THESIS

SAVAGE MODELING ANALYSIS LANGUAGE (SMAL): METADATA FOR TACTICAL SIMULATIONS AND X3D VISUALIZATIONS

by

Travis Rauch

March 2006

Thesis Advisor:	Don Brutzman
Second Reader:	Jeffrey Weekley

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY	**2. REPORT DATE** March 2006	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE: Savage Modeling and Analysis Language (SMAL): Metadata for Tactical Simulations and X3D Visualizations	**5. FUNDING NUMBERS**
6. AUTHOR Travis M. Rauch	
7. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME AND ADDRESS N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT

Visualizing operations environments in three-dimensions is in keeping with the military's drive to increase the speed and accuracy with which warfighters make decisions in the command center and in the field. Three-dimensional (3D) environments support speed in decision-making by presenting complex systems in an integrated, naturalistic display format. Constructing these environments is a time-consuming task requiring specific expertise not typically available in the command center. The future use of 3D visualization in military operations depends on the ability to create virtual environments quickly and accurately by personnel with minimal graphics experience leveraging data available in the command center. It depends on autogeneration.

Assembling and making sense of data necessary to autogenerate a 3D environment requires context and good documentation, best-accomplished using metadata. Metadata supports data-centric, component-based design; key philosophies in promoting interoperability of networked applications.

This thesis proposes an XML metadata standard to collect and organize the information necessary to create and populate a 3D virtual environment. The logical extension of a well-designed standard is the ability to cross the boundaries of usage, allowing simulators to share data with command and control suites and mission planning tools based on the construction of a virtual scene.

14. SUBJECT TERMS 3D, Battlespace Visualization, Autogeneration, Extensible Markup Language, XML, X3D	**15. NUMBER OF PAGES** 264
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

SAVAGE MODELING ANALYSIS LANGUAGE (SMAL): METADATA FOR TACTICAL SIMULATIONS AND X3D VISUALIZATIONS

Travis M. Rauch
Lieutenant Commander, United States Navy
B.A., Lafayette College, 1994

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MODELING, VIRTUAL ENVIRONMENTS, AND SIMULATION (MOVES)

from the

NAVAL POSTGRADUATE SCHOOL
March 2006

Author:	Travis M. Rauch
Approved by:	Don Brutzman Thesis Advisor
	Jeffrey Weekley Second Reader
	Rudy Darken Chair, MOVES Academic Committee

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Visualizing operations environments in three dimensions (3D) supports the warfighters' ability to make rapid, well-informed decisions by presenting complex systems in a naturalistic, integrated display format. Unfortunately, constructing these environments is a time-consuming task requiring specific expertise not typically available in the command center. The future use of 3D visualization in military operations depends on the ability of personnel with minimal graphics experience to create virtual environments quickly and accurately by leveraging data-driven customization of content from model archives with the data available in the command center. Practical 3D visualization depends on standardized scene autogeneration.

The Extensible 3D (X3D) Graphics family of specifications is approved by the International Standards Organization (ISO) as the Web-based format for the interchange and rendering of 3D scenes. Previous work has demonstrated that an archive of X3D scenes, such as the Scenario Authoring and Visualization for Advanced Graphical Environments (SAVAGE) library, can be used to autogenerate sophisticated 3D tactical environments. Assembling and making sense of the data necessary to autogenerate a 3D environment requires context and good documentation, best accomplished through metadata. Metadata also supports data-centric, component-based design; key philosophies in promoting interoperability of networked applications. Coupled with recent developments in X3D, enhanced features of the Savage X3D Model archives are now sufficiently mature to support rapid generation of tactical environments.

This thesis proposes an XML metadata standard to collect and organize the information necessary to create and populate a tactical 3D virtual environment: the Savage Modeling and Analysis Language (SMAL). The logical extension of a well-designed standard is the ability to cross the boundaries of usage, allowing simulators to share data with command and control (C2) suites and mission planning tools based on the construction of a virtual scene. SMAL provides the informational "glue" necessary to perform tactical modeling, simulation, and analysis using networked, physics-based X3D virtual environments.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACRONYMS AND ABBREVIATIONS

2D	Two-Dimensional
3D	Three-Dimensional
AFATDS	Advanced Field Artillery Tactical Data System
AFWA	Air Force Weather Agency
ANSI	American National Standards Institute
AT/FP	Anti-Terrorism/Force Protection
ATO	Air Tasking Order
AUTL	Army/Marine Corps Unified Task List
AUV	Autonomous Underwater Vehicle
BBIE	Basic Business Identification Element
BCS3	Battle Command Sustainment Support System
BIM	Building Informational Model
BML (CBML)	Battle Management Language (Coalition Battle Management Language)
C2	Command and Control
C2IEDM	Command and Control Information Exchange Data Model
C4I	Command, Control, Communications, Computers, and Intelligence
CAD	Computer Aided Design
CAPES	Combined Arms Planning and Execution/Monitoring System
CIO	Chief Information Officer
COG	Center of Gravity
COLLADA	Collaborative Design Activity
COP	Common Operational Picture
DC	Dublin Core
DCMI	Dublin Core Metadata Initiative
DES	Discrete-Event Simulation
DIS	Distributed Interactive Simulation
DM	Decision-Making
DMS	Defense Messaging System
DMSO	Defense Modeling and Simulation Office

DoD	Department of Defense
DoN	Department of the Navy
DTD	Document Type Definition
DTED	Digital Terrain Elevation Data
ESPDU	Entity State Protocol Data Unit
ESS	Embedded Simulation Systems
FAA	Federal Aviation Administration
FAADC2I	Forward Area Air Defense Command, Control and Intelligence
FBCB2	Force XXI Battle Command Brigade and Below
FOUO	For Official Use Only
GML	Geography Markup Language
H-Amin	Human Animation
HCI	Human-Computer Interface
HSI	Human-Systems Integration
HTML	Hypertext Markup Language
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IR	Infrared
ISO	International Standards Organization
JC3IEDM	Joint Command, Control, and Communications Information Exchange Data Model
JMBL	Joint METOC Broker Language
JMPS	Joint Mission Planning Software
MGRS	Military Grid Reference System
MPEG	Moving Pictures Experts Group
MSDL	Military Scenario Description Language
NATO	North Atlantic Treaty Organization
NAVO	Naval Oceanographic Office
NAVFAC	Naval Facilities Command
NC3A	NATO Command, Control, and Consultation Authority
NFESC	Naval Facilities Engineering Service Center
NIRIS	NATO Interoperable Recognized Air Picture Information System

NISO	National Information Standards Organization
NMSO	Navy Modeling and Simulation Office
NVG	Night Vision Goggles
OneSAF	One Semi-Automated Forces
OO	Object-Oriented
OODA	Observe-Orient-Decide-Act
OPORD	Operations Order
OS	Operating System
OWL	Web Ontology Language
PC	Personal Computer
PDU	Protocol Data Unit
PFPS (N-PFPS)	Portable Flight-Planning Software (Navy Portable Flight-Planning Software)
RDF	Resource Description Framework
SA	Situational Awareness
SAVAGE	Scenario Authoring and Visualization for Advanced Graphical Environments
SISO	Simulation Interoperability Standards Organization
SMAL	Savage Modeling and Analysis Language
SOMT	Savage Object Metadata Template
STMT	Savage Terrain Metadata Template
SVMT	Savage Vehicle Metadata Template
TAML	Tactical Assessment Markup Language
TAMPS	Tactical Automated Mission Planning System
TDL	Tactical Data Link
TOPSCENE	Tactical Operations Scene
URI	Universal Resource Identifier
URL	Universal Resource Locator
USW	Undersea Warfare
VCS	Voluntary Consensus Standard
VRML	Virtual Reality Markup Language
W3C	World Wide Web Consortium

X3D	Extensible 3D
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language Transformation
XTC	XML Tactical Chat

ACKNOWLEDGMENTS

This thesis is the end product of two years of both inspiration and frustration. I would like to take this opportunity to thank my family—my daughters Jillian and Ashlyn, and especially my loving wife, Julie—for tolerating the long hours I spent away from them and the general lack of mental focus in the time I spent with them. They kept me on an even keel, and brought me back to earth when my head was lost in the clouds.

Long discussions with my friends and colleagues brought me professional focus and perspective along with the laughs, and for that I'd like to thank them as well: Major Axel Weber, Major Mounir Sidhom, Major Jon Ellis, Captain Michael Martin, Lieutenant Brian Jones, Lieutenant Pat Sullivan, Lieutenant Patricia Sweat, Captain Jasur Khakimbaev, Terry Norbraten, and Alan Hudson.

I'd also like to thank Don Brutzman, my advisor, for giving me the opportunity to think out loud and for keeping me just enough out of my element to encourage me to approach problems in novel ways. Thanks also to my second reader, Jeff Weekley, for helping me take Don with a grain of salt.

The Naval Postgraduate School MOVES Institute gratefully acknowledges support for this research from the Navy Modeling and Simulation Office (NMSO), the Naval Facilities Engineering Service Center (NFESC). Outstanding support and technical insights are also gratefully acknowledged from Yumetech, Inc., Aniviza, Inc., Planet9 Studios, and Sound and Sea Technology, Inc.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. OVERVIEW

Visualizing operations environments in three dimensions (3D) supports the warfighters' ability to make rapid, well-informed decisions by presenting complex systems in a naturalistic, integrated display format. Unfortunately, constructing these environments is a time-consuming task requiring specific expertise not typically available in the command center. The future use of 3D visualization in military operations depends on the ability of personnel with minimal graphics experience to create virtual environments quickly and accurately by leveraging data-driven customization of content from model archives with the data available in the command center. Practical 3D visualization depends on standardized scene autogeneration.

Using Extensible 3D (X3D) Graphics and the Scenario Authoring and Visualization for Advanced Graphical Environments (SAVAGE) Archive, this thesis proposes an Extensible Markup Language (XML) metadata standard to collect and organize the information necessary to create and populate a tactical 3D virtual environment: the Savage Modeling and Analysis Language (SMAL). The logical extension of a well-designed standard is the ability to cross the boundaries of usage, allowing simulators to share data with command and control (C2) suites and mission planning tools based on the construction of a virtual scene. SMAL provides the informational "glue" necessary to perform tactical modeling, simulation, and analysis using networked, physics-based X3D virtual environments.

B. DISCUSSION

Three-dimensional (3D) virtual environments have many potential uses in the military, but constructing them is a time-consuming task requiring specific expertise not typically available in the command center. The future use of 3D visualization in military operations depends on the ability to quickly and accurately create virtual environments by personnel with minimal graphics experience leveraging data available in the command center. It depends on autogeneration.

As technology advances and new graphics solutions are brought to bear, the goal of an automatically created 3D environment has come ever closer. The question becomes, what has hampered the accomplishment of this goal? The answer lies in the way in which the solution has been sought. Individual, proprietary development of separate—but ultimately connected—applications has brought about a stove-piping of the flow of data from source to visualization, or a fundamental lack of understanding of the data that already exists in other forms or in other applications. In the end, it will be the ability to leverage pre-existing data to perform autogeneration that will bring the technology to maturity.

This thesis proposes a solution to at least one part of this problem. There is a definable set of information necessary to develop a useful visual scene that captures the essence of the actual conditions it is meant to portray. The solution is to collect that set of information, standardize its format and contextualize its content, which allows any application to read the data or translate its own data into a form readable by other applications. The logical extension of this interoperability feat is the ability to cross the boundaries of usage. The creation of a 3D scene for a simulation is not much different than the creation of a 3D scene for a command and control suite, which is not much different than the creation 3D scene for a tactical or operational planning tool. Thus great potential value is likely to come with successful implementation of these approaches.

C. MOTIVATION

The continuing development of high speed communications and the types of information on the battlefield has put increased pressure on the warfighter to be able to absorb it all. The development of command and control technology has made great strides in the ability to connect the front-line warrior with the command center, but it has lacked in the area of making all this new information understandable to the commander. The typical military response to this sort of problem is to change or increase the training—in this case for commanders and their staff—to more efficiently operate the equipment. Perhaps, in this case, training is not the most effective solution, but instead improving or augmenting the equipment is preferable. Better methods are needed for improving the fusion and presentation of data coming to the commander.

One possible method of doing this is to develop 3D representations of the battlefield. The problem is in that development of 3D scenes is a labor-intensive process for graphics professionals, typically not suited for the time-pressured, rapidly changing environment of the battlefield. Increasing the speed and ease of construction of 3D scenes can bring visualization technology to the warfighter in ways that are not currently possible augmenting (not replacing) 2D displays. Autogeneration is the key to the next step of battlespace visualization.

D. APPROACHES TAKEN

The road to procedural 3D scene generation has several lanes, down which research has brought the technology to varying levels of achievement. The lane taken by this thesis supports setting up the visualization side of the problem. Many of the research advances have come from the data supply side, standardizing content of orders and battlefield communications. Once communications standardization is at an acceptable level of maturity, the remaining problem will be to translate the data resident in that format to a visual environment.

Not all of the data represented in battlefield communication is required or even wanted for visualization purposes, so defining what *is* needed remains a part of the process bringing the technology to its fruition. Because the actual schemata of battlefield communications are not a completely mature technology, and are to some extent Byzantine, the most straightforward method of determination of this data is to start with the end product, namely the scene. Understanding what goes into a scene to develop a useful visualization allows the user to then categorize and select the most appropriate inputs to feed into it.

It is not always possible to wade through the multitude of data sources available to the warfighter to create a fully representative proof-of-concept solution. Nevertheless, it is possible to use well-documented data provided by a realistic simulation. There are direct correspondences between the world of simulation and the world of C2, so it is possible to determine requirements and test results of visualization-based simulations before attempting to do the same with actual C2 systems.

The development of visualizations from available data is not remarkably different from one usage to the other, so a common interface grammar can be created. Given a common interface grammar, it follows that more of the overlap between simulation and command and control can be explored, possibly creating a well-defined pipeline between the two, which can be used to capture data from command and control equipment for use in developing analytical simulations. Once captured this data will be transportable to any simulation that has knowledge of the grammar and format.

What might eventually result from this development is an avenue of interoperability between simulations and C2 suites, or other simulations, as well as a method to efficiently store scenario data for examination and re-use.

E. THESIS ORGANIZATION

The processes involved in developing this thesis included thought experiments in data collection, a review of available data sets and the formats used, a review of various visualization and data storage formats, an exemplar study of scenario creation in the Anti-Terrorism Force Protection (AT/FP) Toolkit currently under development at the Naval Postgraduate School and, finally, the development of an exemplar data collection format used to gather the information necessary to fully define a scenario in the AT/FP tool. The resultant format presented here is named the Savage Modeling and Analysis Language (SMAL).

Chapter II details the background information gathered and used in the development of the SMAL, and discusses some of the related work in the area of 3D visualization and supporting technologies as well as possible sources of data, which may be used in conjunction with SMAL. Chapter III covers the design principles behind using metadata, designing for interoperability and autogeneration, and the benefits of XML realizes for all of them. Chapter IV walks through the methodology of building 3D scenes and simulations and discusses some of the considerations when attempting to link the two. Chapter V covers the development of SMAL and its supporting technologies. Chapter VI covers some of the standardization requirements for 3D scene autogeneration, and Chapter VII presents conclusions, future avenues of research, and possible uses for this technology in part or as a whole.

Appendix A is the SMAL schema and included sub-schema, Appendix B contains an example document created in SMAL, Appendix C, D, and E are the three Savage Metadata Templates and Appendix F contains some examples of completed templates. Appendix G is the schema documentation generated by <oXygen>.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND AND RELATED WORK

A. INTRODUCTION

Visualizing the operations environment in 3D is in keeping with the military's drive to increase the speed and accuracy with which warfighters make decisions in the command center and on the battlefield. The 3D environment supports speed in decision-making by presenting a complex environment in an integrated naturalistic display format. Assembling and making sense of the data necessary to build a 3D environment requires that it is well documented and presented in context, which is best accomplished through the use of metadata. Metadata supports data-centric, component-based design, which are key philosophies in the effort to promote interoperability of applications in military networks. The metadata necessary for interoperability also supports automatic parsing, which is useful for autogenerating 3D scenes. XML provides an excellent framework for including metadata in the structure of data, and is in broad use in all sectors of government and business.

B. OVERVIEW

There is a steadily increasing interest in computer-generated visualizations in the realm of military planning and operations. Part of this interest is directed toward developing 3D environments to support better understanding of the battlespace by the warfighter. Ideally, these complex environments can be procedurally built and controlled using data already present in command center, saving time and reducing errors through autogeneration. The modern joint and coalition battlefield has degree of heterogeneity heretofore unseen in the business of networking, thus requiring that interoperability is established from the very beginnings of equipment design. The success of interoperability in this sort of environment hinges on open standards and architecture, and a universal understanding of the data that will flow through it. One intriguing method of establishing standardization and understanding is by using a data-centric model with thorough, accessible data markup. This sort of tool is already mature and available in the form of XML.

C. VISUALIZING THE OPERATIONS ENVIRONMENT

In the realms of planning, simulation, and operations, there is an equivalent need for visualization. Human systems interactions tell us that visual representations of complex systems go a long way toward making them understandable. Not only does visual representation increase the level of understanding, it also increases the speed at which understanding comes.[1] When one considers the battlefield as a complex system, the need for enabling rapid, profound understanding becomes readily apparent. It is so apparent, in fact, that understanding of the need for visualization of the battlefield is not a new concept. Battlefield visualization has existed as long as war itself, in the form of wooden models, paper maps, and simple lines in the dirt. What novelty there still exists in the concept comes from the technology used to implement it. Computers and display technology augment the simple map, providing automatic, near-real-time updates and access to a depth of information not possible with ink and paper.

Every modern command and control suite has some form of visualization built into it (or onto it as the case may be) but nearly all of them are completely constrained to the two-dimensional representation of the battlefield. These “augmented maps” rely on the user’s mental modeling skills and pre-existing knowledge of two-dimensional map symbology to get a complete picture of the battlefield environment. As a rule, there is only so much information that can be placed in a two dimensional representation of a 3D world without hopelessly cluttering it with symbols and keys. This depth of information brings with it a drain on mental resources necessary to glean and manipulate the information required to make decisions. The advantage that maps provide in decision-making is precisely what is lost when overloading the medium with information.

1. Decision Making in the Information Age

While command and control technology has increased greatly over the past twenty years, the theory and methodology behind it has not. The publishing of the Organic Design for Command and Control by Colonel John Boyd[2] signaled the last major shift in Decision Making (DM) theory with significant impact on the military.

[1] Macklin, 2001

[2] Boyd, 1987

While there have been refinements and expansions on Boyd's concept, none have had the same paradigm shifting effect. One refinement in particular bears scrutiny in understanding the role of visualization in DM; the Lawson-Moose Cycle.

a. Observe-Orient-Decide-Act (OODA)

In his work, Boyd presented the Observe-Orient-Decide-Act model of DM. The advent of the "OODA loop" (Figure 1) concept drove the development of tactical doctrine toward the speed of DM as the primary focus for gaining battlefield superiority. Research and development for each individual portion of the loop has been cast on improving the speed and reliability of communications and the fusion of data from disparate sources, essentially increasing the rate and quality of data consolidation. This has spurred the advent of Net-Centric warfare and the perceived need for real-time data as the cornerstone of military supremacy. Tactical data links and modems have become standard fare in equipment requirement documents and voice reporting of information is rapidly being replaced with electronic mail and "tactical chat" which can include still imagery or streaming video. Measuring the effectiveness of the decision-making process has become dependent on evaluating the density and correctness of the data in data-rich environments.

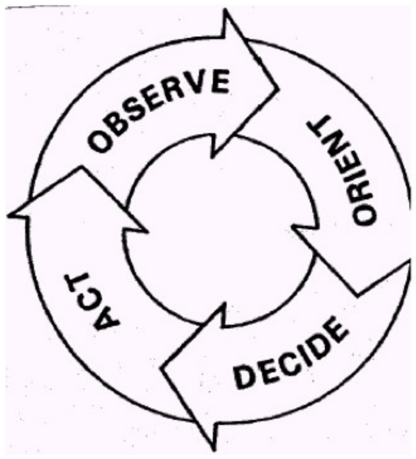


Figure 1. The "Observe-Orient-Decide-Act" or OODA Loop.

The critical second phase of the OODA-loop, Orientation, involves building and maintaining a mental model of the current situation from which possible courses of action can be determined and then decisions can be made.

b. The Lawson-Moose Cycle

Lawson introduced the idea that the entire purpose behind C2 was for the commander to understand and somehow affect change on his environment. This basic relationship at the core of C2 was not specifically addressed in Boyd's model, so Lawson incorporated it in his.

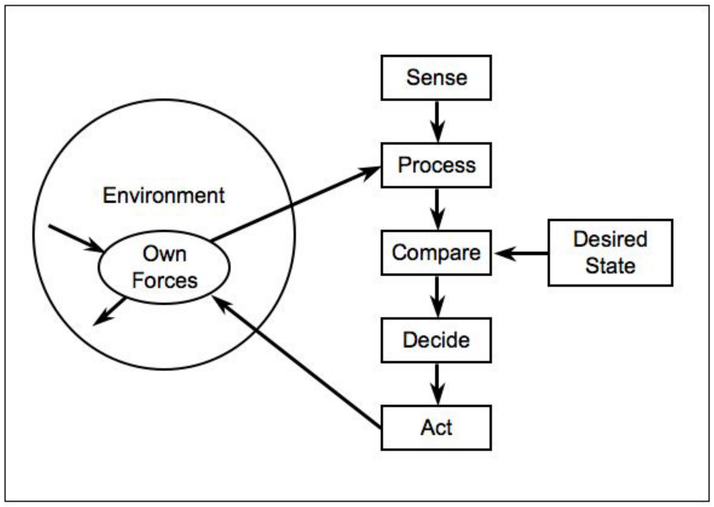


Figure 2. The Lawson-Moose Cycle demonstrating a closer link to the environment as the basis for decision-making.

Lawson broke Boyd's "Observe" and "Orient" steps into "Sense," "Process," and "Compare" reflecting the physical and cognitive differences between collecting data, turning it into information useful for decision-making, and weighing the information against expectations. In terms of C2, this explicit breakdown of cognitive processes is better suited to understand and categorize the basic processes of command than the OODA loop, but it is still a broad model, which must be mapped onto the finer points of real-world C2 support systems.

2. Command and Control (C2)

In both the Boyd and the Lawson model, the majority of C2 processes occur inside the commander's head. Other than the data the commander receives and the orders he or she issues, there is little connection between the cycles and the real world. This interface becomes a key issue in the actual processes of C2 as they relate to operations.

There are seven key functional tasks involved in C2 as identified by Kaye and Galdorisi:[3]

- Focused Sensing and Data Acquisition
- Dynamic Interoperable Connectivity
- Universal Information Access
- Information Operations Assurance
- Consistent Situation Representation
- Distributed Collaboration
- Resource Planning and Management

Sensing, information operations assurance, and resource planning fall outside the scope of this thesis, but the remaining four are rooted in the distribution, collation and display of data. Using the consideration that every user in a C2 network is both a provider and a consumer of data, the four remaining functional areas can be cast in a data model.

a. Dynamic Interoperable Connectivity

Dynamic Interoperable Connectivity involves ensuring every node in a communication network has access to all other data sources and consumers, unblocked by language or system incompatibilities.

b. Universal Information Access

The concept of Universal Information Access prohibits enforced hierarchical or interest-subset filtering of the data available on a network, leaving the individual user to determine what data are important and what is not.

c. Consistent Situation Representation

Consistent Situation Representation defines the Common Operational Picture (COP) concept, in which all participants in a C2 network are able to look at an identical picture of the current situation based on a common set of data, interpreted in a common manner.

d. Distributed Collaboration

Distributed Collaboration requires equivalent access rights to the data on a network at a given command level or job functionality for the purposes of data

[3] Kaye, 2002

manipulation, which enables planning and coordination among dispersed users. Using the two models, it is possible to understand how a system can be built to support the C2 process.

3. Command, Control, Communication, Computers, and Intelligence (C4I)

If C2 is to be effective in a complex, rapidly evolving environment, the communications methods by which C2 is accomplished must also be effective in a complex, rapidly evolving environment. The computers and communications networks needed to transmit and process the data on the battlefield have a heavy impact on the success or failure of an operation, so special attention must be paid to the effectiveness of these systems in the transmission and display of data.

a. C4I Architecture

Current theory behind the design of C4I architecture actually separates the command structure from communications. Command structure is set by the operational requirements of the units involved, and is within the domain of military leaders to design. Communications experts are free to design the communication structure separately, according to user requirements and the specifications and availability of equipment.

The rather loose association between the two is defined by what are called "lines of need." Where a particular "line of need" occurs, such as communication between command elements, the communication architecture must provide at least one path (but ideally, multiple paths) to match. The number and type of communications nodes and the medium over which communications travel are made as transparent as possible to the user.

b. Supporting the Functional Tasks

It stands to reason that a C4I network must adequately support each of the seven functional areas of C2 in order to meet the needs of the commander. In regards to the four areas covered in this thesis, C4I networks must be able to do the following:

- Enforce standards of data format and context or be able to adapt at the user level to the variations that exist.
- Maintain a repository of vetted data in some form—centralized, distributed, or some combination—which can be accessed by users as needed.

- Establish the contextual basis upon which interpretation of the data in the repository can be performed. This is needed to force a logical consistency to the interpretation, no matter what application performs it.
- Control the types of data access (i.e. read, write, delete) a given user has to the repository, and determine what type and level of processing must be applied to data received from that user.

c. The Display of Information

In order to augment the warfighter's situational awareness, computer tactical data displays have become a heavily integrated part of C4I equipment suites. The fluid nature of the battlefield coupled with the real-time data revolution has resulted in mountains of data that must be navigated to maintain an accurate situational model. As such, C4I suites have made a parallel shift to automation, which leaves no choice but to maintain a visual model (abstract or concrete) "in the world". The display and manipulation of data on a computer has become integral to large-scale command and control operations and the demand for increased speed of decision-making requires that the data must be rapidly and accurately digested.

Collation and display of data has had to be automated in order to keep up with the sheer volume of data entering the command center. This requires equipment and methodologies which essentially autogenerate a picture from disparate sources of data. In a well-established network, data inputs are all of known format and content, which are then checked for accuracy and associated in time and space to create an integrated picture. Once integration has been accomplished, all that remains is to sort the data into what must be displayed in a raw form (text) and that which can be translated into other formats (visual in the form of lights and symbology or auditory in the form of horns, bells, and spoken words). The translation of data into visual format is meant to distill data into a quickly and easily digestible form, whereas the auditory format is primarily reserved for emphasis or cueing.

4. Operational and Tactical Planning

Planning for operations, either deliberate or as crisis action response, is an integral part of C2, and one that is not particularly well supported graphically. It stands to reason that planning is most naturally be done, if not on the C4I equipment itself, then using an interface that closely matches it.

In the realm of aviation tactical planning, there exists many separate mission planning software suites, each with a particular strong suit. The PC-based Portable Flight Planning System (PFPS) and the follow-on system Joint Mission Planning Software (JMPS), in use by the Air Force and Navy, are mission planning software suites using electronic charts and Digital Terrain Elevation Data (DTED) for two- and 3D visual review capability. The older, UNIX-based Tactical Automated Mission Planning Software (TAMPS) is an integrated 3D flight planning and rehearsal system designed to interface with aircraft mission computers to upload flight, weapons, and systems data. Tactical Operation Scene (TOPSCENE) is a well-integrated visualization suite which can be used to “fly-through” JMPS, PFPS, or TAMPS generated mission sets using real imagery, 3D cultural features, and alternate viewing modes (simulated Night-Vision Goggle (NVG), infrared (IR), and radar). Each of these tools is well designed and executed within its area of specialty, given the training and contractor support necessary to operate it.

One possible reason why this situation exists is the difficulty inherent in scenario development. Where the aforementioned applications and others like them fall apart is their separation from the warfighter, either by complexity or cost. When a soldier or Marine sits down to plan an operation in the field, he or she still uses the “sandbox” approach, essentially using sticks and bottle-caps to rough out a scene in the dirt, which is then used for planning and orientation. Ideally, a warfighter is able to sit down in front of a computer screen and, using his or her own parlance, be able to construct a scene or scenario that is close enough to the real-world equivalent to be useful for planning or rehearsal. Parlance can be defined as the terms and context in which the warfighter normally receives and understands the data that surrounds planning. Given relief from complexity, there still remains the issue of cost. TAMPS is delivered as a complete, standalone workstation with specialized equipment. Its replacements are PC-based and procured under an enterprise license, and will run on any relatively up-to-date computer in the DoD.

D. THREE-DIMENSIONAL (3D) VIRTUAL ENVIRONMENTS

The preponderance of battlespace visualizations in both real-world and simulated environments is two-dimensional, approximating a paper map or chart. While this type of visualization is familiar and long since standardized, it requires a great degree of training and experience to be able to properly read and translate it to a mental model useful to a planner or warfighter.[4] 3D representations of the real world environment are an order of magnitude easier to translate into a mental model, since the representation more closely approximates the real world. Numerous studies over the past twenty years have developed a set of standards for designing—or evaluating the efficacy of—a human-computer interface. Several studies suggest that 3D displays are not appropriate for certain types of interfaces. In the 2001 Human Factors Design Guideline report from the Federal Aviation Administration (FAA), three dimensional displays were found to be inferior to two dimensional displays for a number of tasks “requiring focus on a single axis,” but were found to be superior for “a visual search and a tracking task performed using [traffic displays]” and “three-dimensional, spatial, dynamic task[s]”.[5] As a rule, integrated tasks get the most benefit from 3D displays, whereas tasks that require a single channel of attention and close tolerances benefit least. This suggests that for most applications, 3D displays are a value-added component to a system, not a central requirement. It follows that 3D displays are not be considered a replacement for two-dimensional displays, but rather they will be used to augment or be augmented by two-dimensional displays.

Understanding this, it remains advantageous to gather data for construction of a 3D environment even if it is not actually rendered as a 3D scene. Because 3D scenes require more data to create, they can be distilled to any number of two-dimensional projections as required.

1. Human Computer Interaction (HCI)

3D visualization takes the extra step of mentally transferring the abstract symbology of two-dimensional representations out of the process of creating a mental model of a situation. 2D representations of terrain use abstract symbology and coloration

[4] Macklin, 1998

[5] Mejdal, 2001, p. 38

to represent the third dimension. Two-dimensional displays depicting highly spatialized environments such as air traffic control use a “two-and-a-half-dimensional” representation, replacing depth or field with a numerical or symbolic representation of altitude. The process of translating data into a clear, concise, and easily readable visual format has become a science in the information age—Human-Computer Interaction (HCI), a sub-specialty of Human Systems Integration (HSI). “Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them.” [6]

2. Collaborative Situational Awareness (SA)

The current trend among the military services is to increase the utilization of network enabled applications to collaborate or share information among units separated in space from one another. The idea is to give a common situational picture to the various units participating in an exercise or operation. With a common picture of the battlespace, each participating unit requires less specific direction from higher authority for matters of de-confliction or mutual support. This can decrease the volume of verbal or written traffic on the various communication networks.

E. RELATED XML-BASED LANGUAGES

XML and data management tools abound in the graphics industry and in the military, this chapter describes a set of example schemata and a data management concept that bear some conceptual relationship to the work in this thesis.

XML has been readily accepted in government and industry as a solution to interoperability and reuse issues associated with data. So it is not coincidence that a majority of the work related to the development of SMAL and its supporting technologies is also in XML. The idea that XML would be used to solve a problem—which is presented in some part by the proliferation of data enabled by XML—is also not surprising, given the enthusiasm with which its proponents extol its virtues. It has been said that XML is like violence; if a little doesn’t solve the problem, use more. The work

[6] Hewitt, 2001, p. 6

listed in this chapter falls under three categories, data aggregation, data sources, or data management. The first two are subcategories of XML, while the third is a concept rather than a format onto itself.

Developed in 1996, XML is hardly a new technology, and it has rapidly infused government and industry, but it is not without its cautionary tales. Use cases abound for XML, each meeting with success or failure based largely on its applicability to the exchange of data between provider and consumer. The creation of a schema does not guarantee an improvement in the exchange of data, only that it is structured. The following is an incomplete list of current XML schemas designed for broad use in various communities. They represent a portion of the possible naming conventions upon which information are drawn to create new schema for autogenerated battlespace visualization.

1. Simulation and Visualization

a. Military Scenario Definition Language (MSDL) for Simulation

MSDL is a schema currently under review—nearing acceptance as of 2006—by a multi-national group of interested parties from government, industry, and the military under the banner of the Simulation Interoperability Standards Organization (SISO). It is designed to fully describe the initialization parameters for a military scenario in a way that is application independent and:

- Improves Scenario Quality
- Reduces Time to Develop Scenarios
- Reduces Costs of Developing Scenarios
- Reduces Tool Development Costs
- Improves Tool Interoperability[7]

MSDL is an in-depth and all-encompassing review of military simulation. As such it has extremely limited use outside of military circles. Its primary strength has been its focus on the broadest usability inside those circles. Its naming and structure are based on the Command and Control Information Exchange Data Model, (C2IEDM), the Battle Management Language (BML), the Coalition BML (CBML), the Army and Marine Corps Universal Task List (AUTL), and what NATO standards are applicable. It

[7] Lacy, 2001

has as one of its goals, the ability to link simulation technologies like the Army's One Semi-Automated Force (OneSAF) to current C4I suites (FBCB2, AFATDS, BCM3, etc.).

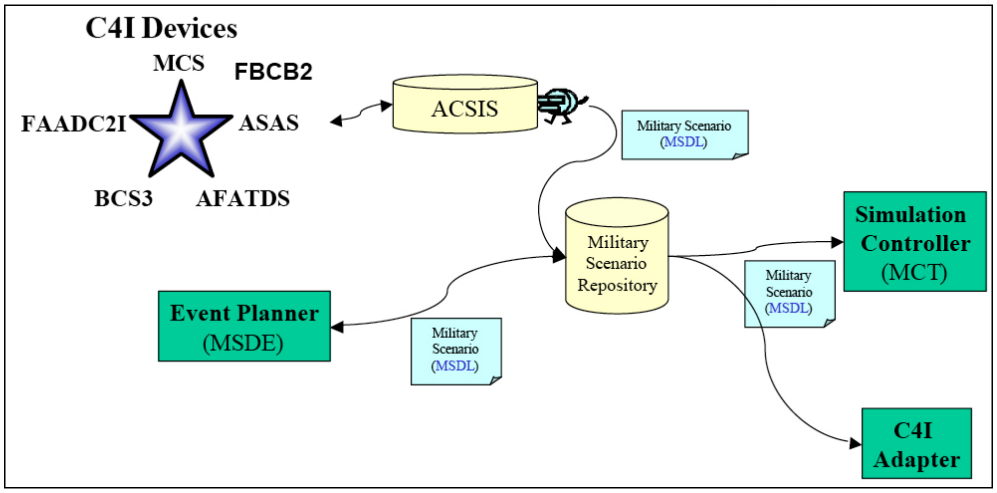


Figure 3. Military Scenario Definition Language (MSDL) is used to link C4I equipment and simulators by providing identical initialization data.[8]

b. *Extensible 3D (X3D)*

The premiere XML schema designed for the creation and display of 3D scenes, X3D is an ISO standard (ISO/IEC FDIS 19775) created by the Web3D Consortium. As an open-source, open-standards, XML-based technology it is data-centric, platform-independent, and web-portable. Its design was modeled after the Virtual Reality Markup Language (VRML) 2.0 standard with the added benefit of extensibility, making it a powerful tool for application independent 3D scene development. It is an expanding standard, recently adding extensions for shader compatibility and continuing development on Computer Aided Design (CAD) visualization support. Because it is data-centric, it utilizes data controlled software (browsers) to visualize its content, which is both its great strength and primary weakness. In order to get consistent results between browsers and between platforms on a given browser, application developers must meet strict conformance standards, which is not always the case. On the other hand, properly conforming browsers are capable of supporting X3D scene-author developed prototype functionality immediately without updating software. X3D was chosen as the visualization technology to be used by the

[8] ACSIS, located at the top of the figure, is the Army C4ISR and Simulation Initialization System.

Autonomous Underwater Vehicle (AUV) Workbench and Anti-Terrorism Force Protection (AT/FP) Toolkit projects at the Naval Postgraduate School for its scene graph-based ease of authoring and web-portability. The concurrent development of the AT/FP Toolkit using X3D makes it an excellent candidate testbed for the technology developed in this thesis. In the following chapters, X3D examples will be used when discussing 3D scene development, and the AT/FP Toolkit will figure prominently in discussing Simulation development.

c. Collaborative Design Activity (COLLADA)

The Khronos Group, a consortium of graphics companies and academics, has sponsored the development of a "schema for the source data for…digital assets," especially games. COLLADA is a schema designed to facilitate the packaging and exchange of digital assets such as 3D models, graphics, shader effects, and physics engine controller code. It is designed to capture the metadata necessary for an application to understand the construction and layout of this data no matter what application was used to generate it. It is reportedly an open source project, and caters primarily to 3D graphics artists and application developers helping to add enough metadata to an asset to overcome the difficulties conflicting standards and formats (which abound in the industry) present to interoperability.

2. Battlefield Information Exchange

a. Joint Command Control and Communication Information Exchange Data Model (JC3IEDM) and C2IEDM

JC3IEDM is a follow on project to C2IEDM, developed by the Multilateral Interoperability Programme, and international consortium of military C2 support commands designed to enhance the joint capabilities of C2IEDM. The two data models were built with the concept of standardizing data exchange between joint warfighting commands on a broad range of combat functional areas. The C2IEDM came into service in 2003 after nearly ten years in development with a special emphasis on the ability to translate the communications between legacy C2 systems from different countries. For this reason, C2IEDM is not meant to be a language of use, but instead act as a language for intermediate translation, allowing legacy applications to map their data to C2IEDM, which can then be translated into any of the other legacy systems.

JC3IEDM is designed to build on the concept and broaden its application by merging it with the NATO Reference Model standard, which was built in parallel.

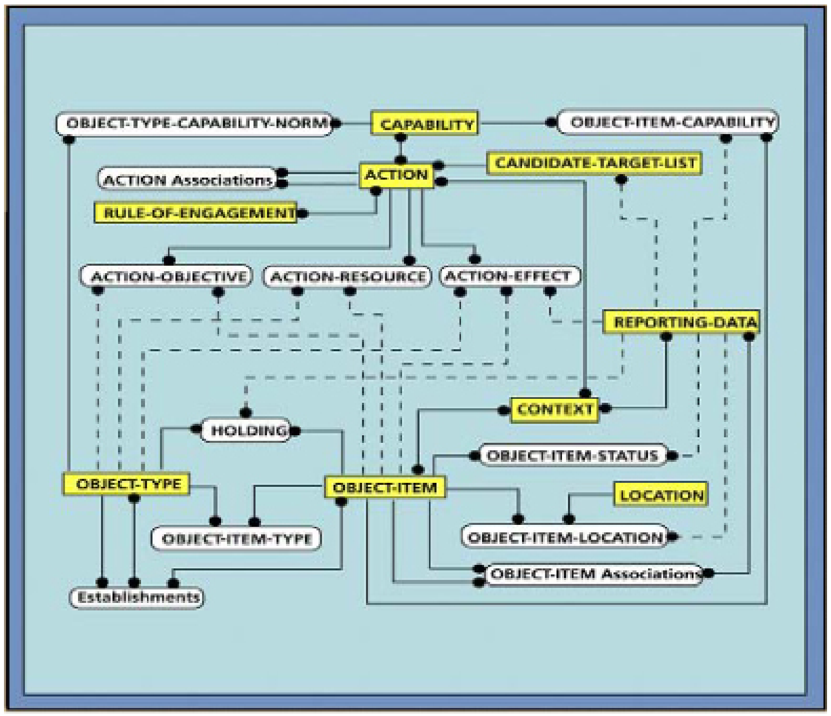


Figure 4. High-Level Overview of Command and Control Information Exchange Data Model (C2IEDM) Classes and their Interrelationships Showing the Abstract Nature of the Language[9]

b. *Battle Management Language (BML)*

Where C2IEDM and JC3IEDM are high languages, which describe, but do not participate in, actual combat operations, BML is to become the common language of battle. BML is destined to be the standard language of communication between and among C4I elements, simulations, and robotic forces. It attempts to codify the current BML, which is essentially human interaction through verbal and written orders, and match it with the type of BML found in simulation. This work is inspired by the same ideas as Kleiner et al, in that there is a drive to standardize free-text to make it less ambiguous to people and more easily parsed by computer. Figure 7 shows the comparative use cases for BML and MSDL. BML and MSDL are related technologies

[9] Johnson, 2004

when used for the purpose of linking live-virtual-constructive simulations for training. MSDL provides the initialization data for all three participant types and BML handles the real-time data interchange.

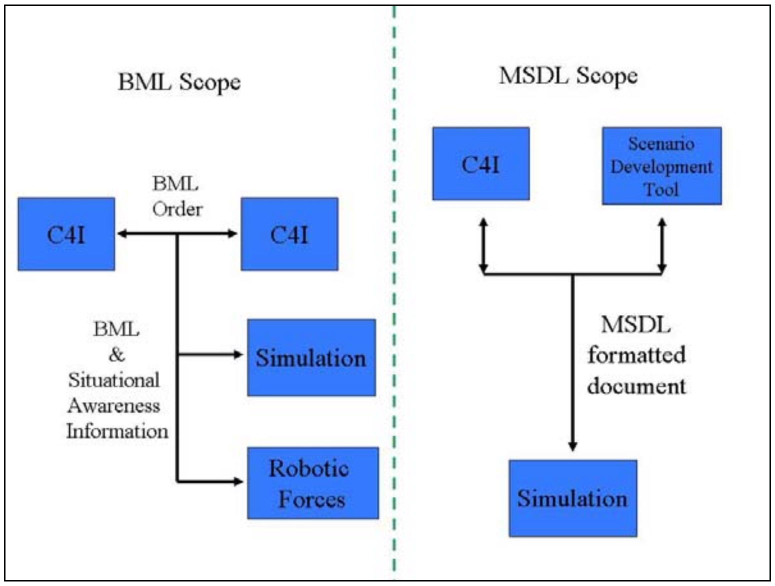


Figure 5. Battle Management Language (BML) & Military Scenarion Definition Language (MSDL) scopes of application showing BML as communication transport and MSDL as intermediary document.

c. *Tactical Assessment Markup Language (TAML)*

The product of an effort by the Under Sea Warfare (USW) XML Working Group, TAML is a grammar designed to represent tactical track information with the context necessary to make it understandable to a wide range of applications. The Working Group is made up of representatives from the Anti-Submarine Warfare, Under Sea Warfare, and Carrier Tactical Support communities. The language was built using a consensus representation approach, attempting to capture the widest acceptance among the naval warfare communities, eventually providing the major avenue of interoperability between them.[10]

d. *NATO XML for Link-16 Data*

NATO Consultation, Command and Control Agency (NC3A) is in development of a set of XML schema and web-services used to transport Tactical Data Link (TDL) information across networks. It began in 1994 as an ad-hoc setup to allow

[10] Brutzman, 2006

NATO commanders in Brussels to view in-theater Link-11 data from the Bosnian Theater of operations, which is not possible over the traditional radio-net upon which Link-11 works. The utility of sharing tactical data between distant elements became apparent, and the NC3A started the NATO Interoperable Recognized Air and Surface Picture (RASP) Information System (NIRIS) to create this functionality for Link-16.

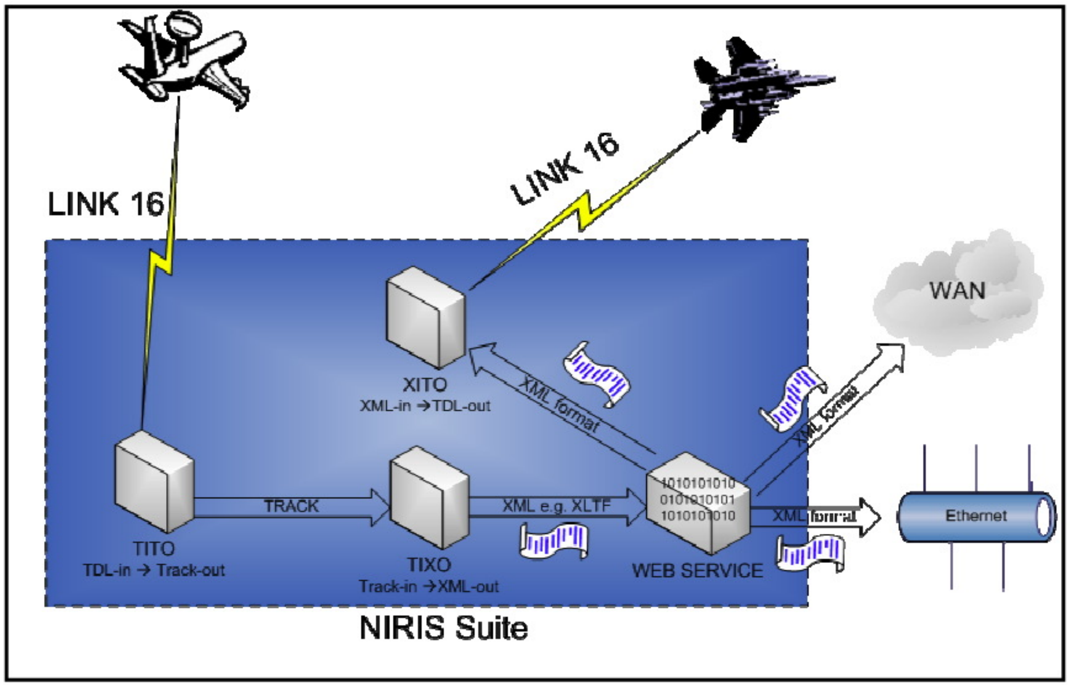


Figure 6. XML translation of Link-16 designed to expand Link-16 functionality by making it network-transportable.[11]

e. XML Tactical Chat (XTC)

The use of tactical chat has grown rapidly in the past several years, but the transport mechanism in text-based Internet Relay Chat lacks structure, making it difficult to leverage the data implicit in the chat. The Jabber Extensible M Packet Protocol has emerged as the preferred standard. XML structure of messaging allows procedural storage, cataloging, and search of messages for specific text. The XML will likely be suitable as a transport for other XML code as a payload for distribution to networked collaborators.

