
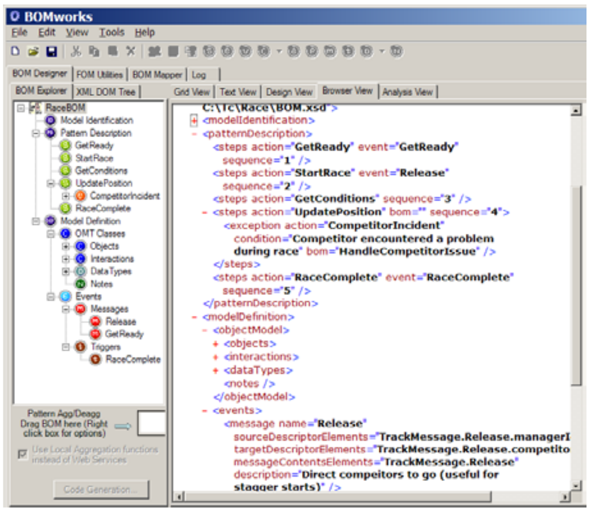


Figure 9: UML Sequence Diagram of Race Pattern

After the roles and sequence of our race pattern are defined, an IF BOM can be constructed which defines the classes, attributes, and events for the race sequence. A portion of the parsed XML representing the race IF BOM is shown in Figure 10 using a tool called BOMworks™.



```
   C:\Tc\Race\BOM.xsd">
+ <modelIdentification>
- <patternDescription>
    <steps action="GetReady" event="GetReady"
      sequence="1" />
    <steps action="StartRace" event="Release"
      sequence="2" />
    <steps action="GetConditions" sequence="3" />
  - <steps action="UpdatePosition" bom="" sequence="4">
      <exception action="CompetitorIncident"
        condition="Competitor encountered a problem
        during race" bom="HandleCompetitorIssue" />
    </steps>
    <steps action="RaceComplete" event="RaceComplete"
      sequence="5" />
  </patternDescription>
- <modelDefinition>
  - <objectModel>
    + <objects>
    + <interactions>
    + <dataTypes>
      <notes />
    </objectModel>
  - <events>
      <message name="Release"
        sourceDescriptorElements="TrackMessage.Release.managerI
        targetDescriptorElements="TrackMessage.Release.competito
        messageContentsElements="TrackMessage.Release"
        description="Direct compeitors to go (useful for
        stagger starts)" />
```

Figure 10: XML Syntax of Race BOM

Next we can begin to create ECAP BOMs to prescribe the behavior needed for modeling the conceptual entities. The various ECAP BOMs created for this example include the following:

- A race manager,
- several types of tracks,
- various types of competitors (including cars and horses), and
- environmental models

These conceptual entities that are intended to be modeled for the race competition can be represented as classes as illustrated in Figure 11.

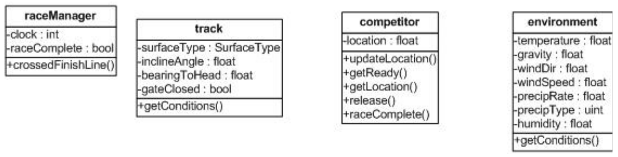


Figure 11: UML Class Diagram of Conceptual Entities

Note: One of the benefits of separating the interface from the encapsulated behavior (the implementation) is that it allows a variety of behavior models to be created and used to support a common conceptual entity within the pattern of interplay. Incidentally, this capability allows for multi-resolution modeling, where each federate can have multiple models to support the appropriate behavior and these models could be of varying resolution.

Once we've captured the behavior description for each ECAP BOM, which includes the conceptual entity state activities that are impacted by the IF BOM events, we can begin to create the necessary ECAP BOM Implementations (EBIs) necessary for our federate to model the conceptual entities. For the purposes of our demonstration represented in this paper, the ECAP BOM Implementations produced are .NET Assemblies developed under the Microsoft Visual Studio environment using C#.