[11] Bayer, 2005, p. 54

3. Supporting Data and Metadata Standards

a. Dublin Core Metadata Initiative (DCMI)

DCMI was created in an effort to develop a "standard for cross-domain information resource description" through the use of metadata. The Simple DC schema (ISO 15836) contains only fifteen elements, which are used to establish identity and authorship of a particular resource. The larger Qualified schema contains refinement elements, enumerations, and different encoding schemes to better describe a resource. The DCMI is a self-described consensus group, not a standards organization, so it must work within the framework of another organization to publish standards. It currently works with the Internet Engineering Task Force (IETF) as well as several other standards organizations to publish its standards. The Simple DC schema is one of the smallest in widespread use, which is a testament to the process used to distill the properties required for identification down to their bare minimum while still retaining usability. The Dublin Core schema is the prototype behind the development of the World-Wide Web Consortium (W3C) Resource Description Format.

b. Resource Description Framework (RDF)

Less a related work, than a possible avenue of advance for SMAL, RDF was developed by the W3C as a metadata standard for web-based resources (W3C Recommendation, 10 February 2004). It is in part a refinement of the Uniform Resource Locator (URL)—the ubiquitous method of addressing for the web—called a Uniform Resource Identifier (URI). The URI is how properties and property values are described over the web, and they are not limited to web-retrievable information; they can also describe other resources that are not on the web, such as books, people, etc. It is also in part a refinement of XML, called RDF/XML designed to remove the strict ordering requirement of XML elements and to be able to describe all the URIs associated with a web resource in a single block of code. The combination of these two refinements allows applications to process web-based content descriptions in order to enable procedural resource discovery. "Reasoners" are the applications used to process query requests and build resource lists for delivery to an end user. As long as the reasoner is appropriately configured, its operation is relatively transparent to the end users of RDF.

c. Web Ontology Language (OWL)

Built on top of RDF, OWL goes beyond describing resources on the web to enable processing information about the web. Unlike RDF, it is not meant to be human-readable; it is meant to allow computers to read and interpret data for the purpose of integrating information. Creating an OWL for a particular domain on the web involves defining classes of information and the properties within those classes. Once the formal semantics are set, a logical process can be applied to derive further information about them.

d. The MPEG-7 Video Metadata Standard

There was a DCMI working group established in 1999 which explored the creation of a set of XML sub-elements derived from the Dublin Core set of elements to annotate digital video content, but it was deactivated in 2002. In 2004, the Moving Pictures Experts Group (MPEG), through ISO, was the first to establish a video markup standard, referred to as MPEG-7 (ISO/IEC TR 15938 series). It is beginning to take hold in the video industry in an attempt to track the huge repositories of digital video data that have developed. MPEG-7 is actually a set of relatively loose standards against which "profile schemas" are created to standardize the development of individual industry versions. The idea is that if a candidate schema is validated against the profile schema, it would be considered valid against the MPEG-7 standard. The problem is, it might still be valid against the standard and not validate under the profile schema. This defeats much of the purpose of strongly typing and strictly validating data for interoperability. The content of the MPEG-7 standard annotates format, scenes, shots, and time-stamps, audio specific (range, key, tempo, sampling frequency, etc.), visual specific (color, shape, texture, motion, face-recognition, etc.), and multi-media specific (content, media features, navigation summaries, etc.). No industry-wide effort remains in effect, leaving a fractured set of slightly inconsistent "standards" among the various digital video producers. The lack of tight standardization can either make full discovery of resources much more difficult and inconsistent, or promote vendor lock-in for large-scale video resource managers.

e. Geography Markup Language (GML)

The OpenGIS Consortium developed the GML schema from the ISO 19118 standard for transport and storage of geographic information. GML itself (ISO/DIS 19136) was designed as a data transport standard, but left its use as a storage format to the discretion of the content developer. The grammar describes geographic and cultural features, locations, coordinate systems, map, chart, and imagery types, storage format, and extent of coverage. There are detailed sets of elements and attributes designed to hold information for geographic phenomena such as curvature anomalies, and magnetic phenomena, well beyond the usage of the typical map user. On the other hand, information important to the typical map user *can* be gleaned from the data available in a GML document if necessary. For that reason the typical layperson's use of GML is as a resource discovery tool for geospatial data.

f. Joint METOC Broker Language (JMBL)

JMBL was created by the DoD Meterology and Oceanography Data Administration with input from the Navy, Marine Corps, Air Force, and Army. It attempts to define a language of transaction for weather products including—but not limited to—forecasts, observations, imagery, and climatology. The standard suffers from its wide scope of purpose, since the schema is expected to accommodate all use cases rather than a well-defined subset. The schema committee is constantly under pressure to include more use-cases that are found unaddressed by its current iteration. Early adoption of the schema by the Air Force Weather Agency (AFWA) and the Naval Oceanographic Office (NAVO) put counter pressure on the committee to keep the schema as it is or at least keep it backward compatible. The language is also aimed at weather professionals who have in-depth knowledge of the forecast models and observation types would be necessary to include in a web-query to extract the right information for a given need. As such, the broker language is not for use by end-users of weather products like pilots, operational planners, and other warfighters.

F. BUILDING INFORMATIONAL MODEL (BIM) APPROACHES

Placing information into models is also not a new idea. A current effort with a similar goal as SMAL and SAVAGE Metadata but with an extremely different scope is the Building Information Model (BIM) project. This concept, which was developed nearly 30 years ago by Charles Eastman of the Georgia Tech College of Architecture and Computing, is being used by Planet9, Inc. and the Naval Facilities Command (NAVFAC) to put together a method of storing and accessing detailed engineering and structural data about buildings, bridges, and other infrastructure on Naval facilities using 3D models. The model is designed to mimic the actual structures and provide information in much the same way the building would provide in a physical survey, including things like maintenance schedules, schematics, and materials, but doing so through links with relational databases.

The process of combining the data in one location, accessed through one interface, forces data consumers to fully develop an understanding of their need, and providers to standardize the context and format of the data they present to reduce or eliminate incompatibilities and inconsistent information about any given item.

This metadata architecture brings with it the requirements, 1) that the data in the database is well documented, just like any other data application, so the consumer is able to understand its context and use it appropriately, and 2) the precise makeup of the data set contents is available in a way that is both discoverable and understandable to the consumer. This concept is highly compatible with the RDF/OWL initiatives currently underway in the web-services arena.

The greatest benefit of this type of data linkage is the practicality of distributed database maintenance. A trusted web-service data provider maintains the database over which it has expertise, relieving the consumer from the problem of keeping local databases up-to-date. As long as the provider maintains a consistent pathway to the data, it will be accessible to the model user. Further work is recommended to compare and contrast BIM approaches with X3D and SMAL.

G. SUMMARY

3D visualization promotes rapid understanding of complex systems, making it a potential tool for battlefield command and control visualization. Virtual environments relieve some of the cognitive load from the commander and, if properly interconnected, can increase collaborative situational awareness. XML languages abound in the military and industry, some of which have applicability to battlefield visualization.

III. DESIGN PRINCIPLES

A. INTRODUCTION

In order to make the best use of data, it is necessary to have an understanding of its format and context. If a system keeps its data exposed and well documented, it is more likely to be highly interoperable, since it is the data, and not the application, that has the most value. The methodology of component-based design places data separate from logic and interface, which puts the emphasis on the architecture of the application rather than its component parts, including the data. The same qualities that make a data-centric or component-based system interoperable, also make it well-suited to supporting autogeneration. The interface and data parts of component-based design are both well-suited to the use of XML.

B. DATA AND METADATA

So what is involved in leveraging data? There are only two things: format and context. Format involves the form and method in which data are organized and stored; how it is *written to* a file. This presents a problem wherever databases are used, proprietary formats can, at best, be unreadable to other applications using the same media and, at worst, be unreachable because they are maintained within an application instead of separate from it or even on a completely different kind of media (e.g. punch cards or paper tape). The second, context, involves the understanding that come with the data, the circumstances and conditions that allows the data to make sense; how it is *read from* a file. A number is useless without context. Looking at a number without context brings up the questions, “What kind of number is this, decimal, hexadecimal, octal?” “Is this number a value or a reference?” “If this number is a value, what is it evaluating?” “What are its units of evaluation?” “If this number is a reference, to what does it refer?” Lists of numbers are even more ambiguous, since there is no inherent way of knowing if the list is meant to represent different values describing different things, the same value describing different things, or different values describing the same thing.

1. How Metadata Adds Utility to Data

The simple answer is, “metadata *is* context.” It is defined as, "structured information that describes, explains, locates, or otherwise makes it easier to retrieve, use, or manage an information resource. Metadata is often called data about data or information about information."[12] By providing illuminating facts about the data—such as type, units, use, and source—it can make data more understandable to the detached observer or new user. Developing a coherent set of metadata can increase the usage of data simply by making it accessible and understandable to more potential users. Metadata can also allow a user to differentiate between usable and unusable data based on source or type. If it is not usable, knowing why may provide the ability for it to be corrected or translated so that it is.

2. Standards of Practice for Creation of Metadata

Ideally, metadata turns implicit information into explicit data, which removes ambiguity. Determining what metadata to include with a particular set of data has two major standards considerations, naming and format. it is important to consider the widest possible audience, meaning it can cross usage boundaries if at all possible

C. STANDARDS OF INTEROPERABILITY

A number of methodologies have arisen to implement the goal of interoperability, but the most flexible and useful of these is the idea of separating interface from application from data. Establishing the working data set as a separate entity from the application and the interface, there exists the ability to access the database without the intervention or knowledge of the code involved in the application portion of the program. Additionally, the creation of an interface separate from the application reduces the application's computational overhead and allows for multiple interfaces to be developed without altering the core application. This can be useful for creating multiple language-specific interfaces, or simultaneously pulling different sets of output for display out of a common application.

1. Data-centric Applications

One of the greatest difficulties in transitioning to new applications is the transfer of data from an old application into a new one. Many legacy systems were built to contain the data as well as operate on it, effectively hiding the data from outside use, preventing easy upgrade or migration, and—most troubling of all—preventing access for the purpose of data sharing. In this type of programming, the application is viewed as the

[12] NISO

key asset, which leads easily to proprietary control of code and data. The antithesis to this idea is the data-centric application, in which the data are viewed as the key asset. The thought is, the program that manipulates the data is replaceable at a given cost, but the data itself—if lost—is irreplaceable. In effect, application-based programming of databases holds the data hostage to the program and its creator, requiring data flow in and out of the database through the database itself in the form chosen by (or requested of) the creator.

The first and most important step in making an application work with others is exposing the data. In the data-centric approach, data are separate and so it is by definition exposed. The exposure of data for use by other applications can be accompanied by an open, standardized format for anyone to use, or by a proprietary format, which must be reverse engineered or purchased for use by other applications through licensing. The business model behind these two cases is not a subject of this thesis, but it is the first model which is most favored by the Armed Services for both current development and future work and is the second step in creating truly interoperable applications.

Once the data are exposed and available for use, it is merely a matter of accessing and manipulating the data. If an application can write the data to an external database, it can also send it out over the network. If it can send it out over the network, it can send it in a standardized, contextualized format (just like the database population), which can be read by other applications. If the format for the database is too large or unwieldy for network use, any separate network format used needs to at least be able to be translated to and from the database format and have similar properties of standardization and contextualization in order to preserve a chain of understanding. At worst other applications would be able to know the database format and the network format and perform its own translations as needed to faithfully replicate the data for its own uses.

2. Component-Based Design

A parallel movement to the Object-Oriented (OO) school of design is Component-Based design. In a broad sense, component based design is a companion concept or an extension to object-oriented design in that it works outside a particular program or code-base. The idea is that a component is a complete “black-box” with no pressing architectural requirement to know what occurs within its processes. The key to development in Component-Based Design is the "loose-coupling" of its various parts, which require no knowledge of the workings of other components and no modifications required to interface with them. As a result, mapping tasks to components, developing a well-defined set of common interfaces, and establishing a communications standard over which the interfaces operate are the primary tasks involved in Component-Based design.[13]

This concept can be extended to the larger classes of program functionalities as well. Any application can be componentized along the following lines: Interface, Logic, and Data. The more easily an application can be split into these three portions, the more closely it can be said to follow the component-based design. The separation of logic and interface requires a well-defined communication standard between the two components, such as the Distributed Interactive Simulation (DIS) protocol. The separation of logic and interface from data requires a well-defined method of communication between them as well, hence the establishment of visualization metadata standards.

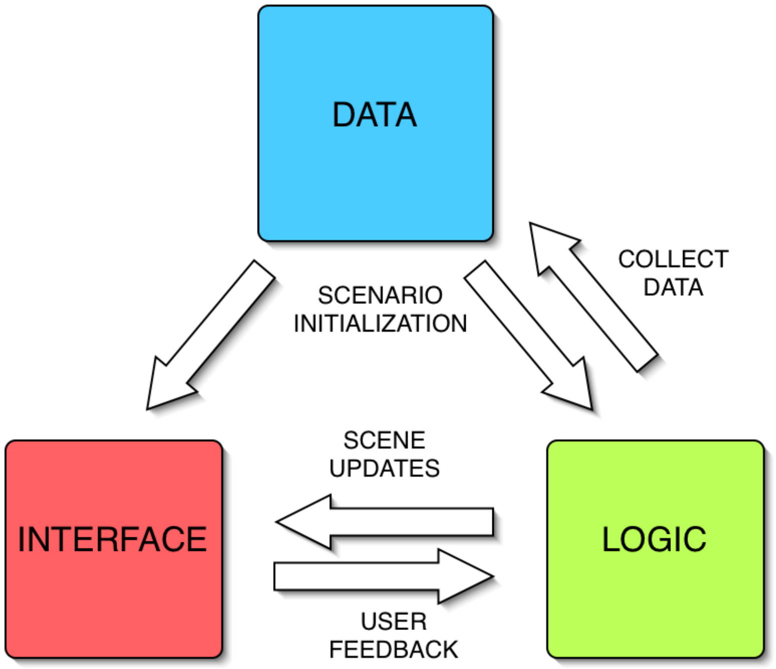


Figure 7. The Component-Based high-level design of a simulation and the interaction between the components.

[13] Buss, 2000

3. C4I Interoperability with Simulation

Developing rapid and accurate understanding of large amounts of data is part training issue and part interface issue; the better and more intuitive the interface, the less training will be necessary for a basic level of competency and the more quickly users will be able to advance to proficiency.

The separate architecture approach to C4I makes it possible to insert simulation control elements into the communication net either to monitor operations, drive sensor readouts, or provide simulated opposition forces, as required, for the purposes of training. For network architectures without embedded simulation capability, it is necessary to "trick" the network by replacing what would normally be another sensor node with simulation equipment that is designed to send properly formatted data to the other nodes and cause them to display simulated artifacts. For legacy applications, this can require substantial reverse engineering and testing for successful implementation. Networks with multiple equipment types may additionally require large amounts of code to translate for multiple operating systems, languages, or data format requirements.

Systems with built-in simulation capability either in the hardware or software make it relatively easy to set up and conduct distributed simulator training. While heterogeneous equipment types remain a challenge, embedded simulation technology has shown that one of the largest difficulties in networked simulation and training can be removed. Simulations that can be run directly on the C4I equipment itself, using an interface that feeds simulator data into the real-world consoles, creates opportunity for excellent training. Some modern equipment, such as that created under the Army's Embedded Simulation Systems (ESS) initiative, not only has simulator interoperability in mind, but actually included in the C4I equipment in each vehicle. Separate architecture and interface necessary for an ESS-equipped vehicle to become part of a larger exercise are not required.

4. Simulation as Decision Support

The idea that simulators can be used to feed data to modern C4I equipment begs the question, what about the reverse flow? Is it possible for C4I equipment to feed information to a simulator, which can then be analyzed using current modeling techniques? This requires a look at mapping the availability of data in a C4I suite to the data requirements for scenario construction in a simulation engine. If enough data exists in the suite to set up the initial conditions for a simulation engine with only minor additions or adjustments, all that remains is enabling the appropriate data to flow from source to consumer—in this case, C4I suite to simulation engine. Enabling that flow is a matter of identifying the necessary data and making sure that it exists in, or can be translated into, the appropriate format and context. The transport of the data is a matter of network design; the extraction, formatting, and context is a matter of application design.

D. AUTOGENERATION: TRANSLATING DATA INTO PRESENTATION

Automatic development of 3D environments is useful in any number of applications, not the least of which is battlespace visualization. One of two major difficulties lies in extracting the right data for inclusion in a scene from the cornucopia of data sources that exist in an operational environment. The other lies in using that data to faithfully reproduce the conditions it actually expresses. Autogeneration has several advantages, which continue to drive developers despite the magnitude of the difficulties it presents.

1. Speed

The ability to turn textual or byte-coded data into a visual format with minimum operator intervention will speed up the process of entering data into C4I suites, planning software, or simulations. Simulation software designed for operators need to provide a simple method for constructing a scenario, and once a scenario is built, it must be easily leveraged for the construction of other scenarios. This benefit translates across the spectrum of military planning and operations applications. Even more advantageous is the ability to directly translate existing documents such as tactical manuals, Operational Orders (OPORD), Air Tasking Orders (ATO), and similar documents into scenes or scenarios. Direct translation of familiar documents of the operating environment opens the will of simulation to the warfighter in ways that are currently not possible. The quick turnaround of simulation data and results would bring decision aids to the operational and tactical level.

2. Precision

In the strictest sense of the word, precision is a term describing the ability to consistently produce conditions that closely match desired results.[14] It is essentially a state of repeatable accuracy. The military has need of precision in much of what it does, not least of which is planning and executing combat operations. Developing the annotated map materials necessary for an operation is painstaking work, involving many man-hours poring over documents and charts. Once a set of orders has been published, the preparation of the supporting maps and charts occurs simultaneously in operations and intelligence shops throughout a theater. Mistranslations and mistakes in two separate staffs can result in serious consequences, not the least of which is friendly fire incidents. Collaborative tools in the C2 community have greatly decreased this sort of error, but it is still possible for two different staffs, working with two different command and control suites to come up with slightly different visualizations of the same set of orders. It is possible to distill the necessary information to create an accurate, repeatable set of visual documents (including 3D representations) no matter what application creates them. Collecting the data is only half the solution. The other half is painstakingly and explicitly specifying the use case and context of the data in the collection.

3. Data availability

Data are abundant in the typical theater of operations; the problem often comes in the form of a lack of ability to translate that data into usable information. Creating usable information is a matter of understanding the necessary interface between the data and the user. Data are generated in raw form by sensors and is communicated in the form of reports to users from the data supplier: the sensor operator or the sensor itself. It must be supplied in a concise, timely, and understandable form to the data consumer: the commander or warfighter. These reports are formatted in a way that is readable by the consumer and ideally contains only information the consumer requires to perform the actions necessary to complete a mission. Information comes electronically in the form of data-link or message traffic, or verbally in the form of radio-net reports.

[14] Oxford American Dictionary

a. Format

Defense Messaging System traffic is of particular interest, because it is an electronic form that is translated into human-readable format then interpreted in such a way that it can be added to the battlespace picture. This is a problem of format. It is a waste of time and resources if the same information can come in a human and machine-readable format, which can be directly translated into battlespace visualization. A recent study of this issue from the Army perspective[15] found that orders which are formatted as a “commander’s intent” with the information contained within a “free-text” field of a standard message are extremely difficult for a computer to accurately parse into useful data. The study concluded that perhaps the solution to this issue is that the “free text” command format is not the best method of command dissemination. The suggestion is to tighten up the format by restricting the text to doctrinal verbiage, which makes orders easier to parse into computer-usable commands. This has the added benefit of reducing ambiguity of language in orders to live units as well. The thrust of this result is that perhaps there needs be no difference in the way the user interacts with a live subordinate and a simulated one, and what needs to change (or improve) is not the human-computer interface, but the human-human interface. Computer technology improvement is meant to make human-computer interaction more natural. Perhaps that interaction still needs improvement, but there also exists room for improvement in the “natural” state of human interaction.

A sample order of 400 words of free text was examined in this report. Reformatting by removing superfluous adjectives and redundant language resulted in a 214-word text message, still human readable, with no appreciable loss in detail. This was further reduced to less than 100 words when placing it into a table format containing only pertinent points, however this was far more difficult to read and relatively incompatible with current messaging technology. The middle ground of fixed-format text is be an acceptable alternative to “man-in-the-loop” intervention. It also suggests an avenue of improvement for current operations. This signals a possible shift in the simulation to

[15] Kleiner, 1998

operation relationship. Simulation requirements do not necessarily drive operational change, but it is possible that efficiencies found in simulators can be transferred to real-world operations. Perhaps other interface efficiencies can be translated as well.

b. Context

Just knowing *what* is transmitted leaves much to be interpreted. Interpretation leaves room for ambiguity and error, which translates to inaccurate results and a lack of precision between different C4I suites on a network. The cure for this problem is standardizing context and providing an appropriate depth of documentation for the data. The example of fixed-format text in commander's intent orders may demonstrate a possible solution to the issue of format, but context is not established by creating standardized format alone. The example falls short of true contextualization because the context is assumed by the fact that it is based on a pre-existing "schema," a U.S. Army Order. The context of the report was provided by the fact that those who read and studied it understood U.S. Army Order format and content. A German Army unit receiving the modified order on the same operations network would have some difficulty in gleaning the information from that order, despite the strict standardization of content. Contextualization involves a specific knowledge of each piece of data, i.e. the meaning of "forcefully attack" or the understanding that, doctrinally, an engaged American armor unit typically has a responsibility for XX kilometers of front or is expected to move XX kilometers per day. The amount of contextualization the document requires depends on the application, which means it must be created with an eye to interoperability and the informational requirements behind joint operations. The creators of the document and its supporting context cannot work in a vacuum, but rather in communication with their peer community of interest across the joint and coalition spectra. Understandably, this is not always possible or practical, so all that remains is to ensure that the context is openly published in sufficient detail to allow individual members of that community of interest to easily translate it into meaningful documents under their own context with equivalent precision.

4. Relationship Between Autogeneration and Interoperability

Much of what goes into establishing interoperability on heterogeneous networks also enables autogeneration. Strict, open standardization provides both a common language and a common, precise understanding of data content among nodes on a network in much the same way that the Transmission Control Protocol / Internet Protocol provides the ability to transmit the packets containing that data among nodes the internet. The trick is to remove human translation as much as from the process in both realms. If the flow of information requires human intervention anywhere in the transfer chain, the ability to autogenerate is greatly reduced. The very same things that introduce the possibility of error into the creation of visual content also inhibit accurate recreation of visual content via autogeneration.

E. EXTENSIBLE MARKUP LANGUAGE (XML) IN THEORY

The data-related technology most applicable to solving the problems of interoperability and autogeneration is XML. XML is a public-licensed, extensible, platform-independent technology used to structure data. Essentially, it is a language used to create languages called schema, which are meant to give context to data. As an ISO standard, it has a well-defined methodology behind its construction and broad acceptance in government and industry. Visually it is similar to Hypertext Markup Language (HTML), but its relationship to HTML is in its structure. XML uses tags and attributes to describe the data it holds, just like HTML, but its implementation is far stricter, using well-defined standards of form and strongly typed definitions to force XML users to adhere to good, consistent formatting practices.

1. Relevant Benefits and Limitations

As with all technologies, XML is not without faults for all its features. For the purposes outlined in this thesis, however, the benefits far outweigh the limitations.[16]

a. Validation

Strong typing and strict implementation of structure allow an XML document to be "validated" against the schema under which it was created. According to the specification, processing an invalid XML document is forbidden, so programs designed to read documents (called parsers) are required to stop reading and report an error as soon as one is found. This helps reduce the overhead of format checking within the application.

[16] XML in 10 Points, 2004

b. Text-based

XML is text-based, meaning the data is stored in text form rather than binary, so the files are human-readable if necessary for debugging. Because XML is Unicode compliant, schema and documents can be constructed using any Unicode character set, and are readable by any operating system or application that understands the Unicode standard. The problem with the text format is the considerable overhead it adds to the file size. This is not a new consideration, and there have been avenues of research and advances in creating a binary compression format to be used in transmitting XML documents over networks, and current standard methods like .zip and .gzip compression can be used to shrink local files sizes.[17]

c. Modular

Schemas created using XML are not immutable end products once they have been created. XML schemas can be used to create new schema by combination or using the *inclusion* tag in the new schema. The mechanism of XML *namespaces* provides separation within a schema by adding distinct prefixes to the elements coming from different sub-schemas if they have identical tag names; this is done using the *import* tag. By combining schemas, data from different sources typically described using different schemas can be linked and easily combined to form one document type, which funnels all the desired data into one document. Well-designed schema may also utilize this ability to separate out portions of the schema which are more likely to require future modification, or which may have a limited audience.

d. Extensible

Using the *import* tag, appropriately written XML schemas can be extended to include new tags and attributes that were not originally envisioned to be included in the language. This allows a schema to be robust against the pressure of future development and still maintain forward and backward compatibility. Extensions also allow for parallel development using a common base schema. Specialized applications can be supported using extensions that add narrow functionalities to the schema without creating all-new schemas. As long as these extension schemas are written and documented appropriately, they may be used in parallel with the original schema.

[17] Norbraten, 2004

e. Translatable

One part of the XML family of technologies is the Extensible Stylesheet Language (XSL), which is used to create logical links between the tags and attributes of two different schemas. Using documents created using one schema to create new documents valid under another is called XSL for Transformation (XSLT). This is an extremely powerful feature of XML, which prevents parallel or insular development of schemas from precluding the interoperability of XML databases. A limitation of XSLT is that it is a work intensive process to create an XSL document and the resultant transformation is only one way.

f. Cross-compatibility

It deserves special consideration to point out that the features that make XML modular, extensible, and translatable can be combined to create a powerful methodology for enhancing the interoperability of schema-aware applications. Applications that use the same basic schema to store their data—with application-specific extensions—may still be made interoperable, perhaps using most or all of the same data. If the data in the extended portion of the schema is not central to operation or does not in some way attempt to replace the core schema's data, other applications can be set to ignore the extensions and still use the core data in a document.

If the document's extended data are necessary for the other application, though, it is still not a complete loss. As long as the extension schemas are well defined, documented, and freely available, it is possible to map documents from one to the other using XSLT, or merely make the application aware of the extension. It is also a useful tool to guide versioning: if enough of the extensions in use among consumers of the base schema have similar data inclusions, it may be useful to update the base schema to include that data.

2. Navy Standards

The office of the Department of the Navy Chief Information Officer (DoN CIO) has promulgated policy on the development practice and use of XML in the Navy, which places great emphasis on the following:

a. Pre-existing Standards

The DoN CIO requires adherence to pre-existing standards with special consideration to those recommended by the World-wide Web Consortium (W3C). Pre-existing standards can include Federal standards or those put forth as Business Voluntary Consensus Standards (VCS) by the ISO, the Institute for Electrical and Electronic Engineers (IEEE), American National Standards Institute (ANSI), or others.

b. Public License

To prevent entanglements with private contractors over the use and distribution of XML intellectual property the emphasis is on non-proprietary formats. The use of proprietary schema and extensions schema or extensions that require licensing with corporate entities for enterprise use are strictly forbidden.

c. Participation in Standards Development

Representation in the development of XML standards is preferable to consumer status when the Navy has a specific interest in the functional area covered by the standard. Where no standard or standards working group exists, Navy development guidance has been published to substitute.

d. Enterprise Management of XML

High-level oversight of use and development, including the registration of schema, has been established the enterprise level to promote Navy-wide access and re-use. Functional Area Managers have been designated and charged with the oversight, management, funding, and registration of emergent XML standards within specific areas of interest, and the office of the CIO provides overarching review. Furthermore, development guidance systematically spells out appropriate authoring practices for making schema reusable throughout the Navy and DoD

F. SUMMARY

Visualizing the operations environment in three-dimensions is a worthy goal for the continued development of command and control technology because it helps commanders more quickly understand the subtleties of the battlespace. Research to improve the ease and speed with which 3D visualization are created can foster improved interoperability and collaborative planning as well, which will be the key to its acceptance and further development. The most likely enabling technology for this sort of advancement is an already established one: XML.

There is a proliferation of XML grammars that can be used as a basis for the development of an interface grammar. These schemas have different use-cases, scopes, target audiences, and directions from which they approach the problem of standardization. No single one of the schemas discussed in this chapter contains all the information, in the right form, at the right level to develop a visual scene—with the possible exception of MSDL. The problem with MSDL is its sheer size and depth of detail. It is not necessary to capture all data necessary to run a simulation in all situations; it is just as useful to capture a core set of data and merely refer to the rest, thus keeping the size and complexity of the resulting documents to a minimum.

IV. X3D SCENE AND SIMULATION SCENARIO CONSIDERATIONS

A. INTRODUCTION

Developing visual environments involves using a 3D modeling language to establish order on an abstract space, and then map reality onto it. Once the basic frame of reference is established, creating a scene is a matter of systematically positioning and orienting 3D models in the space to represent terrain, scenery, and entities, and then setting the conditions that effect the visibility in the scene. In the end, a scene is just an interface, only one component in an application that requires other components to drive it. The driving component in this thesis is a Discrete Event Simulation (DES) generated by the Viskit authoring tool, which uses a graphical authoring interface to allow users to create simulation program by manipulating icons. The overlapping elements needed to create both a scene and a scenario are the starting point for developing a grammar to capture the process.

B. OVERVIEW

There are three distinct processes at work in the procedural linking of simulations (or battlefield networks) to a 3D visualization: establishing a virtual environment, populating the virtual environment to create a dynamic scene, and then linking the actors and processes in the scene to the sources of data which drive it. Each of these sub processes has a separate set of operations and requirements necessary to successfully achieve its part of the equation, which means each has its own set of inputs and outputs. The design of component-based systems dictates that these inputs and outputs (as well as the tasks within each component) be well defined. This chapter outlines these interrelationships and tasks.

Figure 8 shows the component-based breakdown of the AT/FP Toolkit in terms of Interface, Logic, and Data. The Savage Modeling and Analysis Language is the transport mechanism bringing the data to the other two components, and Distributed Interactive Simulation handles data exchange between the Logic and Interface components.

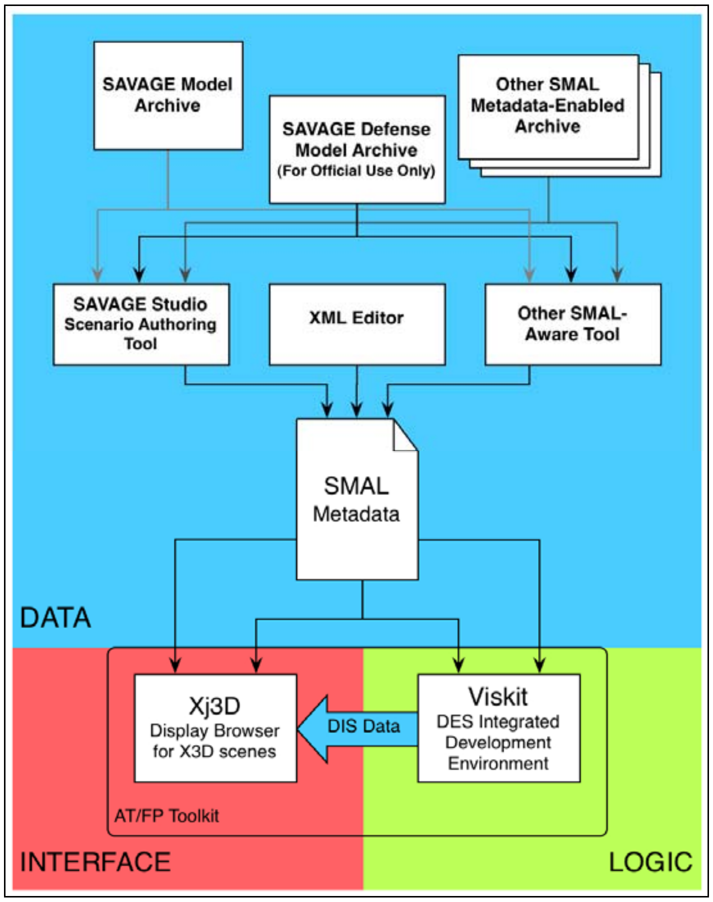

SAVAGE Model Archive
SAVAGE Defense Model Archive (For Official Use Only)
Other SMAL Metadata-Enabled Archive
SAVAGE Studio Scenario Authoring Tool
XML Editor
Other SMAL-Aware Tool
SMAL Metadata
DATA
Xj3D Display Browser for X3D scenes
DIS Data
Viskit DES Integrated Development Environment
AT/FP Toolkit
INTERFACE
LOGIC

Figure 8. The role of Savage Modeling and Analysis Language (SMAL) metadata in the Anti-Terrorism/Force Protection (AT/FP) Toolkit. The SMAL Metadata is shown as able to be generated by different tools using a variety of resources.

C. DEVELOPING VISUAL ENVIRONMENTS

Virtual environments have a unique distinction from any other type of environment in that they start from absolute zero. There is no predetermined natural order or frame of reference inherent in virtual 3D space; the builder imposes all order and reference on the space using the 3D modeling language. The modeling language is, in some respect, an interface with the abstract world of the virtual environment. Because there is no inherent set of standards to the virtual environment, all standards begin with the modeling language. The language itself controls how objects are constructed, their visual and material properties, and the way in which they interact with each other and the environment. None of these interactions is required to have any basis in reality unless it is reality itself that the builder is trying to emulate.

Given that the builder is attempting to construct a virtual representation of reality, there arises a need to map actual reality onto the virtual reality. The computational workings behind the interaction between a virtual object and either the virtual environment or another virtual object is not within the scope of this discussion. The model by which the visual representation of virtual objects is accomplished is also outside the scope. Instead, when discussing the construction of a 3D virtual environment, the primary topic for consideration in this Section is the placement of virtual objects in a scene in a way that makes visual sense.

Both virtual worlds and the virtual objects within them are created in the form of computer files using a modeling language. The world and objects can be created together and concurrently in one file or—in some modeling languages—separately and be pieced together as needed using features of the language which allow accessing multiple files. The piece-wise method of construction is less efficient computationally, since it requires accessing multiple files and all the operations attendant to that, but it is more efficient for the process of scene development, since it allows for easy object re-use between worlds.

The two methods of construction have some common issues involved in their development. Those issues can be handled by following 3D development conventions. However, piece-wise construction has some special issues that must be addressed using development standards. The difference between a convention and a standard is that conventions are the result of agreements by default, while standards are the result of specific agreements in writing.

1. Position

3D modeling languages typically provide a three-value coordinate reference system with an arbitrary origin, or (0, 0, 0) point, upon which objects can be positioned by assigning a three-value number representing the distance the object is to be displaced from the origin. This linear displacement is called a *translate* transformation, and it is the first of three types of transformations that can be applied to virtual objects during scene construction. The three-value number defining an object’s position requires a point of reference, which is the second consideration in scene creation. When that point of reference is the world origin, or (0, 0, 0) point of the scene, it is termed *globally* positioned. However, if the position uses a point of reference other than the world origin—a point on another object, for example—it is termed *relatively* positioned. It is fairly easy and fast for a computer to translate the relative position value of an object to global coordinates and vice versa given that the scale is constant, and it is only slightly more difficult if it is not. The frame of reference used to position an object can, logically, be different for every object in a scene, and—as long as the resultant global coordinates can be found for every object—is relatively immaterial. The number of reference points for an object is limited only by the numeric precision of the computer calculating its position. Therefore, the explicitly recording the point of reference used to position an object is not necessary, the information is better found in the structure of the scene.

2. Scale

The units of distance along the coordinate axes in a virtual environment have no meaning other than to the modeling language in its attempt to impose structure on the void. It is left to the builder of the scene to establish the scale, that is, to assign a meaning or value to the units of the coordinate system. In order to make visual sense and be useful, objects to be placed in the scene must conform to the virtual world's established scale. If an object does not conform to that scale, it is possible to perform the second type of transformation, a *scale* transformation, on the object to bring it into conformity. Scale, once set, is an absolute throughout the scene, so it deserves explicit documentation in standards or in metadata.

3. Axis-Orientation

The third consideration in dealing with object placement is orientation. Typically, real objects have a natural orientation to which a viewer is accustomed, a sort of "right-side-up" quality. Virtual objects do not have a "natural" orientation, even though they have a coordinate reference system, since they are unaffected in their virtual existence by gravity or other forces. Thus, the directions of the reference coordinate axes must be assigned real-world directions such as "up" or "left" in order to make any consistent sense. Axis naming has a strong convention behind it, which uses a mnemonic borrowed from the world of physics known as the "Right-Hand Rule." Holding the right hand with index finger and thumb at right angles, then extending the middle finger, bent at the first knuckle ninety degrees to the index finger and thumb; a set of orthogonal axes is formed.

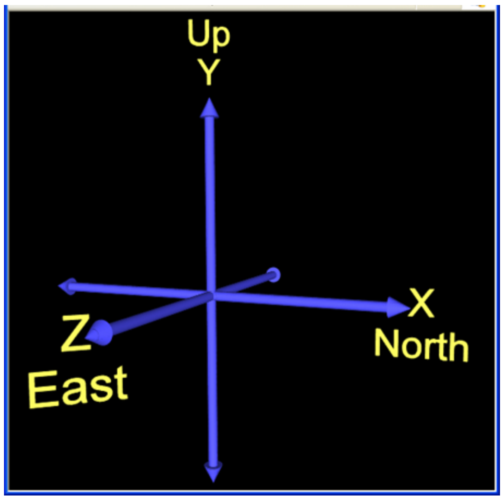
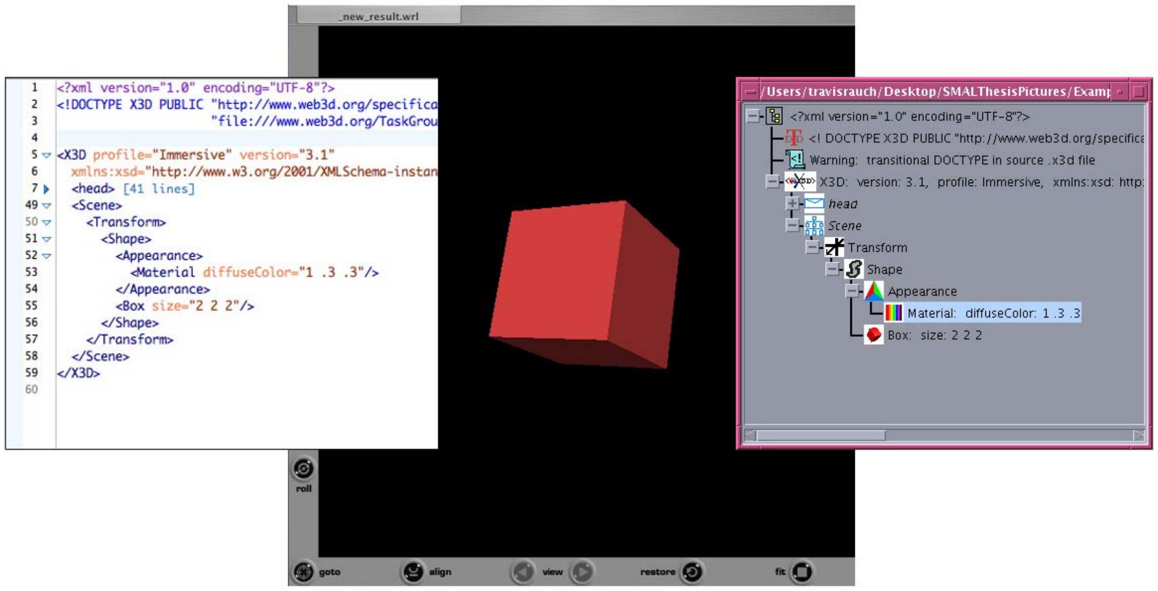


Figure 9. Coordinate axes showing the “positive Y up” axis orientation standard.

To use the Right-Hand-Rule, the thumb is named “X”, the index finger “Y”, and the middle finger “Z.” Given the axes named using this convention, there are two conflicting conventions currently used in the 3D graphics world: the so-called “positive-Y-up” and “negative-Z-up” standards. Looking at a scene using the positive-Y-up standard, the ground plane is defined by the X- and Z-axes. Units of X increase moving to the right, units of Z increase moving toward the viewer, and the Y-axis extends vertically up with units increasing as they move upward. This convention is utilized throughout the graphics community. The positive-Z-up standard had its origins in robotics and has the ground plane defined by the X- and Y-axes with units of Y increasing toward the viewer and units of Z increasing in the downward direction. The existence of two standards can create conflicts if it is not known which one was used when a virtual object was created. If it is known, it is possible to rotate, or rotate transformation (the third type of transformation), on the object to bring it into accordance with the world’s axis-orientation standard.

D. X3D SCENE CONSTRUCTION

There are many 3D graphics languages in current use, and most are proprietary. In the development of this thesis, all work was done in X3D. Because it is XML-based, it is stored in text form. Its visible structure mimics its logical structure and its features are representative of graphics languages in general, making it a worthwhile language to use as an example.


```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "http://www.web3d.org/specifica
                     "file:///www.web3d.org/TaskGrou

<X3D profile="Immersive" version="3.1"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema-instan
  <head> [41 lines]
  <Scene>
    <Transform>
      <Shape>
        <Appearance>
          <Material diffuseColor="1 .3 .3"/>
        </Appearance>
        <Box size="2 2 2"/>
      </Shape>
    </Transform>
  </Scene>
</X3D>
```

```
/Users/travisrauch/Desktop/SMALThesisPictures/Examp
<?xml version="1.0" encoding="UTF-8"?>
  <! DOCTYPE X3D PUBLIC "http://www.web3d.org/specifica
  Warning:  transitional DOCTYPE in source .x3d file
  X3D:  version: 3.1,  profile: Immersive,  xmlns:xsd: http:
    head
    Scene
      Transform
        Shape
          Appearance
            Material:  diffuseColor: 1 .3 .3
          Box:  size: 2 2 2
```

Figure 10. An example X3D scene showing the underlying XML on the left, the Scene graph equivalent from X3D-Edit on the right, and the resulting scene in the center.

Understanding the three basic considerations that go into generically structuring a virtual world is only part of the equation. Once the world has some order imposed on it, constructing a scene, or set of virtual objects placed in a virtual world becomes a matter of managing transformations and geometry rendering. The complex operations behind rendering 3D geometry on a computer are not within the scope of this paper, the focus for scene construction is on the inputs required to build a specific 3D scene, which do not differ greatly between use cases.

There are three basic things that must be known to construct a scene:

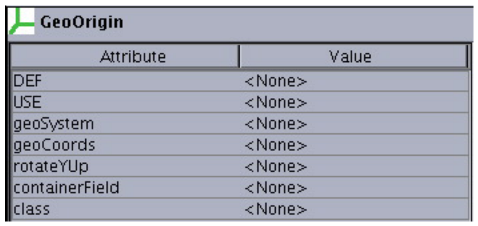
- The point of reference from which everything in the scene is evaluated
- The basic environmental conditions
- The type, scale, position, and orientation of objects that are present

These are the base minimum, and may have different levels of detail required based on the application. For example, a simple scene may have one universal set of environmental conditions, and another may have several localized sets. The reference point may be purely virtual, or have some connection with the real world.

1. Origin Point of Reference

As discussed in the previous section, the stage is set for developing a 3D scene by establishing its scale and the context. If the scene is to be a land battle, placing a 3D model of terrain will often be sufficient to set scale and context. If the terrain model represents actual terrain, then the scale has been set, and all other models must conform to that scale. It is helpful if the scale on which the terrain model was constructed is known, but it is possible to determine it by matching points on the terrain to real-world points. On the other hand, if the terrain is randomly or manually generated with no particular location in mind, the scale has not been set, and cannot be known unless it has been explicitly established by some other means. As a rule, the first object placed in the scene that represents a known quantity is what will set the scale. The most common non-terrain object that can be used to set scale is a human figure, but any known quantity will suffice, such as a vehicle or building. For the purposes of representing actual environments, terrain is normally sufficient.

Terrain has the added feature of adding locality to a scene. Using a terrain model to anchor a scene provides a direct mapping to the real world, which will also set the location parameter of vertical displacement. That is, if the actual location on the earth represented by the terrain model is known, then the altitude of at least one point on that scene can also be known or determined. By setting the terrain model in the scene, and knowing the altitude and position of at least one point on that terrain, the altitude and position of all other points on the terrain—or any other point in the entire scene—can be determined. An entire set of extensions has been integrated in X3D to handle the mapping of the world onto a virtual scene, called the X3D GeoSpatial Component. This set of extensions maps the original Virtual Reality Markup Language (VRML) GeoVRML extension into X3D. The basic functionality is to establish a GeoOrigin point with a specified geospatial reference system and real world coordinate.



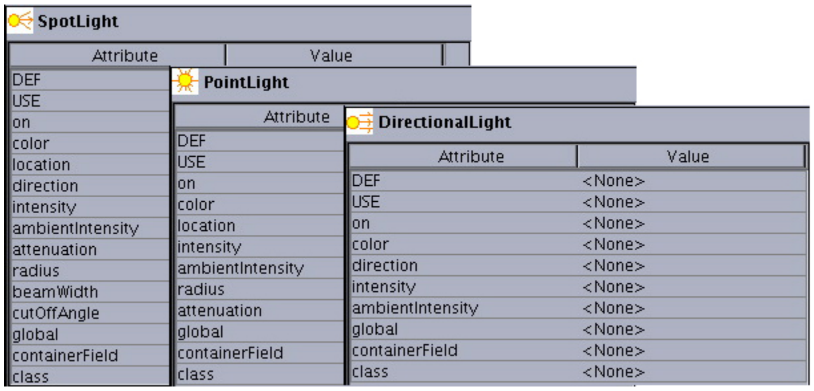
GeoOrigin

Attribute	Value
DEF	<None>
USE	<None>
geoSystem	<None>
geoCoords	<None>
rotateYUp	<None>
containerField	<None>
class	<None>

Figure 11. X3D GeoOrigin Node illustrating the geoSystem, geoCoords, and rotateYUp fields used to establish a reference frame for the virtual environment.

2. Basic Environmental Conditions

This refers to visibility in the scene. Visibility is developed in a graphical environment by using lights, fog, and visual effects. Virtual lights—like the PointLight, SpotLight, and DirectionalLight nodes in X3D—have basic properties of color, direction, and range of effect.



SpotLight

Attribute	Value
DEF	
USE	
on	
color	
location	
direction	
intensity	
ambientIntensity	
attenuation	
radius	
beamWidth	
cutOffAngle	
global	
containerField	
class	

PointLight

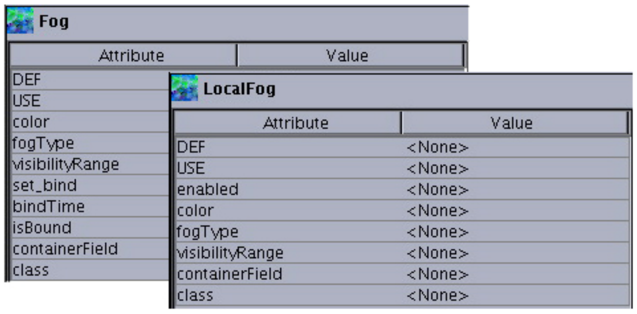
Attribute
DEF
USE
on
color
location
intensity
ambientIntensity
radius
attenuation
global
containerField
class

DirectionalLight

Attribute	Value
DEF	<None>
USE	<None>
on	<None>
color	<None>
direction	<None>
intensity	<None>
ambientIntensity	<None>
global	<None>
containerField	<None>
class	<None>

Figure 12. X3D SpotLight, PointLight, and DirectionalLight illustrating the color, intensity, attenuation, and location fields common to all three. PointLight includes a radius of effect, while PointLight and SpotLight include directional modifiers.

Virtual fog has properties of color and distance-based level of obscuration, which are implemented along with others in the X3D Fog and LocalFog nodes. The type of fog representation used in X3D adjusts visibility, once the viewer comes under its effect, by shifting the color of objects in the scene toward the color of the fog based on the distance to the viewer. Other fog-representation models may use volume-based color change that change the color of every pixel on the screen based on distance through the fog volume, including background pixels.



Fog

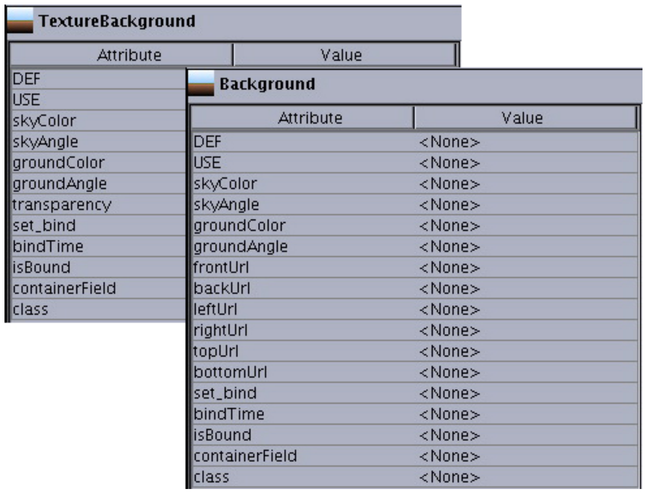
Attribute	Value
DEF	
USE	
color	
fogType	
visibilityRange	
set_bind	
bindTime	
isBound	
containerField	
class	

LocalFog

Attribute	Value
DEF	<None>
USE	<None>
enabled	<None>
color	<None>
fogType	<None>
visibilityRange	<None>
containerField	<None>
class	<None>

Figure 13. X3D Fog and LocalFog nodes illustrating the color, visibilityRange, and fogType fields used to configure the visibility in the scene.

Using combinations of these two factors, plus visual effects like backgrounds and animations or particle systems representing rain or snow, any number of weather and daylight conditions can be mimicked.



TextureBackground

Attribute	Value
DEF	
USE	
skyColor	
skyAngle	
groundColor	
groundAngle	
transparency	
set_bind	
bindTime	
isBound	
containerField	
class	

Background

Attribute	Value
DEF	<None>
USE	<None>
skyColor	<None>
skyAngle	<None>
groundColor	<None>
groundAngle	<None>
frontUrl	<None>
backUrl	<None>
leftUrl	<None>
rightUrl	<None>
topUrl	<None>
bottomUrl	<None>
set_bind	<None>
bindTime	<None>
isBound	<None>
containerField	<None>
class	<None>

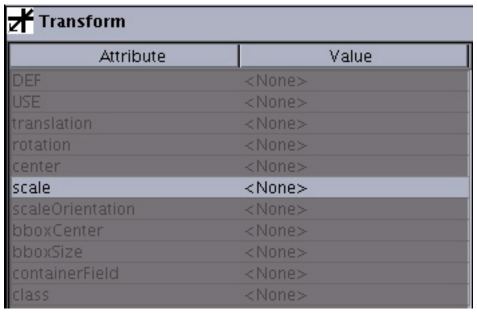
Figure 14. X3D TextureBackground and Background nodes illustrating the skyColor, groundColor, skyAngle, and groundAngle fields used to set a horizon and simulate atmospheric effects in the sky. The Background node additionally allows the use of images as a backdrop for the scene.

X3D provides nodes for backgrounds (Background and TextureBackground) and advantage has been taken of X3D's extensible nature to create a particle system prototype node. The Background node essentially assigns colors to every pixel not already assigned a color by an object. This can be done using images or by using explicitly assigned colors.

In scene development, other environmental factors such as wind speed and direction or ocean current are only important if there is some visual effect on the scene, such as the orientation of a windsock or flag, or the angle at which rain falls. Non-visual effects may be extremely important for other parts of the application, but that will be treated as a component of behavior, which is separate from the visual aspect of the scene.

3. Type, Scale, Position, and Orientation of Objects

Visualizing terrain and environmental conditions is not typically the end goal of scene development. At a minimum, cultural features must be included to make the scene interesting and germane to most use cases. Following an object-oriented approach to modeling, the objects to be included in the scene are built separately or selected from a repository of object models for positioning in the scene. Because every object in a scene that is not terrain has been modeled separately with its own internal scale, the scale of each object must also be known so it can be compared and adjusted to match the scale set for the scene. Scaling transformations on objects in X3D are done with the scale field of the Transform node.

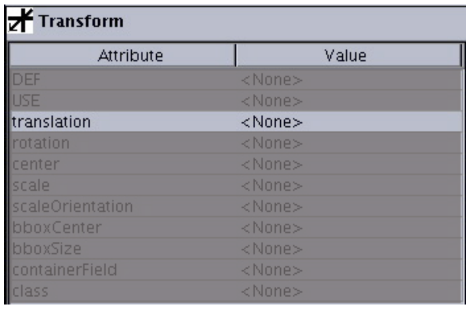

Transform

Attribute	Value
DEF	<None>
USE	<None>
translation	<None>
rotation	<None>
center	<None>
scale	<None>
scaleOrientation	<None>
bboxCenter	<None>
bboxSize	<None>
containerField	<None>
class	<None>

Figure 15. X3D Transform node scale field used to modify the size of the object in all three dimensions without specifically modifying the geometry of the object.

Once the scale of the object matches the scale of the scene, the next area of concern is the actual position of the object. Position information can take one of three forms: the two discussed in Section 1 (global-virtual and relative-virtual) and geographic. Global-virtual and relative-virtual positioning are functionally identical, the only difference being the point from which the object's position is referenced. Virtual positioning is performed in X3D using the *translation* field of the Transform node.

Whether the positioning is global or relative depends on the location of the node on the scene graph. If the node is accessed directly from the root, it is a global position. If it is accessed from another Transform that performs a translation, it is a relative position.

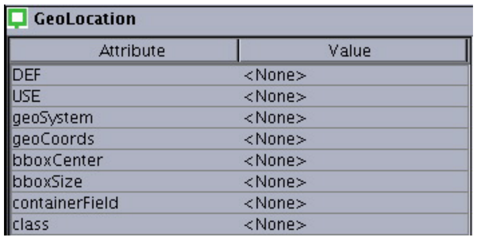


Transform

Attribute	Value
DEF	<None>
USE	<None>
translation	<None>
rotation	<None>
center	<None>
scale	<None>
scaleOrientation	<None>
bboxCenter	<None>
bboxSize	<None>
containerField	<None>
class	<None>

Figure 16. X3D Transform node translation field used to position the object in 3D virtual space using virtual coordinates.

Geographic positioning requires real-world mapping to be useful, which is to say, there must be a geographic point mapped to some virtual point, preferably the origin, and an established scale set in the virtual world. Once those two items are known, it is possible to translate geographic positions (latitude-longitude pairs, Military Grid Reference System (MGRS) coordinates, etc.) into virtual coordinates for object placement. As discussed, X3D uses the GeoOrigin node to set the reference point, but actual coordinate-based positioning is accomplished using the GeoLocation node, which is a specialized transform that hold the same type of coordinate information as the GeoOrigin that is translated to virtual coordinates at run-time.

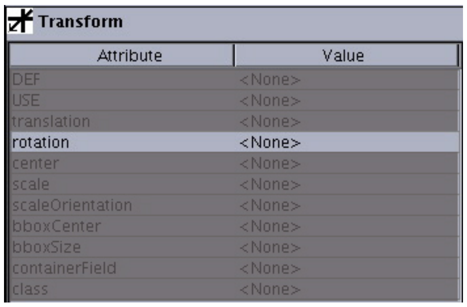


GeoLocation

Attribute	Value
DEF	<None>
USE	<None>
geoSystem	<None>
geoCoords	<None>
bboxCenter	<None>
bboxSize	<None>
containerField	<None>
class	<None>

Figure 17. X3D GeoLocation node used to position an object in 3D virtual space using a latitude and longitude in the geoCoords field. This node can only be used in conjunction with a GeoOrigin node.

Assuming the objects have been made or adjusted to have a natural orientation (the “up” direction) that matches the natural orientation of the scene, the particular orientation of the object must be established to make it visually correct. An object representing an aircraft in the air may have three explicit angles of orientation (pitch, roll, and yaw). An object representing a vehicle may have one explicit angle of orientation (yaw, in the form of heading) and two implicit angles determined by the terrain on which it sits. Finally, an object representing a building may have one explicit angle of orientation (yaw, in the form of "north") and two fixed angles based on the assumption that the building is plumb and level. X3D accomplishes this using the Transform node's *rotation* attribute, which accepts a quaternion value.



Transform

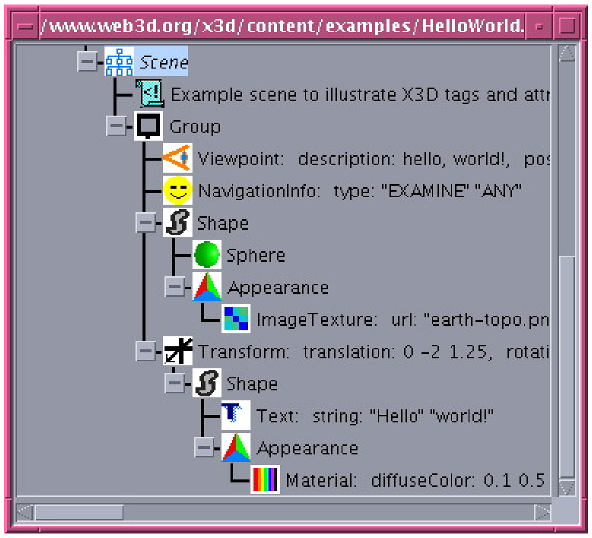
Attribute	Value
DEF	<None>
USE	<None>
translation	<None>
rotation	<None>
center	<None>
scale	<None>
scaleOrientation	<None>
bboxCenter	<None>
bboxSize	<None>
containerField	<None>
class	<None>

Figure 18. X3D Transform Node rotation field uses a quaternion value to orient the object in space around the point located in the center field (default is the current point in virtual space).

4. Effect of Structure on Data Access

Once objects have been properly placed and oriented within a scene, there is little else that must be done for scene construction. The next steps are viewing and interacting with a scene, which require further explanation. The information required to enable the proper viewing of a scene using virtual cameras is a function of the modeling language and the equipment used to render it and is outside the scope of this thesis. The ability to interact with the scene, on the other hand, is quite topical. In order to interact with a scene, there must be some method of selecting objects and manipulating their data in some way. The simplest form is merely to read or display an object's associated data such as position, orientation, or the geometry data that was used to render it. One step beyond reading this data is to change it in some way. An obvious example would be to change the position information, effectively moving the object from one place to another. This requires the ability to select a specific object in the scene, possibly out of thousands, access the values used to determine its position, and change them.

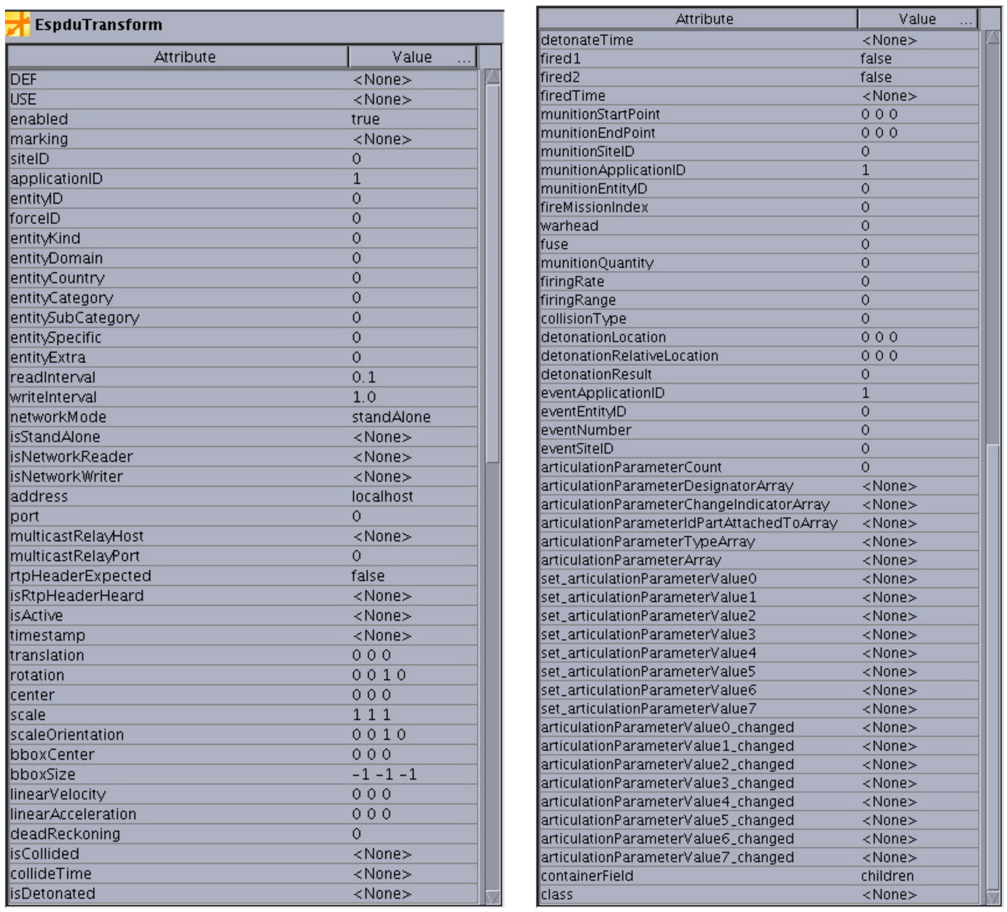
In the realm of 3D scene construction, keeping track of thousands of objects and their associated transformations is kept manageable by imposing structure on the scene. One method of imposing structure is using an organization tool known as a scene graph. Scene graphs are tree structures upon which transformation and geometry data are placed as branches of information called nodes.



```
/www.web3d.org/x3d/content/examples/HelloWorld.
Scene
  Example scene to illustrate X3D tags and attr
  Group
    Viewpoint:  description: hello, world!,  pos
    NavigationInfo:  type: "EXAMINE" "ANY"
    Shape
      Sphere
      Appearance
        ImageTexture:  url: "earth-topo.pn
    Transform:  translation: 0 -2 1.25,  rotati
      Shape
        Text:  string: "Hello" "world!"
        Appearance
          Material:  diffuseColor: 0.1 0.5
```

Figure 19. A Scene graph as depicted in X3D-Edit showing its tree-like structure.

The last node on a particular branch (one that does not link to any further nodes) is called a leaf node, and it is the leaf node position in which object models and descriptors are found. The structure of branching nodes above the model is made of transformation nodes. Extensive research has gone into the study and development of tree-style data structures in computer science. The result has been fast, efficient search and sort algorithms, which can be used to access the information found in a tree. As the computer moves from node to node down the length of a branch, it reads the transforms and establishes the position orientation and scale for the object it eventually finds in the leaf node position. The tree structure of a scene graph thus facilitates the understanding and rendering of a 3D scene. These same algorithms can be used to access specific information with the intention of manipulating it, thus also facilitating interaction with a scene. For example, in X3D specialized transforms called Entity State Protocol Data Unit (ESPDU) Transforms are designed specifically to receive change instructions in the form of Distributed Interactive Simulation (DIS) data packets, which can be used to “drive” objects around within a scene. Of note is that the DIS axis-orientation standard is negative-Z-up, which is handled in X3D with the rotateYUp field in the EspduTransform node.


EspduTransform

Attribute	Value
DEF	<None>
USE	<None>
enabled	true
marking	<None>
siteID	0
applicationID	1
entityID	0
forceID	0
entityKind	0
entityDomain	0
entityCountry	0
entityCategory	0
entitySubCategory	0
entitySpecific	0
entityExtra	0
readInterval	0.1
writeInterval	1.0
networkMode	standAlone
isStandAlone	<None>
isNetworkReader	<None>
isNetworkWriter	<None>
address	localhost
port	0
multicastRelayHost	<None>
multicastRelayPort	0
rtpHeaderExpected	false
isRtpHeaderHeard	<None>
isActive	<None>
timestamp	<None>
translation	0 0 0
rotation	0 0 1 0
center	0 0 0
scale	1 1 1
scaleOrientation	0 0 1 0
bboxCenter	0 0 0
bboxSize	-1 -1 -1
linearVelocity	0 0 0
linearAcceleration	0 0 0
deadReckoning	0
isCollided	<None>
collideTime	<None>
isDetonated	<None>

Attribute	Value
detonateTime	<None>
fired1	false
fired2	false
firedTime	<None>
munitionStartPoint	0 0 0
munitionEndPoint	0 0 0
munitionSiteID	0
munitionApplicationID	1
munitionEntityID	0
fireMissionIndex	0
warhead	0
fuse	0
munitionQuantity	0
firingRate	0
firingRange	0
collisionType	0
detonationLocation	0 0 0
detonationRelativeLocation	0 0 0
detonationResult	0
eventApplicationID	1
eventEntityID	0
eventNumber	0
eventSiteID	0
articulationParameterCount	0
articulationParameterDesignatorArray	<None>
articulationParameterChangeIndicatorArray	<None>
articulationParameterIdPartAttachedToArray	<None>
articulationParameterTypeArray	<None>
articulationParameterArray	<None>
set_articulationParameterValue0	<None>
set_articulationParameterValue1	<None>
set_articulationParameterValue2	<None>
set_articulationParameterValue3	<None>
set_articulationParameterValue4	<None>
set_articulationParameterValue5	<None>
set_articulationParameterValue6	<None>
set_articulationParameterValue7	<None>
articulationParameterValue0_changed	<None>
articulationParameterValue1_changed	<None>
articulationParameterValue2_changed	<None>
articulationParameterValue3_changed	<None>
articulationParameterValue4_changed	<None>
articulationParameterValue5_changed	<None>
articulationParameterValue6_changed	<None>
articulationParameterValue7_changed	<None>
containerField	children
class	<None>

Figure 20. X3D EspduTransform node illustrating the large number of parameters available for modification, which are matched to the data found in a Distributed Interactive Simulation (DIS) Protocol Data Unit (PDU). The EspduTransform node has all the functionality of a Transform node.

An additional benefit of the tree structure is in the realm of object-oriented programming. Once an object on the tree has been defined to the satisfaction of the user, it is possible to create a handle or pointer to that object and merely reference the pointer to re-use the information rather than recreate the object or data from zero. X3D uses the DEF and USE attributes for this function, which implement the XML types ID and IDREF respectively.

5. Effect of Structure on Visualization

Modern computers can perform each of the three types of transformations (translation, scale, and orientation) quickly enough to have little impact on the viewing or manipulation of a scene, but as scenes get larger, the number of transformations that must be performed can have a slowing effect on rendering when interacting with the scene, resulting in “choppy” or visually disconcerting behavior. The number of transformations an algorithm has to negotiate to render a scene or component of a scene exacerbates this slowing effect.

In relatively flat tree structures, the sheer number of and complexity of objects has the greatest impact on the usability of the scene. In deeper tree structures, with more nodes between the root and a given leaf, the number of transformational steps between any the leaf and the root of the tree can have additional impact on usability.

E. COMPONENT-BASED MODELS AND SIMULATIONS

The component-based software design relies on collecting monolithic software components and connecting them through a common interface to develop applications. The term monolithic is used to describe individual components because they are the fundamental building blocks of a component-based design; there is no requirement to understand or access the processes inside the component. Instead, the interesting part of the component is its interface. The interface of a component is comprised of a number of methods, divided into three types: accessor/mutator methods, action methods, and event-handler methods. The accessor/mutator methods are used to read or manipulate single properties within the component for initialization or run-time adjustment. Action methods are used to perform some function beyond a simple value change, such as changing the state of the component, performing a calculation, or activating a subsystem. Event handler methods are the primary method of reacting to the actions of other components.

When an accessor/mutator method or action method is activated in a component, the component may broadcast the change, action, or event to the other components in the application, which receive the status change and react using their event handler methods. This paradigm removes the requirement, found in non-component-based applications, for one process to know the internal workings of another in order to access its methods and get a result. In effect, each component sits and passively awaits a signal from the others, acts if the signal is one in which it is interested, then broadcasts the results of its own action. This is known as the Listener Design Pattern, for obvious reasons.

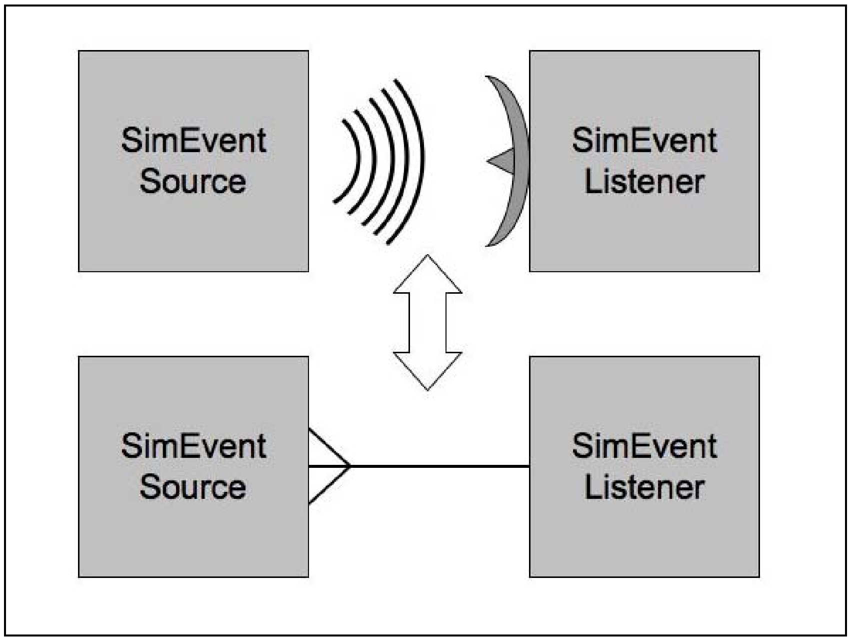


Figure 21. The Listener Design Pattern as diagrammed for Viskit showing a Listener Component listening for a Source Component to "broadcast" an event.[18]

Components may not always have perfectly aligned interfaces, most likely because they have been built by different coders or for different purposes. If this is the case and a developer wishes to retain the functionality of one component without modifying it—for whatever reason—he or she can apply the Adaptor Design Pattern. Components called adaptors listen for specific events and, when they occur, immediately respond with a modified version of the event designed to trigger a response in the intended receiver. This pattern is essential to ensure compatibility among components when the appropriate functionality already exists, but the required component's interface does not match the rest of the application. Adaptors are used mostly when it is not practical to create a new component.

While custom coding is discouraged in true component-based design, components may still need to be created to handle exceptional or unorthodox situations, so

[18] Adapted from Buss, 2004, p.4

modification of components is not without its own methodology. If the adaptor pattern cannot be applied or interface is not the problem, the preferred method of component redesign is through polymorphism, which allows the analyst to change only that behavior which needs changing, rather than rebuild the entire component from scratch, including its interface.

For the purposes of this paper, the primary discussion on component-based simulation will center on analytic simulation (specifically the Viskit-based AT/FP Toolkit), which does not accept human interaction at runtime. This in no way suggests that component-based design does not apply to real-time simulators or even command and control suites. To some extent, real-time simulators have component-based design elements in them already, especially those designed to drive actual operational equipment with computer generated input. Network or web-based operations have a component-based philosophy behind them as well (e.g. web browsers connected by a common data transfer protocol and markup language to web servers), although not always purely implemented (e.g. CORBA, RPC, and the like).

1. Discrete Event Simulation (DES) Modeling

DES is one of two time management paradigms in simulation. The more familiar paradigm, time-step, involves advancing simulator time by a pre-set discrete amount, checking all entities, performing required calculations, updating all entity states, then advancing simulator time again to start the process over. In DES, events requiring calculations are scheduled, simulator time is advanced to the first event (no matter what time increment that requires), calculations are made, entity states are updated, resultant events are added to the schedule, and then simulator time is advanced to the next event. DES is a powerful concept, because dead time (i.e. time in which nothing important happens) is completely removed, and calculations are only performed when something occurs that requires them. This reduces computational overhead, removes the effect of time-step granularity (where events occur between steps and effects must be interpolated), and radically reduces the run-time of a simulation.

2. Building a Simulation Application

Simkit and Viskit are two APIs used to create discrete event simulations in Java. Their design is based largely on the Listener Pattern found in component-based design.

The Listener Pattern comes into practice in DES tools in two major ways. The central component of a DES is the Schedule, which is essentially a signaling device broadcasting the occurrence of events, to which other components respond, these components are considered Event Listeners. Event Listeners perform their response actions, change their state in some way, report their changes to the rest of the application, and then return to listening. If its internal state change requires it, a component can place a new event on the schedule to be executed at a future time determined by the component. The purpose of reporting changes is to allow components designed to listen for data, called Property Change Listeners to capture the change and, if they are so designed, either record or act upon it.

Scalability is one of the major strengths of the Listener Design Pattern. Adding more components to the design does not affect the basic structure, as long as the breakdown of fundamental tasks has not changed with the addition of the new components. The effect of the new components on the output may be considerable, but the original components of the application do not need to be adjusted in any way to accept them, which can enable rapid spiral development. Additionally, components tasked with data collection can be added and removed with no appreciable change to the application's basic operation.

Construction of the application is thus equivalent to building with Legos™[19]. Each of the components has a defined interface, and that interface allows a component to be combined in a particular way with other components with an identical (or similar) interface. Once a core set of components has been assembled to enable the core functionality, it is possible to continue to add functionality without adjusting the core. If it is necessary to adjust the core because it is not possible to manipulate the core's output, it is best done by swapping core components with different components, but possessing the same interface (created by polymorphism from the original component or implementing a common interface).

[19] The component-design elements in Viskit are sometimes referred to as Listener Event Graph Objects (LEGO).

3. Viskit Visual Development Environment for Simkit Discrete-Event Analysis

The building block approach lends itself to procedural code creation, which is the idea behind the creation of Viskit. Viskit provides a visual interface layer over Simkit, using event graph symbology to allow a user to take components of their own design—or selected from a repository of pre-built components—and connect them to create a simulation by manipulating icons and entering parameter values. The structure of the resulting event graphs and assemblies are stored in XML format, which is used by the tool as a map for the creation of Java code.

Simkit Event Graph	*XML Representation*
Parameter	Parameter Element
State Variable	State Variable Element
Event	Event Element
Scheduling edge	Schedule Element

Table 1. The mapping of Event Graph elements to XML elements by Viskit.[20] This mapping allows Viskit to store a compact description of an Event Graph, which can be procedurally turned to Java code for use in a simulation.

The XML files are compact and can be read or transformed to extract data about the interface of the component or the structure of the assembly.

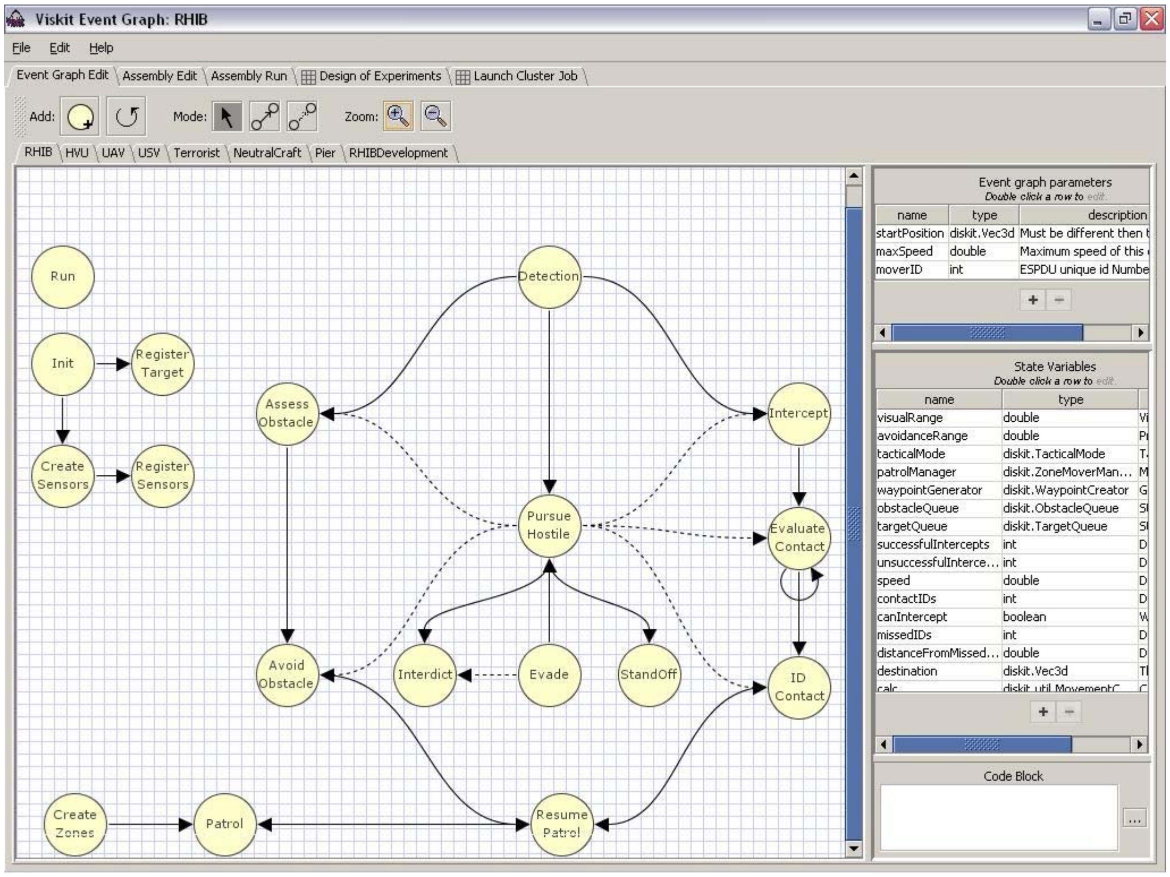
a. Viskit Event Graph Editor

Event graphs are the smallest functional component of a DES, constructed from events, scheduling edges, and canceling edges. Events are represented in an event graph as circles, scheduling edges as arrows, and canceling edges as arrows with dashed lines. When an event "occurs," it may schedule another event after a given time delay, represented by an arrow pointing to the event being scheduled, or cancel another event, represented by a dashed arrow pointing to the event.

The Event Graph editor pane in Viskit contains an area where Event Graphs can be constructed by dragging and dropping events onto the workspace, selecting the scheduling edge tool and dragging an arrow from the scheduling event to the one to be scheduled. Canceling edges work in a similar fashion.

[20] Buss 2004 p.5

Next to the graphical event-graph layout is a window that allows the user to add simulation parameters and state variables to the graph, which can be modified by the events. To set an event to modify a variable, the user selects the event and an editor window pops up, allowing the user to select the variable and write a snippet of code to manipulate it. Once the event graph is complete, the user saves it, and it becomes available for use in an assembly.



Event graph parameters
Double click a row to edit.

name	type	description
startPosition	diskit.Vec3d	Must be different then t
maxSpeed	double	Maximum speed of this
moverID	int	ESPDU unique id Numbe

State Variables
Double click a row to edit.

name	type	
visualRange	double	Vi
avoidanceRange	double	Pi
tacticalMode	diskit.TacticalMode	T.
patrolManager	diskit.ZoneMoverMan…	M
waypointGenerator	diskit.WaypointCreator	G
obstacleQueue	diskit.ObstacleQueue	Sl
targetQueue	diskit.TargetQueue	Sl
successfulIntercepts	int	D
unsuccessfulInterce…	int	D
speed	double	D
contactIDs	int	D
canIntercept	boolean	W
missedIDs	int	D
distanceFromMissed…	double	D
destination	diskit.Vec3d	Tl
calc	diskit.util.MovementC	C

Figure 22. Viskit authoring tool Event Graph Editor panel, depicting a completed event graph. Event Graphs are reusable patterns similar to class definitions.

b. *Viskit Assembly Editor*

Once the event graphs have been created, the user must connect them together to create a simulation. This is performed using the Assembly Editor. It works in much the same way as the Event Graph Editor, with the user selecting Event Graph files from the top file selection box and dragging them onto the graphic assembly editor area, then connecting them using the connector types provided at the top of the panel.

Event graphs are connected to each other by Event Listener edges, but to gather statistics, Property Change Listeners from the bottom file selector box must be connected to the Event graphs by Property Change Listener edges.

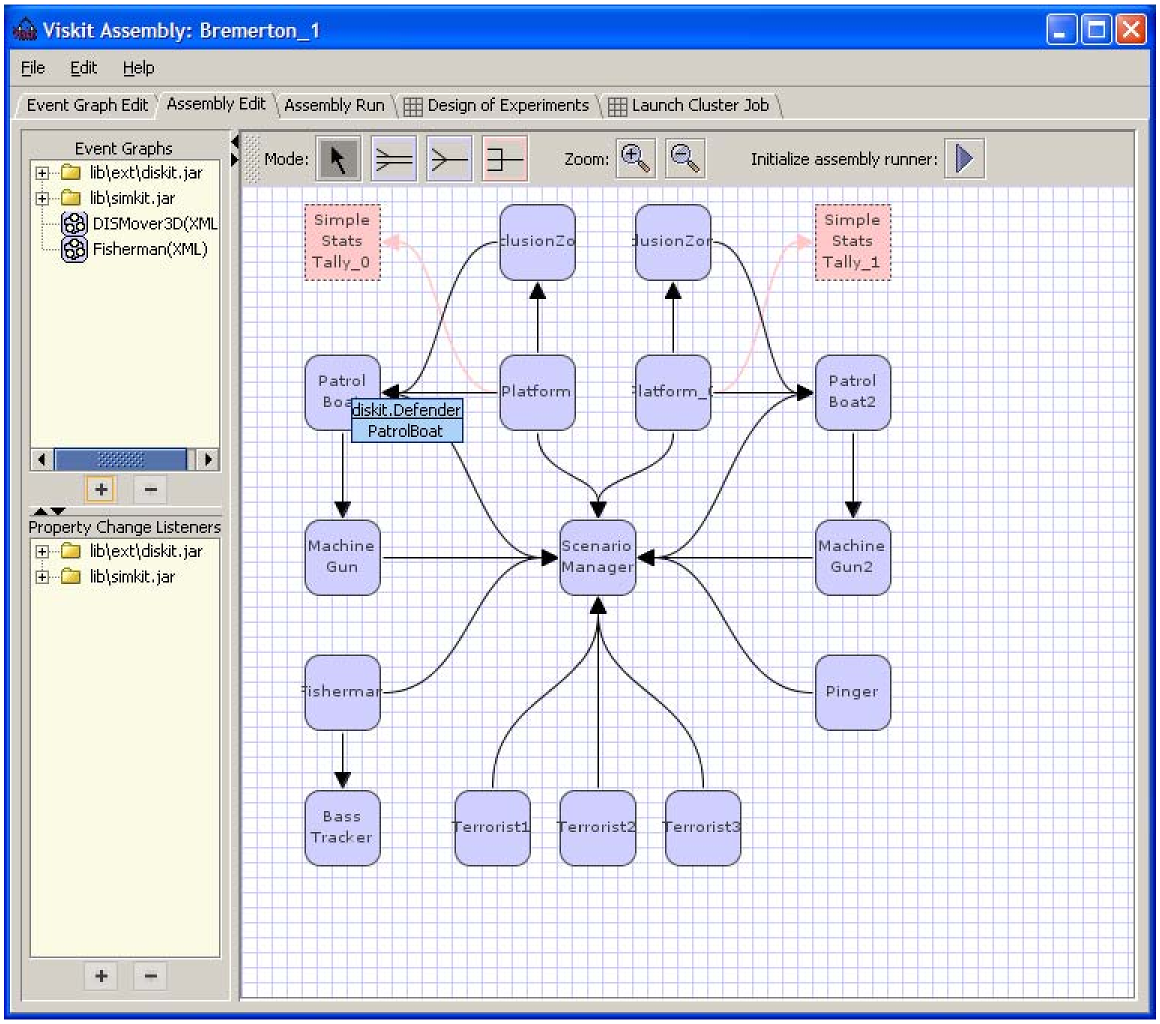
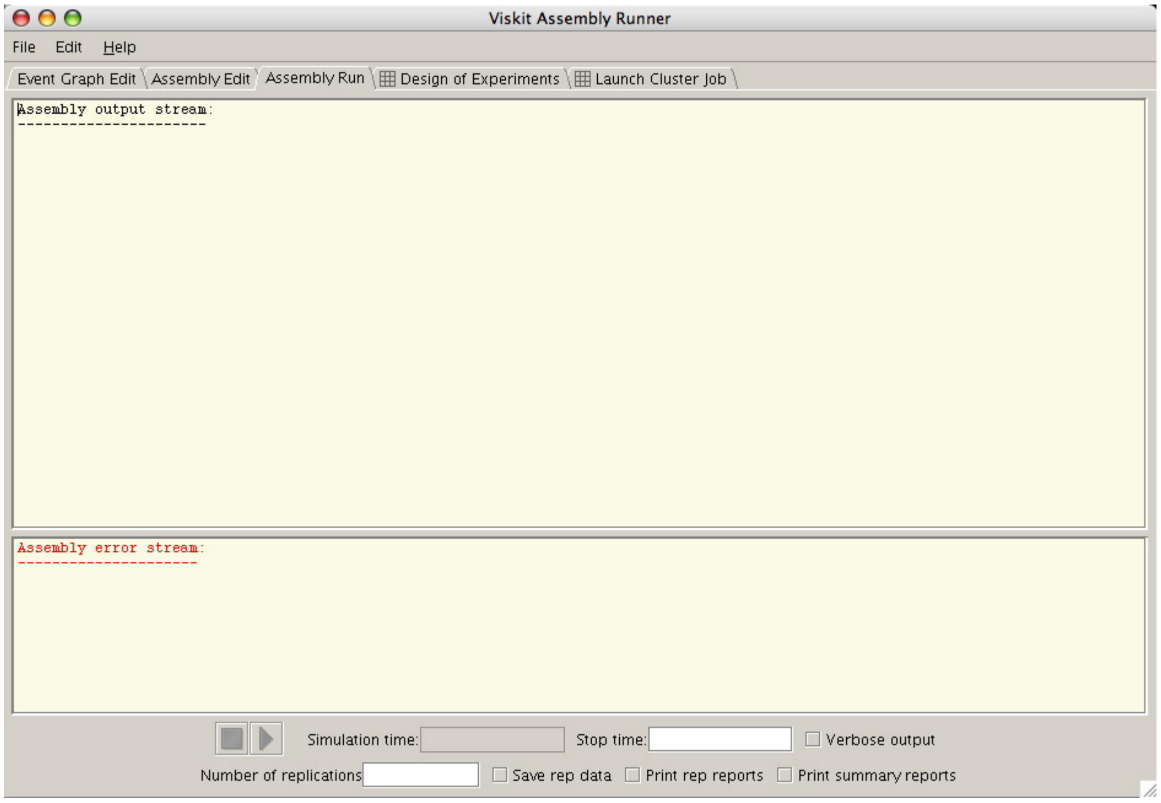


Figure 23. The Assembly Editor panel, showing event graphs connected to each other and PropertyChange Listeners (PCL) connected to the event graphs.

c. Viskit Assembly Run Panel

After the assembly is built and saved, it can be run using the Assembly Run panel. The duration of the simulation, the number of runs, and output options can be adjusted at the bottom of the panel, and the output streams are exposed in the two large text areas. Simple play and stop controls are used to control the simulation if necessary.



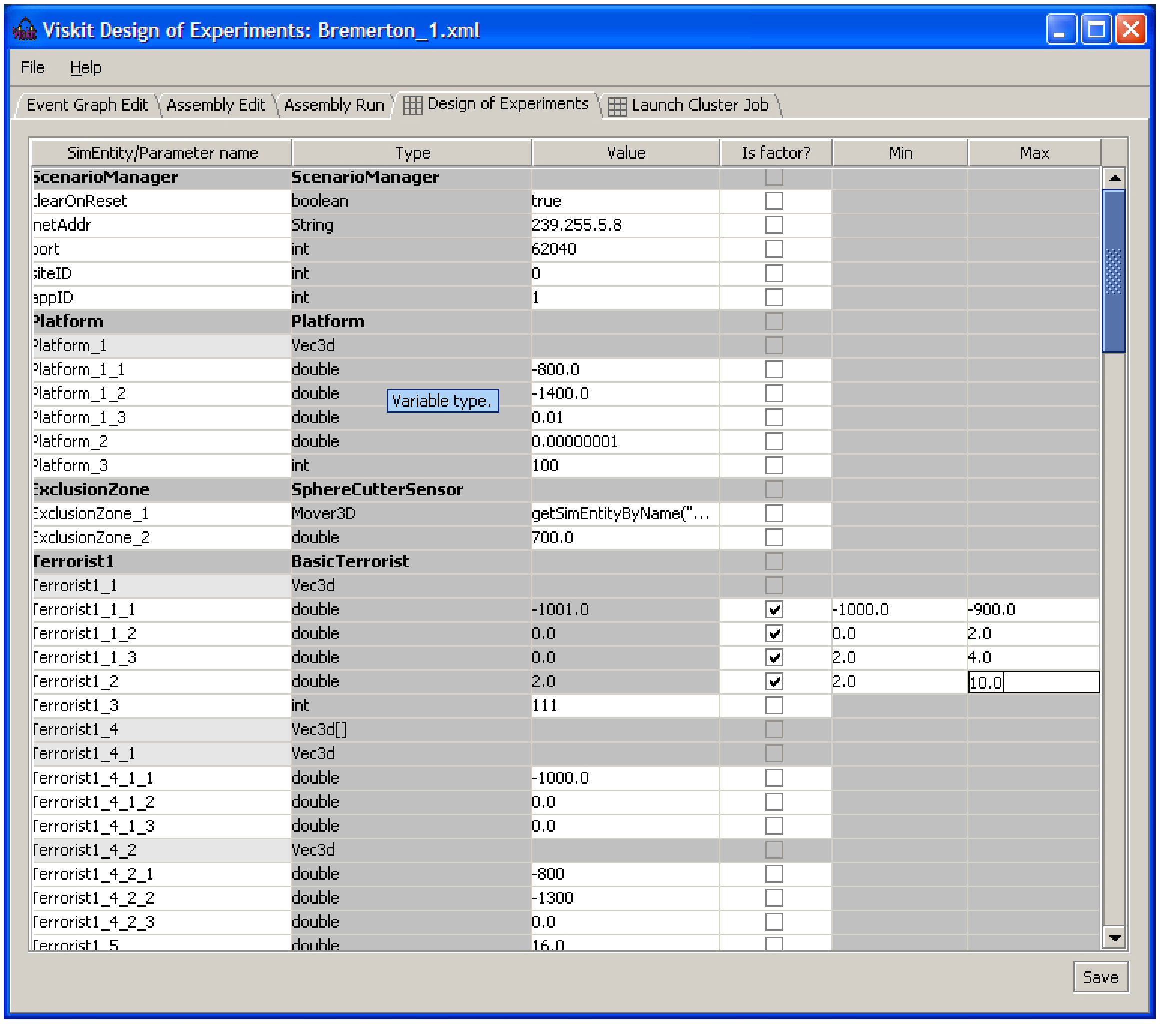
```
Assembly output stream:
----------------------
```

```
Assembly error stream:
---------------------
```

Figure 24. The Assembly Run panel, showing the output panes and the simulation replication, duration, and printout options.

d. Viskit Design of Experiments (DOE) Panel

All of the state variables and parameters that were designed into the various event graphs and the overall assembly are exposed in the field list on this panel, for a quick overview of the simulation and to allow parameters to be adjusted to perform sensitivity analyses or the explore the effects at the boundary conditions of the parameters.