A screen capture of a federate interoperating with other federates that adhere to our Race IF BOM is shown in Figure 12. This federate is capable of dynamically loading the .NET assembly ECAP BOM Implementations (EBI) during execution.

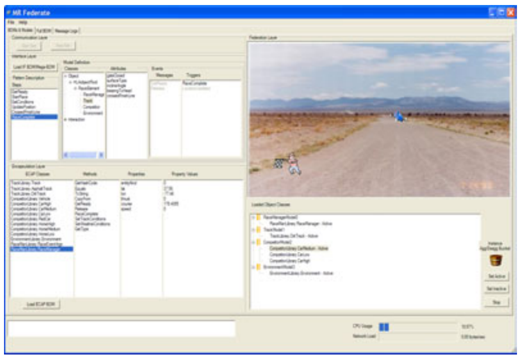


Figure 12: Race Federate Employing .NET EBIs

Another option for our ECAP BOM Implementations (EBIs) is to use the Simulation Reference Markup Language (SRML). Figure 13 below illustrates the execution of SRML-based EBIs executing within a web browser. In this example, a browser plug-in developed by Boeing is used to execute the SRML EBIs for a race manager, track and two competitors.

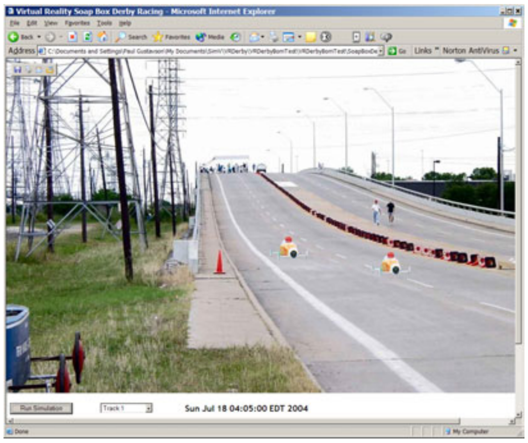

Virtual Reality Soap Box Derby Racing - Microsoft Internet Explorer
Run Simulation
Track 1
Sun Jul 18 04:05:00 EDT 2004

Figure 13: Race Simulation Using SRML in a Browser

8 POTENTIAL OPPORTUNITIES

Let us now explore the potential opportunities BOMs and XML-based standards can provide for supporting composability, web-based simulation and training.

8.1 Enriching Web Services

One of the biggest areas of influence is the reuse aspect and cross platform accessibility that web services provide for both the development network and the federation network. For instance web services can be used to establish a conduit for automation and collaboration and possibly used, earlier on, to support conceptual modeling. The services that are envisioned include web enabled repositories containing and/or providing the following:

- database elements and enumeration information,
- IF BOMs as reusable patterns that can be fetched,
- ECAP BOMs as reusable component models that can be dynamically loaded by federates during an execution,
- 3D models that can be utilized for visually representing entities, and
- the discovery and deployment of self attaching process agents used to support the adaptability of systems based on "different" interfaces (represented using IF BOMs and Mega BOMs).

These agents would enable communication and interoperability among disparate systems and could provide federate support during a federation execution. Web services can also be used to support pattern aggregation (model composition) and dynamic entity aggregation.

8.2 Enabling Adaptability

It has already been recognized that mechanisms are needed to enable federates to participate in unique federations without the manual effort required in re-compiling and re-linking each and every unique Federation, which is represented by a FOM. One of the unique opportunities the BOM architecture provides is the potential ability to facilitate this type of adaptability. For instance, Mega-BOMs produced by BOM compositions can be used to represent the standard data exchange interface for systems and simulations. The opportunity is that this Mega-BOM, or player interface, can be leveraged and used for each federation by mapping to common elements. Adaptability can be accomplished by deploying and applying the appropriate XML-based transformations (XSLT), which reflects mappings between common BOMs within a Mega-BOM, by the receiving federate. This would minimize the effort typically spent in re-tooling federates associated to complying with a specific FOM. Unlike the current HLA approach in which all federates must comply with a common FOM, federates can continue to use their specific Mega-BOM interface to adapt and play within environments comprised of other simulations and systems, which are represented by their own unique Mega-BOM interface.