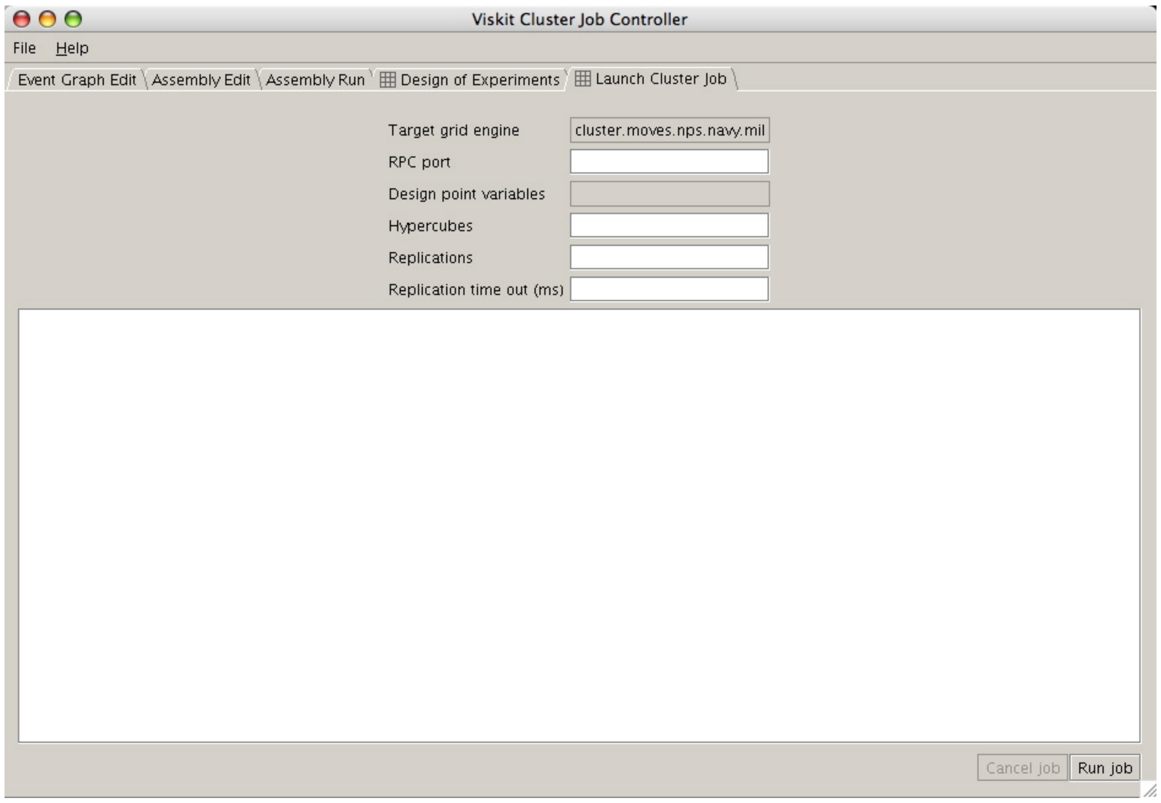


SimEntity/Parameter name	Type	Value	Is factor?	Min	Max
ScenarioManager	**ScenarioManager**		☐		
clearOnReset	boolean	true	☐		
netAddr	String	239.255.5.8	☐		
port	int	62040	☐		
siteID	int	0	☐		
appID	int	1	☐		
Platform	**Platform**		☐		
Platform_1	Vec3d		☐		
Platform_1_1	double	-800.0	☐		
Platform_1_2	double	-1400.0	☐		
Platform_1_3	double	0.01	☐		
Platform_2	double	0.00000001	☐		
Platform_3	int	100	☐		
ExclusionZone	**SphereCutterSensor**		☐		
ExclusionZone_1	Mover3D	getSimEntityByName("...	☐		
ExclusionZone_2	double	700.0	☐		
Terrorist1	**BasicTerrorist**		☐		
Terrorist1_1	Vec3d		☐		
Terrorist1_1_1	double	-1001.0	✓	-1000.0	-900.0
Terrorist1_1_2	double	0.0	✓	0.0	2.0
Terrorist1_1_3	double	0.0	✓	2.0	4.0
Terrorist1_2	double	2.0	✓	2.0	10.0
Terrorist1_3	int	111	☐		
Terrorist1_4	Vec3d[]		☐		
Terrorist1_4_1	Vec3d		☐		
Terrorist1_4_1_1	double	-1000.0	☐		
Terrorist1_4_1_2	double	0.0	☐		
Terrorist1_4_1_3	double	0.0	☐		
Terrorist1_4_2	Vec3d		☐		
Terrorist1_4_2_1	double	-800	☐		
Terrorist1_4_2_2	double	-1300	☐		
Terrorist1_4_2_3	double	0.0	☐		
Terrorist1_5	double	16.0	☐		

Figure 25. The Design of Experiments Panel with the exposed state variable and parameter fields from an assembly.

e. Viskit Launch Cluster Job Panel

Despite the speed advantage of DES, analysis of complex systems over thousands of runs can still take inordinate amounts of time. Viskit has the capability to take advantage of cluster computing to achieve supercomputer speeds by using groups of connected processors that distribute the computational load. The code necessary to utilize a computer cluster can be generated using a few customizable variables on the Launch Cluster Job panel.


Viskit Cluster Job Controller

File Help

Event Graph Edit | Assembly Edit | Assembly Run | Design of Experiments | Launch Cluster Job

Target grid engine	cluster.moves.nps.navy.mil
RPC port	
Design point variables	
Hypercubes	
Replications	
Replication time out (ms)	

Figure 26. The Launch Cluster Job Panel, showing the variables necessary to set up a cluster computing session to handle the analysis of an assembly.

4. Scenario Construction

Once built, a component-based simulation application has as its interface the aggregate of all the mutator interfaces of all the components. Some of those interfaces are for initialization, and some will be for internal communication only, not designed for access by the analyst at all. In this design philosophy, the base scenario is hard-wired by the assembly of the components, and the only effect the analyst can have is through adjusting the initialization parameters. There can be considerable leeway for adjustment in these parameters, but if the base scenario is to be changed, such as adding actions or event-handling capabilities, then components must be added or replaced. The idea of adjusting a component to change the simulation is not the preferred methodology, because it falls outside the component-based paradigm into the realm of custom coding. A well-designed suite of scenario creation tools would have a repository of components that covered the majority of actions and events necessary to develop scenarios in the target area of interest.

The tenets of the component-based design philosophy make it ideal for the purposes of autogenerating scenarios. Since the components are designed with a common interface, their purposes are specific, their inputs and outputs are established, and no single component needs to know of the existence of any other to operate, the parts can be procedurally combined to perform a wide range of analyses. This idea is still in its infancy, but research is moving forward with the development of, among others, Viskit and the AT/FP Toolkit.

F. CONNECTING NETWORKED INTERACTIONS

The virtual environment, scene construction, and scenario construction are linked in a visual simulation. Precisely how they are linked deserves discussion. In a component-based design, the components of a simulation can be divided into three parts by their basic functions: Interface, Logic, and Data. For the purposes of this paper, the virtual environment is the Interface, the simulation application is the Logic, and the Data to initialize them is supplied by the analyst. An appropriately componentized version of this application is able to easily switch or add interfaces without affecting the simulation engine. The virtual environment can just as easily run on any given simulation engine. The networked data communication are the driving force behind the ability for the two to interact. The exemplar application, AT/FP Toolkit, has little in the way of interaction once the simulation begins, but the ability may be required in the future to actively adjust parameters or extract data during run-time, so the interface must allow both components to transmit *and* receive data.

1. Distributed Interactive Simulation (DIS)

The primary network interface chosen by the AT/FP designers is the Distributed Interactive Simulation (DIS) protocol. This communications standard is made up of a set of Protocol Data Units (PDUs), which contains packets of data about specific events for specific objects in a simulation. Some may contain data about the precise state of a vehicle entity; others may contain the information that an explosion just occurred. DIS cannot handle initialization using its own protocols; the receiving component must have the entities to update before PDUs (especially Entity State PDUs or ESPDUs) can be used to update anything.

a. DIS-XML

The format of the standard DIS PDU is a package of 144 bytes in a strictly ordered format, which allows a DIS-enabled receiver to know precisely how to handle it. The DIS data received by a component can be used to update the state of its scene at any given interval. The benefit of this form of the protocol is its economy of size. The problem is its fragility in the face of line noise in the network or other forms of interference. NPS can now successfully translate DIS packets into a proposed XML format, which brings the benefits of a more robust format at the relatively low cost of a larger per-packet size. DIS-XML packets can be routed through firewalls over Extensible Messaging and Presence Protocol (XMPP) chat, enabling distributed collaboration among analysts.[21]

b. Packet Smoothing and Dead-Reckoning

DIS applications must be able to handle long periods between updates and remain visually intact. This is achieved by dead-reckoning calculations at the visualization end of the pipeline. Reasonable visual fidelity is achieved by "packet-smoothing" in which updates and dead-reckoning positions are averaged against each other to maintain smooth motion when packets arrive which differ greatly from the dead-reckoned position. It also forces DIS applications to limit the time between updates to some maximum amount, creating a "heartbeat" update for objects, which have not significantly changed their state.

c. Timing Effects When Coupling DIS with DES

This is a problem when using DIS to visualize a discrete event simulation, because DES event-queue time-steps are irregular while DIS PDU time-steps are real time or faster than real time, a timing mismatch must be resolved. Attempting to directly visualize a discrete event simulation results in a disjoint, inscrutable series of connected frames, so it is necessary to insert time-step artificiality into the simulation in order to update the scene in a fashion more fitting the DIS requirements. This creates a drag on the simulation, partially canceling out its benefit of limiting calculation overhead for entities that might not otherwise need to schedule an event. The advantage of the component-based simulation design exerts itself in this architectural decision, though.

[21] McGregor, 2006

Adding DIS capability to the simulation is a simple as establishing a PCL to collect position information from each of the entity components and set a component to schedule regular heartbeat events that cause the entities to report their positions, then send the data collected by the PCL as an ESPDU whenever there is a state change or the heartbeat component requests one.

2. Initialization and Data Capture

The communications between the application and the interface is only part of the triad of interface specifications necessary to fully describe this as a component-based system. Some architecture for data to flow from repository to scene and from repository to scenario is also required. Ideally, once the required data are collected and organized in one location, metadata and context are used to enable each component to extract the data it needs. This is the direction taken by this thesis to assist in the development of the AT/FP Toolkit.

G. SUMMARY

Each of the three processes involved in developing a 3D visualization application has its particular requirements and steps to accomplish, but the central inquiry of this thesis is the interplay of these steps with each other. The three processes can be loosely mapped to the functional components of an application: Interface, Logic, and Data. The task mapping is straightforward, so all that remains is to define the interface.

The Logic component—the simulation engine—is the core component of the application, making the other components primarily listeners. The Interface is currently for visual effect only, but it is possible to augment it to enable data from the visual representation to feed back to the Logic portion without significantly changing the current application.

THIS PAGE INTENTIONALLY LEFT BLANK

V. SAVAGE MODELING AND ANALYSIS LANGUAGE (SMAL) DESIGN AND IMPLEMENTATION

A. INTRODUCTION

Using a suite of tools available on the Macintosh comprised of two open-source projects and one proprietary editor, an XML schema was created and X3D structure were created to provide the data transport architecture for the AT/FP Toolkit. A standardized structure was designed to insert data into 3D models for runtime extraction on an as needed basis. The XML schema was designed to capture that data as well as other general data necessary to build a scene and set up a scenario in a notional application, but with specific support to the AT/FP Toolkit.

B. OVERVIEW

Creating the SAVAGE Modeling Analysis Language (SMAL) was a three-step process. First was the conceptualization and creation of the MetadataTemplate feature. The concept was to place information into 3D object models, which describes information not readily available in the visual representation of the object, but which may have some impact on the behavior of the object the model represents in simulation. The second step was the development of the SMAL schema, which involved integrating the kind of data to be found in the Metadataset feature with simulation-specific or scene-specific information and any information found in other sources that are germane to the process of generating a 3D scene and its scenario. The final step was to build the supporting infrastructure and code-base, which might allow applications to easily annotate, discover and utilize the metadata in the X3D MetadataSet nodes and functionally equivalent SMAL files.

The specific applications in mind were the suite of tools found in the AT/FP toolkit; specifically Savage Studio, VisKit, and Xj3D. These tools represented the exemplar application of SMAL, but its development was not limited to the scope of the AT/FP toolkit. The likely alternate applications of SMAL also drove its development, so the processes necessary in the development of scenes and scenarios for the AT/FP toolkit were generalized where possible, or else made expandable or replaceable where they were not. Specific attention was paid to the ability of other applications to leverage SMAL to collect and store the data necessary to reproduce 3D scenes and scenarios with minimum loss of detail.

C. DEVELOPMENT TOOLS

Partly due to the construction of the AT/FP Toolkit and partly by independent choice, the following tools were chosen to develop SMAL. The majority of applications under construction for military simulations are Windows-based and developed on computers using the Windows operating system. The development environment for SMAL and the majority of the associated code in this thesis was Apple's Macintosh OS 10.4.5. The XML editor was <oXygen> 7.1 and the X3D editor was a version of X3D-Edit, which was modified (slightly) to run on the Macintosh. Because each of the SMAL-related products are based on XML, zero dependencies on specific operating systems were produced.

1. Extensible 3D (X3D) and X3D-Edit

Using X3D as the example 3D modeling language had some inherent advantages, not least of which is that it an open-source, open-standards format. X3D-Edit was chosen as the primary X3D editor because it provides the greatest standards conformance, most control over scene structure, and is built using Java on the Xeena XML editor provided under public license by IBM's Alphaworks Software, making it freely available. X3D-Edit also has useful features of built-in error checking, VRML97 import, and XSLT export to VRML97, Classic VRML, and HTML PrettyPrint formats.

2. Xj3D

Developed by Yumetech in conjunction with the Web3D Consortium, Xj3D is a Java-based native-X3D browser with the capability of operating as a stand-alone browser or a plug-in to bring 3D visualization to Java applications. It has been developed, using the spiral-development method, to be completely standards-compliant by functionality as it is introduced. It is the only native X3D browser currently released which works on the Macintosh OS.

3. <oXygen>

This is one of two pieces of proprietary software used in the development of this thesis. Its capabilities include XML document and schema editing, XSLT development, XQuery debugging, and is equipped to handle editing of specific standard schema such as XHTML, SVG, and WSDL.

4. Altova XMLSpy

XMLSpy, a Windows operating system-only XML editor, was used to automatically generate example documents, HTML documentation, and translate early versions of the schema into a Document Type Definition (DTD) format. A variety of other XML development and verification was also performed using the Altova suite of tools. The Naval Postgraduate School MOVES Institute continues to benefit from an academic partnership with Altova

D. THE SAVAGE MODEL ARCHIVES

Based on a 2001 thesis work[22], two repositories of 3D object models and authoring tools were established at the Naval Postgraduate School. The Savage Model Archive is an open, public X3D model resource, while Savage Defense is a restricted-access, For Official Use Only (FOUO) archive reserved for unclassified but sensitive models. The archives were created to provide support for the creation of military-themed 3D scenes for planning support and scenario visualization. In order to expose the models for use by planners and scene authors, a content catalog generation program was updated to procedurally list, describe, and index the models as a set of XML documents based on the file structure of the archive and any descriptive metadata in the headers of the models. The catalog generator application has been leveraged to also expose the SAVAGE Metadata Template information within the catalog by transforming it into a SMAL metadata fragment within the catalog. Applications like Savage Studio can read the catalog and extract the SMAL metadata when accessing the models in the Archive.

E. SAVAGE METADATA TEMPLATE

It quickly became apparent that the attempt to insert the kinds of data into a model an application might use for simulation is a monumental task, not necessarily by process, but certainly by volume. For every level of simulation there is a corresponding depth of

[22] Nicklaus, 2001

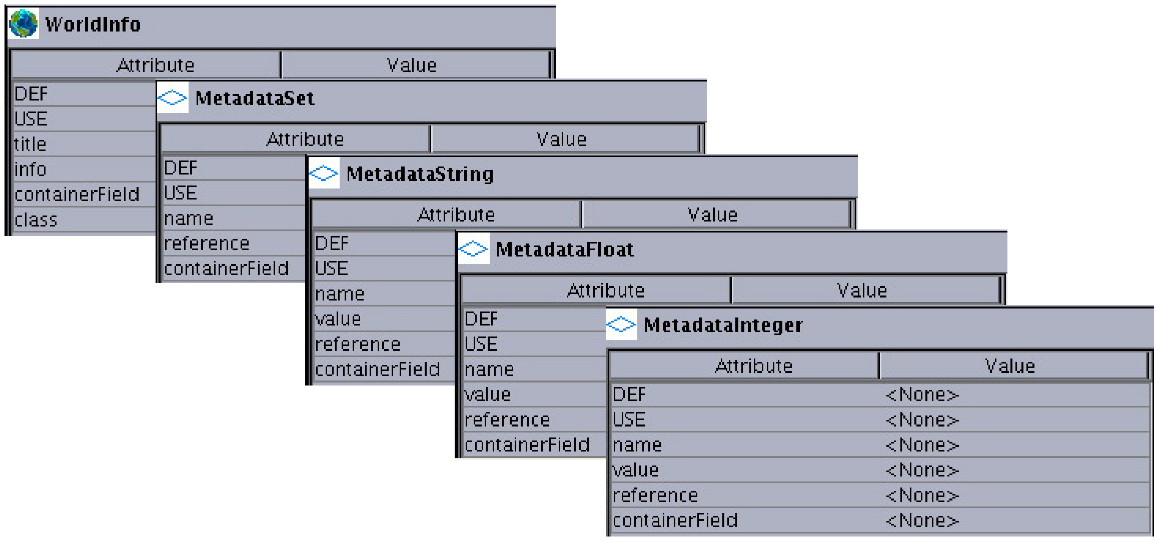
detail necessary to properly initialize it and allow it to perform its calculations. It is necessary to establish a cutoff point for the detail that is included in an object model, lest that model become too bloated with data to be used efficiently within authoring tools for producing visual simulations. As a result the detail cap placed on the metadata inserted in X3D models is at a fairly low level. Gross parametric data was the perceived limit, with more detailed, in-depth metadata left as a matter of future work or custom extensions. The included metadata was grouped by usage and availability where possible.

Because X3D is an XML language, there are several potential methods of data inclusion into object model files, but there were only two basic paradigms for those methods. The first was to work within the X3D specification and use built-in features to hold metadata in a way that is accessible once the model is loaded at runtime. Pertinent X3D metadata nodes implement the X3DMetadataObject interface. This type of metadata storage brings with it some overhead in the form of its forced structure. X3DMetadataNodes, because they are pre-built to capture a single datum as an attribute of the node, can only hold the information as individual name-value pairs, which significantly expands the number of nodes required to hold the entire set. The second paradigm was to make use of the extensibility feature of XML in some way, such as including another XML schema into the document to allow the data to be held in a much more space-efficient manner. The problem with this method is that, because it is not a part of the core X3D schema, browsers are not able to access the data at runtime, and the documents may fail validation depending on the way in which unrecognized tags are handled by the underlying XML parser. Although XML namespaces[23] can handle this situation, multiple namespace support is not implemented by any of the current X3D browsing or visualization tool. Thus this second approach is deferred as a future work.

1. X3DMetadataObject

In order to ensure that Savage Metadata-enabled models are readable, and to make it possible to access the data at runtime, the X3DMetadataObject approach was chosen. X3DMetadataObjects are sub-classed as five node types: MetadataSet, MetadataDouble, MetadataFloat, MetadataInteger, and MetadataString.

[23] Namespaces in XML 1.1, 2004



WorldInfo

Attribute	Value
DEF	
USE	
title	
info	
containerField	
class	

MetadataSet

Attribute	Value
DEF	
USE	
name	
reference	
containerField	

MetadataString

Attribute	Value
DEF	
USE	
name	
value	
reference	
containerField	

MetadataFloat

Attribute	Value
DEF	
USE	
name	
value	
reference	
containerField	

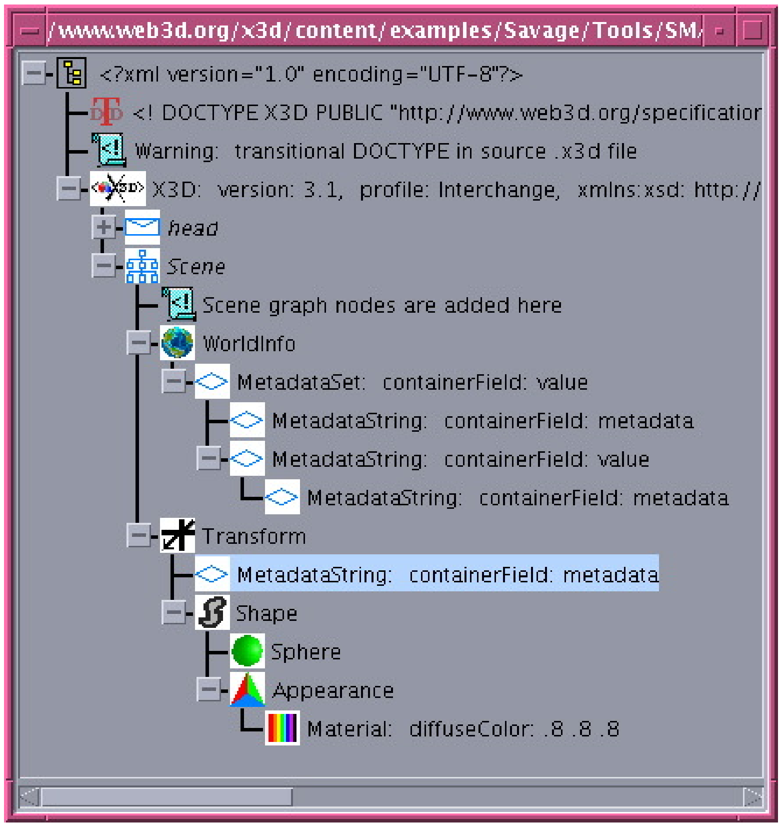
MetadataInteger

Attribute	Value
DEF	<None>
USE	<None>
name	<None>
value	<None>
reference	<None>
containerField	<None>

Figure 27. The X3D WorldInfo and Metadata nodes use simple "name-value" pair to capture metadata and provide a field to include a reference locator.

Rather than setting each piece of Metadata separately as a name-value pair in a long, flat list structure, it is beneficial for parsing and maintainability to provide a more tree-like structure. Using MetadataSet nodes as collection nodes (one-to-many relationship), metadata nodes containing data of similar categories are placed into a separate tree-structure within the X3D document.

MetadataSet's peculiarity is in the fact that it can hold X3DMetadataObjects both as self-referential metadata and also as child nodes. This can cause a disruption in reading the metadata under the top-level MetadataSet node. If data that is to be accessed is considered metadata for MetadataSet, it is handled differently than if it is considered a value. For the purposes of Savage Metadata, it is important to ensure the X3DMetadataObjects are added to the top-level MetadataSet with containerField "value" rather than containerField "metadata." In X3D-Edit this distinction is handled using the containerField pull-down menu, which allows either metadata to be chosen or value to be typed in. X3D-Edit further warns the user that if a single X3DMetadataObject child of an X3DMetadataObject has its containerField set to metadata, the parent X3DMetadataObject is empty (has no value). If the X3DMetadataObject parent has two X3DMetadataObject children, one set to metadata and the other to value, then the two are handled as if they are in different trees, rather than two branches on the same tree.


/www.web3d.org/x3d/content/examples/Savage/Tools/SM/

```
<?xml version="1.0" encoding="UTF-8"?>
  <! DOCTYPE X3D PUBLIC "http://www.web3d.org/specificatior
  Warning:  transitional DOCTYPE in source .x3d file
  X3D:  version: 3.1,  profile: Interchange,  xmlns:xsd: http://
    head
    Scene
      Scene graph nodes are added here
      WorldInfo
        MetadataSet:  containerField: value
          MetadataString:  containerField: metadata
          MetadataString:  containerField: value
            MetadataString:  containerField: metadata
      Transform
        MetadataString:  containerField: metadata
        Shape
          Sphere
          Appearance
            Material:  diffuseColor: .8 .8 .8
```

Figure 28. An example tree structure of metadata using MetadataSet in an X3D Scene graph.

2. WorldInfo

X3DMetadataObjects are not allowed to be placed directly under the root node of the scene, and so must be set as a metadata child of another node. Most X3D nodes have the ability to contain X3DMetadataObjects but, to set the Savage Metadata Template apart, it is set under a WorldInfo node. The WorldInfo node is specified to "contain information about the world. This node is strictly for documentation purposes and has no effect on the visual appearance or behavior of the world."[24] This solution works well and is the recommended approach for adding SMAL to an X3D scene.

Setting the WorldInfo node as the root for the Metadata poses some difficulties when translating the X3D into VRML97 format—which is necessary for some web-browser plug-ins to visualize 3D scenes—since VRML97 does not support metadata under its equivalent WorldInfo node. The XSLT document used by X3D-Edit to translate

[24] X3D Specification

X3D files into VRML97 format was modified to place the entire metadataset under a standard VRML97 group node and publish a non-blocking warning to the user that the shift had occurred. For this reason, the top-level MetadataSet node's containerField must not be set to value.

3. The SAVAGE Metadata Template

The Savage Metadata Template (see Appendix A) format is built as a scene consisting of the entire set of required Metadata nodes with pre-set name attributes, value attributes set to default values, MetadataString descriptors of the value content, and no geometry or other scene structure. Using the Template is a matter of copying the top WorldInfo node and associated Metadata node children, pasting it into the X3D object model, filling in the appropriate values, and removing non-required nodes that are not germane to the object depicted in the model.

There are three Savage Metadata Template types, the Vehicle template, the StaticModel template, and the Terrain Template. Each allows type-specific metadata to be inserted to describe the virtual object. There may be need in the future to expand this catalog of templates if other kinds of entities, like munitions or sensor objects, are to be modeled and visualized. The Metadata in the .x3d model is not in a form that is usable for other applications, unless they are made aware of the X3D metadata standards, so it must be translated for use outside X3D into a more universal format, namely XML. The universal format is part of the design requirement for SMAL, which uses built-in XML data-types and restricted sub-types to validate the data. The specific structure of the metadata aids in translation to SMAL code snippets. Bi-directional round-trip conversion between X3D and XML without loss of metadata semantics is thus an essential design requirement.

F. SMAL

Researching the required and available content for inclusion in the SMAL schema took the most time and effort in this project, with naming and design considerations coming in a close second. This section will outline the naming and design considerations, the content collection affected development of both the Savage Metadata Template and SMAL.

The Navy-published XML development standards contain specific guidance on naming and documentation, were deliberately only loosely followed in the development of SMAL.[25] Complete compliance is left to future work if it is determined that this schema should be registered with the DoN CIO. The inclusion of Basic Business Identifier Elements (BBIE) and the supporting documentation style might be a thesis level effort in itself to determine whether significant value is added to SMAL. This and the development history of every other standard XML schema discussed in this thesis say something for the process of schema development: it is typically a committee process with a collaborative approach to design, and for good reason. Because standardization is a typical use for XML, it is fitting that using a consortium development team is the standard of practice.

1. Basic SMAL Concepts

SMAL development attempted to take a long view toward interoperability with the understanding that the initial work is exactly that; initial. Further innovations are expected. Extensibility and modularity were built into structure of the schema from the ground-up, so the resulting content will not be hindered by the format in which it was built.

a. XML Schema-based Extensibility Features

The basic element type is based on its type of extensibility: Full, Element, or Attribute. Element extensibility is provided by using the <xs:any> and <xs:anyAttribute> tags requiring new elements to be in another namespace and providing lax validation. If universal attributes need to be added at some point, they can be added to these base types, and propagate throughout the document.

b. Type Declarations

Where appropriate, element types are declared outside their corresponding element declarations to encourage type re-use and avoid overloading with different eponymous types. Abstract elements and substititionGroups are used when a choice between multiple element types is desired, allowing new elements to be added to the

[25] NDR, 2005

substitutionGroup if necessary to expand the palette of choices.   Enumerations and derived simpleTypes are broken out into separate includes to allow easy access to them for the purposes of modification.

c. Naming Conventions

Some simple rules went into element and attribute naming, which were followed as much as practicable throughout development. Elements that are used to collect multiple sub-elements of one type are named <*ElementName***Set**>, where *ElementName* is the kernel name of the sub-element. Attributes designed to identify a particular element out of a set are named *elementName***Identifier**, and are of type xs:ID. Attributes designed to refer to a particular element out of a set are named *elementName***Reference** and are of type xs:IDREF. Definitions of simpleTypes are named *descriptor***Type** if they are simple restrictions, *descriptor***Pattern** if they restrict by using a regular expression pattern, or *descriptor***Enumeration** if they restrict by using enumerations.

2. Element-wise Structure of SMAL Simulation Metadata

a. Overview

SMAL is a metadata storage format, so it attempts to sort the data it holds into logical groups with temporal characterizations. The temporal categories were Inherent, Parametric, and Instantaneous. Inherent metadata is that which is resistant to conditional change. This data category includes the physical size and weight of the object, the maximum values of inherent capabilities such as speed, acceleration, and detection or engagement range. Parametric data includes conditions that are set to determine the scenario—environmental conditions, associations between entities, terrain and objects in the terrain, behaviors, and identifiers. The Instantaneous data consists of items required to describe the situation at a given moment in time. A description of the scene at the very first moment of the scenario is both an instantaneous description of the scene and a large part of the initialization parameters. Position and orientation, current speed, acceleration, fuel state, and time are all Instantaneous data.

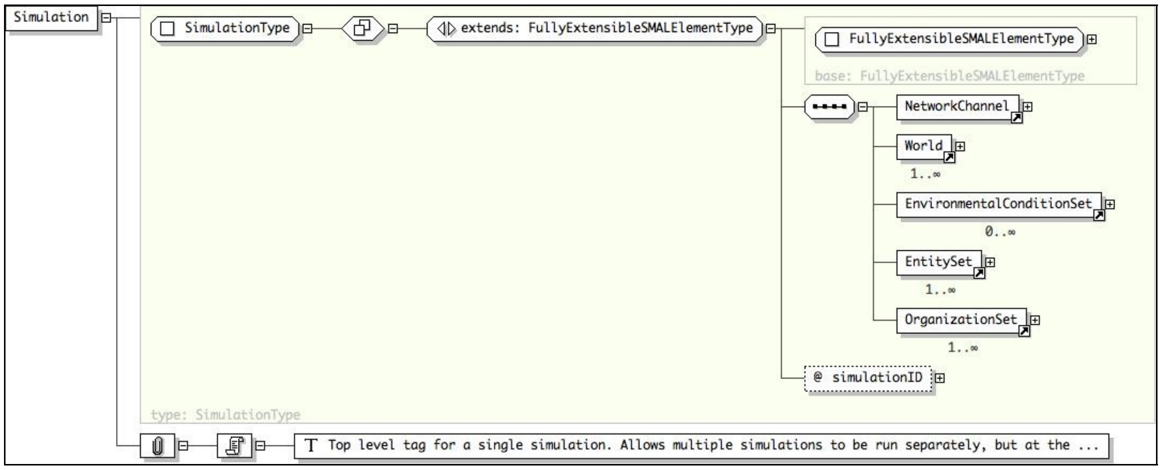

Simulation
SimulationType
extends: FullyExtensibleSMALElementType
FullyExtensibleSMALElementType
base: FullyExtensibleSMALElementType
NetworkChannel
World
1..∞
EnvironmentalConditionSet
0..∞
EntitySet
1..∞
OrganizationSet
1..∞
@ simulationID
type: SimulationType
Top level tag for a single simulation. Allows multiple simulations to be run separately, but at the ...

Figure 29. The task-oriented division of data into five groups, Network, World, Environment, Entities, and Organization.

The task-oriented division of data was based on what was necessary to run a single simulation. The first category, Network information, included the data necessary to set up communications for participation in a DIS network. The second category is used to collect all static objects, including terrain models, which are to be placed in the scene but are not to be controlled by a simulation element. EnvironmentalConditions holds weather conditions, background color information, and time data. EntitySet contains the objects in the scene that are driven by simulation elements. Finally, the experimental Organization element is available to define the hierarchy and basic associations between entities, if required.

b. Classification

Before covering specific data elements, there are two more global design considerations that deserve mention. First, classification of the files is based on the paragraph-wise classification of documents in government publications. An element with an enumeration for classification level and attributes for reference documentation and rationale behind the classification is made available at various points throughout the schema to aid in properly classifying a document which may contain some amount of sensitive information. While no security is provided by this scheme, it provides the ability to extract sensitive information from a scenario to make it clearable for public release. For example, a scenario using a classified set of data about a proposed weapons system is run for analysis purposes. None of the other entities in the simulation contain classified data. As a whole, the simulation takes on the classification of the weapon system data, even though that is the only sensitive data in the entire document. If that set of data can be identified and removed, the rest of the scenario can be re-used for unclassified purposes.

c. unitSystem

A highly important piece of metadata about any number is the units in which it was measured. For this purpose a universal attribute, unitSystem, is made available to provide a gross breakdown of Metric versus English unit systems by including it in appropriate elements. Metric units are assumed to be meters, kilometers per hour, meters per second, kilograms, and liters. English units are assumed to be feet, miles per hour, feet per second, pounds, and U.S. gallons. Enumerations have been created to identify specific measurement schemes, but are not yet validatable directly, pending further development of the schema. Further SMAL schema work is recommended.

d. NetworkChannel

One NetworkChannel is allowed per Simulation. It holds all networking information for the simulation, such as multicast address, ports, relay hosts, and the high-level DIS information such as site and application identification.

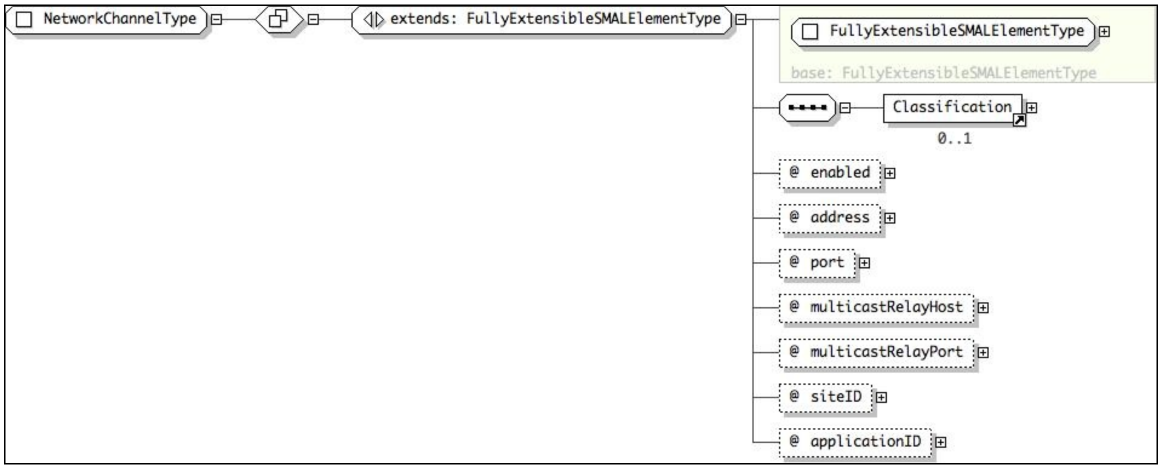


Figure 30. The NetworkChannel element showing the direct mapping to high-level Discrete-Event Simulation (DIS) data found in DIS Protocol Data Units (PDUs).

e. World

The World element contains the GeoOrigin, which is the information to set the geographic reference point of the simulation. It also contains the TerrainTileSet, and StaticModelSet elements, which in turn hold the terrain models in the form of TerrainTiles and all the non-terrain scenery in the form of StaticModelDefinitions.

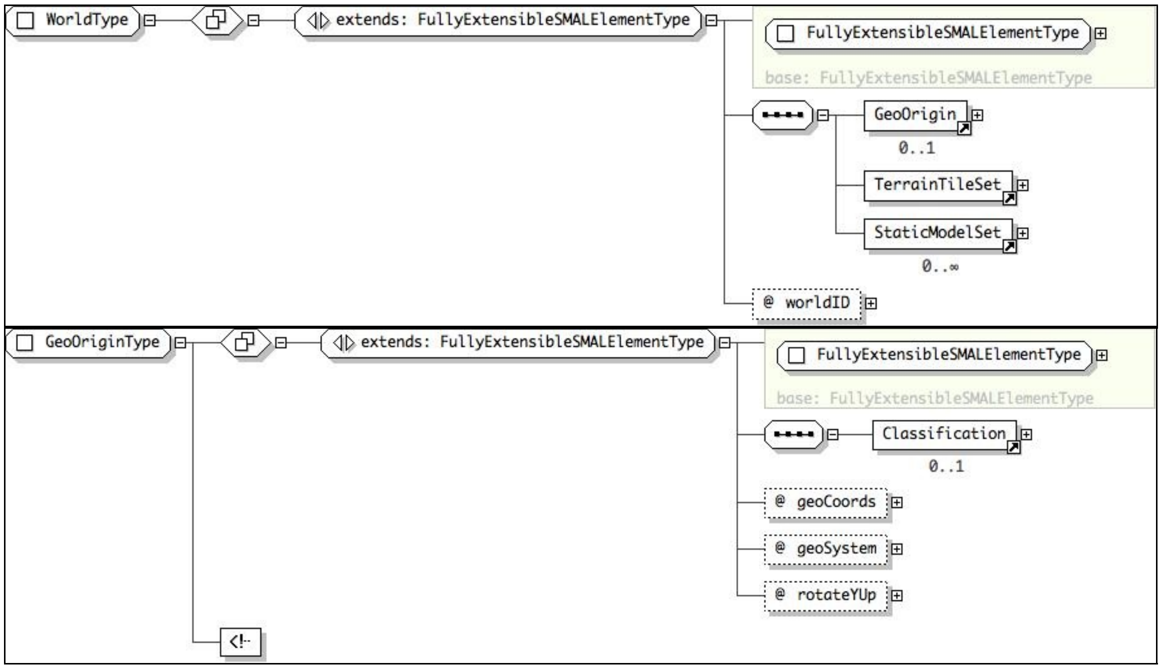

WorldType
extends: FullyExtensibleSMALElementType
FullyExtensibleSMALElementType
base: FullyExtensibleSMALElementType
GeoOrigin
0..1
TerrainTileSet
StaticModelSet
0..∞
@ worldID
GeoOriginType
extends: FullyExtensibleSMALElementType
FullyExtensibleSMALElementType
base: FullyExtensibleSMALElementType
Classification
0..1
@ geoCoords
@ geoSystem
@ rotateYUp

Figure 31. The World and GeoOrigin element types, which contain the static models and geographic reference points in a simulation.

f. EntitySet and EntityDefinition

The EntitySet element holds the EntityDefinitions for every actor in a simulation. The EntityDefinitions in turn hold all inherent, parametric and instantaneous data about the actor.

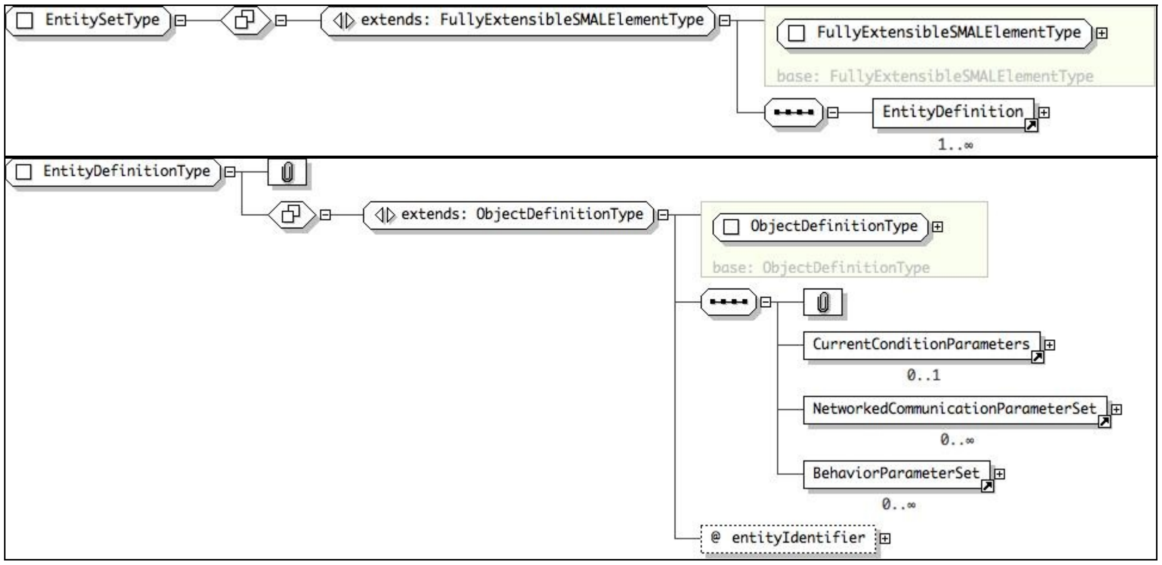
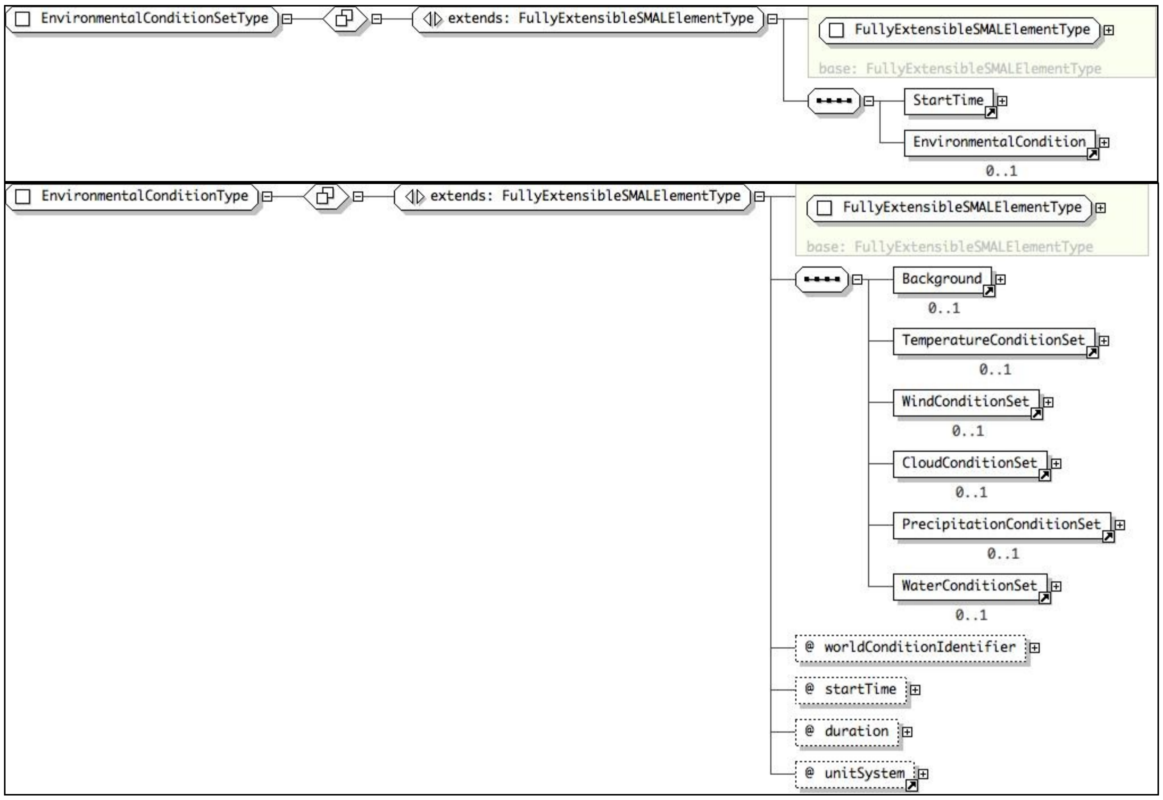

EntitySetType
extends: FullyExtensibleSMALElementType
FullyExtensibleSMALElementType
base: FullyExtensibleSMALElementType
EntityDefinition
1..∞
EntityDefinitionType
extends: ObjectDefinitionType
ObjectDefinitionType
base: ObjectDefinitionType
CurrentConditionParameters
0..1
NetworkedCommunicationParameterSet
0..∞
BehaviorParameterSet
0..∞
@ entityIdentifier

Figure 32. EntitySet and EntityDefinition elements used to hold the data for every actor in a scene.

g. *EnvironmentalConditionSet*

Because environmental conditions can change over the course of a simulation, the EnvironmentalConditionSet holds the StartTime element of the simulation and a set of EnvironmentalCondition elements. The StartTime element provides the ability to set the start time and virtual run-time of the simulation and allows specification of the day and month in anticipation of procedurally identifying light color and direction simulating daylight conditions. Each EnvironmentalCondition element has a start time and duration and holds all data about temperature, wind, clouds, precipitation, and water current conditions. The Background element included in the EnvironmentalCondition element matches the X3D Background node field to field, since it is necessary to manage background colors to mimic changing sky conditions, including nightfall.

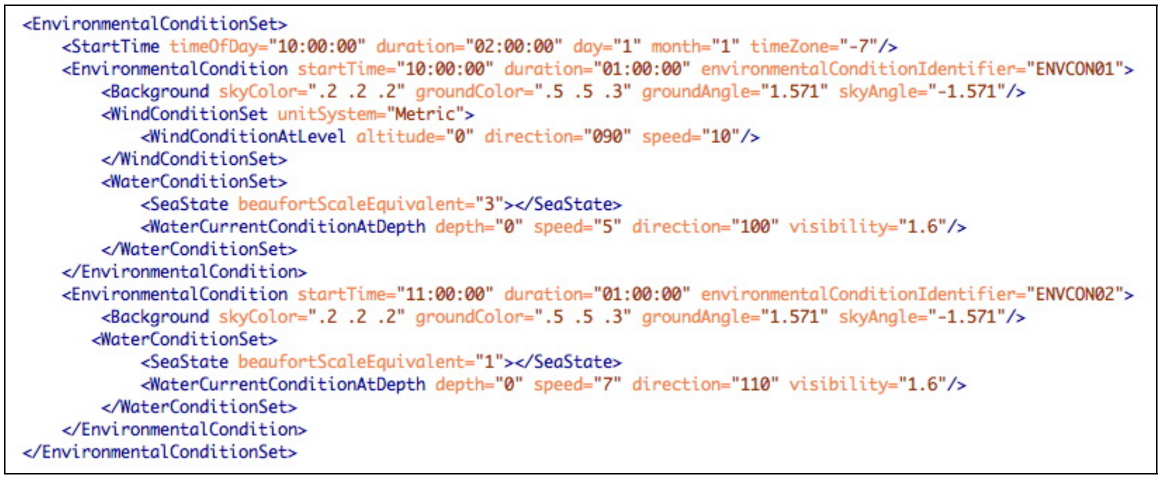
The TemperatureConditionSet provides altitude-based control of air temperature and depth-based control of water temperature in the simulation. WindConditionset provides altitide-based control of wind direction and speed (also allowing for gusting conditions). The CloudConditionSet can contain multiple cloud layers by altitude, modifiable using National Weather Service cloud coverage enumerations. It also contains a visibility modifier, which can be mapped to a Fog node in X3D. Precipitation also contains a visibility modifier, NWS standard precipitation type enumerations, and allows precipitation to be localized by providing a GeographicExtent element. Finally the WaterConditionSet allows control of visibility, current, and salinity data by depth.



EnvironmentalConditionSetType
extends: FullyExtensibleSMALElementType
FullyExtensibleSMALElementType
base: FullyExtensibleSMALElementType
StartTime
EnvironmentalCondition
0..1
EnvironmentalConditionType
extends: FullyExtensibleSMALElementType
FullyExtensibleSMALElementType
base: FullyExtensibleSMALElementType
Background
0..1
TemperatureConditionSet
0..1
WindConditionSet
0..1
CloudConditionSet
0..1
PrecipitationConditionSet
0..1
WaterConditionSet
0..1
@ worldConditionIdentifier
@ startTime
@ duration
@ unitSystem

Figure 33. EnvironmentConditionSet and EnvironmentCondition elements used to collect temporal and weather effect data.

Using multiple EnvironmentalConditions allows weather to be shifted from one set to another by setting the durations by individual Condition sub-element or as a whole. Individual Conditions, once set, can be considered to continue until their duration runs out or they are superseded by another Condition of the same type.



```xml
<EnvironmentalConditionSet>
    <StartTime timeOfDay="10:00:00" duration="02:00:00" day="1" month="1" timeZone="-7"/>
    <EnvironmentalCondition startTime="10:00:00" duration="01:00:00" environmentalConditionIdentifier="ENVCON01">
        <Background skyColor=".2 .2 .2" groundColor=".5 .5 .3" groundAngle="1.571" skyAngle="-1.571"/>
        <WindConditionSet unitSystem="Metric">
            <WindConditionAtLevel altitude="0" direction="090" speed="10"/>
        </WindConditionSet>
        <WaterConditionSet>
            <SeaState beaufortScaleEquivalent="3"></SeaState>
            <WaterCurrentConditionAtDepth depth="0" speed="5" direction="100" visibility="1.6"/>
        </WaterConditionSet>
    </EnvironmentalCondition>
    <EnvironmentalCondition startTime="11:00:00" duration="01:00:00" environmentalConditionIdentifier="ENVCON02">
        <Background skyColor=".2 .2 .2" groundColor=".5 .5 .3" groundAngle="1.571" skyAngle="-1.571"/>
       <WaterConditionSet>
            <SeaState beaufortScaleEquivalent="1"></SeaState>
            <WaterCurrentConditionAtDepth depth="0" speed="7" direction="110" visibility="1.6"/>
        </WaterConditionSet>
    </EnvironmentalCondition>
</EnvironmentalConditionSet>
```

Figure 34. An example EnvironmentalConditionSet showing two consecutive sets of conditions. In the second hour, the winds time-out and go to zero, but the sea state and current shift.

h. OrganizationSet

This element is an experimental attempt to codify the basic relationships between entities in a simulation. This is not necessary in the AT/FP Toolkit, since the basic organization of the simulation is captured in the Assembly XML file generated by Viskit. The functionality of OrganizationSet is for simulation engines that do not have a similar construct for defining the hierarchy and interrelationships among entities.

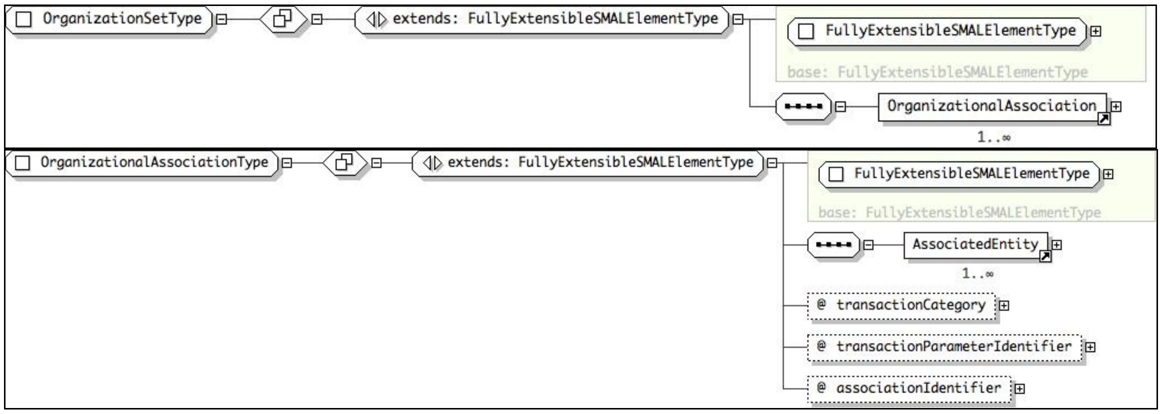
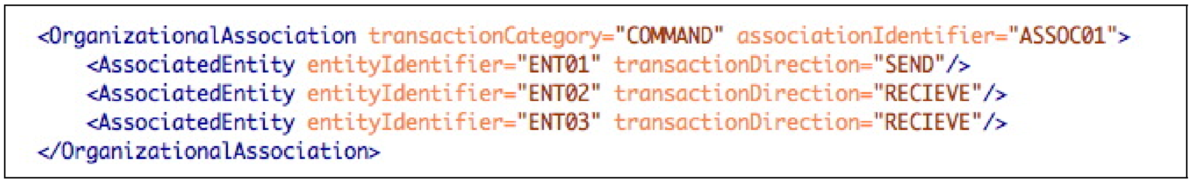

OrganizationSetType
extends: FullyExtensibleSMALElementType
FullyExtensibleSMALElementType
base: FullyExtensibleSMALElementType
OrganizationalAssociation
1..∞
OrganizationalAssociationType
extends: FullyExtensibleSMALElementType
FullyExtensibleSMALElementType
base: FullyExtensibleSMALElementType
AssociatedEntity
1..∞
@ transactionCategory
@ transactionParameterIdentifier
@ associationIdentifier

Figure 35. OrganizationSet and OrganizationAssociation types used to establish the interrelationships between entities in a scenario.

The basic concept is to create OrganizationalAssociations among a set of entities. The type of transaction is set using the transactionCategory enumeration and the AssociatedEntities are named by entity IDREF with their place in the hierarchy of the association set by the transactionDirection enumeration. For example, a command relationship between ENT01, ENT02, and ENT03 are set up as:



```
<OrganizationalAssociation transactionCategory="COMMAND" associationIdentifier="ASSOC01">
    <AssociatedEntity entityIdentifier="ENT01" transactionDirection="SEND"/>
    <AssociatedEntity entityIdentifier="ENT02" transactionDirection="RECIEVE"/>
    <AssociatedEntity entityIdentifier="ENT03" transactionDirection="RECIEVE"/>
</OrganizationalAssociation>
```

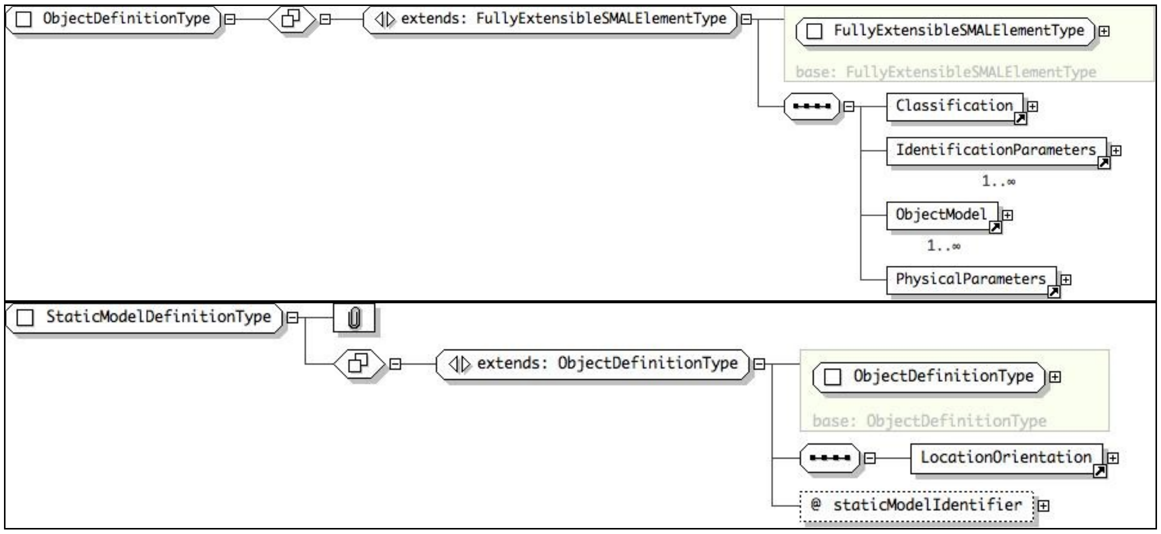
Figure 36. An example OrganizationalAssociation showing a command relationship in which ENT01 has command over ENT02 and ENT03.

3. StaticModelDefinition, EntityDefinition, and TerrainTile

A SMAL file can have either a set of Simulation elements or a single EntityDefinition, StaticModelDefinition, or TerrainTile as the child of the root element. The single Definition-type element allows populated descriptions of objects to exist outside their models. This is useful if it is not practical to otherwise maintain a set of metadata in a particular model, especially if the model is not under the control of the user. The Definition-type files can be maintained with a full set of data but act as pointers to the actual models using the X3DModelArchive element with a fully qualified alternateBaseURL attribute. It also facilitates piece-wise validation of the SMAL documents during construction.

a. StaticModelDefinition

The StaticModelDefinition is the SMAL equivalent to the Savage Object Metadata Template (SOMT). It holds inherent and parametric data, but does not require instantaneous data beyond its initialization data. Position and orientation are established on initialization and do not get updated again unless there is some physics-base action performed on it. Because a simulator agent does not drive it, its visual model can be placed under a regular Transform or GeographicTransform in the scene.



ObjectDefinitionType
extends: FullyExtensibleSMALElementType
FullyExtensibleSMALElementType
base: FullyExtensibleSMALElementType
Classification
IdentificationParameters
1..∞
ObjectModel
1..∞
PhysicalParameters
StaticModelDefinitionType
extends: ObjectDefinitionType
ObjectDefinitionType
base: ObjectDefinitionType
LocationOrientation
@ staticModelIdentifier

Figure 37. The StaticModelDefinition type and its super-type the ObjectDefinition type showing the largely inherent and instantaneous data they contain.

b. EntityDefinition

The EntityDefinition is the SMAL equivalent to the Savage Vehicle Metadata Template (SVMT). This holds all the same data as the StaticModelDefinition plus additional data to describe its DIS network identification, any inherent and instantaneous data associated with movement, and a set of BehaviorParameters. Because simulator agent will drive it, its visual model will be placed under an ESPDU transform. When paired with Viskit in the AT/FP Toolkit, the SimulationAgent element holds the fully qualified URL of the XML behavior file. Additional elements are available under SimulationAgent to hold name-value pairs matched to any extra parameter values required by Viskit to run the agent.

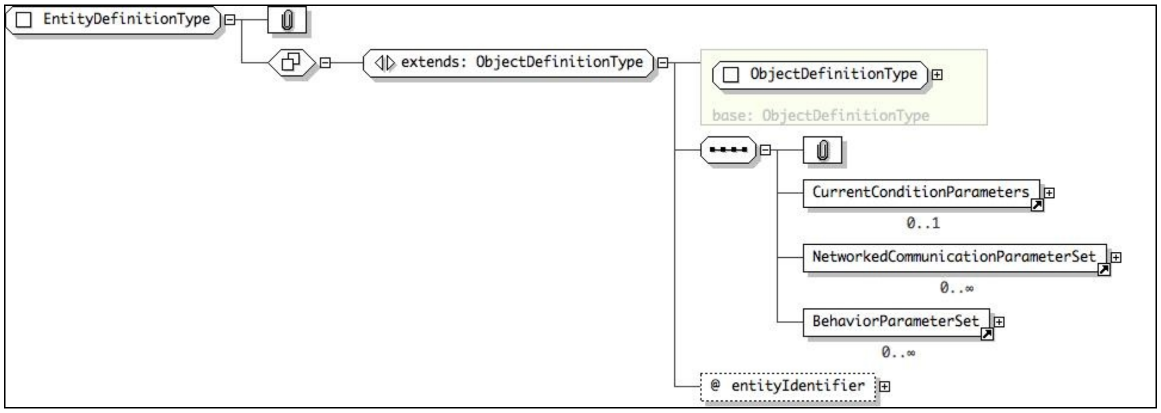

EntityDefinitionType
extends: ObjectDefinitionType
ObjectDefinitionType
base: ObjectDefinitionType
CurrentConditionParameters
0..1
NetworkedCommunicationParameterSet
0..∞
BehaviorParameterSet
0..∞
@ entityIdentifier

Figure 38. The EntityDefinition type showing the addition of more parametric and instantaneous data than the StaticModelDefinition

c. TerrainTile

The TerrainTile is the SMAL equivalent to the Savage Terrain Metadata Template (STMT). The TerrainTile expresses the size and vertical extent of the terrain described in the visual model and contains the fully qualified URLs of any graphic overlays that are associated with it. The overlay elements contain positioning data to feed a texture node in X3D, which allows the graphic to be placed on the terrain model if desired.

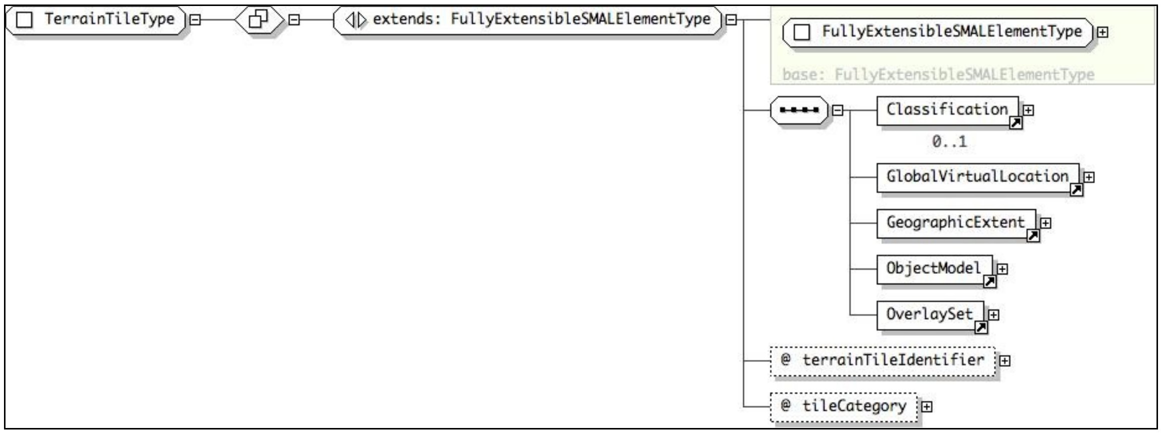


Figure 39. The TerrainTile type showing the locator, size, location, and image file elements used to establish terrain and texture.

G. SUMMARY

The working pair of Savage Metadata and SMAL are designed to hold the data necessary to initialize and then capture the visualization data generated by a simulation. SMAL collects the data found in the various SVMTs, SOMTs, and STMTs that are assembled to construct a 3D virtual scene, accepts the additional data stipulated by the user for the scenario construction, and establishes the baseline for the Interface and Logic components of the application. By updating the instantaneous data in the SMAL file over the course of a simulation, SMAL can also be used as a capture document to save information about intermediate or final outcomes of a simulation run to be used as the basis for analysis and additional study.

Using a mix of commercial off-the-shelf software and free, open-source tools on a Macintosh operating system, the visualization grammar of SMAL was created. It was designed using the AT/FP Toolkit as a use-case example, in order to test the viability of the grammar in a real-world application. The schema was designed from the ground up to be extensible and adaptable for use in a wide range of simulation tools, as well as eventually for use in planning tools and C2 suites. It splits scene construction along use-cases and temporal access.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. STANDARDIZATION IN SUPPORT OF AUTOGENERATION

A. INTRODUCTION

The major consideration in procedural construction of 3D scenes is that there can be no creativity in the act of creation. Building a visualization based on a set of rules requires that the rules are highly explicit and the initial conditions are exact. This means there must be a strict standardization of content in size, orientation, and origin. Alternately, if procedural creation is secondary to the act of scenario development, tools must be developed which makes scene construction an intuitive procedure. Savage Studio is such a tool.

B. INPUTS

Autogenerated visualizations require a strong content standardization scheme to make consistent visual sense. The obvious standardizations of scale and material appearance are only part of the problem space. Placing objects correctly on a procedural basis is central to automatically creating a 3D scene, so object model development must therefore include strict orientation and placement standards. Considerations for procedural object placement can be broken into two categories; global positioning and local positioning. The current Savage standard specifies appropriate origin location on models, with the primary longitudinal axis of the vehicle aligned with the X-axis. This standard is not quite sufficient because, for proper placement in a scene, consistent relative location of the object's origin point with respect to some reference plane such as the ground or water surface must be known.[26]

It can be argued that using the center of gravity or the centroid of the object as the origin point is appropriate because it makes physical sense when applying equations of motion or collision detection to the object. Knowing the position of the center of gravity of an object has some bearing on visualization when motion must be resolved around it (as in ships afloat or aircraft and munitions in flight) but its relative importance declines from there. In these cases, it is not so much the visual location of the center of gravity (COG) that is important, but its location with respect to the geometry of the model. Simple, efficient matrix transform calculations can be used to obtain the COG's reference

[26] X3D Scene Authoring Hints

frame from any point on the model as long as the relative position of the COG from that point is known. Thus, even if the standard reference point of a model is established for entirely visual effect, careful consideration in establishing the standard can allow for relatively uncomplicated translations when required to perform physical calculations and equally uncomplicated translations can be made for applying their results.

1. Global Placement Considerations

For global placement, there are three primary considerations: geographic location, vertical displacement, and 3D orientation. For airborne or submerged vehicles, depiction of geographic location has the greater fidelity requirement since vertical displacement is often explicitly laid out as altitude or depth, and 3D orientation can be generalized from the knowledge that the object is climbing or descending if it is not explicitly available in the form of pitch, roll, and yaw. For zero-altitude or ground placement, elevation is not often explicitly available as a data point, but must instead be resolved by associating geographic location with terrain data, which makes fidelity in the depiction of vertical displacement a more stringent requirement. A more difficult problem is ascertaining the proper object orientation based on the terrain, since each vehicle has a specific number and position of ground contact points forming its ground contact plane, which must then be compared to the overall slope of the terrain with which it comes in contact. Poor correlation of the object's elevation and orientation with the terrain results in objects that appear to be floating or buried in the ground (or half-sunk for sea-going vessels). Procedural placement adds the additional problem of accurately and explicitly defining the geometry of the object and the terrain in such a way that the vertical location and orientation of the placed object can be properly calculated. One solution has been created in the Savage Model Archive: a prototype node that reads the terrain height and surface normal values of the point on a terrain model and adjusts the height and orientation of an object model placed at that point. This is a procedural solution native to X3D, and not a panacea for the problem, but it is an example of how the problem can be addressed.[27]

An intuitive standard must be applied which requires a minimum of additional data not easily gleaned from the physical parameters of the model to relieve some of the problems inherent in procedural placement: model geometry must be placed such that its

[27] Hittner, 2003

local origin—the (0, 0, 0) point—occurs at the object's point of support and rotation. For example, a ship model must have its origin located on the navigational draft waterline at the ship's minimum steerage pivot point. An aircraft model's origin point must be at its center of gravity if it is depicted with its landing gear up. A land vehicle (or a landed aircraft depicted with its landing gear down) must have its origin located on the center-point of the plane defined by the ground-contact points of its wheels or tracks (for wheels it must be on the longitudinal center-line exactly halfway between the front-most and rear-most axles, for tracks it is on the longitudinal center-line at the halfway point of the track contact patch). By the Human Animation (H-Anim) Specification, a human model must have its origin at its sacrum.[28] However, for procedural placement, this requires that the computer must know the distance between the figure's feet and sacrum to place it properly in the scene.

Once an appropriate origin has been established, the relative position of various points of interest for physics calculation—such as the center of gravity, center of buoyancy, or aerodynamic center—can be included as metadata in the model. The majority of physical geometry information required for game-level physics can be calculated from those simple points.

2. Local Placement Considerations

Some objects will require placement based on the location of other objects in a scene. For example, an aircraft entity may have to be collocated with a carrier entity until it is launched as part of a scenario. Just prior to the aircraft launch, its 3D representation must be placed at a specific location on the flight deck at the end of the catapult gear before it begins its flight. In another application, a removable, crew-served weapon may have to be placed at a specific hard-point aboard a given ship as part of the development of a force-protection planning scenario. In both of these situations, it is necessary to define a specific location to place an object based on the parent object's origin. By pre-defining locations of interest based on the origin of the parent object, it is relatively simple to place the child objects, as long as there is some method of standardizing the location of the origin of the child objects as well. For objects that are designed to be component parts of vehicles, such as weapons and sensors, the point of attachment (by

[28] X3D H-Anim Specification

including in the model any typically associated mounting hardware) is an appropriate origin. At the same time, pre-defined locations of interest on the parent object must be defined as attachment points. For purposes of report generation or plain-language description, these points must be appropriately documented in the models themselves, either by explicitly naming them either in the metadata or in the 3D modeling language's node identification scheme.

3. Savage Studio Authoring Tool

Savage Studio is a drag-and-drop scenario-authoring tool developed by Yumetech, Inc. in partnership with NPS. Savage Studio is designed around the SMAL metadata model, which leverages the metadata found in the Savage archive in conjunction with user input to populate a SMAL simulation metadata file. The file is then used as the initialization document for the AT/FP Toolkit.

The Savage Model Archive is read by the catalog creator tool, which reads the X3D files and translates any Savage Metadata into snippets of SMAL code and places it in the catalog file. Savage Studio in turn reads the catalog file and creates customized menus of available terrain and vehicle models from which the user may choose.

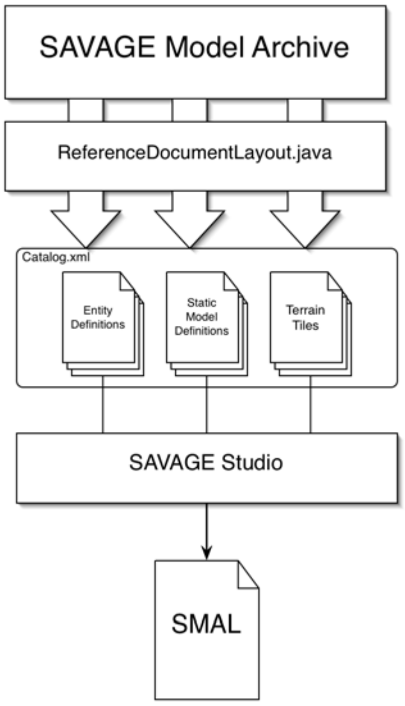
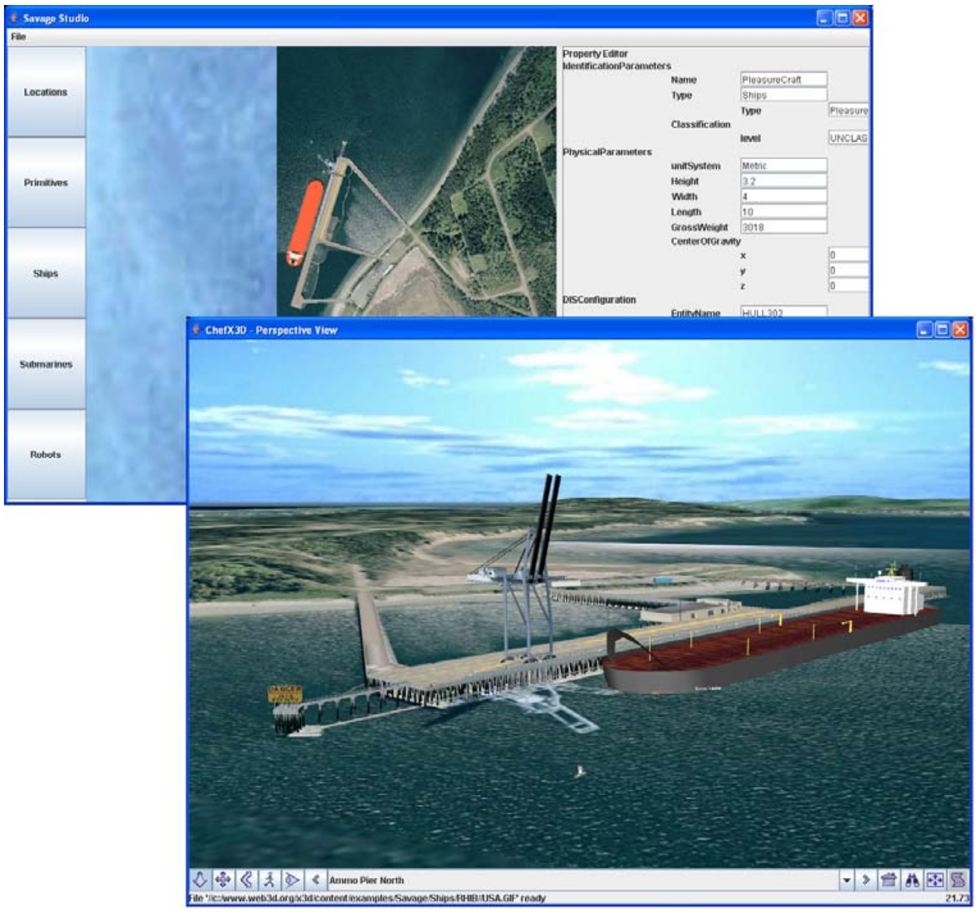


Figure 40. The flow of metadata from the Savage Model Archive to a SMAL file through Savage Studio.

When terrain is selected, the Savage metadata is used to populate a TerrainTile element while a 2D overhead view appears in a panel on-screen. The user is permitted to make any adjustments to the exposed metadata. When a model is selected the Savage metadata is used to populate an EntityDefinition or StaticModelDefinition. The user may then use the drag-and-drop functionality to place it on the terrain and orient it as desired. If the model is SMAL-enabled, the tool exposes the Savage Metadata to allow the user to change parameters if desired.



Property Editor
IdentificationParameters
Name PleasureCraft
Type Ships
Type Pleasure
Classification
level UNCLAS
PhysicalParameters
unitSystem Metric
Height 3.2
Width 4
Length 10
GrossWeight 3018
CenterOfGravity
x 0
y 0
z 0
DISConfiguration
EntityName HULL302

File '//c:/www.web3d.org/x3d/content/examples/Savage/Ships/RHIB/USA.GIF' ready 21.73

Figure 41. Screen captures of Savage Studio showing the model categories on the buttons at left, exposed SMAL metadata on the right, a supertanker positioned on the overhead view, and the resulting 3D scene in the foreground.

At this point, the user may select a Viskit behavior for the model from a list of event-graph model behaviors, which would cause the tool to create an EntityDefinition, instead of a StaticModelDefinition, so that a simulation agent can drive it. When the scene is constructed, the user moves on to the assembly of the entities into a coherent scenario using Viskit. Savage Studio saves all the TerrainTiles, StaticModelDefinitions, and EntityDefinitions into a working SMAL document and generates individual IDs for them. When the user has assembled the scenario in Viskit, a reference to the assembly's XML file is also included in the SMAL file.

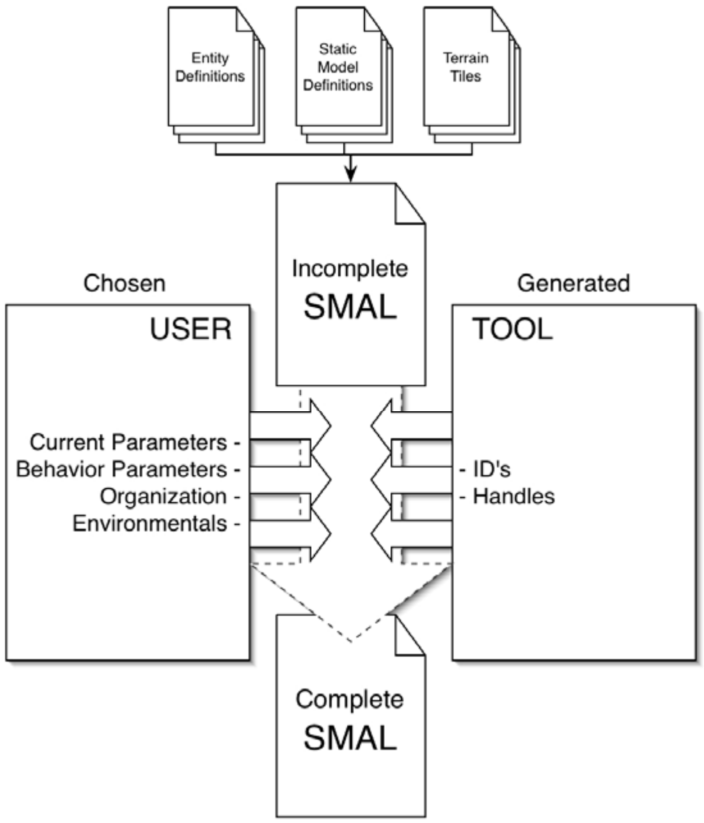


Figure 42. Once the Terrain, Static Models, and Entities have been collected and positioned in the scene, other parameters are added to complete the scenario definition, some chosen by the user, and others generated automatically by the tool.

C. SUMMARY

A strict content standardization scheme supports the goal of autogeneration by removing unacceptable variations from the methods of placement and orientation of models. Where variations occur or can't be avoided, it must be codified in the metadata so the application has some means to adjust its procedural actions to compensate for the irregularity. Parallel to complete autogeneration is the partial autogeneration of scenes possible with the use of scene authoring tools like Savage Studio. Savage Studio brings with the scene construction the ability to continue and develop an entire scenario, which is still out of the reach of full autogeneration.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

Several different projects are expected to benefit from SMAL. Multiple conclusions and recommendations for future work follow.

1. Well-Defined Metadata Structures the Relationships within an Application, and Data Improves Its Value

Interoperability is a feature defined at many levels in an application, but a consistent design philosophy can satisfy it across all levels. At the highest level is the idea that the data (collected, processed, or both) are the important part of an application. Creating a well-defined and task-oriented division of labor among the components of an application will facilitate interoperability by forcing the designer to closely specify its interfaces rather than its processes. Once the interface is specified, the exact processes of the components are immaterial, as long as each one produces accurate results. The data transport architecture between the components is therefore the most important part because it is the most persistent. Components can be replaced so long as their interface matches the rest of the application, but the interface, once set, is far more permanent. So it is the data, and the way components relate the data with other components, that enable interoperability and resist obsolescence.

2. "Non-Lossy" Equivalence Through Strong Typing Supports Data Sharing

A data standard that is able to be translated to another format and then back into its original format with no loss of data has the underpinnings necessary to support interoperability. It requires strong typing and good documentation on both sides of the translation to be useful, since precision lost by a poor translation scheme cannot be regained. If precision is lost, then data can become useless to an application, which discourages data sharing and collaborative analysis.

3. Interoperability Benefits from Limited Data Standardization

The many-to-one-to-many concept of standards establishment is not a new one, but it is not easily accomplished, since competing priorities of its "customer" standards can cause it to bloat with specialized data, making differing implementations all but unusable. Discouraging specialization is not the answer, but consensus-building among interested parties in a data standard can reduce perceived requirements for superfluous data and encourage the adoption of more universal data sets.

4. Interoperability Through Well-Defined Vocabularies

Non-standard data that are well defined and contextualized are not lost to the general community's interests, but they require conversion or translation to become useful. "Non-Lossy" translation is only possible when the data and their sources are well-understood. Once the data is understood, and translation is possible, interoperability can be achieved, if necessary, through multiple chained translations.

B. RECOMMENDATIONS FOR FUTURE WORK

1. Improved Metadata Storage and Access Methods for Models

In determining the set of data appropriate to include in a model, it became apparent that there is always more data that can be included, but only at the expense of bloating the metadata-enabled model. A more compact and extensible method of data-inclusion is to link rather than include data. Instead of storing data directly in 3D models, it is possible to use links to databases that hold the pertinent data, essentially using a URI to locate the database and a set of keywords or pointers to get at the necessary data. The URI can point to either local databases, or web-based information services. This type of data linkage creates a highly customizable set of metadata, limited only by the richness and depth of the database itself. It also leaves the database maintenance to the owners of the data, rather than the customers.

2. Develop Stylesheets to Directly Translate Between SMAL and X3D

The current method of translating Savage Metadata Template data to SMAL data is a work-around, and it has no return translation ability. Direct translation is a superior method, which will encourage wider use and make it easier to enable Savage Models with SMAL Metadata. The Savage Studio Tool currently under development will need to include this ability.

3. SMAL-Enable the Entire Savage and SavageDefense Catalogs

The current set of SMAL-Enabled models is limited, but easily expandable. The process, however, is time consuming if the model is not already annotated in some way to describe it. Easily recognizable objects can be researched to determine their parameters, but the more inscrutable models required direct measurement to establish their parameters. A set of virtual measurement tools could be developed to facilitate this process.

4. Explore the Use of SMAL Snippets Over XML Tactical Chat Channels or as DIS-XML Payloads

XML grammars share a common trait that may prove useful in future communications modes, namely the fact that they are all XML. The XML Tactical Chat channel can be leveraged to send SMAL snippets to the other participants in a chat session to update their catalogs. Similarly, DIS-XML packets can carry a SMAL snippet with it as a payload, which in turn can be used to initialize a scene.

5. Create XML-based DIS Enumerations for Use With SMAL

The current method of DIS entity identification uses a cryptic series of numbers to specify the type of object the entity refers to. Using processing instructions and the <xs:enumeration> method, it is possible to create a set of "xs:NMTOKEN" literal enumerations so that the entity can be categorized without using a reference document. These enumerations may be suitable for further use within SMAL.

6. Determine the Feasibility/Utility of Mapping SMAL to JC3IEDM

There is some overlap in functionality between SMAL and JC3IEDM, which may be leveraged to create direct visualizations of battlefield command and control, either through double translation from BML or direct translation on a network, which uses a JC3IEDM sub-language.

7. Determine the Feasibility/Utility of Mapping SMAL to TAML

There is also some overlap in functionality between SMAL and TAML. It may be possible to use TAML documents to create a submarine track history, which can be visualized through translation to SMAL

8. Multiple Namespace Support

As these various data languages are used and integrated, support for XML namespaces[29] is needed. This approach will allow SMAL documents to embed XML elements and attributes from seperately defined schema for DIS-XML, JC3IEDM, TAML, BML, X3D, etc. SMAL 2.0 needs to be able to compose and correlate these diverse and complementary standards, avoiding the redefinition of terms.

[29] Namespaces in XML, 2004

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A.   SMAL SCHEMA SOURCE CODE

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with oXygen v7.1 (http://www.oXygenxml.com) by Travis Rauch (Naval Postgraduate Schoo) -->
<!-- edited with XMLSpy v2006 sp1 U (http://www.altova.com) by Don Brutzman (Naval Postgraduate School) -->
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" version="1.0"
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <!-- SavageModelingAnalysisLanguage1.0.xsd -->
        <!-- version:  1.0 -->
        <!-- revised:  26 March 2006 -->
        <!-- authors:  Travis Rauch and Don Brutzman -->
        <xs:annotation>
                <xs:appinfo>This language is intended to provide a lightweight, human- and machine-readable collection of data which
can be used to create or recreate a 3D scene using SAVAGE library elements and any event-driven source of data.</xs:appinfo>
        </xs:annotation>

        <xs:include schemaLocation="SavageModelingAnalysisLanguageEnumerations1.0.xsd">
                <xs:annotation>
                        <xs:appinfo>Contains all simpleType restrictions by enumeration for the attributes in SMAL.</xs:appinfo>
                </xs:annotation>
        </xs:include>
        <xs:include schemaLocation="SavageModelingAnalysisLanguageDataTypes1.0.xsd">
                <xs:annotation>
                        <xs:appinfo>Contains all simpleType restrictions by regular expression or bound limits for the attributes in
SMAL.</xs:appinfo>
                </xs:annotation>
        </xs:include>

        <xs:complexType abstract="true" name="FullyExtensibleSMALElementType">
                <xs:annotation>
                        <xs:appinfo>Super-type that includes the 'any' and 'anyAttribute' tags for full extensibility of elements of this
type.</xs:appinfo>
                </xs:annotation>
                <xs:sequence>
                        <xs:any maxOccurs="unbounded" minOccurs="0" namespace="##other" processContents="lax"/>
                </xs:sequence>
                <xs:anyAttribute namespace="##other" processContents="lax"/>
        </xs:complexType>
        <xs:complexType abstract="true" name="ElementExtensibleSMALElementType">
                <xs:annotation>
                        <xs:appinfo>Super-type that includes only the 'any' tag for element-wise extensibility of elements of this
type.</xs:appinfo>
                </xs:annotation>
                <xs:sequence>
                        <xs:any maxOccurs="unbounded" minOccurs="0" namespace="##other" processContents="lax"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType abstract="true" name="AttributeExtensibleSMALElementType">
                <xs:annotation>
                        <xs:appinfo>Super-type that includes only the 'anyAttribute' tag for attribute-wise extensibility of elements of this
type.</xs:appinfo>
                </xs:annotation>
                <xs:anyAttribute namespace="##other" processContents="lax"/>
        </xs:complexType>

        <xs:element name="head">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element minOccurs="0" ref="Classification"/>
                                <xs:element maxOccurs="unbounded" minOccurs="0" ref="meta"/>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
        <xs:element name="meta">
                <xs:annotation>
                        <xs:documentation source="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"/>
                </xs:annotation>
```

```
		<xs:complexType mixed="false">
			<xs:attribute name="name" type="xs:string" use="required"/>
			<xs:attribute name="content" type="xs:string" use="required"/>
			<xs:attribute name="http-equiv" type="xs:string"/>
			<xs:attribute name="lang" type="xs:string"/>
		</xs:complexType>
	</xs:element>

	<!-- ******************************************** -->
	<!-- SMAL -->
	<!-- ******************************************** -->
	<xs:element name="SMAL">
		<xs:annotation>
			<xs:appinfo>SAVAGE Model Analysis Language: collects data necessary to create a 3D scene out of disparate
elements</xs:appinfo>
		</xs:annotation>
		<xs:complexType>
			<xs:complexContent>
				<xs:extension base="FullyExtensibleSMALElementType">
					<xs:sequence>
						<xs:element minOccurs="0" ref="head"/>
						<xs:choice>
							<xs:annotation>
								<xs:appinfo>A SMAL file will contain either a full-fledged Simulation or a
partial/default metadata corresponding to a simulation component file such as an entity, static model, or terrain tile.</xs:appinfo>
							</xs:annotation>
							<xs:element maxOccurs="unbounded" ref="Simulation"/>
							<xs:element maxOccurs="1" minOccurs="1" ref="EntityDefinition"/>
							<xs:element maxOccurs="1" minOccurs="1" ref="TerrainTile"/>
							<xs:element maxOccurs="1" minOccurs="1" ref="StaticModelDefinition"/>
						</xs:choice>
					</xs:sequence>
					<xs:attribute fixed="1.0" name="version" type="xs:string"/>
				</xs:extension>
			</xs:complexContent>
		</xs:complexType>
	</xs:element>

	<!-- ******************************************** -->
	<!-- SMAL		-->
	<!--   Simulation -->
	<!-- ******************************************** -->
	<xs:element name="Simulation">
		<xs:annotation>
			<xs:appinfo>Top level tag for a single simulation. Allows multiple simulations to be run separately, but at the same
time under the root SMAL tag.</xs:appinfo>
		</xs:annotation>
		<xs:complexType>
			<xs:complexContent>
				<xs:extension base="FullyExtensibleSMALElementType">
					<xs:sequence>
						<xs:element maxOccurs="1" minOccurs="0" ref="Classification"/>
						<xs:element maxOccurs="1" minOccurs="1" ref="NetworkChannel"/>
						<xs:element maxOccurs="1" minOccurs="1" ref="World"/>
						<xs:element maxOccurs="unbounded" minOccurs="1" ref="EntitySet"/>
						<xs:element maxOccurs="unbounded" minOccurs="0" ref="EnvironmentalConditionSet"/>
						<xs:element maxOccurs="unbounded" minOccurs="0" ref="AssociationSet"/>
					</xs:sequence>
					<xs:attribute name="simulationIdentifier" type="xs:ID" use="optional"/>
				</xs:extension>
			</xs:complexContent>
		</xs:complexType>
	</xs:element>

	<xs:element name="NetworkChannel">
		<xs:annotation>
			<xs:appinfo>Contains all data necessary to define the multicast port address, simulation, and local computer ID. A
single Simulation has only one network channel.</xs:appinfo>
			<xs:documentation source="http://www.web3d.org/x3d/specifications/ISO-IEC-19775-FDIS-
X3dAbstractSpecification/Part01/components/dis.html">See X3D Specification Part I 28.2.3</xs:documentation>
```

```
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="FullyExtensibleSMALElementType">
                    <xs:sequence>
                        <xs:element minOccurs="0" ref="Classification"/>
                    </xs:sequence>
                    <xs:attribute default="true" name="enabled" type="xs:boolean" use="optional"/>
                    <xs:attribute default="127.0.0.1" name="address" type="ipAddressPattern" use="optional"/>
                    <xs:attribute default="62040" name="port" type="xs:nonNegativeInteger" use="optional"/>
                    <xs:attribute default="127.0.0.1" name="multicastRelayHost" type="ipAddressPattern" use="optional"/>
                    <xs:attribute default="62040" name="multicastRelayPort" type="xs:nonNegativeInteger"
use="optional"/>
                    <xs:attribute default="false" name="rtpHeaderExpected" type="xs:boolean"/>
                    <xs:attribute name="siteID" type="xs:nonNegativeInteger" use="optional"/>
                    <xs:attribute name="applicationID" type="xs:nonNegativeInteger" use="optional"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>

    <xs:element name="World">
        <xs:annotation>
            <xs:appinfo>Defines the largely static environment in which the simulation will run. It will contain all terrain
features (terrain and bathymetry), associated charts or maps, and any buildings or other significant model data which will not be
tracked for movement or other actions.</xs:appinfo>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="FullyExtensibleSMALElementType">
                    <xs:sequence>
                        <xs:element maxOccurs="1" minOccurs="0" ref="GeoOrigin"/>
                        <xs:element maxOccurs="1" minOccurs="1" ref="TerrainTileSet"/>
                        <xs:element maxOccurs="unbounded" minOccurs="0" ref="StaticModelSet"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>

    <xs:element name="GeoOrigin">
        <xs:annotation>
            <xs:appinfo>The GeoOrigin node defines an absolute geographic location and an implicit local coordinate frame
against which geometry is referenced. This node is used to translate from geographical coordinates into a local Cartesian coordinate
system which can be managed by the browser.</xs:appinfo>
            <xs:documentation source="http://www.web3d.org/x3d/specifications/ISO-IEC-19775-FDIS-
X3dAbstractSpecification/Part01/components/geodata.html">See X3D specification Part I 25.3.6</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="FullyExtensibleSMALElementType">
                    <xs:sequence>
                        <xs:element minOccurs="0" ref="Classification"/>
                    </xs:sequence>
                    <xs:attribute default="N00 0.0 W00 0.0" name="geoCoords" type="xs:string" use="optional"/>
                    <xs:attribute default="'GD', 'WE'" name="geoSystem" type="xs:string" use="optional">
                        <xs:annotation>
                            <xs:appinfo>Two enumerations, one for spatial reference frames, the other for earth ellipsoid
i.e. ["GD", "WE"] See X3D specification 25.2.3.</xs:appinfo>
                        </xs:annotation>
                    </xs:attribute>
                    <xs:attribute default="true" name="rotateYUp" type="xs:boolean" use="optional"/>
                    <xs:attribute name="geoOriginIdentifier" type="xs:ID" use="required"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>

    <xs:element name="TerrainTileSet">
        <xs:annotation>
```

```
			<xs:appinfo>Contains the set of all Topography and Bathymetry models and their associated chart/imagery
overlays.</xs:appinfo>
		</xs:annotation>
		<xs:complexType>
			<xs:complexContent>
				<xs:extension base="ElementExtensibleSMALElementType">
					<xs:sequence>
						<xs:element maxOccurs="unbounded" ref="TerrainTile"/>
					</xs:sequence>
				</xs:extension>
			</xs:complexContent>
		</xs:complexType>
	</xs:element>

	<!-- ******************************************** -->
	<!-- SMAL		-->
	<!--	TerrainTile -->
	<!-- ******************************************** -->
	<xs:element name="TerrainTile">
		<xs:annotation>
			<xs:appinfo>A single terrain model containing size, vertical extent information, and references to the charts, maps,
and/or image overlays associated with it.</xs:appinfo>
		</xs:annotation>
		<xs:complexType>
			<xs:complexContent>
				<xs:extension base="FullyExtensibleSMALElementType">
					<xs:sequence>
						<xs:element maxOccurs="1" minOccurs="0" ref="Classification"/>
						<xs:element minOccurs="1" maxOccurs="1" ref="GeoOrigin"/>
						<xs:element maxOccurs="1" minOccurs="1" ref="GeographicExtent"/>
						<xs:element maxOccurs="1" minOccurs="0" ref="OverlaySet"/>
						<xs:element maxOccurs="1" minOccurs="1" ref="ObjectModel"/>
						<xs:element maxOccurs="1" minOccurs="0" ref="Location"/>
					</xs:sequence>
					<xs:attribute name="terrainTileIdentifier" type="xs:ID" use="optional"/>
					<xs:attribute default="landTerrain" name="tileCategory" type="tileCategoryEnumeration"/>
				</xs:extension>
			</xs:complexContent>
		</xs:complexType>
	</xs:element>

	<xs:complexType name="BoundingPolygonType">
		<xs:annotation>
			<xs:appinfo>A base complexType used to describe a two-and-a-half dimensional shape for the purposes of
establishing boundaries.  Description of the shape is either by defining the points of a polygon or by establishing a centerpoint and
radius for a circular boundary.</xs:appinfo>
		</xs:annotation>
		<xs:complexContent>
			<xs:extension base="FullyExtensibleSMALElementType">
				<xs:choice>
					<xs:annotation/>
					<xs:element maxOccurs="1" ref="PointRadiusCircular"/>
					<xs:element maxOccurs="unbounded" minOccurs="3" ref="MapCoordinate"/>
				</xs:choice>
				<xs:attribute ref="unitSystem"/>
				<xs:attribute default="0.0" name="area" type="xs:float" use="optional"/>
				<xs:attribute default="0.0" name="verticalExtent" type="xs:float" use="optional"/>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:element name="GeographicExtent" type="BoundingPolygonType">
		<xs:annotation>
			<xs:appinfo>A BoundingPolygonType element used to describe the boundaries of a geographic element such as a
TerrainTile.</xs:appinfo>
		</xs:annotation>
	</xs:element>

	<xs:element name="PointRadiusCircular">
		<xs:annotation>
			<xs:appinfo>A shape descriptor used to specify circular shapes in BoundingPolygonType elements.</xs:appinfo>
```

```
                </xs:annotation>
                <xs:complexType>
                        <xs:complexContent>
                                <xs:extension base="FullyExtensibleSMALElementType">
                                        <xs:sequence>
                                                <xs:element maxOccurs="1" minOccurs="1" ref="MapCoordinate"/>
                                        </xs:sequence>
                                        <xs:attribute ref="unitSystem"/>
                                        <xs:attribute name="radius" type="nonNegativeFloat" use="required"/>
                                </xs:extension>
                        </xs:complexContent>
                </xs:complexType>
        </xs:element>