8.3 Supporting Dynamic Composability

The distribution and dynamic loading of component payloads by players during run-time execution is a potential need. Repositories can serve as a real-time distribution vehicle of ECAP BOM Implementations (EBIs), which can be dynamically loaded by BOM-enabled systems, as illustrated in Figure 14.

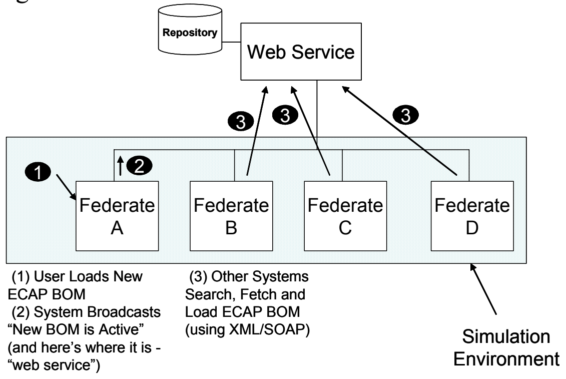

Repository
Web Service
3
3
3
1
2
Federate A
Federate B
Federate C
Federate D
(1) User Loads New ECAP BOM
(2) System Broadcasts "New BOM is Active" (and here's where it is - "web service")
(3) Other Systems Search, Fetch and Load ECAP BOM (using XML/SOAP)
Simulation Environment

Figure 14: ECAP BOM Dynamic Composability

Through the use of web services, next generation federates can request and retrieve the necessary EBIs from a repository, which could then be dynamically loaded during execution by the federate. If execution time would allow, one system that loads a platform specific EBI during an execution could broadcast to other members of a simulation that a new ECAP BOM is being used.

The systems associated to the other members could then take the information, and, if the ECAP BOM capability is not already within their local environment, fetch the same platform specific EBI (in the platform and language) needed from the BOM repository, via XML/SOAP, and dynamically load it.

8.4 Automated Repository

As more and more platform independent ECAP BOMs, commonly referred as PIMs are developed and platform specific ECAP BOMs, which are identifies as PSMs, are generated to support specific languages and platforms, the need for storage increases. Repositories need to be optimized for speed and scalability so that they can be accessed at runtime to support dynamic composabilty. Consider storing only PIMs, and have the repository produce PSMs as needed. A more efficient method of storage would be to store platform independent ECAP BOMs within the BOM repository. These platform-independent ECAP BOMs could be used to create multiple platform-specific ECAP BOMs for various languages and platforms as needed.

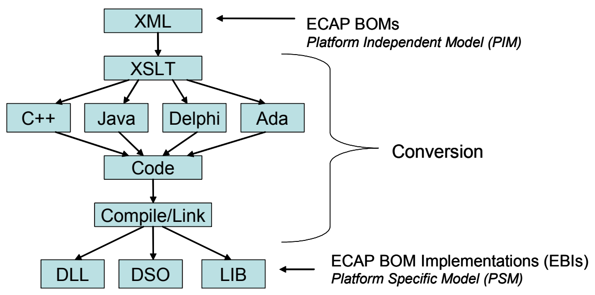

XML
ECAP BOMs
Platform Independent Model (PIM)
XSLT
C++
Java
Delphi
Ada
Code
Compile/Link
Conversion
DLL
DSO
LIB
ECAP BOM Implementations (EBIs)
Platform Specific Model (PSM)

Figure 15: Web-Based ECAP BOM Generation

This capability can be accomplished by embedding a PIM to PSM specific XML Transformation capability as part of the repository. As illustrated in Figure 15, when an on-line federate requests an ECAP BOM Implementation (EBI) from the BOM repository, the web service associated to the repository can locate the equivalent platform independent ECAP BOM (if the specified EBI is not available) and invoke the PIM to PSM transformation. This adds to the Dynamic Composability capability described earlier since the EBI is composed on the fly for a specific need. Once provided to the federate, the EBI can be dynamically loaded by the system. A benefit with this approach is that all subscribers would be using the same compiler and transformation tool provided by the server side application cohabitated with the repository, ensuring common builds for all users. In other words, the production of models is all based on the same validated models.