        <xs:element name="OverlaySet">
                <xs:annotation>
                        <xs:appinfo>Collects all imagery overlays associated with a TerrainTile.</xs:appinfo>
                </xs:annotation>
                <xs:complexType>
                        <xs:complexContent>
                                <xs:extension base="FullyExtensibleSMALElementType">
                                        <xs:sequence>
                                                <xs:element maxOccurs="1" minOccurs="0" ref="Classification"/>
                                                <xs:element maxOccurs="unbounded" minOccurs="0" ref="OverlayImageDescriptor"/>
                                        </xs:sequence>
                                </xs:extension>
                        </xs:complexContent>
                </xs:complexType>
        </xs:element>
        <xs:complexType name="OverlayImageDescriptorType">
                <xs:annotation>
                        <xs:appinfo>Super-type used to describe an image overlay for a TerrainTile.  The image is referenced by a URI, and
its geographic location and extent is described using centerpoint and bounding latitudes/longitudes.</xs:appinfo>
                </xs:annotation>
                <xs:complexContent>
                        <xs:extension base="FullyExtensibleSMALElementType">
                                <xs:attribute default="http://www.web3d.org/x3d/content/examples/" name="fileLocationURL"
type="xs:anyURI"/>
                                <xs:attribute default="N00 00.0" name="centerPointLatitude" type="latitudeOrdinatePattern" use="optional"/>
                                <xs:attribute default="W00 00.0" name="centerPointLongitude" type="longitudeOrdinatePattern"
use="optional"/>
                                <xs:attribute default="N00 00.0" name="northBoundLatitude" type="latitudeOrdinatePattern" use="optional"/>
                                <xs:attribute default="N00 00.0" name="southBoundLatitude" type="latitudeOrdinatePattern" use="optional"/>
                                <xs:attribute default="W00 00.0" name="westBoundLongitude" type="longitudeOrdinatePattern"
use="optional"/>
                                <xs:attribute default="W00 00.0" name="eastBoundLongitude" type="longitudeOrdinatePattern"
use="optional"/>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:element abstract="true" name="OverlayImageDescriptor">
                <xs:annotation>
                        <xs:appinfo>Abstract placeholder for specific overlay elements in an OverlaySet element.</xs:appinfo>
                </xs:annotation>
        </xs:element>
        <xs:element name="OverlaySetChart" substitutionGroup="OverlayImageDescriptor" type="OverlayImageDescriptorType">
                <xs:annotation>
                        <xs:appinfo>A chart image used as a TerrainTile overlay.</xs:appinfo>
                </xs:annotation>
        </xs:element>
        <xs:element name="OverlaySetImagery" substitutionGroup="OverlayImageDescriptor" type="OverlayImageDescriptorType">
                <xs:annotation>
                        <xs:appinfo>A satellite image or graphic used as a TerrainTile overlay.</xs:appinfo>
                </xs:annotation>
        </xs:element>
        <xs:element name="OverlaySetMap" substitutionGroup="OverlayImageDescriptor" type="OverlayImageDescriptorType">
                <xs:annotation>
                        <xs:appinfo>A map image used as a TerrainTile overlay.</xs:appinfo>
                </xs:annotation>
        </xs:element>
```

```
    <xs:element name="EnvironmentalConditionSet">
        <xs:annotation>
            <xs:appinfo>Holds information about changeable world conditions such as time of day and weather.</xs:appinfo>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="FullyExtensibleSMALElementType">
                    <xs:sequence>
                        <xs:element maxOccurs="1" minOccurs="1" ref="TimeParameters"/>
                        <xs:element maxOccurs="unbounded" minOccurs="1" ref="EnvironmentalCondition"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>

    <xs:element name="TimeParameters">
        <xs:annotation>
            <xs:appinfo>Holds temporal data about the scenario such as start time, time of year, time zone, and duration of the
simulation.</xs:appinfo>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="FullyExtensibleSMALElementType">
                    <xs:attribute ref="startTime" use="required"/>
                    <xs:attribute default="1" name="day" type="dayType" use="optional"/>
                    <xs:attribute default="1" name="month" type="monthType" use="optional"/>
                    <xs:attribute default="-7" name="timeZone" type="xs:int" use="optional"/>
                    <xs:attribute ref="duration" use="optional"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>

    <xs:element name="EnvironmentalCondition">
        <xs:annotation>
            <xs:appinfo>Contains a list of weather conditions such as cloud layers, temperature, precipitation, winds, and water
currents, as well as the start time and duration of this set of conditions. Multiple sets of conditions can be established to create
complex weather patterns.</xs:appinfo>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="FullyExtensibleSMALElementType">
                    <xs:sequence>
                        <xs:element maxOccurs="1" minOccurs="1" ref="Background"/>
                        <xs:element minOccurs="0" ref="TemperatureConditionSet"/>
                        <xs:element minOccurs="0" ref="WindConditionSet"/>
                        <xs:element minOccurs="0" ref="CloudConditionSet"/>
                        <xs:element minOccurs="0" ref="PrecipitationConditionSet"/>
                        <xs:element minOccurs="0" ref="WaterConditionSet"/>
                    </xs:sequence>
                    <xs:attribute name="environmentalConditionIdentifier" type="xs:ID"/>
                    <xs:attribute name="startTime" type="xs:time" use="required"/>
                    <xs:attribute ref="duration" use="required"/>
                    <xs:attribute ref="unitSystem" use="optional"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>

    <xs:element name="Background">
        <xs:annotation>
            <xs:appinfo>Holds information for filling the X3D Background node which is used to set up a skybox or simple
background colors.</xs:appinfo>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="FullyExtensibleSMALElementType">
                    <xs:attribute name="skyColor" type="xs:string"/>
                    <xs:attribute name="groundColor" type="xs:string"/>
                    <xs:attribute name="skyAngle" type="xs:float"/>
                    <xs:attribute name="groundAngle" type="xs:float"/>
                    <xs:attribute name="frontURL" type="xs:anyURI"/>
                    <xs:attribute name="backURL" type="xs:anyURI"/>
                    <xs:attribute name="leftURL" type="xs:anyURI"/>
                    <xs:attribute name="rightURL" type="xs:anyURI"/>
                    <xs:attribute name="topURL" type="xs:anyURI"/>
                    <xs:attribute name="bottomURL" type="xs:anyURI"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>

    <xs:element name="TemperatureConditionSet">
        <xs:annotation>
            <xs:appinfo>Air and water temperatures can be established based on altitude and depth. Includes a duration attribute
to allow for changes over time.</xs:appinfo>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="FullyExtensibleSMALElementType">
                    <xs:sequence>
                        <xs:element maxOccurs="unbounded" minOccurs="1" ref="TemperatureCondition"/>
                    </xs:sequence>
                    <xs:attribute ref="startTime"/>
                    <xs:attribute ref="duration"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="TemperatureConditionType">
        <xs:annotation>
            <xs:appinfo>Super-type describing the base attributes required for describing a temperature profile.</xs:appinfo>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="AttributeExtensibleSMALElementType">
                <xs:attribute default="15" name="degrees" type="xs:float"/>
                <xs:attribute default="false" name="isothermal" type="xs:boolean"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:element abstract="true" name="TemperatureCondition">
        <xs:annotation>
            <xs:appinfo>Abstract placeholder for specific temperature elements in a TemperatureConditionSet
element.</xs:appinfo>
        </xs:annotation>
    </xs:element>

    <xs:element name="AirTemperatureCondition" substitutionGroup="TemperatureCondition">
        <xs:annotation>
            <xs:appinfo>Holds the air temperature value and the base altitude at which this temperature begins. Temperature is
assumed to decrease as altitude increases from the base altitude unless isothermal is set to true.</xs:appinfo>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="TemperatureConditionType">
                    <xs:attribute default="0" name="altitude" type="nonNegativeFloat"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>

    <xs:element name="WaterTemperatureCondition" substitutionGroup="TemperatureCondition">
        <xs:annotation>
            <xs:appinfo>Holds the water temperature and the depth at which this temperature occurs. Temperature is assumed to
decrease as depth increases unless isothermal is set to true.</xs:appinfo>
        </xs:annotation>
        <xs:complexType>
```

```
            <xs:complexContent>
                <xs:extension base="TemperatureConditionType">
                    <xs:attribute default="0" name="depth" type="nonNegativeFloat"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>

    <xs:element name="WindConditionSet">
        <xs:complexType>
            <xs:annotation>
                <xs:appinfo>Collects all wind condition elements for a single EnvironmentalCondition element.</xs:appinfo>
            </xs:annotation>
            <xs:complexContent>
                <xs:extension base="FullyExtensibleSMALElementType">
                    <xs:sequence>
                        <xs:element maxOccurs="unbounded" minOccurs="0" ref="WindConditionAtLevel"/>
                    </xs:sequence>
                    <xs:attribute ref="startTime"/>
                    <xs:attribute ref="duration"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="SpeedDirectionType">
        <xs:annotation>
            <xs:appinfo>Super-type containing the base attributes for elements which describe the speed and direction of an
object or phenomena using attribute values.</xs:appinfo>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="FullyExtensibleSMALElementType">
                <xs:attribute name="speed" type="nonNegativeFloat" use="required"/>
                <xs:attribute name="direction" type="degreeType" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:element name="WindConditionAtLevel">
        <xs:annotation>
            <xs:appinfo>Winds are defined by altitude, speed, maximum gust speed, and direction. The value for altitude
represents the lowest altitude at which this condition exists.  Wind conditions at a given altitude are considered persistent over the
course of an EnvironmentCondition.  Changes are made using multiple WindConditionSets.</xs:appinfo>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="SpeedDirectionType">
                    <xs:attribute name="altitude" type="nonNegativeFloat" use="required"/>
                    <xs:attribute default="0.0" name="gustSpeed" type="nonNegativeFloat" use="optional"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>

    <xs:element name="WaterCurrentConditionAtDepth">
        <xs:annotation>
            <xs:appinfo>Water currents are defined by depth, speed, and direction.  Additional data for salinity and visibility are
included.  The value for depth represents the shallowest depth at which this condition exists. Water currents, salinity, and visibility by
depth are considered persistent over the course of an EnvironmentalCondition.  Changes can be made by using multiple
WaterCurrentConditionSets.</xs:appinfo>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="SpeedDirectionType">
                    <xs:attribute name="depth" type="nonNegativeFloat" use="required"/>
                    <xs:attribute default="0.0" name="visibility" type="nonNegativeFloat" use="optional"/>
                    <xs:attribute default="0.0" name="salinity" type="nonNegativeFloat" use="optional"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
```

```
	</xs:element>

	<xs:element name="WaterConditionSet">
		<xs:annotation>
			<xs:appinfo>Collects the sea state and all water current condition elements for a single EnvironmentalCondition
element.  Currents may change over the course of a single EnvironmentCondition, so there is a duration attribute.</xs:appinfo>
		</xs:annotation>
		<xs:complexType>
			<xs:complexContent>
				<xs:extension base="FullyExtensibleSMALElementType">
					<xs:sequence>
						<xs:element maxOccurs="1" minOccurs="1" ref="SeaState"/>
						<xs:element maxOccurs="unbounded" minOccurs="1" ref="WaterCurrentConditionAtDepth"/>
					</xs:sequence>
					<xs:attribute ref="startTime"/>
					<xs:attribute ref="duration"/>
				</xs:extension>
			</xs:complexContent>
		</xs:complexType>
	</xs:element>

	<xs:element name="SeaState">
		<xs:annotation>
			<xs:appinfo>Describes water surface conditions to optionally include the Beaufort Scale equivalent condition for
winds and wave action, and/or specific wave conditions such as crest conditions and wave height, length, and period.  Sea state is
considered persistent and for a given EnvironmentCondition so there is no duration attribute.</xs:appinfo>
		</xs:annotation>
		<xs:complexType>
			<xs:complexContent>
				<xs:extension base="FullyExtensibleSMALElementType">
					<xs:attribute name="beaufortScaleEquivalent" type="xs:nonNegativeInteger" use="optional"/>
					<xs:attribute name="waveHeight" type="nonNegativeFloat" use="optional"/>
					<xs:attribute name="wavePeriod" type="nonNegativeFloat" use="optional"/>
					<xs:attribute name="waveLength" type="nonNegativeFloat" use="optional"/>
					<xs:attribute name="crest" type="xs:NMTOKEN" use="optional"/>
				</xs:extension>
			</xs:complexContent>
		</xs:complexType>
	</xs:element>

	<xs:element name="CloudConditionSet">
		<xs:annotation>
			<xs:appinfo>Collects the cloud condition elements for a single EnvironmentalCondition element.</xs:appinfo>
		</xs:annotation>
		<xs:complexType>
			<xs:complexContent>
				<xs:extension base="FullyExtensibleSMALElementType">
					<xs:sequence>
						<xs:element maxOccurs="unbounded" minOccurs="0" ref="CloudLayer"/>
					</xs:sequence>
					<xs:attribute ref="startTime"/>
					<xs:attribute ref="duration"/>
				</xs:extension>
			</xs:complexContent>
		</xs:complexType>
	</xs:element>

	<xs:element name="CloudLayer">
		<xs:annotation>
			<xs:appinfo>Clouds are defined in layers and can optionally be localized by using the LocationOrientation and
GeographicExtent elements. Altitude represents the cloud base for that layer and its depth as defined in the GeographicExtent element
extends upward.  Cloud conditions by altitude are considered persistent in a given Environmental condition.  Changes may be made
using multiple CloudConditionSets.</xs:appinfo>
		</xs:annotation>
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="LocationOrientation" minOccurs="0" maxOccurs="1"/>
				<xs:element ref="GeographicExtent" minOccurs="0" maxOccurs="1"/>
			</xs:sequence>
			<xs:attribute name="altitude" type="xs:float"/>
```

```
                    <xs:attribute default="CLR" name="coverage" type="cloudCoverageEnumeration"/>
                    <xs:attribute default="CUMULUS" name="cloudType" type="cloudTypeEnumeration"/>
                    <xs:attribute name="visibility" type="nonNegativeFloat"/>
                </xs:complexType>
        </xs:element>

        <xs:element name="PrecipitationConditionSet">
                <xs:annotation>
                    <xs:appinfo>Collects the precipitation elements for a single EnvironmentalCondition element.</xs:appinfo>
                </xs:annotation>
                <xs:complexType>
                    <xs:complexContent>
                        <xs:extension base="FullyExtensibleSMALElementType">
                            <xs:sequence>
                                <xs:element maxOccurs="unbounded" minOccurs="1" ref="LocalPrecipitation"/>
                            </xs:sequence>
                            <xs:attribute ref="startTime"/>
                            <xs:attribute ref="duration"/>
                        </xs:extension>
                    </xs:complexContent>
                </xs:complexType>
        </xs:element>

        <xs:element name="LocalPrecipitation">
                <xs:annotation>
                    <xs:appinfo>Describes a local precipitation event with Location, GeographicExtent, type, degree, starting altitued,
and associated visibility.  Precipitation may change over the course of an EnvironmentalCondition, so there is a duration
attribute.</xs:appinfo>
                </xs:annotation>
                <xs:complexType>
                    <xs:complexContent>
                        <xs:extension base="FullyExtensibleSMALElementType">
                            <xs:sequence>
                                <xs:element ref="GeographicExtent"/>
                            </xs:sequence>
                            <xs:attribute name="precipitationType" type="precipitationTypeEnumeration"/>
                            <xs:attribute name="precipitationDegree" type="precipitationDegreeEnumeration"/>
                            <xs:attribute name="altitude" type="xs:float"/>
                            <xs:attribute name="visibility" type="nonNegativeFloat"/>
                            <xs:attribute ref="startTime"/>
                            <xs:attribute ref="duration"/>
                        </xs:extension>
                    </xs:complexContent>
                </xs:complexType>
        </xs:element>

        <xs:element name="StaticModelSet">
                <xs:annotation>
                    <xs:appinfo>These models are not entities, but rather unchanging objects in the X3D scene: buildings, signs, roads,
bridges, etc.</xs:appinfo>
                </xs:annotation>
                <xs:complexType>
                    <xs:complexContent>
                        <xs:extension base="FullyExtensibleSMALElementType">
                            <xs:sequence>
                                <xs:element maxOccurs="unbounded" ref="StaticModelDefinition"/>
                            </xs:sequence>
                        </xs:extension>
                    </xs:complexContent>
                </xs:complexType>
        </xs:element>

        <xs:element name="EntitySet">
                <xs:annotation>
                    <xs:appinfo>Contains the active entities which operate in the virtual environment.</xs:appinfo>
                </xs:annotation>
                <xs:complexType>
                    <xs:complexContent>
                        <xs:extension base="FullyExtensibleSMALElementType">
                            <xs:sequence>
                                <xs:element maxOccurs="unbounded" ref="EntityDefinition"/>
                            </xs:sequence>
                        </xs:extension>
                    </xs:complexContent>
                </xs:complexType>
        </xs:element>

        <xs:complexType name="ObjectDefinitionType">
                <xs:annotation>
                    <xs:appinfo>Super-type which contains the base elements for all non-terrain models, including identification, 3D
model, physical parameters, and attachment points, if any.</xs:appinfo>
                </xs:annotation>
                <xs:complexContent>
                    <xs:extension base="FullyExtensibleSMALElementType">
                        <xs:sequence>
                            <xs:element maxOccurs="1" minOccurs="1" ref="Classification"/>
                            <xs:element maxOccurs="unbounded" minOccurs="1" ref="IdentificationParameters"/>
                            <xs:element maxOccurs="1" minOccurs="1" ref="PhysicalParameters"/>
                            <xs:element maxOccurs="1" minOccurs="0" ref="AttachmentPointSet"/>
                            <xs:element maxOccurs="unbounded" minOccurs="1" ref="ObjectModel"/>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
        </xs:complexType>

        <!-- ********************************************** -->
        <!-- SMAL                  -->
        <!--      StaticModelDefinition -->
        <!-- ********************************************** -->
        <xs:element name="StaticModelDefinition">
                <xs:annotation>
                    <xs:appinfo>Defines a single static model.</xs:appinfo>
                </xs:annotation>
                <xs:complexType>
                    <xs:complexContent>
                        <xs:extension base="ObjectDefinitionType">
                            <xs:sequence>
                                <xs:element maxOccurs="1" minOccurs="1" ref="LocationOrientation"/>
                            </xs:sequence>
                            <xs:attribute name="staticModelIdentifier" type="xs:ID"/>
                        </xs:extension>
                    </xs:complexContent>
                </xs:complexType>
        </xs:element>

        <!-- ********************************************** -->
        <!-- SMAL              -->
        <!--      EntityDefinition -->
        <!-- ********************************************** -->
        <xs:element name="EntityDefinition">
                <xs:annotation>
                    <xs:appinfo>Defines a single entity down to network identification. Also includes default physical parameters and
behavioral data for that particular entity type</xs:appinfo>
                </xs:annotation>
                <xs:complexType>
                    <xs:complexContent>
                        <xs:extension base="ObjectDefinitionType">
                            <xs:sequence>
                                <xs:element maxOccurs="1" minOccurs="0" ref="CurrentConditionParameters"/>
                                <xs:element maxOccurs="1" minOccurs="0" ref="NetworkedCommunicationParameterSet"/>
                                <xs:element maxOccurs="unbounded" minOccurs="0" ref="BehaviorParameterSet">
                                    <xs:annotation>
                                        <xs:appinfo>Multiple BehaviorParameters sets allow for sensitivity analysis using
different parameter settings or to provide unclassified and classified parameters for the purpose of creating unclassified products from
otherwise classified datasets.</xs:appinfo>
                                    </xs:annotation>
                                </xs:element>
                            </xs:sequence>
                            <xs:attribute name="entityIdentifier" type="xs:ID"/>
                        </xs:extension>
```

```
			</xs:complexContent>
		</xs:complexType>
	</xs:element>

	<xs:element name="IdentificationParameters">
		<xs:annotation>
			<xs:appinfo>Human-readable identifying data for model. NOTE: Identification parameters are distinct from DIS
enumerations.</xs:appinfo>
		</xs:annotation>
		<xs:complexType>
			<xs:complexContent>
				<xs:extension base="FullyExtensibleSMALElementType">
					<xs:attribute name="name" type="xs:string" use="required"/>
					<xs:attribute name="unit" type="xs:string" use="optional"/>
				</xs:extension>
			</xs:complexContent>
		</xs:complexType>
	</xs:element>

	<xs:element name="PhysicalParameters">
		<xs:annotation>
			<xs:appinfo>Contains all parameters which are dependent on the physical characteristics and performance of the
model.</xs:appinfo>
		</xs:annotation>
		<xs:complexType>
			<xs:complexContent>
				<xs:extension base="FullyExtensibleSMALElementType">
					<xs:sequence>
						<xs:element maxOccurs="1" minOccurs="0" ref="Classification"/>
						<xs:element maxOccurs="1" minOccurs="1" ref="PhysicalConstraints"/>
						<xs:element maxOccurs="1" minOccurs="0" ref="DynamicResponseConstraints"/>
						<xs:element maxOccurs="1" minOccurs="0" ref="TacticalConstraints"/>
					</xs:sequence>
				</xs:extension>
			</xs:complexContent>
		</xs:complexType>
	</xs:element>

	<xs:element name="PhysicalConstraints">
		<xs:annotation>
			<xs:appinfo>Physics modeling data for collision detection and physics engine use. NOTE: AVCL parameters can be
included optionally here. </xs:appinfo>
		</xs:annotation>
		<xs:complexType>
			<xs:complexContent>
				<xs:extension base="FullyExtensibleSMALElementType">
					<xs:attribute ref="unitSystem"/>
					<xs:attribute name="height" type="xs:float"/>
					<xs:attribute name="width" type="xs:float"/>
					<xs:attribute name="length" type="xs:float"/>
					<xs:attribute name="grossWeight" type="xs:float"/>
					<xs:attribute name="draft" type="xs:float" use="optional"/>
					<xs:attribute name="trackWidth" type="xs:float" use="optional"/>
					<xs:attribute name="wheelbase" type="xs:float" use="optional"/>
				</xs:extension>
			</xs:complexContent>
		</xs:complexType>
	</xs:element>

	<xs:element name="DynamicResponseConstraints">
		<xs:annotation>
			<xs:appinfo>Performance characteristics including speed, acceleration, and turn rate and radius.</xs:appinfo>
		</xs:annotation>
		<xs:complexType>
			<xs:complexContent>
				<xs:extension base="FullyExtensibleSMALElementType">
					<xs:attribute ref="unitSystem"/>
					<xs:attribute default="0 0 0" name="centerOfGravity" type="threeTuplePattern"/>
					<xs:attribute name="aerodynamicCenter" type="threeTuplePattern" use="optional"/>
					<xs:attribute name="centerOfBuoyancy" type="threeTuplePattern" use="optional"/>
```

```
                                        <xs:attribute name="maximumSpeed" type="xs:float"/>
                                        <xs:attribute name="cruiseSpeed" type="xs:float"/>
                                        <xs:attribute name="maximumAltitude" type="xs:float" use="optional"/>
                                        <xs:attribute name="cruiseAltitude" type="xs:float" use="optional"/>
                                        <xs:attribute name="maximumDepth" type="xs:float" use="optional"/>
                                        <xs:attribute name="cruiseDepth" type="xs:float" use="optional"/>
                                        <xs:attribute name="maximumAcceleration" type="xs:float" use="optional"/>
                                        <xs:attribute name="maximumDeceleration" type="xs:float" use="optional"/>
                                        <xs:attribute name="minimumTurnRadius" type="xs:float" use="optional">
                                                <xs:annotation>
                                                        <xs:appinfo>Tactical Radius for ships or Best Cornering Speed turn radius for
aircraft.</xs:appinfo>
                                                </xs:annotation>
                                        </xs:attribute>
                                        <xs:attribute name="maximumTurnRate" type="xs:float" use="optional"/>
                                        <xs:attribute name="maximumFuelCapacity" type="xs:float" use="optional"/>
                                </xs:extension>
                        </xs:complexContent>
                </xs:complexType>
        </xs:element>

        <xs:element name="TacticalConstraints">
                <xs:annotation>
                        <xs:appinfo>Domain-specific threat and detection ranges for attack and defense capabilities.</xs:appinfo>
                </xs:annotation>
                <xs:complexType>
                        <xs:complexContent>
                                <xs:extension base="FullyExtensibleSMALElementType">
                                        <xs:attribute ref="unitSystem"/>
                                        <xs:attribute name="maximumAirThreatRange" type="xs:float" use="optional"/>
                                        <xs:attribute name="maximumSurfaceThreatRange" type="xs:float" use="optional"/>
                                        <xs:attribute name="maximumSubsurfaceThreatRange" type="xs:float" use="optional"/>
                                        <xs:attribute name="maximumAirDetectionRange" type="xs:float" use="optional"/>
                                        <xs:attribute name="maximumSurfaceDetectionRange" type="xs:float" use="optional"/>
                                        <xs:attribute name="maximumSubsurfaceDetectionRange" type="xs:float" use="optional"/>
                                </xs:extension>
                        </xs:complexContent>
                </xs:complexType>
        </xs:element>

        <xs:element name="CurrentConditionParameters">
                <xs:annotation>
                        <xs:appinfo>Parameters used to describe the location, orientation, and state of an entity.</xs:appinfo>
                </xs:annotation>
                <xs:complexType>
                        <xs:complexContent>
                                <xs:extension base="FullyExtensibleSMALElementType">
                                        <xs:sequence>
                                                <xs:annotation/>
                                                <xs:element maxOccurs="1" ref="LocationOrientation"/>
                                        </xs:sequence>
                                        <xs:attribute default="0" name="currentSpeed" type="xs:float">
                                                <xs:annotation>
                                                        <xs:appinfo>Assumed to be along the current vehicle track.</xs:appinfo>
                                                </xs:annotation>
                                        </xs:attribute>
                                        <xs:attribute name="currentAcceleration" type="threeTuplePattern" use="optional">
                                                <xs:annotation>
                                                        <xs:appinfo>Linear acceleration along global-aligned coordinate axes.</xs:appinfo>
                                                </xs:annotation>
                                        </xs:attribute>
                                        <xs:attribute name="currentTurnRate" type="xs:float" use="optional">
                                                <xs:annotation>
                                                        <xs:appinfo>Degrees of heading change per second around global-aligned coordinate
axes.</xs:appinfo>
                                                </xs:annotation>
                                        </xs:attribute>
                                        <xs:attribute name="currentFuelState" type="xs:float" use="optional">
                                                <xs:annotation>
                                                        <xs:appinfo>The volumetric amount of usable fuel aboard the vehicle.</xs:appinfo>
                                                </xs:annotation>
                                        </xs:attribute>
                                        <xs:attribute name="currentPayloadWeight" type="xs:float" use="optional">
                                                <xs:annotation>
                                                        <xs:appinfo>All weight not attributable to the vehicle or internal fuel.</xs:appinfo>
                                                </xs:annotation>
                                        </xs:attribute>
                                </xs:extension>
                        </xs:complexContent>
                </xs:complexType>
        </xs:element>

        <xs:element name="LocationOrientation">
                <xs:annotation>
                        <xs:appinfo>Contains data on position, orientation, and (optionally) direction of motion for the object it is
describing.</xs:appinfo>
                </xs:annotation>
                <xs:complexType>
                        <xs:complexContent>
                                <xs:extension base="FullyExtensibleSMALElementType">
                                        <xs:sequence>
                                                <xs:element maxOccurs="1" minOccurs="1" ref="Location"/>
                                                <xs:element maxOccurs="1" minOccurs="1" ref="Orientation"/>
                                                <xs:element maxOccurs="1" minOccurs="0" ref="DirectionOfMotion"/>
                                        </xs:sequence>
                                </xs:extension>
                        </xs:complexContent>
                </xs:complexType>
        </xs:element>

        <xs:element abstract="true" name="Location">
                <xs:annotation>
                        <xs:appinfo>Abstract placeholder for a specific locator element.</xs:appinfo>
                </xs:annotation>
        </xs:element>

        <xs:element abstract="true" name="MapCoordinate">
                <xs:annotation>
                        <xs:appinfo>Abstract placeholder for coordinate-reference-system-specific location value element.</xs:appinfo>
                </xs:annotation>
        </xs:element>

        <xs:element name="MgrsCoordinate" substitutionGroup="MapCoordinate">
                <xs:annotation>
                        <xs:appinfo>Military Grid Reference System coordinate location value element.</xs:appinfo>
                </xs:annotation>
                <xs:complexType>
                        <xs:complexContent>
                                <xs:extension base="AttributeExtensibleSMALElementType">
                                        <xs:attribute name="coordinate" type="mgrsCoordinatePattern"/>
                                </xs:extension>
                        </xs:complexContent>
                </xs:complexType>
        </xs:element>

        <xs:complexType name="CartesianCoordinateType" final="restriction">
                <xs:annotation>
                        <xs:appinfo>A complexType used to describe a three-value coordinate location.</xs:appinfo>
                </xs:annotation>
                <xs:complexContent>
                        <xs:extension base="AttributeExtensibleSMALElementType">
                                <xs:attribute ref="unitSystem"/>
                                <xs:attribute default="0" name="xValue" type="xs:float"/>
                                <xs:attribute default="0" name="yValue" type="xs:float"/>
                                <xs:attribute default="0" name="zValue" type="xs:float"/>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>

        <xs:element name="CartesianCoordinate" substitutionGroup="MapCoordinate">
```

```
		<xs:annotation>
			<xs:appinfo>Cartesian coordinate system coordinate-value location element.</xs:appinfo>
		</xs:annotation>
	</xs:element>

	<xs:element name="LatLongCoordinate" substitutionGroup="MapCoordinate">
		<xs:annotation>
			<xs:appinfo>Latitude-Longitude coordinate system coordinate-value location element.  References a GeoOrigin
element.</xs:appinfo>
		</xs:annotation>
		<xs:complexType>
			<xs:complexContent>
				<xs:extension base="AttributeExtensibleSMALElementType">
					<xs:attribute default="N00 00.0" name="latitude" type="latitudeOrdinatePattern"/>
					<xs:attribute default="W00 00.0" name="longitude" type="longitudeOrdinatePattern"/>
					<xs:attribute name="geoOriginReference" type="xs:IDREF" use="required"/>
				</xs:extension>
			</xs:complexContent>
		</xs:complexType>
	</xs:element>

	<xs:element name="AttachmentPointLocation" substitutionGroup="Location">
		<xs:annotation>
			<xs:appinfo>Object AttachmentPoint reference-based location element. The @attachmentPointReference value is
used to reference the @attachmentPointIdentifier value from an AttachmentPoint element.</xs:appinfo>
		</xs:annotation>
		<xs:complexType>
			<xs:complexContent>
				<xs:extension base="AttributeExtensibleSMALElementType">
					<xs:attribute name="attachmentPointReference" type="xs:IDREF"/>
				</xs:extension>
			</xs:complexContent>
		</xs:complexType>
	</xs:element>

	<xs:element name="RelativeVirtualLocation" substitutionGroup="Location">
		<xs:annotation>
			<xs:appinfo>Object-reference based location element used when there is no specific attachment point specified on
that object. The @baseLocationObject IDREF value is used to reference the @entityIdentifier ID value of an EntityDefinition
element.</xs:appinfo>
		</xs:annotation>
		<xs:complexType>
			<xs:complexContent>
				<xs:extension base="CartesianCoordinateType">
					<xs:attribute name="baseObjectReference" type="xs:IDREF" use="required"/>
				</xs:extension>
			</xs:complexContent>
		</xs:complexType>
	</xs:element>

	<xs:element name="GlobalVirtualLocation" substitutionGroup="Location" type="CartesianCoordinateType">
		<xs:annotation>
			<xs:appinfo>Current point of origin based virtual location element.</xs:appinfo>
		</xs:annotation>
	</xs:element>

	<xs:element name="GeographicLocation" substitutionGroup="Location">
		<xs:annotation>
			<xs:appinfo>Global geographic reference based geographic location element. References a GeoOrigin
element.</xs:appinfo>
		</xs:annotation>
		<xs:complexType>
			<xs:complexContent>
				<xs:extension base="FullyExtensibleSMALElementType">
					<xs:attribute ref="unitSystem"/>
					<xs:attribute name="latitude" type="latitudeOrdinatePattern"/>
					<xs:attribute name="longitude" type="longitudeOrdinatePattern"/>
					<xs:attribute name="verticalPosition" type="xs:float"/>
					<xs:attribute name="geoOriginReference" type="xs:IDREF" use="required"/>
				</xs:extension>
				</xs:complexContent>
			</xs:complexType>
		</xs:element>

		<xs:element abstract="true" name="Orientation">
			<xs:annotation>
				<xs:appinfo>Abstract placeholder for a specific orientor element.</xs:appinfo>
			</xs:annotation>
		</xs:element>
		<xs:element abstract="true" name="DirectionOfMotion">
			<xs:annotation>
				<xs:appinfo>Abstract placeholder for a specific element describing direction of travel.</xs:appinfo>
			</xs:annotation>
		</xs:element>


		<xs:element name="XYZOrientation" substitutionGroup="Orientation" type="CartesianCoordinateType">
			<xs:annotation>
				<xs:appinfo>A simple set of orthogonal angles based on the current reference coordinate axes. Use fractional degree
measurements for the angles vice radians.</xs:appinfo>
			</xs:annotation>
		</xs:element>
		<xs:element name="VectorDirection" substitutionGroup="DirectionOfMotion" type="CartesianCoordinateType">
			<xs:annotation>
				<xs:appinfo>A simple set of orthogonal angles based on the current reference coordinate axes. Use fractional degree
measurements for the angles vice radians.</xs:appinfo>
			</xs:annotation>
		</xs:element>

		<xs:complexType name="QuaternionType">
			<xs:annotation>
				<xs:appinfo>A set of four values based on the current reference coordinate axes describing an arbitrary axis and the
rotation around it.  Use fractional values -1 to 1 for the xValue, yValue, and zValues and radians for the wValue</xs:appinfo>
			</xs:annotation>
			<xs:complexContent>
				<xs:extension base="FullyExtensibleSMALElementType">
					<xs:attribute ref="unitSystem"/>
					<xs:attribute default="0" name="xValue" type="unitValueFloat"/>
					<xs:attribute default="0" name="yValue" type="unitValueFloat"/>
					<xs:attribute default="0" name="zValue" type="unitValueFloat"/>
					<xs:attribute default="0" name="wValue" type="nonNegativeFloat"/>
				</xs:extension>
			</xs:complexContent>
		</xs:complexType>

		<xs:element name="QuaternionOrientation" substitutionGroup="Orientation" type="QuaternionType">
			<xs:annotation>
				<xs:appinfo>A set of four values based on the current reference coordinate axes. Use fractional values -1 to 1 for the
first three numbers and a radian value for the fourth.</xs:appinfo>
			</xs:annotation>
		</xs:element>
		<xs:element name="QuaternionDirection" substitutionGroup="DirectionOfMotion" type="QuaternionType">
			<xs:annotation>
				<xs:appinfo>A set of four values based on the current reference coordinate axes. Use fractional values -1 to 1 for the
first three numbers and a radian value for the fourth.</xs:appinfo>
			</xs:annotation>
		</xs:element>

		<xs:complexType name="HeadingType">
			<xs:annotation>
				<xs:appinfo>An angular direction in degrees based on either Magnetic or True north, and a pitch angle to describe
two-and-a-half-dimensional orientation</xs:appinfo>
			</xs:annotation>
			<xs:complexContent>
				<xs:extension base="FullyExtensibleSMALElementType">
					<xs:attribute default="0" name="angle" type="xs:float"/>
					<xs:attribute default="MAGNETIC" name="northReference" type="northReferenceEnumeration"/>
					<xs:attribute default="0" name="pitchAngle" type="xs:float"/>
				</xs:extension>
			</xs:complexContent>
```

```
	</xs:complexType>
	<xs:element name="Heading" substitutionGroup="Orientation" type="HeadingType">
		<xs:annotation>
			<xs:appinfo>The current geographic direction of the longitudinal axis of this object, or where it is pointed, as
separate from the direction in which it is moving.</xs:appinfo>
		</xs:annotation>
	</xs:element>
	<xs:element name="Track" substitutionGroup="DirectionOfMotion" type="HeadingType">
		<xs:annotation>
			<xs:appinfo>The current geographic direction of travel of this object.</xs:appinfo>
		</xs:annotation>
	</xs:element>

	<xs:element name="NetworkedCommunicationParameterSet">
		<xs:annotation>
			<xs:appinfo>Entity-specific network communication parameters for use in message construction. Parameters
common to all entities are held in the ancestor NetworkChannel element. The protocol-specific tags will eventually be replaced by
namespace references.</xs:appinfo>
		</xs:annotation>
		<xs:complexType>
			<xs:complexContent>
				<xs:extension base="FullyExtensibleSMALElementType">
					<xs:sequence>
						<xs:element minOccurs="0" ref="DisConfiguration"/>
						<xs:element minOccurs="0" ref="LinkConfiguration"/>
						<xs:element minOccurs="0" ref="HlaConfiguration"/>
					</xs:sequence>
				</xs:extension>
			</xs:complexContent>
		</xs:complexType>
	</xs:element>

	<xs:element name="DisConfiguration">
		<xs:annotation/>
		<xs:complexType>
			<xs:complexContent>
				<xs:extension base="FullyExtensibleSMALElementType">
					<xs:attribute name="marking" type="xs:string" use="optional"/>
					<!-- marking is limited to 11 characters -->
					<xs:attribute name="entityID" type="xs:nonNegativeInteger" use="optional"/>
					<xs:attribute name="forceID" type="xs:nonNegativeInteger" use="optional"/>
					<xs:attribute name="entityName" type="xs:NMTOKEN" use="optional"/>
					<xs:attribute name="entityCountry" type="xs:nonNegativeInteger" use="required"/>
					<xs:attribute name="entityKind" type="xs:nonNegativeInteger" use="required"/>
					<xs:attribute name="entityDomain" type="xs:nonNegativeInteger" use="required"/>
					<xs:attribute name="entityCategory" type="xs:nonNegativeInteger" use="required"/>
					<xs:attribute name="entitySubCategory" type="xs:nonNegativeInteger" use="optional"/>
					<xs:attribute name="entityExtra" type="xs:nonNegativeInteger" use="optional"/>
					<xs:attribute name="entitySpecific" type="xs:nonNegativeInteger" use="optional"/>
				</xs:extension>
			</xs:complexContent>
		</xs:complexType>
	</xs:element>

	<xs:element name="LinkConfiguration">
		<!-- Hypothetical example only, projected for use with Link11 and Link16 protocols -->
		<xs:complexType>
			<xs:attributeGroup ref="attlist.SampleConfigurationParameters"/>
		</xs:complexType>
	</xs:element>

	<xs:element name="HlaConfiguration">
		<!-- Hypothetical example only, not yet developed. -->
		<xs:complexType>
			<xs:attributeGroup ref="attlist.SampleConfigurationParameters"/>
		</xs:complexType>
	</xs:element>

	<xs:element name="BehaviorParameterSet">
		<xs:annotation>
```

```
            <xs:appinfo>Data required for proper agent-based behavior simulation. NOTE: Knots and nautical miles are not used
in parameters or modeling, only Metric or English units are used in SMAL files.</xs:appinfo>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="FullyExtensibleSMALElementType">
                    <xs:sequence>
                        <xs:element minOccurs="0" ref="Classification"/>
                        <xs:element maxOccurs="unbounded" minOccurs="1" ref="SimulationAgent"/>
                    </xs:sequence>
                    <xs:attribute ref="unitSystem"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>

    <xs:element name="SimulationAgent">
        <xs:annotation>
            <xs:appinfo>Agent-specific information required to run the simulation. Includes behavioral probability parameters
and similar information not specifically enumerated in the tactical or dynamic response constraints.</xs:appinfo>
        </xs:annotation>
        <xs:complexType>
            <xs:annotation>
                <xs:appinfo>The entity name should be found in the parent EntityDefinition/@entityIdentifier
attribute.</xs:appinfo>
            </xs:annotation>
            <xs:complexContent>
                <xs:extension base="FullyExtensibleSMALElementType">
                    <xs:sequence>
                        <xs:element maxOccurs="unbounded" minOccurs="0" ref="Parameter"/>
                    </xs:sequence>
                    <xs:attribute name="simulationAgentIdentifier" type="xs:ID" use="optional"/>
                    <xs:attribute name="agent" type="simulationEngineEnumeration" use="required"/>
                    <xs:attribute name="type" type="xs:anyURI" use="required">
                        <xs:annotation>
                            <xs:appinfo>The filename or other identifier for the agent behavior file to be attached to this
entity.</xs:appinfo>
                        </xs:annotation>
                    </xs:attribute>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>

    <xs:element name="Parameter">
        <xs:annotation>
            <xs:appinfo>A generic name-value pair, used to hold a datum the agent requires that is not already established in the
metadata.</xs:appinfo>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="FullyExtensibleSMALElementType">
                    <xs:attribute name="name" type="xs:NMTOKEN" use="required"/>
                    <xs:attribute name="value" type="xs:string" use="required"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>

    <xs:element name="AttachmentPointSet">
        <xs:annotation>
            <xs:appinfo>Collects the set of locations at which other objects can be attached to this one.</xs:appinfo>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="FullyExtensibleSMALElementType">
                    <xs:sequence>
                        <xs:element maxOccurs="unbounded" ref="AttachmentPoint"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>

    <xs:element name="AttachmentPoint">
        <xs:annotation>
            <xs:appinfo>Allows the attachment to this unit of independently operating entities by type and location. This
AttachmentPoint can be uniquely identified using the attachmentPointIdentifier attribute.</xs:appinfo>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="FullyExtensibleSMALElementType">
                    <xs:sequence>
                        <xs:annotation>
                            <xs:appinfo>The GlobalVirtualLocation is used to modify the position of the AttachmentPoint
from the origin of the object to which it belongs.</xs:appinfo>
                            <xs:appinfo>The Orientation element is used to modify the orientation of the AttachmentPoint
from the orientation of the object to which it belongs.</xs:appinfo>
                        </xs:annotation>
                        <xs:element maxOccurs="1" minOccurs="1" ref="GlobalVirtualLocation"/>
                        <xs:element maxOccurs="1" minOccurs="1" ref="Orientation"/>
                    </xs:sequence>
                    <xs:attribute name="attachmentCategory" type="attachmentCategoryEnumeration"/>
                    <xs:attribute name="attachmentPointIdentifier" type="xs:ID"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>

    <!-- vvvvv UNDER CONSTRUCTION vvvvv -->

    <xs:element name="AssociationSet">
        <xs:annotation>
            <xs:appinfo>Collects all Association elements for a Simulation.</xs:appinfo>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="FullyExtensibleSMALElementType">
                    <xs:sequence>
                        <xs:element maxOccurs="unbounded" minOccurs="1" ref="Association"/>
                        <xs:element minOccurs="0" maxOccurs="1" name="Assembly">
                            <xs:complexType>
                                <xs:attribute name="viskitAssembly" type="xs:anyURI"/>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>

    <xs:element name="Association">
        <xs:annotation>
            <xs:appinfo>Allows entities to be associated with each other by command relationship and/or as an embarking entity
(onboard or attached to an entity).</xs:appinfo>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="FullyExtensibleSMALElementType">
                    <xs:sequence>
                        <xs:element maxOccurs="unbounded" minOccurs="2" ref="AssociatedEntity"/>
                    </xs:sequence>
                    <xs:attribute name="transactionCategory" type="transactionCategoryEnumeration"/>
                    <xs:attribute name="associationIdentifier" type="xs:ID"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>

    <xs:element name="AssociatedEntity">
```

```
            <xs:annotation>
                <xs:appinfo>A reference to a single entity member of an association which describes its heirarchical position in the
relationship and its embarked status, if any.</xs:appinfo>
            </xs:annotation>
            <xs:complexType>
                <xs:complexContent>
                    <xs:extension base="FullyExtensibleSMALElementType">
                        <xs:attribute name="entityReference" type="xs:IDREF"/>
                        <xs:attribute name="transactionDirection" type="transactionDirectionEnumeration"/>
                        <xs:attribute name="isEmbarked" type="xs:boolean"/>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
        </xs:element>

        <!-- ^^^^^ UNDER CONSTRUCTION ^^^^^ -->

        <xs:element name="Classification">
            <xs:annotation>
                <xs:appinfo>The level of classification for the line item this element describes, similar to classification of documents
by section or paragraph.</xs:appinfo>
            </xs:annotation>
            <xs:complexType>
                <xs:complexContent>
                    <xs:extension base="FullyExtensibleSMALElementType">
                        <xs:attribute default="UNCLASSIFIED" name="level" type="classificationLevelEnumeration"
use="optional"/>
                        <xs:attribute name="reference" type="xs:string" use="optional"/>
                        <xs:attribute name="rationale" type="xs:string" use="optional"/>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
        </xs:element>

        <xs:element abstract="true" name="ObjectModel">
            <xs:annotation>
                <xs:appinfo>Abstract placeholder for 3D Model elements. Only X3D Model elements are currently
defined.</xs:appinfo>
            </xs:annotation>
        </xs:element>

        <xs:element name="X3DArchiveModel" substitutionGroup="ObjectModel">
            <xs:annotation>
                <xs:appinfo>The descriptive requirements necessary to find a particular model in an X3D Model Archive.  Built
upon the SAVAGE Model Archive structure.</xs:appinfo>
            </xs:annotation>
            <xs:complexType>
                <xs:complexContent>
                    <xs:extension base="FullyExtensibleSMALElementType">
                        <xs:sequence>
                            <xs:element minOccurs="0" ref="Classification"/>
                        </xs:sequence>
                        <xs:attribute default="Savage" name="archive" type="modelArchiveEnumeration"/>
                        <xs:attribute name="section" type="xs:NMTOKEN"/>
                        <xs:attribute name="chapter" type="xs:NMTOKEN"/>
                        <xs:attribute name="subChapter" type="xs:NMTOKEN"/>
                        <xs:attribute name="model" type="xs:NMTOKEN"/>
                        <xs:attribute default="http://www.web3d.org/x3d/content/examples/" name="alternateBaseURL"
type="xs:anyURI">
                            <xs:annotation>
                                <xs:appinfo>Required only when using non-SAVAGE Archive X3D models.</xs:appinfo>
                            </xs:annotation>
                        </xs:attribute>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
        </xs:element>

        <xs:attributeGroup name="attlist.SampleConfigurationParameters">
            <xs:attribute name="name" type="xs:NMTOKEN"/>
            <xs:attribute name="country" type="xs:NMTOKEN"/>
            <xs:attribute name="kind" type="xs:NMTOKEN"/>
            <xs:attribute name="domain" type="xs:NMTOKEN"/>
            <xs:attribute name="category" type="xs:NMTOKEN"/>
            <xs:attribute name="subCategory" type="xs:NMTOKEN"/>
            <xs:attribute name="extra" type="xs:NMTOKEN"/>
        </xs:attributeGroup>