9 SUMMARY

BOMs are a key enabler for supporting composability in the following specific ways:

- encourages the development and reuse of certified components
- facilitates rapid development of federations and federates in a cost-effective manner
- contributes to conceptual modeling aiding in the identification of shortfalls, gaps and deficiencies
- provides opportunities for greater joint collaboration and feedback, which can support a wide-breadth of initiatives including an M&S enabled Global Information Grid (GIG)

These capabilities are possible because BOMs marry the concept of design patterns with HLA and offers the extensibility and flexibility provided by XML. While HLA is a key technology for establishing interoperability, identifying and codifying Patterns helps us to make sure we have our conceptual models right thereby improving our awareness and analysis needed for supporting the requirements of the warfighter and the systems which we build and use in protecting our interests here and abroad. Furthermore, an XML-based standard allows composabilty to be achieved for any number of platforms and for any number of purposes.

10 ADDITIONAL INFORMATION

Additional information and downloadable material on BOMS can be found on the Base Object Model Specification Information website at `<www.boms.info>`.

ACKNOWLEDGMENTS

This document was prepared based on the work produced by the SISO BOM Product Development Group (PDG) and the results and findings of SimVentions in support of the Defense Modeling and Simulation Office (DMSO) in the area of composability, and for the U.S. Army RDECOM in the areas of aggregation and multi-resolution modeling.

REFERENCES

BOM Product Development Group. July 2004. Base Object Model (BOM) Template Specification Volume I - Interface BOM. SISO-STD-003.1-TRIAL-USE-V0.9. Simulation Interoperability Standards Organization (SISO). Available online via `<http://www.boms.info>` [accessed August 30, 2004].

Reichenthal, S. 18 December 2002. SRML - Simulation Reference Markup Language, *W3C Note*, Available online via `<http://www.w3.org/TR/SRML>` [accessed August 30, 2004].

Gustavson, P., K. Morse, R. Lutz, and S. Reichenthal. April 2004. Applying Design Patterns for Enabling Simulation Interoperability, In *Proceedings of the Spring 2004 Simulation Interoperability Workshop (SIW).* 04S-SIW-111.

Zimmerman, P. DMSO Perspective (Vision). September 2002. Briefed at the XMSF Workshop at George Mason University.

FEDEP Product Development Group. April 2003. High Level Architecture (HLA) Federation Development and Execution Process (FEDEP). Institute of Electrical and Electronics Engineers (IEEE). IEEE 1516.3.

AUTHOR BIOGRAPHIES

PAUL GUSTAVSON is Chief Scientist and co-founder of SimVentions, Inc. He has over 15 years experience supporting a wide variety of modeling and simulation, system engineering, and web technology efforts within the DoD and software development communities. Mr. Gustavson has been a long-time advocate and pioneer of the Base Object Model (BOM) concept for enabling simulation composability, interoperability and reuse. He has also co-authored and edited several software development books and articles related to C++, UML and mobile computing. His e-mail address is `<pgustavson@simventions.com>`.

TRAM CHASE is a software engineer at SimVentions, Inc. (http://www.simventions.com) and is focused on the development and integration of technology for creating innovative and engaging experiences and solutions. In support of BOMs, Tram has been the lead developer of BOMworks™, a tool used to build, edit and compose BOMs. Tram is a graduate of Virginia Tech, with a B.S. in Mathematics (1994), and has supported a wide variety of modeling and simulation and system engineering efforts within the DoD. Tram lives in Virginia with his wife. His e-mail address is `<tchase@simventions.com>`.