        <xs:attribute default="Metric" name="unitSystem" type="unitSystemEnumeration"/>

        <xs:attribute default="00:00:00" name="startTime" type="xs:time"/>
        <xs:attribute default="01:00:00" name="duration" type="xs:time"/>

        <xs:attribute default="squareKilometers" name="areaUnits" type="areaUnitEnumeration"/>
        <xs:attribute default="meters" name="linearUnits" type="linearUnitEnumeration"/>
        <xs:attribute default="kilometersPerHour" name="speedUnits" type="speedUnitEnumeration"/>
        <xs:attribute default="kilograms" name="weightUnits" type="weightUnitEnumeration"/>
        <xs:attribute default="Celsius" name="temperatureUnits" type="temperatureUnitEnumeration"/>

</xs:schema>
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. SMAL EXAMPLE

```xml
<?xml version="1.0" encoding="UTF-8"?>
<SMAL xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="file:/www.web3d.org/x3d/content/examples/Savage/Tools/SMAL/SavageModelingAnalysis
      Language1.0.xsd" version="1.0">
  <head>
    <Classification level="UNCLASSIFIED"/>
    <meta name="" content=""/>
  </head>
  <Simulation simulationIdentifier="ONE">
    <NetworkChannel address="127.0.0.1" applicationID="1" enabled="true" siteID="1" multicastRelayHost="215.125.0.0"
          multicastRelayPort="215" port="25"/>
    <World>
      <GeoOrigin geoCoords="N00 0.0 W00 0.0" geoSystem="GD WE" rotateYUp="false"
          geoOriginIdentifier="GEOORIGIN01"/>
      <TerrainTileSet>
        <TerrainTile terrainTileIdentifier="CENTER" tileCategory="landTerrain">
          <Classification level="UNCLASSIFIED"/>
          <GeoOrigin geoOriginIdentifier="GEOORIGINTILE01" geoCoords="N00 0.0 W00 0.0" geoSystem="'GD', 'WE'"
              rotateYUp="true"/>
          <GeographicExtent area="100" unitSystem="English" verticalExtent="5000">
            <PointRadiusCircular unitSystem="English" radius="500">
              <CartesianCoordinate zValue="0" yValue="0" xValue="0"/>
            </PointRadiusCircular>
          </GeographicExtent>
          <OverlaySet>
            <Classification level="UNCLASSIFIED"/>
            <OverlaySetImagery centerPointLatitude="N00 0.0" centerPointLongitude="W00 0.0"
              fileLocationURL="http://web.nps.navy.mil/~brutzman/Savage/Harbors/Bremerton/textures/BremertonSat.jpg"/>
          </OverlaySet>
          <X3DArchiveModel archive="Savage" section="Harbors" chapter="Bremerton" model="Bremerton.x3d">
            <Classification level="UNCLASSIFIED"/>
          </X3DArchiveModel>
          <GlobalVirtualLocation xValue="0" yValue="0" zValue="0"/>
        </TerrainTile>
      </TerrainTileSet>
      <StaticModelSet>
        <StaticModelDefinition staticModelIdentifier="HULK">
          <Classification level="UNCLASSIFIED"/>
          <IdentificationParameters name="Grounded Hulk"/>
          <PhysicalParameters>
            <Classification level="UNCLASSIFIED"/>
            <PhysicalConstraints height="19" width="20" length="300" draft="3" unitSystem="Metric"/>
          </PhysicalParameters>
          <X3DArchiveModel archive="Savage" section="ShipsCivilian" chapter="CargoShips" model="Freighter.x3d">
            <Classification level="UNCLASSIFIED"/>
          </X3DArchiveModel>
          <LocationOrientation>
            <GeographicLocation latitude="N00 0.0" longitude="W01 0.0" verticalPosition="-30"
              geoOriginReference="GEOORIGIN01"/>
            <Heading angle="020" northReference="MAGNETIC"/>
          </LocationOrientation>
        </StaticModelDefinition>
      </StaticModelSet>
    </World>
    <EntitySet>
      <EntityDefinition entityIdentifier="ENT01">
        <Classification level="UNCLASSIFIED"/>
        <IdentificationParameters name="USS Ticonderoga"/>
        <PhysicalParameters>
          <Classification level="UNCLASSIFIED"/>
          <PhysicalConstraints height="25" width="20" length="167" grossWeight="45000000" draft="15"
              unitSystem="Metric"/>
          <DynamicResponseConstraints maximumSpeed="30" cruiseSpeed="15" unitSystem="Metric"/>
          <TacticalConstraints maximumAirDetectionRange="320" maximumAirThreatRange="280"
              maximumSurfaceDetectionRange="120" maximumSurfaceThreatRange="160" unitSystem="Metric"/>
  </PhysicalParameters>
  <X3DArchiveModel archive="Savage" section="ShipsMilitary" chapter="Cruiser-UnitedStates" model="CG47.x3d">
    <Classification level="UNCLASSIFIED"/>
  </X3DArchiveModel>
  <CurrentConditionParameters currentAcceleration="0 0 0" currentFuelState="300000" currentSpeed="12"
          currentTurnRate="0">
    <LocationOrientation>
      <GeographicLocation latitude="N01 0.0" longitude="W00 59.0" verticalPosition="0"
        geoOriginReference="GEOORIGIN01"/>
      <Heading angle="270" northReference="MAGNETIC"/>
      <Track angle="270" northReference="MAGNETIC"/>
    </LocationOrientation>
  </CurrentConditionParameters>
  <NetworkedCommunicationParameterSet>
    <DisConfiguration entityCountry="225" entityKind="1" entityDomain="6" entityCategory="3"></DisConfiguration>
  </NetworkedCommunicationParameterSet>
  <BehaviorParameterSet>
    <Classification level="UNCLASSIFIED"/>
    <SimulationAgent agent="Viskit" type="HVU.xml"/>
  </BehaviorParameterSet>
</EntityDefinition>
<EntityDefinition entityIdentifier="ENT02">
  <Classification level="UNCLASSIFIED"/>
  <IdentificationParameters name="USS Arleigh Burke"/>
  <PhysicalParameters>
    <Classification level="UNCLASSIFIED"/>
    <PhysicalConstraints height="20" width="10" length="150" draft="12" grossWeight="340000000"
          unitSystem="Metric"/>
    <DynamicResponseConstraints maximumSpeed="30" cruiseSpeed="15"/>
    <TacticalConstraints maximumAirDetectionRange="320" maximumAirThreatRange="240" unitSystem="Metric"/>
  </PhysicalParameters>
  <X3DArchiveModel archive="Savage" section="ShipsMilitary" chapter="DDG-51_FlightIIA" model="DDG51.x3d">
    <Classification level="UNCLASSIFIED"/>
  </X3DArchiveModel>
  <CurrentConditionParameters currentAcceleration="0 0 0" currentFuelState="300" currentSpeed="12"
   currentTurnRate="0">
    <LocationOrientation>
      <GeographicLocation latitude="N01 0.0" longitude="W01 0.0" verticalPosition="0"
        geoOriginReference="GEOORIGIN01"/>
      <Heading angle="270" northReference="MAGNETIC"/>
      <Track angle="270" northReference="MAGNETIC"/>
    </LocationOrientation>
  </CurrentConditionParameters>
  <NetworkedCommunicationParameterSet>
    <DisConfiguration entityCountry="225" entityKind="1" entityDomain="6" entityCategory="4"/>
  </NetworkedCommunicationParameterSet>
  <BehaviorParameterSet>
    <Classification level="UNCLASSIFIED"/>
    <SimulationAgent agent="Viskit" type="Defender.xml"/>
  </BehaviorParameterSet>
</EntityDefinition>
<EntityDefinition entityIdentifier="ENT03">
  <Classification level="UNCLASSIFIED"/>
  <IdentificationParameters name="USS O'Kane"/>
  <PhysicalParameters>
    <Classification level="UNCLASSIFIED"/>
    <PhysicalConstraints height="20" width="10" length="150" draft="12" grossWeight="340000000"
          unitSystem="Metric"/>
    <DynamicResponseConstraints maximumSpeed="30" cruiseSpeed="15"/>
    <TacticalConstraints maximumAirDetectionRange="320" maximumAirThreatRange="240" unitSystem="Metric"/>
  </PhysicalParameters>
  <X3DArchiveModel archive="Savage" section="ShipsMilitary" chapter="DDG-51_FlightIIA" model="DDG51.x3d">
    <Classification level="UNCLASSIFIED"/>
  </X3DArchiveModel>
  <CurrentConditionParameters currentAcceleration="0 0 0" currentFuelState="300" currentSpeed="12"
   currentTurnRate="0">
    <LocationOrientation>
      <GeographicLocation latitude="N01 0.0" longitude="W01 1.0" verticalPosition="0"
        geoOriginReference="GEOORIGIN01"/>
      <Heading angle="270" northReference="MAGNETIC"/>
```

```
              <Track angle="270" northReference="MAGNETIC"/>
            </LocationOrientation>
          </CurrentConditionParameters>
          <NetworkedCommunicationParameterSet>
            <DisConfiguration entityCountry="225" entityKind="61" entityDomain="1" entityCategory="1"></DisConfiguration>
          </NetworkedCommunicationParameterSet>
          <BehaviorParameterSet>
            <Classification level="UNCLASSIFIED"/>
            <SimulationAgent agent="Viskit" type="Defender.xml"/>
          </BehaviorParameterSet>
        </EntityDefinition>
      </EntitySet>

      <EnvironmentalConditionSet>
        <TimeParameters startTime="10:00:00" duration="02:00:00" day="1" month="1" timeZone="-7"/>
        <EnvironmentalCondition startTime="10:00:00" duration="01:00:00" environmentalConditionIdentifier="ENVCON01">
          <Background skyColor=".2 .2 .2" groundColor=".5 .5 .3" groundAngle="1.571" skyAngle="-1.571"/>
          <WindConditionSet>
            <WindConditionAtLevel altitude="0" direction="090" speed="10"/>
          </WindConditionSet>
          <WaterConditionSet>
            <SeaState beaufortScaleEquivalent="3"></SeaState>
            <WaterCurrentConditionAtDepth depth="0" speed="5" direction="100" visibility="1.6"/>
          </WaterConditionSet>
        </EnvironmentalCondition>
        <EnvironmentalCondition startTime="11:00:00" duration="01:00:00" environmentalConditionIdentifier="ENVCON02">
          <Background skyColor=".2 .2 .2" groundColor=".5 .5 .3" groundAngle="1.571" skyAngle="-1.571"/>
          <WaterConditionSet>
            <SeaState beaufortScaleEquivalent="1"></SeaState>
            <WaterCurrentConditionAtDepth depth="0" speed="7" direction="110" visibility="1.6"/>
          </WaterConditionSet>
        </EnvironmentalCondition>
      </EnvironmentalConditionSet>

      <AssociationSet>
        <Association transactionCategory="COMMAND" associationIdentifier="ASSOC01">
          <AssociatedEntity entityReference="ENT01" transactionDirection="SEND"/>
          <AssociatedEntity entityReference="ENT02" transactionDirection="RECIEVE"/>
          <AssociatedEntity entityReference="ENT03" transactionDirection="RECIEVE"/>
        </Association>
      </AssociationSet>
    </Simulation>
</SMAL>
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. SAVAGE VEHICLE METADATA TEMPLATE

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D profile="Interchange" version="3.1" xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
xsd:noNamespaceSchemaLocation="http://www.web3d.org/specifications/x3d-3.1.xsd">
 <head>
  <meta content="SavageVehicleMetadataTemplate.x3d" name="title"/>
  <meta content="This scene defines the exemplar template for Savage vehicle metadata, allowing further interoperability via SMAL
constructs. Savage Modeling Analysis Language (SMAL) authoring capabilities for X3D assume proper metadata within a scene to
identify an object properly.  A corresponding native-XML .xsd schema for SMAL will also be developed to facilitate conversion and
use of vehicle metadata." name="description"/>
  <meta content="Travis Rauch, Don Brutzman" name="creator"/>
  <meta content="20 May 2005" name="created"/>
  <meta content="01 February 2006" name="modified"/>
  <meta content="SMAL vehicle metadata" name="subject"/>
  <meta content="under development" name="warning"/>
  <meta content="http://web.nps.navy.mil/~brutzman/Savage/Tools/SMAL/SavageVehicleMetadataTemplate.x3d"
name="identifier"/>
  <meta content="X3D-Edit, http://www.web3d.org/x3d/content/README.X3D-Edit.html" name="generator"/>
  <meta content="../../license.html" name="license"/>
 </head>
 <Scene>
  <WorldInfo title="SavageVehicleMetadataTemplate">
   <MetadataSet containerField="metadata" name="SMAL">
    <MetadataString containerField="value" name="version" value="1.0">
     <MetadataString name="appinfo" value="This is the version of SMAL employed, not of the model."/>
    </MetadataString>
    <MetadataSet containerField="value" name="EntityDefinition">
     <!--Identifying metadata for the current simulation of interest-->
     <MetadataSet containerField="value" name="Classification">
      <MetadataString containerField="value" name="level" value="UNCLASSIFIED">
       <MetadataString name="appinfo" value="UNCLASSIFIED, FOUO, CONFIDENTIAL, SECRET, or TOPSECRET"/>
      </MetadataString>
      <MetadataString containerField="value" name="reference" value="">
       <MetadataString name="appinfo" value="The published source of classified information, if any, contained in the
Metadata."/>
      </MetadataString>
      <MetadataString containerField="value" name="rationale" value="">
       <MetadataString name="appinfo" value="The specific element which contains the information classifying this document."/>
      </MetadataString>
     </MetadataSet>
     <MetadataSet containerField="value" name="IdentificationParameters">
      <MetadataString containerField="value" name="name" value="">
       <MetadataString name="appinfo" value="The plain language name of the vehicle this model represents, i.e. the base class
(DDG-51), or vehicle designation (M1A2)."/>
      </MetadataString>
     </MetadataSet>
     <MetadataSet containerField="value" name="PhysicalParameters">
      <MetadataSet containerField="value" name="PhysicalConstraints">
       <MetadataFloat containerField="value" name="height" value="0">
        <MetadataString name="appinfo" value="The maximum structural height of the object in meters. This may be used for
clearance checking or other calculations."/>
       </MetadataFloat>
       <MetadataFloat containerField="value" name="width" value="0">
        <MetadataString name="appinfo" value="The maximum width, beam, or wingspan of the vehicle in meters.  This may be
used for clearance checking or other calculations."/>
       </MetadataFloat>
       <MetadataFloat containerField="value" name="length" value="0">
        <MetadataString name="appinfo" value="The maximum structural length of the object in meters. This may be used for
clearance checking or other calculations."/>
       </MetadataFloat>
       <MetadataFloat containerField="value" name="draft" value="0">
        <MetadataString name="appinfo" value="The vertical distance in meters from the deepest point (keel or other structure) to
the waterline of a vehicle at its stated displacement or gross weight."/>
       </MetadataFloat>
       <MetadataFloat containerField="value" name="wheelbase" value="0">
```

```
          <MetadataString name="appinfo" value="The longitudinal distance in meters from the center of the forwardmost roadwheel
axle to the center of the rearmost roadwheel axle on this vehicle."/>
        </MetadataFloat>
        <MetadataFloat containerField="value" name="trackWidth" value="0">
          <MetadataString name="appinfo" value="The horizontal distance in meters from the rightmost edge of the right wheel or
track to the leftmost edge of the left wheel or track on this vehicle."/>
        </MetadataFloat>
        <MetadataFloat containerField="value" name="grossWeight" value="0">
          <MetadataString name="appinfo" value="The standard operational weight of the vehicle in pounds or kilograms. This may
be used in any number of physics calculations."/>
        </MetadataFloat>
        <MetadataFloat containerField="value" name="fuelCapacity" value="0">
          <MetadataString name="appinfo" value="The maximum usable internal fuel capacity of this vehicle in liters."/>
        </MetadataFloat>
      </MetadataSet>
      <MetadataSet containerField="value" name="DynamicResponseConstraints">
        <MetadataFloat containerField="value" name="centerOfGravity" value="0 0 0">
          <MetadataString name="appinfo" value="Sets the Center of Gravity of the object as an (x, y, z) distance in meters from the
origin of the scene, which is located at (0, 0, 0)."/>
        </MetadataFloat>
        <MetadataFloat containerField="value" name="aerodynamicCenter" value="0 0 0">
          <MetadataString name="appinfo" value="Sets the Aerodynamic Center of the object as an (x, y, z) distance in meters from
the origin of the scene, which is located at (0, 0, 0)."/>
        </MetadataFloat>
        <MetadataFloat containerField="value" name="centerOfBuoyancy" value="0 0 0">
          <MetadataString name="appinfo" value="Sets the Center of Buoyancy of the object as an (x, y, z) distance in meters from
the origin of the scene, which is located at (0, 0, 0)."/>
        </MetadataFloat>
        <MetadataFloat containerField="value" name="maximumSpeed" value="0">
          <MetadataString name="appinfo" value="The maximum rated speed for this vehicle in mph or kph."/>
        </MetadataFloat>
        <MetadataFloat containerField="value" name="cruiseSpeed" value="0">
          <MetadataString name="appinfo" value="The published cruise speed for this vehicle in mph or kph."/>
        </MetadataFloat>
        <MetadataFloat containerField="value" name="maximumAltitude" value="0">
          <MetadataString name="appinfo" value="The absolute ceiling for this aircraft in feet or meters."/>
        </MetadataFloat>
        <MetadataFloat containerField="value" name="cruiseAltitude" value="0">
          <MetadataString name="appinfo" value="The cruise ceiling for this aircraft in feet or meters."/>
        </MetadataFloat>
        <MetadataFloat containerField="value" name="maximumDepth" value="0">
          <MetadataString name="appinfo" value="The absolute depth for this submersible in feet or meters."/>
        </MetadataFloat>
        <MetadataFloat containerField="value" name="cruiseDepth" value="0">
          <MetadataString name="appinfo" value="The cruise depth for this submersible in feet or meters."/>
        </MetadataFloat>
        <MetadataFloat containerField="value" name="maximumAcceleration" value="0">
          <MetadataString name="appinfo" value="The ideal maximum acceleration acheivable by this vehicle in feet or meters per
second squared, as in at maximum Power excess for aircraft."/>
        </MetadataFloat>
        <MetadataFloat containerField="value" name="maximumDeceleration" value="0">
          <MetadataString name="appinfo" value="The ideal best braking performance acheivable by this vehicle in feet or meters
per second squared."/>
        </MetadataFloat>
        <MetadataFloat containerField="value" name="minimumTurnRadius" value="0">
          <MetadataString name="appinfo" value="The minimum turning radius for this vehicle in feet or meters, as in at best
cornering speed for aircraft."/>
        </MetadataFloat>
        <MetadataFloat containerField="value" name="maximumTurnRate" value="0">
          <MetadataString name="appinfo" value="The maximum turning rate for this vehicle in degrees per second, as in at best
cornering speed for aircraft."/>
        </MetadataFloat>
      </MetadataSet>
      <MetadataSet containerField="value" name="TacticalConstraints">
        <MetadataFloat containerField="value" name="maximumAirThreatRange" value="0">
          <MetadataString name="appinfo" value="The maximum effective range in miles or kilometers of the longest-range anti-
aircraft weapon on this platform."/>
        </MetadataFloat>
        <MetadataFloat containerField="value" name="maximumSurfaceThreatRange" value="0">
```

```
              <MetadataString name="appinfo" value="The maximum effective range in miles or kilometers of the longest-range anti-
surface weapon on this platform."/>
            </MetadataFloat>
            <MetadataFloat containerField="value" name="maximumSubsurfaceThreatRange" value="0">
              <MetadataString name="appinfo" value="The maximum effective range in miles or kilometers of the longest-range anti-
submarine weapon on this platform."/>
            </MetadataFloat>
            <MetadataFloat containerField="value" name="maximumAirDetectionRange" value="0">
              <MetadataString name="appinfo" value="The maximum detection range of the longest-range air detection sensor on this
platform."/>
            </MetadataFloat>
            <MetadataFloat containerField="value" name="maximumSurfaceDetectionRange" value="0">
              <MetadataString name="appinfo" value="The maximum detection range of the longest-range surface detection sensor on
this platform."/>
            </MetadataFloat>
            <MetadataFloat containerField="value" name="maximumSubsurfaceDetectionRange" value="0">
              <MetadataString name="appinfo" value="The maximum detection range of the longest-range subsurface detection sensor on
this platform."/>
            </MetadataFloat>
          </MetadataSet>
        </MetadataSet>
        <MetadataSet containerField="value" name="AttachmentPointSet">
          <MetadataString name="appinfo" value="A collection of attachment points for this object.  Attachment points are used to
place objects by direct reference to a defined point.  Multiple AttachmentPoints can be defined."/>
          <MetadataSet containerField="value" name="AttachmentPoint">
            <MetadataString name="appinfo" value="A single AttachmentPoint has category, location, and orientation."/>
            <MetadataString containerField="value" name="attachmentCategory" value="WEAPON_MOUNT">
              <MetadataString name="appinfo" value="SENSOR_MOUNT, WEAPON_MOUNT, EMBARKING_POINT"/>
            </MetadataString>
            <MetadataSet containerField="value" name="GlobalVirtualLocation">
              <MetadataString name="appinfo" value="The GlobalVirtualLocation is used to modify the position of the AttachmentPoint
from the origin of the object to which it belongs."/>
              <MetadataFloat containerField="value" name="xValue" value="0.0">
                <MetadataString name="appinfo" value="Distance from the origin along the local x-axis in meters."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="yValue" value="0.0">
                <MetadataString name="appinfo" value="Distance from the origin along the local y-axis in meters."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="zValue" value="0.0">
                <MetadataString name="appinfo" value="Distance from the origin along the local z-axis in meters."/>
              </MetadataFloat>
            </MetadataSet>
            <MetadataSet containerField="value" name="QuaternionOrientation">
              <MetadataString name="appinfo" value="The QuaternionOrientation element is used to modify the orientation of the
AttachmentPoint from the orientation of the object to which it belongs."/>
              <MetadataFloat containerField="value" name="xValue" value="0.0">
                <MetadataString name="appinfo" value="A value from -1.0 to 1.0 based on local coordinate frame."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="yValue" value="0.0">
                <MetadataString name="appinfo" value="A value from -1.0 to 1.0 based on local coordinate frame."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="zValue" value="0.0">
                <MetadataString name="appinfo" value="A value from -1.0 to 1.0 based on local coordinate frame."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="wValue" value="0.0">
                <MetadataString name="appinfo" value="A radian value describing rotation about the arbitrary axis defined using xValue,
yValue, and zValue."/>
              </MetadataFloat>
            </MetadataSet>
          </MetadataSet>
        </MetadataSet>
        <MetadataSet containerField="value" name="X3DArchiveModel">
          <MetadataString name="appinfo" value="This is a placeholder element which ensures the proper validation of autogenerated
SMAL code."/>
        </MetadataSet>
        <MetadataSet containerField="value" name="CurrentConditionParameters">
          <MetadataString name="appinfo" value="This is a placeholder element which ensures the proper validation of autogenerated
SMAL code."/>
        </MetadataSet>
        <MetadataSet containerField="value" name="NetworkedCommunicationParameters">
```

```
      <MetadataSet containerField="value" name="DisConfiguration">
       <MetadataInteger containerField="value" name="entityKind" value="1">
        <MetadataString name="appinfo" value="The DIS enumeration for the overall type of this object (for vehicles this value
should be 1)."/>
       </MetadataInteger>
       <MetadataInteger containerField="value" name="entityDomain" value="1">
        <MetadataString name="appinfo" value="The DIS enumeration for the domain of operations of this object (air, surface, sub-
surface, etc.)."/>
       </MetadataInteger>
       <MetadataInteger containerField="value" name="entityCountry" value="225">
        <MetadataString name="appinfo" value="The DIS enumeration for the country of origin of this object (the value for United
States is 225)."/>
       </MetadataInteger>
       <MetadataInteger containerField="value" name="entityCategory" value="1">
        <MetadataString name="appinfo" value="The DIS enumeration for the type of this vehicle (cruiser or destroyer, tank or
truck, bomber or fighter, etc.)."/>
       </MetadataInteger>
       <MetadataInteger containerField="value" name="entitySubCategory" value="1">
        <MetadataString name="appinfo" value="The DIS enumeration for the class designation of this vehicle (CG-47
Ticonderoga, DDG-51 Arleigh Burke, M1A2, M880, B-52, F-22."/>
       </MetadataInteger>
       <MetadataInteger containerField="value" name="entitySpecific" value="1">
        <MetadataString name="appinfo" value="The DIS enumeration for the specific unit or variant of this object (CG-68, DDG-
77, HMMVW w/TOW package)."/>
       </MetadataInteger>
       <MetadataInteger containerField="value" name="entityExtra" value="1">
        <MetadataString name="appinfo" value="The DIS enumeration for optional equipment or configurations for this vehicle."/>
       </MetadataInteger>
      </MetadataSet>
     </MetadataSet>
    </MetadataSet>
   </MetadataSet>
  </WorldInfo>
 </Scene>
</X3D>
```

APPENDIX D. SAVAGE OBJECT METADATA TEMPLATE

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D profile="Interchange" version="3.1"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
xsd:noNamespaceSchemaLocation="http://www.web3d.org/specifications/x3d-3.1.xsd">
 <head>
  <meta content="SavageObjectMetadataTemplate.x3d" name="title"/>
  <meta content="This scene defines the exemplar template for Savage Static Model metadata, allowing further interoperability via
SMAL constructs. Savage Modeling Analysis Language (SMAL) authoring capabilities for X3D assume proper metadata within a
scene to identify an object properly.  A corresponding native-XML .xsd schema for SMAL will also be developed to facilitate
conversion and use of vehicle metadata." name="description"/>
  <meta content="Travis Rauch, Don Brutzman" name="creator"/>
  <meta content="20 May 2005" name="created"/>
  <meta content="26 March 2006" name="modified"/>
  <meta content="SMAL vehicle metadata" name="subject"/>
  <meta content="under development" name="warning"/>
  <meta content="http://web.nps.navy.mil/~brutzman/Savage/Tools/SMAL/SavageObjectMetadataTemplate.x3d"
name="identifier"/>
  <meta content="X3D-Edit, http://www.web3d.org/x3d/content/README.X3D-Edit.html" name="generator"/>
  <meta content="../../license.html" name="license"/>
 </head>
 <Scene>
  <WorldInfo title="SavageObjectMetadataTemplate">
   <MetadataSet containerField="metadata" name="SMAL">
    <MetadataString containerField="value" name="version" value="1.0">
     <MetadataString name="appinfo" value="This is the version of SMAL employed, not of the model."/>
    </MetadataString>
    <MetadataSet containerField="value" name="StaticModelDefinition">
     <!--Identifying metadata for the current simulation of interest-->
     <MetadataSet containerField="value" name="Classification">
      <MetadataString containerField="value" name="level" value="UNCLASSIFIED">
       <MetadataString name="appinfo" value="UNCLASSIFIED, FOUO, CONFIDENTIAL, SECRET, or TOPSECRET"/>
      </MetadataString>
      <MetadataString containerField="value" name="reference" value="">
       <MetadataString name="appinfo" value="The published source of classified information, if any, contained in the
Metadata."/>
      </MetadataString>
      <MetadataString containerField="value" name="rationale" value="">
       <MetadataString name="appinfo" value="The specific element which contains the information classifying this document."/>
      </MetadataString>
     </MetadataSet>
     <MetadataSet containerField="value" name="IdentificationParameters">
      <MetadataString containerField="value" name="name" value="">
       <MetadataString name="appinfo" value="The plain language name of the object this model represents."/>
      </MetadataString>
     </MetadataSet>
     <MetadataSet containerField="value" name="PhysicalParameters">
      <MetadataSet containerField="value" name="PhysicalConstraints">
       <MetadataFloat containerField="value" name="height" value="0">
        <MetadataString name="appinfo" value="The maximum structural height of the object in meters. This may be used for
clearance checking or other calculations."/>
       </MetadataFloat>
       <MetadataFloat containerField="value" name="width" value="0">
        <MetadataString name="appinfo" value="The maximum width of the object in meters.  This may be used for clearance
checking or other calculations."/>
       </MetadataFloat>
       <MetadataFloat containerField="value" name="length" value="0">
        <MetadataString name="appinfo" value="The maximum structural length of the object in meters. This may be used for
clearance checking or other calculations."/>
       </MetadataFloat>
       <MetadataFloat containerField="value" name="draft" value="0">
        <MetadataString name="appinfo" value="The vertical distance in meters from the deepest point to the waterline of this
object at its stated displacement or gross weight."/>
       </MetadataFloat>
       <MetadataFloat containerField="value" name="grossWeight" value="0">
```

```
        <MetadataString name="appinfo" value="The standard operational weight of the object in pounds or kilograms. This may
be used in any number of physics calculations."/>
       </MetadataFloat>
      </MetadataSet>
      <MetadataSet containerField="value" name="DynamicResponseConstraints">
       <MetadataFloat containerField="value" name="centerOfGravity" value="0 0 0">
        <MetadataString name="appinfo" value="Sets the Center of Gravity of the object as an (x, y, z) distance in meters from the
origin of the scene, which is located at (0, 0, 0)."/>
       </MetadataFloat>
       <MetadataFloat containerField="value" name="aerodynamicCenter" value="0 0 0">
        <MetadataString name="appinfo" value="Sets the Aerodynamic Center of the object as an (x, y, z) distance in meters from
the origin of the scene, which is located at (0, 0, 0)."/>
       </MetadataFloat>
       <MetadataFloat containerField="value" name="centerOfBuoyancy" value="0 0 0">
        <MetadataString name="appinfo" value="Sets the Center of Buoyancy of the object as an (x, y, z) distance in meters from
the origin of the scene, which is located at (0, 0, 0)."/>
       </MetadataFloat>
      </MetadataSet>
     </MetadataSet>
     <MetadataSet containerField="value" name="AttachmentPointSet">
      <MetadataString name="appinfo" value="A collection of attachment points for this object.  Attachment points are used to
place objects by direct reference to a defined point.  Multiple AttachmentPoints can be defined."/>
      <MetadataSet containerField="value" name="AttachmentPoint">
       <MetadataString name="appinfo" value="A single AttachmentPoint has category, location, and orientation."/>
       <MetadataString containerField="value" name="attachmentCategory" value="WEAPON_MOUNT">
        <MetadataString name="appinfo" value="SENSOR_MOUNT, WEAPON_MOUNT, EMBARKING_POINT"/>
       </MetadataString>
       <MetadataSet containerField="value" name="GlobalVirtualLocation">
        <MetadataString name="appinfo" value="The GlobalVirtualLocation is used to modify the position of the AttachmentPoint
from the origin of the object to which it belongs."/>
        <MetadataFloat containerField="value" name="xValue" value="0.0">
         <MetadataString name="appinfo" value="Distance from the origin along the local x-axis in meters."/>
        </MetadataFloat>
        <MetadataFloat containerField="value" name="yValue" value="0.0">
         <MetadataString name="appinfo" value="Distance from the origin along the local y-axis in meters."/>
        </MetadataFloat>
        <MetadataFloat containerField="value" name="zValue" value="0.0">
         <MetadataString name="appinfo" value="Distance from the origin along the local z-axis in meters."/>
        </MetadataFloat>
       </MetadataSet>
       <MetadataSet containerField="value" name="QuaternionOrientation">
        <MetadataString name="appinfo" value="The QuaternionOrientation element is used to modify the orientation of the
AttachmentPoint from the orientation of the object to which it belongs."/>
        <MetadataFloat containerField="value" name="xValue" value="0.0">
         <MetadataString name="appinfo" value="A value from -1.0 to 1.0 based on local coordinate frame."/>
        </MetadataFloat>
        <MetadataFloat containerField="value" name="yValue" value="0.0">
         <MetadataString name="appinfo" value="A value from -1.0 to 1.0 based on local coordinate frame."/>
        </MetadataFloat>
        <MetadataFloat containerField="value" name="zValue" value="0.0">
         <MetadataString name="appinfo" value="A value from -1.0 to 1.0 based on local coordinate frame."/>
        </MetadataFloat>
        <MetadataFloat containerField="value" name="wValue" value="0.0">
         <MetadataString name="appinfo" value="A radian value describing rotation about the arbitrary axis defined using xValue,
yValue, and zValue."/>
        </MetadataFloat>
       </MetadataSet>
      </MetadataSet>
     </MetadataSet>
     <MetadataSet containerField="value" name="X3DArchiveModel">
      <MetadataString name="appinfo" value="This is a placeholder element which ensures the proper validation of autogenerated
SMAL code."/>
     </MetadataSet>
    </MetadataSet>
   </MetadataSet>
  </WorldInfo>
 </Scene>
</X3D>
```

APPENDIX E.   SAVAGE TERRAIN METADATA TEMPLATE

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D profile="Interchange" version="3.1" xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
xsd:noNamespaceSchemaLocation="http://www.web3d.org/specifications/x3d-3.1.xsd">
  <head>
    <meta content="SavageTerrainMetadataTemplate.x3d" name="title"/>
    <meta content="This scene defines the exemplar template for Savage vehicle metadata, allowing further interoperability via SMAL
constructs. Savage Modeling Analysis Language (SMAL) authoring capabilities for X3D assume proper metadata within a scene to
identify an object properly.  A corresponding native-XML .xsd schema for SMAL will also be developed to facilitate conversion and
use of vehicle metadata." name="description"/>
    <meta content="Travis Rauch, Don Brutzman" name="creator"/>
    <meta content="16 February 2006" name="created"/>
    <meta content="26 March 2006" name="modified"/>
    <meta content="SMAL terrain metadata" name="subject"/>
    <meta content="under development" name="warning"/>
    <meta content="http://web.nps.navy.mil/~brutzman/Savage/Tools/SMAL/SavageTerrainMetadataTemplate.x3d"
name="identifier"/>
    <meta content="X3D-Edit, http://www.web3d.org/x3d/content/README.X3D-Edit.html" name="generator"/>
    <meta content="../../license.html" name="license"/>
  </head>
  <Scene>
    <WorldInfo title="SavageTerrainMetadataTemplate">
      <MetadataSet containerField="metadata" name="SMAL">
        <MetadataString containerField="value" name="version" value="1.0">
          <MetadataString name="appinfo" value="This is the version of SMAL employed, not of the model."/>
        </MetadataString>
        <MetadataSet containerField="value" name="TerrainTile">
          <MetadataString containerField="value" name="tileCategory" value="landTerrain">
            <MetadataString name="appinfo" value="landTerrain, bathymetry, or planetarySurface"/>
          </MetadataString>
          <MetadataSet containerField="value" name="Classification">
            <MetadataString containerField="value" name="level" value="UNCLASSIFIED">
              <MetadataString name="appinfo" value="UNCLASSIFIED, FOUO, CONFIDENTIAL, SECRET, or TOPSECRET"/>
            </MetadataString>
            <MetadataString containerField="value" name="reference" value="">
              <MetadataString name="appinfo" value="The published source of classified information, if any, contained in the
Metadata."/>
            </MetadataString>
            <MetadataString containerField="value" name="rationale" value="">
              <MetadataString name="appinfo" value="The specific element which contains the information classifying this document."/>
            </MetadataString>
          </MetadataSet>
          <MetadataSet containerField="value" name="GeoOrigin">
            <MetadataString containerField="value" name="geoCoords" value="N00 0.0 W00 0.0">
              <MetadataString name="appinfo" value="The latitude and longitude of the origin point (0, 0, 0) of the model."/>
            </MetadataString>
            <MetadataString containerField="value" name="geoSystem" value="'GD', 'WE'">
              <MetadataString name="appinfo" value="Two enumerations, one for spatial reference frame, the other for earth ellipsoid e.g.
['GD', 'WE']. See X3D specification 25.2.3."/>
            </MetadataString>
            <MetadataString containerField="value" name="rotateYUp" value="true">
              <MetadataString name="appinfo" value="The axis orientation is positive Y up."/>
            </MetadataString>
          </MetadataSet>
          <MetadataSet containerField="value" name="GeographicExtent">
            <MetadataFloat containerField="value" name="area" value="0">
              <MetadataString name="appinfo" value="Describes the size and shape of the terrain model in two and a half dimensions;
vertical extent and a polygonal shape defined by a minimum of three LatLongCoordinates."/>
            </MetadataFloat>
            <MetadataFloat containerField="value" name="verticalExtent" value="0">
              <MetadataString name="appinfo" value="Vertical depth of the model in meters from the lowest to the highest point on the
model."/>
            </MetadataFloat>
            <MetadataSet containerField="value" name="LatLongCoordinate">
              <MetadataString name="appinfo" value="A latitude and longitude coordinate pair using the WG84 earth ellipsoid."/>
```

```
       <MetadataString containerField="value" name="latitude" value="N00 00.0"/>
       <MetadataString containerField="value" name="longitude" value="W00 00.0"/>
      </MetadataSet>
      <MetadataSet containerField="value" name="LatLongCoordinate">
       <MetadataString name="appinfo" value="A latitude and longitude coordinate pair using the WG84 earth ellipsoid."/>
       <MetadataString containerField="value" name="latitude" value="N00 00.0"/>
       <MetadataString containerField="value" name="longitude" value="W00 00.0"/>
      </MetadataSet>
      <MetadataSet containerField="value" name="LatLongCoordinate">
       <MetadataString name="appinfo" value="A latitude and longitude coordinate pair using the WG84 earth ellipsoid."/>
       <MetadataString containerField="value" name="latitude" value="N00 00.0"/>
       <MetadataString containerField="value" name="longitude" value="W00 00.0"/>
      </MetadataSet>
      <MetadataSet containerField="value" name="LatLongCoordinate">
       <MetadataString name="appinfo" value="A latitude and longitude coordinate pair using the WG84 earth ellipsoid."/>
       <MetadataString containerField="value" name="latitude" value="N00 00.0"/>
       <MetadataString containerField="value" name="longitude" value="W00 00.0"/>
      </MetadataSet>
     </MetadataSet>
     <MetadataSet containerField="value" name="OverlaySet">
      <MetadataString name="appinfo" value="The collection point for all image file locators that are associated with this
terrain."/>
      <MetadataSet containerField="value" name="OverlaySetMap">
       <MetadataString name="appinfo" value="A Map image."/>
       <MetadataSet containerField="value" name="Classification">
        <MetadataString containerField="value" name="level" value="UNCLASSIFIED">
         <MetadataString name="appinfo" value="UNCLASSIFIED, FOUO, CONFIDENTIAL, SECRET, or TOPSECRET"/>
        </MetadataString>
        <MetadataString containerField="value" name="reference" value="">
         <MetadataString name="appinfo" value="The published source of classified information, if any, contained in the
Metadata."/>
        </MetadataString>
        <MetadataString containerField="value" name="rationale" value="">
         <MetadataString name="appinfo" value="The specific element which contains the information classifying this
document."/>
        </MetadataString>
       </MetadataSet>
       <MetadataString containerField="value" name="fileLocationURL" value="http://www.web3d.org/x3d/content/examples/">
        <MetadataString name="appinfo" value="The URL of the image file."/>
       </MetadataString>
       <MetadataString containerField="value" name="centerPointLatitude" value="N00 0.0">
        <MetadataString name="appinfo" value="The latitude of the center point of the image."/>
       </MetadataString>
       <MetadataString containerField="value" name="centerPointLongitiude" value="W00 0.0">
        <MetadataString name="appinfo" value="The longitude of the center point of the image."/>
       </MetadataString>
       <MetadataString containerField="value" name="northBoundLatitude" value="N00 0.0">
        <MetadataString name="appinfo" value="The northernmost latitude found on the image."/>
       </MetadataString>
       <MetadataString containerField="value" name="southBoundLatitude" value="N00 0.0">
        <MetadataString name="appinfo" value="The southernmost latitude found on the image."/>
       </MetadataString>
       <MetadataString containerField="value" name="eastBoundLongitude" value="W00 0.0">
        <MetadataString name="appinfo" value="The easternmost logitude found on the image."/>
       </MetadataString>
       <MetadataString containerField="value" name="westBoundLongitude" value="W00 0.0">
        <MetadataString name="appinfo" value="The westernmost longitude found on the image."/>
       </MetadataString>
      </MetadataSet>
      <MetadataSet containerField="value" name="OverlaySetChart">
       <MetadataString name="appinfo" value="A Chart image."/>
       <MetadataSet containerField="value" name="Classification">
        <MetadataString containerField="value" name="level" value="UNCLASSIFIED">
         <MetadataString name="appinfo" value="UNCLASSIFIED, FOUO, CONFIDENTIAL, SECRET, or TOPSECRET"/>
        </MetadataString>
        <MetadataString containerField="value" name="reference" value="">
         <MetadataString name="appinfo" value="The published source of classified information, if any, contained in the
Metadata."/>
        </MetadataString>
        <MetadataString containerField="value" name="rationale" value="">
```

```
                <MetadataString name="appinfo" value="The specific element which contains the information classifying this
document."/>
               </MetadataString>
             </MetadataSet>
             <MetadataString containerField="value" name="fileLocationURL" value="http://www.web3d.org/x3d/content/examples/">
               <MetadataString name="appinfo" value="The URL of the image file."/>
             </MetadataString>
             <MetadataString containerField="value" name="centerPointLatitude" value="N00 0.0">
               <MetadataString name="appinfo" value="The latitude of the center point of the image."/>
             </MetadataString>
             <MetadataString containerField="value" name="centerPointLongitiude" value="W00 0.0">
               <MetadataString name="appinfo" value="The longitude of the center point of the image."/>
             </MetadataString>
             <MetadataString containerField="value" name="northBoundLatitude" value="N00 0.0">
               <MetadataString name="appinfo" value="The northernmost latitude found on the image."/>
             </MetadataString>
             <MetadataString containerField="value" name="southBoundLatitude" value="N00 0.0">
               <MetadataString name="appinfo" value="The southernmost latitude found on the image."/>
             </MetadataString>
             <MetadataString containerField="value" name="eastBoundLongitude" value="W00 0.0">
               <MetadataString name="appinfo" value="The easternmost logitude found on the image."/>
             </MetadataString>
             <MetadataString containerField="value" name="westBoundLongitude" value="W00 0.0">
               <MetadataString name="appinfo" value="The westernmost longitude found on the image."/>
             </MetadataString>
           </MetadataSet>
           <MetadataSet containerField="value" name="OverlaySetImage">
             <MetadataString name="appinfo" value=""/>
             <MetadataSet containerField="value" name="Classification">
               <MetadataString containerField="value" name="level" value="UNCLASSIFIED">
                 <MetadataString name="appinfo" value="UNCLASSIFIED, FOUO, CONFIDENTIAL, SECRET, or TOPSECRET"/>
               </MetadataString>
               <MetadataString containerField="value" name="reference" value="">
                 <MetadataString name="appinfo" value="The published source of classified information, if any, contained in the
Metadata."/>
               </MetadataString>
               <MetadataString containerField="value" name="rationale" value="">
                 <MetadataString name="appinfo" value="The specific element which contains the information classifying this
document."/>
               </MetadataString>
             </MetadataSet>
             <MetadataString containerField="value" name="fileLocationURL" value="http://www.web3d.org/x3d/content/examples/">
               <MetadataString name="appinfo" value="The URL of the image file."/>
             </MetadataString>
             <MetadataString containerField="value" name="centerPointLatitude" value="N00 0.0">
               <MetadataString name="appinfo" value="The latitude of the center point of the image."/>
             </MetadataString>
             <MetadataString containerField="value" name="centerPointLongitiude" value="W00 0.0">
               <MetadataString name="appinfo" value="The longitude of the center point of the image."/>
             </MetadataString>
             <MetadataString containerField="value" name="northBoundLatitude" value="N00 0.0">
               <MetadataString name="appinfo" value="The northernmost latitude found on the image."/>
             </MetadataString>
             <MetadataString containerField="value" name="southBoundLatitude" value="N00 0.0">
               <MetadataString name="appinfo" value="The southernmost latitude found on the image."/>
             </MetadataString>
             <MetadataString containerField="value" name="eastBoundLongitude" value="W00 0.0">
               <MetadataString name="appinfo" value="The easternmost logitude found on the image."/>
             </MetadataString>
             <MetadataString containerField="value" name="westBoundLongitude" value="W00 0.0">
               <MetadataString name="appinfo" value="The westernmost longitude found on the image."/>
             </MetadataString>
           </MetadataSet>
         </MetadataSet>
       </MetadataSet>
       <MetadataSet containerField="value" name="X3DArchiveModel">
         <MetadataString name="appinfo" value="This is a placeholder element which ensures the proper validation of autogenerated
SMAL code."/>
       </MetadataSet>
     </MetadataSet>
```

```
    </WorldInfo>
  </Scene>
</X3D>
```

APPENDIX F. SAVAGE METADATA EXAMPLES

1. SAVAGE VEHICLE METADATA TEMPLATE

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D profile="Interchange" version="3.1" xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
xsd:noNamespaceSchemaLocation="http://www.web3d.org/specifications/x3d-3.1.xsd">
  <head>
    <meta content="DDG51.x3d" name="title"/>
    <meta content="Arleigh Burke Class Destroyer" name="description"/>
    <meta content="LT Patrick Sullivan" name="creator"/>
    <meta content="11 January 2006" name="created"/>
    <meta content="23 January 2006" name="modified"/>
    <meta content="DDG51, Destroyer, Arleigh Burke" name="subject"/>
    <meta content="http://web.nps.navy.mil/~brutzman/Savage/ShipsMilitary/DDG51.x3d" name="identifier"/>
    <meta content="VizX3d, www.vizx3d.com" name="generator"/>
  </head>
  <Scene>
    <WorldInfo info="http://www.fas.org/man/dod-101/sys/ship/ddg-51.htm" title="DDG51">
      <MetadataSet DEF="DDG51_SVMT" containerField="metadata" name="SMAL">
        <MetadataString containerField="value" name="version" value="1.0">
          <MetadataString name="appinfo" value="This is the version of SMAL employed, not of the model."/>
        </MetadataString>
        <MetadataSet containerField="value" name="EntityDefinition">
          <!--Identifying metadata for the current simulation of interest-->
          <MetadataSet containerField="value" name="Classification">
            <MetadataString containerField="value" name="level" value="UNCLASSIFIED">
              <MetadataString name="appinfo" value="UNCLASSIFIED, FOUO, CONFIDENTIAL, SECRET, or TOPSECRET"/>
            </MetadataString>
            <MetadataString containerField="value" name="reference" value="http://www.fas.org/man/dod-101/sys/ship/ddg-51.htm">
              <MetadataString name="appinfo" value="The published source of classified information, if any, contained in the
Metadata."/>
            </MetadataString>
            <MetadataString containerField="value" name="rationale" value="">
              <MetadataString name="appinfo" value="The specific element which contains the information classifying this document."/>
            </MetadataString>
          </MetadataSet>
          <MetadataSet containerField="value" name="IdentificationParameters">
            <MetadataString containerField="value" name="name" value="DDG-51 Arleigh Burke">
              <MetadataString name="appinfo" value="The plain language name of the vehicle this model represents, i.e. the base class
(DDG-51), or vehicle designation (M1A2)."/>
            </MetadataString>
          </MetadataSet>
          <MetadataSet containerField="value" name="PhysicalParameters">
            <MetadataSet containerField="value" name="PhysicalConstraints">
              <MetadataFloat containerField="value" name="height" value="56.7">
                <MetadataString name="appinfo" value="The maximum structural height of the object in meters. This may be used for
clearance checking or other calculations."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="width" value="20.117">
                <MetadataString name="appinfo" value="The maximum width, beam, or wingspan of the vehicle in meters.  This may be
used for clearance checking or other calculations."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="length" value="153.924">
                <MetadataString name="appinfo" value="The maximum structural length of the object in meters. This may be used for
clearance checking or other calculations."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="grossWeight" value="18384000">
                <MetadataString name="appinfo" value="The standard operational weight of the vehicle in kilograms. This may be used in
any number of physics calculations."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="draft" value="9.488">
                <MetadataString name="appinfo" value="The vertical distance in meters from the deepest point (keel or other structure) to
the waterline of a vehicle at its stated displacement or gross weight."/>
              </MetadataFloat>
            </MetadataSet>
            <MetadataSet containerField="value" name="DynamicResponseConstraints">
              <MetadataFloat containerField="value" name="maximumSpeed" value="57">
                <MetadataString name="appinfo" value="The maximum rated speed for this vehicle in kph."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="cruiseSpeed" value="42.6">
                <MetadataString name="appinfo" value="The published cruise speed for this vehicle in kph."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="centerOfGravity" value="0 0 0">
                <MetadataString name="appinfo" value="Sets the Center of Gravity of the object as an (x, y, z) distance in meters from the
physical center of the object, which is located at (0, 0, 0)."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="centerOfBuoyancy" value="0 0 0">
                <MetadataString name="appinfo" value="Sets the Center of Buoyancy of the object as an (x, y, z) distance in meters from
the physical center of the object, which is ocated at (0, 0, 0)."/>
              </MetadataFloat>
            </MetadataSet>
            <MetadataSet containerField="value" name="TacticalConstraints">
              <MetadataFloat containerField="value" name="maximumAirThreatRange" value="115">
                <MetadataString name="appinfo" value="The maximum effective range in kilometers of the longest-range anti-aircraft
weapon on this platform."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="maximumSurfaceThreatRange" value="69">
                <MetadataString name="appinfo" value="The maximum effective range in kilometers of the longest-range anti-surface
weapon on this platform."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="maximumSubsurfaceThreatRange" value="6.5">
                <MetadataString name="appinfo" value="The maximum effective range in kilometers of the longest-range anti-submarine
weapon on this platform."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="maximumAirDetectionRange" value="250">
                <MetadataString name="appinfo" value="The maximum detection range of the longest-range air detection sensor on this
platform."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="maximumSurfaceDetectionRange" value="14.7">
                <MetadataString name="appinfo" value="The maximum detection range of the longest-range surface detection sensor on
this platform."/>
              </MetadataFloat>
            </MetadataSet>
          </MetadataSet>
          <MetadataSet containerField="value" name="X3DArchiveModel">
            <MetadataString name="appinfo" value="This is a placeholder element which ensures the proper validation of autogenerated
SMAL code."/>
          </MetadataSet>
          <MetadataSet containerField="value" name="NetworkedCommunicationParameters">
            <MetadataSet containerField="value" name="DisConfiguration">
              <MetadataInteger containerField="value" name="entityKind" value="1">
                <MetadataString name="appinfo" value="The DIS enumeration for the overall type of this object (for vehicles this value
should be &apos;1&apos;)."/>
              </MetadataInteger>
              <MetadataInteger containerField="value" name="entityDomain" value="3">
                <MetadataString name="appinfo" value="The DIS enumeration for the domain of operations of this object (air, surface, sub-
surface, etc.)."/>
              </MetadataInteger>
              <MetadataInteger containerField="value" name="entityCountry" value="225">
                <MetadataString name="appinfo" value="The DIS enumeration for the country of origin of this object (the value for United
States is &apos;225&apos;)."/>
              </MetadataInteger>
              <MetadataInteger containerField="value" name="entityCategory" value="4">
                <MetadataString name="appinfo" value="The DIS enumeration for the type of this vehicle (cruiser or destroyer, tank or
truck, bomber or fighter, etc.)."/>
              </MetadataInteger>
              <MetadataInteger containerField="value" name="entitySubCategory" value="1">
                <MetadataString name="appinfo" value="The DIS enumeration for the class designation of this vehicle (CG-47
Ticonderoga, DDG-51 Arleigh Burke, M1A2, M880, B-52, F-22."/>
              </MetadataInteger>
              <MetadataInteger containerField="value" name="entitySpecific" value="1">
                <MetadataString name="appinfo" value="The DIS enumeration for the specific unit or variant of this object (CG-68, DDG-
77, HMMVW w/TOW package)."/>
              </MetadataInteger>
            </MetadataSet>
```

```
          </MetadataSet>
        </MetadataSet>
      </MetadataSet>
    </WorldInfo>
    <NavigationInfo avatarSize=" .25 1.6 .75 " headlight="true" speed="1" type="&quot;EXAMINE&quot; &quot;ANY&quot;"/>
    <Transform>
      <Group DEF="DDG51_ClassDestroyer">
        <!--Model Scaled for DDG51 Dimensions. Length 153.924 meters, Width 20.11679 meters-->
        <Transform DEF="DDG51" scale="6.1 6.1 6.1" translation="7.486 0 0">
          <Group>
            <Transform DEF="Ship">
              <Group DEF="Import_Base0">
                <Transform DEF="dad_text110_copy113">
                  <Shape DEF="text110_copy113">
                    <Appearance>
                      <Material DEF="numWhite_mat" ambientIntensity="1.000" diffuseColor="1 1 1" shininess="0.000"
specularColor=".001 .001 .001"/>
                    </Appearance>
…
```

2. SAVAGE OBJECT METADATA TEMPLATE

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN"
                   "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D profile="Interchange" version="3.1"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
xsd:noNamespaceSchemaLocation="http://www.web3d.org/specifications/x3d-3.1.xsd">
  <head>
    <meta content="BoxMine.x3d" name="title"/>
    <meta content="A box-shaped mine" name="description"/>
    <meta content="LCDR Travis Rauch" name="creator"/>
    <meta content="13 March 2006" name="created"/>
    <meta content="13 March 2006" name="modified"/>
    <meta content="mine, imaginary, munition, anti-ship" name="subject"/>
    <meta content="http://web.nps.navy.mil/~brutzman/Savage/Imaginary/BoxMine.x3d" name="identifier"/>
    <meta content="X3D-Edit, http://www.web3d.org/x3d/content/README.X3D-Edit.html" name="generator"/>
  </head>
  <Scene>
    <WorldInfo title="SavageObjectMetadataTemplate">
      <MetadataSet containerField="metadata" name="SMAL">
        <MetadataString containerField="value" name="version" value="1.0">
          <MetadataString name="appinfo" value="This is the version of SMAL employed, not of the model."/>
        </MetadataString>
        <MetadataSet containerField="value" name="EntityDefinition">
          <!--Identifying metadata for the current simulation of interest-->
          <MetadataSet containerField="value" name="Classification">
            <MetadataString containerField="value" name="level" value="UNCLASSIFIED">
              <MetadataString name="appinfo" value="UNCLASSIFIED, FOUO, CONFIDENTIAL, SECRET, or TOPSECRET"/>
            </MetadataString>
          </MetadataSet>
          <MetadataSet containerField="value" name="IdentificationParameters">
            <MetadataString containerField="value" name="name" value="Box Mine">
              <MetadataString name="appinfo" value="The plain language name of the vehicle this model represents, i.e. the base class
(DDG-51), or vehicle designation (M1A2)."/>
            </MetadataString>
          </MetadataSet>
          <MetadataSet containerField="value" name="PhysicalParameters">
            <MetadataSet containerField="value" name="PhysicalConstraints">
              <MetadataFloat containerField="value" name="height" value="1">
                <MetadataString name="appinfo" value="The maximum structural height of the object in meters. This may be used for
clearance checking or other calculations."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="width" value="1">
                <MetadataString name="appinfo" value="The maximum width, beam, or wingspan of the vehicle in meters.  This may be
used for clearance checking or other calculations."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="length" value="1">
                <MetadataString name="appinfo" value="The maximum structural length of the object in meters. This may be used for
clearance checking or other calculations."/>
```

```xml
              </MetadataFloat>
              <MetadataFloat containerField="value" name="draft" value=".75">
                <MetadataString name="appinfo" value="The vertical distance in meters from the deepest point (keel or other structure) to
the waterline of a vehicle at its stated displacement or gross weight."/>
              </MetadataFloat>
            </MetadataSet>
            <MetadataSet containerField="value" name="DynamicResponseConstraints">
              <MetadataFloat containerField="value" name="centerOfGravity" value="0 0 0">
                <MetadataString name="appinfo" value="Sets the Center of Gravity of the object as an (x, y, z) distance in meters from the
physical center of the object, which is located at (0, 0, 0)."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="centerOfBuoyancy" value="0 .25 0">
                <MetadataString name="appinfo" value="Sets the Center of Buoyancy of the object as an (x, y, z) distance in meters from
the physical center of the object, which is ocated at (0, 0, 0)."/>
              </MetadataFloat>
             </MetadataSet>
            <MetadataSet containerField="value" name="TacticalConstraints">
              <MetadataFloat containerField="value" name="maximumAirThreatRange" value="0">
                <MetadataString name="appinfo" value="The maximum effective range in miles or kilometers of the longest-range anti-
aircraft weapon on this platform."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="maximumSurfaceThreatRange" value="0">
                <MetadataString name="appinfo" value="The maximum effective range in miles or kilometers of the longest-range anti-
surface weapon on this platform."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="maximumSubsurfaceThreatRange" value="0">
                <MetadataString name="appinfo" value="The maximum effective range in miles or kilometers of the longest-range anti-
submarine weapon on this platform."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="maximumAirDetectionRange" value="0">
                <MetadataString name="appinfo" value="The maximum detection range of the longest-range air detection sensor on this
platform."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="maximumSurfaceDetectionRange" value="0">
                <MetadataString name="appinfo" value="The maximum detection range of the longest-range surface detection sensor on
this platform."/>
              </MetadataFloat>
              <MetadataFloat containerField="value" name="maximumSubsurfaceDetectionRange" value="0">
                <MetadataString name="appinfo" value="The maximum detection range of the longest-range subsurface detection sensor on
this platform."/>
              </MetadataFloat>
            </MetadataSet>
          </MetadataSet>
          <MetadataSet containerField="value" name="X3DArchiveModel">
            <MetadataString name="appinfo" value="This is a placeholder element which ensures the proper validation of autogenerated
SMAL code."/>
          </MetadataSet>
        </MetadataSet>
      </MetadataSet>
    </WorldInfo>
    <Transform>
      <Shape>
        <Appearance>
          <Material diffuseColor="0.8 0.8 0.8"/>
        </Appearance>
        <Box size="1 1 1"/>
      </Shape>
    </Transform>
  </Scene>
</X3D>
```

APPENDIX G. SAVAGE MODELING ANALYSIS LANGUAGE DOCUMENTATION

TABLE OF CONTENTS

OverlayImageDescriptor
OverlaySet
OverlaySetChart
OverlaySetImagery
OverlaySetMap
Parameter
PhysicalConstraints
PhysicalParameters
PointRadiusCircular
PrecipitationConditionSet
QuaternionDirection
QuaternionOrientation
RelativeVirtualLocation
SeaState
Simulation
SimulationAgent
SMAL
StaticModelDefinition
StaticModelSet
TacticalConstraints
TemperatureCondition
TemperatureConditionSet
TerrainTile
TerrainTileSet
TimeParameters
Track
VectorDirection
WaterConditionSet
WaterCurrentConditionAtDepth
WaterTemperatureCondition
WindConditionAtLevel
WindConditionSet
World
X3DArchiveModel
XYZOrientation

SCHEMA DOCUMENT PROPERTIES

Target Namespace	None
Version	1.0
Element and Attribute Namespaces	• Global element and attribute declarations belong to this schema's target namespace. • By default, local element declarations belong to this schema's target namespace. • By default, local attribute declarations have no namespace.

Schema Composition	• This schema includes components from the following schema document(s): o SavageModelingAnalysisLanguageEnumerations1.0.xsd o SavageModelingAnalysisLanguageDataTypes1.0.xsd
Application Data	This language is intended to provide a lightweight, human- and machinereadable collection of data which can be used to create or recreate a 3D scene using SAVAGE library elements and any event-driven source of data.

Declared Namespaces

Prefix	Namespace
xml	http://www.w3.org/XML/1998/namespace
xs	http://www.w3.org/2001/XMLSchema

Schema Component Representation

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" version="1.0">
   <xs:include schemaLocation="SavageModelingAnalysisLanguageEnumerations1.0.xsd"/>
   <xs:include schemaLocation="SavageModelingAnalysisLanguageDataTypes1.0.xsd"/>
...
</xs:schema>
```

GLOBAL DECLARATIONS

attribute **areaUnits**

type	**areaUnitEnumeration**	
properties	default	squareKilometers
facets	enumeration	squareFeet
	enumeration	squareMeters
	enumeration	squareMiles
	enumeration	squareKilometers
	enumeration	squareNauticalMiles
source	`<xs:attribute name="areaUnits" type="areaUnitEnumeration" default="squareKilometers"/>`	

attribute **duration**

type	**xs:time**	
properties	default	01:00:00
source	`<xs:attribute name="duration" type="xs:time" default="01:00:00"/>`	

attribute **linearUnits**

type	**linearUnitEnumeration**	
properties	default	meters
facets	enumeration	feet
	enumeration	meters

enumeration	miles
enumeration	kilometers
enumeration	nauticalMiles

source `<xs:attribute name="linearUnits" type="linearUnitEnumeration" default="meters"/>`

attribute **speedUnits**

type **speedUnitEnumeration**

properties

default	kilometersPerHour

facets

enumeration	feetPerSecond
enumeration	metersPerSecond
enumeration	milesPerHour
enumeration	kilometersPerHour
enumeration	knots

source `<xs:attribute name="speedUnits" type="speedUnitEnumeration" default="kilometersPerHour"/>`

attribute **startTime**

type **xs:time**

properties

default	00:00:00

source `<xs:attribute name="startTime" type="xs:time" default="00:00:00"/>`

attribute **temperatureUnits**

type **temperatureUnitEnumeration**

properties

default	Celsius

facets

enumeration	Celsius
enumeration	Fahrenheit

source `<xs:attribute name="temperatureUnits" type="temperatureUnitEnumeration" default="Celsius"/>`

attribute **unitSystem**

type **unitSystemEnumeration**

properties

default	Metric

facets

enumeration	English
enumeration	Metric

source `<xs:attribute name="unitSystem" type="unitSystemEnumeration" default="Metric"/>`

attribute **weightUnits**

type **weightUnitEnumeration**

properties

default	pounds

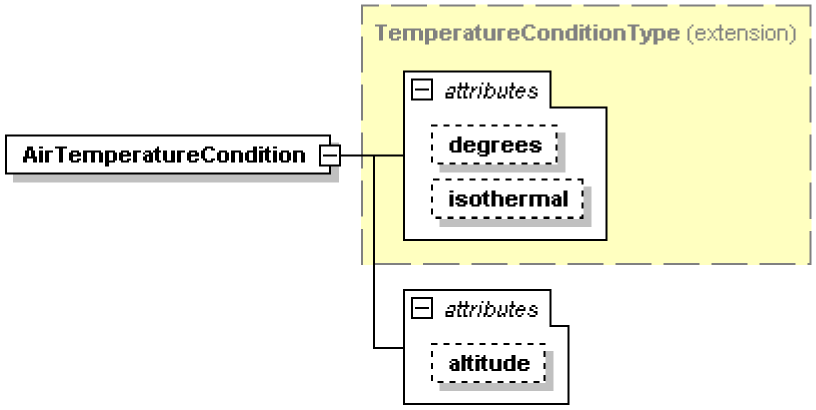
facets

enumeration	pounds
enumeration	kilograms
enumeration	shortTons
enumeration	metricTons

source `<xs:attribute name="weightUnits" type="weightUnitEnumeration" default="pounds"/>`

element **AirTemperatureCondition**

diagram



type extension of **TemperatureConditionType**

properties

content	complex
substGrp	TemperatureCondition

attributes

Name	Type	Use	Default	Fixed	Annotation
degrees	**xs:float**		15		
isothermal	**xs:boolean**		false		
altitude	**nonNegativeFloat**		0		

annotation

appInfo
Holds the air temperature value and the base altitude at which this temperature begins. Temperature is assumed to decrease as altitude increases from the base altitude unless isothermal is set to true.

source

```
<xs:element name="AirTemperatureCondition" substitutionGroup="TemperatureCondition">
 <xs:annotation>
  <xs:appinfo>Holds the air temperature value and the base altitude at which this temperature begins. Temperature is
assumed to decrease as altitude increases from the base altitude unless isothermal is set to true.</xs:appinfo>
 </xs:annotation>
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="TemperatureConditionType">
    <xs:attribute name="altitude" type="nonNegativeFloat" default="0"/>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
```

attribute **AirTemperatureCondition/@altitude**

type **nonNegativeFloat**

properties

isRef	0
default	0

facets

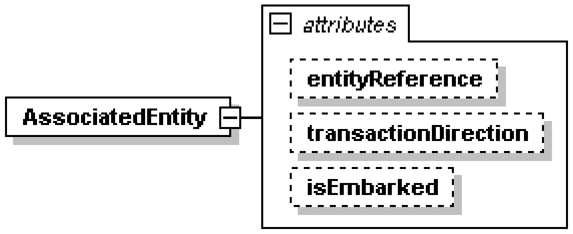
minInclusive	0.0

source

```
<xs:attribute name="altitude" type="nonNegativeFloat" default="0"/>
```

element **AssociatedEntity**

diagram



type extension of **FullyExtensibleSMALElementType**

properties

content	complex

used by

element	**Association**

attributes

Name	Type	Use	Default	Fixed	Annotation
entityReference	**xs:IDREF**				
transactionDirection	**transactionDirectionEnumeration**				
isEmbarked	**xs:boolean**				

source

```
<xs:element name="AssociatedEntity">
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="FullyExtensibleSMALElementType">
    <xs:attribute name="entityReference" type="xs:IDREF"/>
    <xs:attribute name="transactionDirection" type="transactionDirectionEnumeration"/>
    <xs:attribute name="isEmbarked" type="xs:boolean"/>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
```

attribute **AssociatedEntity/@entityReference**

type **xs:IDREF**

properties

isRef	0

source

```
<xs:attribute name="entityReference" type="xs:IDREF"/>
```

attribute **AssociatedEntity/@transactionDirection**

type **transactionDirectionEnumeration**

properties

isRef	0

facets

enumeration	SEND
enumeration	RECIEVE
enumeration	BIDIRECTIONAL

source

```
<xs:attribute name="transactionDirection" type="transactionDirectionEnumeration"/>
```

attribute **AssociatedEntity/@isEmbarked**

type **xs:boolean**

properties

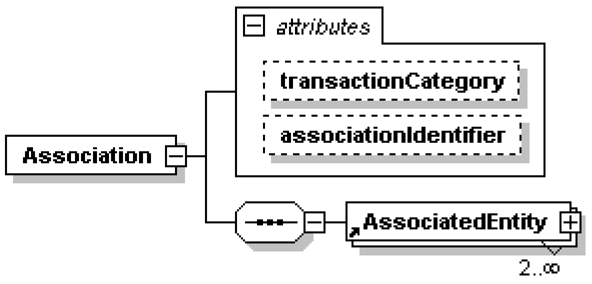
isRef	0

source

```
<xs:attribute name="isEmbarked" type="xs:boolean"/>
```

element **Association**

diagram



type extension of **FullyExtensibleSMALElementType**

properties	content	complex

children **AssociatedEntity**

used by	element	**AssociationSet**

attributes

Name	Type	Use	Default	Fixed	Annotation
transactionCategory	**transactionCategoryEnumeration**				
associationIdentifier	**xs:ID**				

annotation appInfo
Allows entities to be associated with each other by command relationship and/or as an embarking entity (onboard or attached to an entity).

source
```
<xs:element name="Association">
  <xs:annotation>
    <xs:appinfo>Allows entities to be associated with each other by command relationship and/or as an embarking entity
(onboard or attached to an entity).</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:sequence>
          <xs:element ref="AssociatedEntity" minOccurs="2" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="transactionCategory" type="transactionCategoryEnumeration"/>
        <xs:attribute name="associationIdentifier" type="xs:ID"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

attribute **Association/@transactionCategory**

type **transactionCategoryEnumeration**

properties	isRef	0
facets	enumeration	COMMAND
	enumeration	SUPPORT
	enumeration	COMMUNICATION

source
```
<xs:attribute name="transactionCategory" type="transactionCategoryEnumeration"/>
```

attribute **Association/@associationIdentifier**

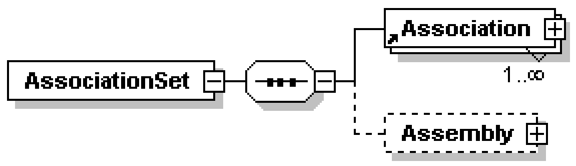
type **xs:ID**

properties	isRef	0

source
```
<xs:attribute name="associationIdentifier" type="xs:ID"/>
```

element **AssociationSet**

diagram



type extension of **FullyExtensibleSMALElementType**

properties	content	complex

children **Association** **Assembly**

used by	element	**Simulation**

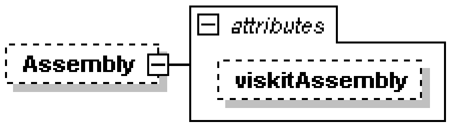
annotation appInfo
Collects all Association elements for a Simulation.

source

```
<xs:element name="AssociationSet">
  <xs:annotation>
    <xs:appinfo>Collects all Association elements for a Simulation.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:sequence>
          <xs:element ref="Association" maxOccurs="unbounded"/>
          <xs:element name="Assembly" minOccurs="0">
            <xs:complexType>
              <xs:attribute name="viskitAssembly" type="xs:anyURI"/>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

element **AssociationSet/Assembly**

diagram



properties

isRef	0
minOcc	0
maxOcc	1
content	complex

attributes

Name	Type	Use	Default	Fixed	Annotation
viskitAssembly	**xs:anyURI**				

source

```
<xs:element name="Assembly" minOccurs="0">
  <xs:complexType>
    <xs:attribute name="viskitAssembly" type="xs:anyURI"/>
  </xs:complexType>
</xs:element>
```

attribute **AssociationSet/Assembly/@viskitAssembly**

type **xs:anyURI**

properties

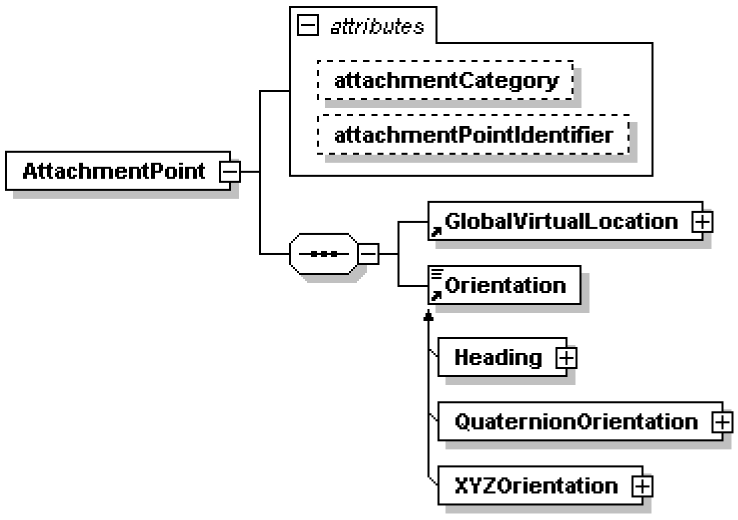
isRef	0

source

```
<xs:attribute name="viskitAssembly" type="xs:anyURI"/>
```

element **AttachmentPoint**

diagram



type	extension of **FullyExtensibleSMALElementType**
properties	content complex
children	**GlobalVirtualLocation** **Orientation**
used by	element **AttachmentPointSet**

attributes

Name	Type	Use	Default	Fixed	Annotation
attachmentCategory	**attachmentCategoryEnumeration**				
attachmentPointIdentifier	**xs:ID**				

annotation

appInfo
Allows the attachment to this unit of independently operating entities by type and location. This AttachmentPoint can be uniquely identified using the attachmentPointIdentifier attribute.

source

```
<xs:element name="AttachmentPoint">
  <xs:annotation>
    <xs:appinfo>Allows the attachment to this unit of independently operating entities by type and location. This
AttachmentPoint can be uniquely identified using the attachmentPointIdentifier attribute.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:sequence>
          <xs:annotation>
            <xs:appinfo>The GlobalVirtualLocation is used to modify the position of the AttachmentPoint from the origin of the
object to which it belongs.</xs:appinfo>
            <xs:appinfo>The Orientation element is used to modify the orientation of the AttachmentPoint from the orientation of
the object to which it belongs.</xs:appinfo>
          </xs:annotation>
          <xs:element ref="GlobalVirtualLocation"/>
          <xs:element ref="Orientation"/>
        </xs:sequence>
        <xs:attribute name="attachmentCategory" type="attachmentCategoryEnumeration"/>
        <xs:attribute name="attachmentPointIdentifier" type="xs:ID"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

attribute **AttachmentPoint/@attachmentCategory**

type	**attachmentCategoryEnumeration**
properties	isRef 0

facets	enumeration	WEAPON_MOUNT
	enumeration	SENSOR_MOUNT
	enumeration	EMBARKING_POINT

source `<xs:attribute name="attachmentCategory" type="attachmentCategoryEnumeration"/>`

attribute **AttachmentPoint/@attachmentPointIdentifier**

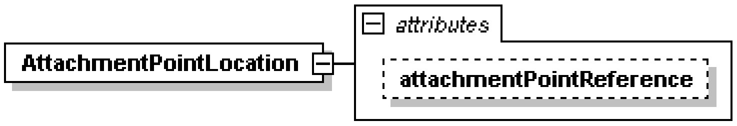
type **xs:ID**

properties isRef 0

source `<xs:attribute name="attachmentPointIdentifier" type="xs:ID"/>`

element **AttachmentPointLocation**

diagram



type extension of **AttributeExtensibleSMALElementType**

properties content complex; substGrp Location

attributes

Name	Type	Use	Default	Fixed	Annotation
attachmentPointReference	**xs:IDREF**				

annotation appInfo
Object AttachmentPoint reference-based location element. The @attachmentPointReference value is used to reference the @attachmentPointIdentifier value from an AttachmentPoint element.

source
```
<xs:element name="AttachmentPointLocation" substitutionGroup="Location">
 <xs:annotation>
  <xs:appinfo>Object AttachmentPoint reference-based location element. The @attachmentPointReference value is used to
reference the @attachmentPointIdentifier value from an AttachmentPoint element.</xs:appinfo>
 </xs:annotation>
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="AttributeExtensibleSMALElementType">
    <xs:attribute name="attachmentPointReference" type="xs:IDREF"/>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
```

attribute **AttachmentPointLocation/@attachmentPointReference**

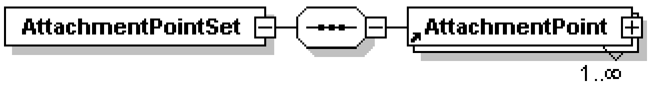
type **xs:IDREF**

properties isRef 0

source `<xs:attribute name="attachmentPointReference" type="xs:IDREF"/>`

element **AttachmentPointSet**

diagram



type extension of **FullyExtensibleSMALElementType**

properties content complex

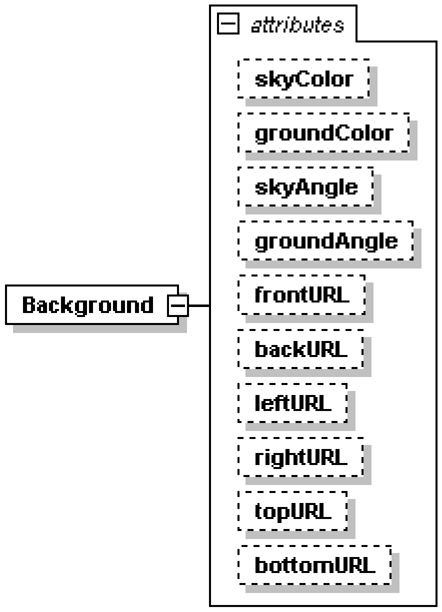
children **AttachmentPoint**

annotation appInfo
Collects the set of locations at which other objects can be attached to this one.

source
```
<xs:element name="AttachmentPointSet">
 <xs:annotation>
  <xs:appinfo>Collects the set of locations at which other objects can be attached
                                to this one.</xs:appinfo>
 </xs:annotation>
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="FullyExtensibleSMALElementType">
    <xs:sequence>
     <xs:element ref="AttachmentPoint" maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
```

element **Background**

diagram



type extension of **FullyExtensibleSMALElementType**

properties content complex

used by element **EnvironmentalCondition**

attributes

Name	Type	Use	Default	Fixed	Annotation
skyColor	**xs:string**				
groundColor	**xs:string**				
skyAngle	**xs:float**				
groundAngle	**xs:float**				
frontURL	**xs:anyURI**				
backURL	**xs:anyURI**				
leftURL	**xs:anyURI**				
rightURL	**xs:anyURI**				
topURL	**xs:anyURI**				
bottomURL	**xs:anyURI**				

annotation appInfo
Holds information for filling the X3D Background node which is used to set up a skybox or simple background colors.

source
```
<xs:element name="Background">
 <xs:annotation>
  <xs:appinfo>Holds information for filling the X3D Background node which is used to set up a skybox or simple background
```

```
colors.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:attribute name="skyColor" type="xs:string"/>
        <xs:attribute name="groundColor" type="xs:string"/>
        <xs:attribute name="skyAngle" type="xs:float"/>
        <xs:attribute name="groundAngle" type="xs:float"/>
        <xs:attribute name="frontURL" type="xs:anyURI"/>
        <xs:attribute name="backURL" type="xs:anyURI"/>
        <xs:attribute name="leftURL" type="xs:anyURI"/>
        <xs:attribute name="rightURL" type="xs:anyURI"/>
        <xs:attribute name="topURL" type="xs:anyURI"/>
        <xs:attribute name="bottomURL" type="xs:anyURI"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

attribute **Background/@skyColor**

type	**xs:string**
properties	isRef 0
source	`<xs:attribute name="skyColor" type="xs:string"/>`

attribute **Background/@groundColor**

type	**xs:string**
properties	isRef 0
source	`<xs:attribute name="groundColor" type="xs:string"/>`

attribute **Background/@skyAngle**

type	**xs:float**
properties	isRef 0
source	`<xs:attribute name="skyAngle" type="xs:float"/>`

attribute **Background/@groundAngle**

type	**xs:float**
properties	isRef 0
source	`<xs:attribute name="groundAngle" type="xs:float"/>`

attribute **Background/@frontURL**

type	**xs:anyURI**
properties	isRef 0
source	`<xs:attribute name="frontURL" type="xs:anyURI"/>`

attribute **Background/@backURL**

type	**xs:anyURI**

properties	isRef 0
source	`<xs:attribute name="backURL" type="xs:anyURI"/>`

attribute **Background/@leftURL**

type	**xs:anyURI**
properties	isRef 0
source	`<xs:attribute name="leftURL" type="xs:anyURI"/>`

attribute **Background/@rightURL**

type	**xs:anyURI**
properties	isRef 0
source	`<xs:attribute name="rightURL" type="xs:anyURI"/>`

attribute **Background/@topURL**

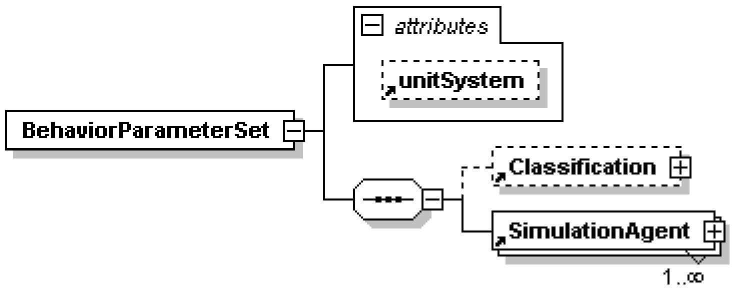
type	**xs:anyURI**
properties	isRef 0
source	`<xs:attribute name="topURL" type="xs:anyURI"/>`

attribute **Background/@bottomURL**

type	**xs:anyURI**
properties	isRef 0
source	`<xs:attribute name="bottomURL" type="xs:anyURI"/>`

element **BehaviorParameterSet**

diagram



type extension of **FullyExtensibleSMALElementType**

properties content complex

children **Classification** **SimulationAgent**

used by element **EntityDefinition**

attributes

Name	Type	Use	Default	Fixed	Annotation
unitSystem			Metric		

annotation appInfo
Data required for proper agent-based behavior simulation. NOTE: Knots and nautical miles are not used in parameters or modeling, only Metric or English units are used in SMAL files.

source
```
<xs:element name="BehaviorParameterSet">
  <xs:annotation>
    <xs:appinfo>Data required for proper agent-based behavior simulation. NOTE: Knots
                                        and nautical miles are not used in parameters or modeling, only Metric or English
                                        units are used in SMAL files.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:sequence>
          <xs:element ref="Classification" minOccurs="0"/>
          <xs:element ref="SimulationAgent" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute ref="unitSystem"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

element **CartesianCoordinate**

diagram

properties	substGrp	MapCoordinate

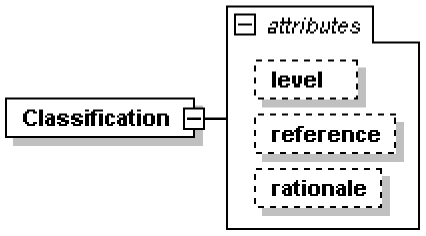
annotation

appInfo
Cartesian coordinate system coordinate-value location element.

source
```
<xs:element name="CartesianCoordinate" substitutionGroup="MapCoordinate">
  <xs:annotation>
    <xs:appinfo>Cartesian coordinate system coordinate-value location element.</xs:appinfo>
  </xs:annotation>
</xs:element>
```

element **Classification**

diagram



type extension of **FullyExtensibleSMALElementType**

properties	content	complex

used by

elements	**BehaviorParameterSet** **GeoOrigin** **head** **NetworkChannel** **OverlaySet** **PhysicalParameters** **Simulation** **TerrainTile** **X3DArchiveModel**
complexTypes	**ObjectDefinitionType** **OverlayImageDescriptorType**

attributes

Name	Type	Use	Default	Fixed	Annotation
level	**classificationLevelEnumeration**	optional	UNCLASSIFIED		
reference	**xs:string**	optional			
rationale	**xs:string**	optional			

source
```
<xs:element name="Classification">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:attribute name="level" type="classificationLevelEnumeration" use="optional" default="UNCLASSIFIED"/>
        <xs:attribute name="reference" type="xs:string" use="optional"/>
        <xs:attribute name="rationale" type="xs:string" use="optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

attribute **Classification/@level**

type	**classificationLevelEnumeration**
properties	isRef 0 default UNCLASSIFIED use optional
facets	enumeration UNCLASSIFIED enumeration FOUO enumeration CONFIDENTIAL enumeration SECRET enumeration TOPSECRET
source	`<xs:attribute name="level" type="classificationLevelEnumeration" use="optional" default="UNCLASSIFIED"/>`

attribute **Classification/@reference**

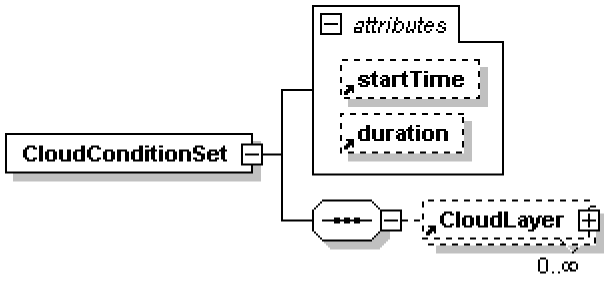
type	**xs:string**
properties	isRef 0 use optional
source	`<xs:attribute name="reference" type="xs:string" use="optional"/>`

attribute **Classification/@rationale**

type	**xs:string**
properties	isRef 0 use optional
source	`<xs:attribute name="rationale" type="xs:string" use="optional"/>`

element **CloudConditionSet**

diagram



type extension of **FullyExtensibleSMALElementType**

properties content complex

children **CloudLayer**

used by element **EnvironmentalCondition**

attributes

Name	Type	Use	Default	Fixed	Annotation
startTime			00:00:00		
duration			01:00:00		

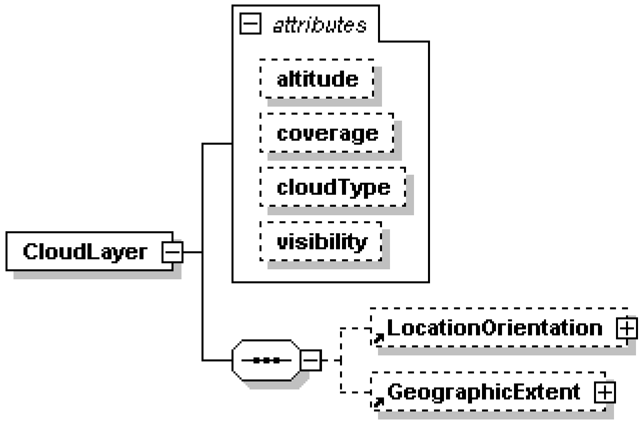
annotation appInfo
Collects the cloud condition elements for a single EnvironmentalCondition element.

source
```
<xs:element name="CloudConditionSet">
  <xs:annotation>
    <xs:appinfo>Collects the cloud condition elements for a single EnvironmentalCondition element.</xs:appinfo>
  </xs:annotation>
```

```
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="FullyExtensibleSMALElementType">
          <xs:sequence>
            <xs:element ref="CloudLayer" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute ref="startTime"/>
          <xs:attribute ref="duration"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
```

element **CloudLayer**

diagram



properties	content	complex
children	**LocationOrientation** **GeographicExtent**	
used by	element	**CloudConditionSet**

attributes

Name	Type	Use	Default	Fixed	Annotation
altitude	**xs:float**				
coverage	**cloudCoverageEnumeration**		CLR		
cloudType	**cloudTypeEnumeration**		CUMULUS		
visibility	**nonNegativeFloat**				

annotation
appInfo
Clouds are defined in layers and can optionally be localized by using the LocationOrientation and GeographicExtent elements. Altitude represents the cloud base for that layer and its depth as defined in the GeographicExtent element extends upward. Cloud conditions by altitude are considered persistent in a given Environmental condition. Changes may be made using multiple CloudConditionSets.

source

```
<xs:element name="CloudLayer">
  <xs:annotation>
    <xs:appinfo>Clouds are defined in layers and can optionally be localized by using the LocationOrientation and GeographicExtent elements. Altitude represents the cloud base for that layer and its depth as defined in the GeographicExtent element extends upward.  Cloud conditions by altitude are considered persistent in a given Environmental condition.  Changes may be made using multiple CloudConditionSets.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="LocationOrientation" minOccurs="0"/>
      <xs:element ref="GeographicExtent" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="altitude" type="xs:float"/>
    <xs:attribute name="coverage" type="cloudCoverageEnumeration" default="CLR"/>
    <xs:attribute name="cloudType" type="cloudTypeEnumeration" default="CUMULUS"/>
    <xs:attribute name="visibility" type="nonNegativeFloat"/>
  </xs:complexType>
</xs:element>
```

attribute **CloudLayer/@altitude**

type	**xs:float**
properties	isRef 0
source	`<xs:attribute name="altitude" type="xs:float"/>`

attribute **CloudLayer/@coverage**

type	**cloudCoverageEnumeration**
properties	isRef 0 default CLR
facets	enumeration OVC enumeration BKN enumeration SCT enumeration CLR
source	`<xs:attribute name="coverage" type="cloudCoverageEnumeration" default="CLR"/>`

attribute **CloudLayer/@cloudType**

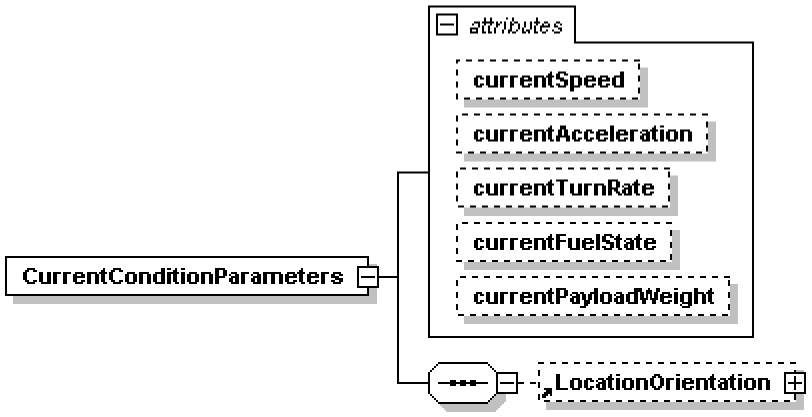
type	**cloudTypeEnumeration**
properties	isRef 0 default CUMULUS
facets	enumeration CUMULUS enumeration STRATUS enumeration CIRRUS
source	`<xs:attribute name="cloudType" type="cloudTypeEnumeration" default="CUMULUS"/>`

attribute **CloudLayer/@visibility**

type	**nonNegativeFloat**
properties	isRef 0
facets	minInclusive 0.0
source	`<xs:attribute name="visibility" type="nonNegativeFloat"/>`

element **CurrentConditionParameters**

diagram



type	extension of **FullyExtensibleSMALElementType**
properties	content complex
children	**LocationOrientation**

used by element **EntityDefinition**

attributes

Name	Type	Use	Default	Fixed	Annotation
currentSpeed	**xs:float**		0		appInfo Assumed to be along the current vehicle track.
currentAcceleration	**threeTuplePattern**	optional			appInfo Linear acceleration along global-aligned coordinate axes.
currentTurnRate	**xs:float**	optional			appInfo Degrees of heading change per second around global-aligned coordinate axes.
currentFuelState	**xs:float**	optional			appInfo The volumetric amount of usable fuel aboard the vehicle.
currentPayloadWeight	**xs:float**	optional			appInfo All weight not attributable to the vehicle or internal fuel.

annotation appInfo
Parameters used to describe the location, orientation, and state of an entity.

source
```
<xs:element name="CurrentConditionParameters">
 <xs:annotation>
  <xs:appinfo>Parameters used to describe the location, orientation, and state of an entity.</xs:appinfo>
 </xs:annotation>
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="FullyExtensibleSMALElementType">
    <xs:sequence>
     <xs:element ref="LocationOrientation" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="currentSpeed" type="xs:float" default="0">
     <xs:annotation>
      <xs:appinfo>Assumed to be along the current vehicle track.</xs:appinfo>
     </xs:annotation>
    </xs:attribute>
    <xs:attribute name="currentAcceleration" type="threeTuplePattern" use="optional">
     <xs:annotation>
      <xs:appinfo>Linear acceleration along global-aligned coordinate axes.</xs:appinfo>
     </xs:annotation>
    </xs:attribute>
    <xs:attribute name="currentTurnRate" type="xs:float" use="optional">
     <xs:annotation>
      <xs:appinfo>Degrees of heading change per second around global-aligned coordinate axes.</xs:appinfo>
     </xs:annotation>
    </xs:attribute>
    <xs:attribute name="currentFuelState" type="xs:float" use="optional">
     <xs:annotation>
      <xs:appinfo>The volumetric amount of usable fuel aboard the vehicle.</xs:appinfo>
     </xs:annotation>
    </xs:attribute>
    <xs:attribute name="currentPayloadWeight" type="xs:float" use="optional">
     <xs:annotation>
      <xs:appinfo>All weight not attributable to the vehicle or internal fuel.</xs:appinfo>
     </xs:annotation>
    </xs:attribute>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
```

attribute **CurrentConditionParameters/@currentSpeed**

type **xs:float**

properties

isRef	0
default	0

annotation appInfo
Assumed to be along the current vehicle track.

source
```
<xs:attribute name="currentSpeed" type="xs:float" default="0">
  <xs:annotation>
    <xs:appinfo>Assumed to be along the current vehicle track.</xs:appinfo>
  </xs:annotation>
</xs:attribute>
```

attribute **CurrentConditionParameters/@currentAcceleration**

type	**threeTuplePattern**
properties	isRef 0 use optional
facets	pattern (\+|\-)?([0-9]*.[0-9]*)((\+|\-)?([0-9]*.[0-9]*)){2}
annotation	appInfo Linear acceleration along global-aligned coordinate axes.

source
```
<xs:attribute name="currentAcceleration" type="threeTuplePattern" use="optional">
  <xs:annotation>
    <xs:appinfo>Linear acceleration along global-aligned coordinate axes.</xs:appinfo>
  </xs:annotation>
</xs:attribute>
```

attribute **CurrentConditionParameters/@currentTurnRate**

type	**xs:float**
properties	isRef 0 use optional
annotation	appInfo Degrees of heading change per second around global-aligned coordinate axes.

source
```
<xs:attribute name="currentTurnRate" type="xs:float" use="optional">
  <xs:annotation>
    <xs:appinfo>Degrees of heading change per second around global-aligned coordinate axes.</xs:appinfo>
  </xs:annotation>
</xs:attribute>
```

attribute **CurrentConditionParameters/@currentFuelState**

type	**xs:float**
properties	isRef 0 use optional
annotation	appInfo The volumetric amount of usable fuel aboard the vehicle.

source
```
<xs:attribute name="currentFuelState" type="xs:float" use="optional">
  <xs:annotation>
    <xs:appinfo>The volumetric amount of usable fuel aboard the vehicle.</xs:appinfo>
  </xs:annotation>
</xs:attribute>
```

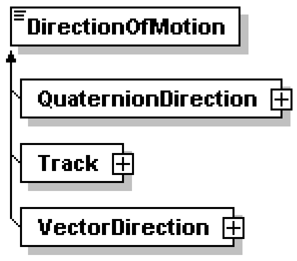
attribute **CurrentConditionParameters/@currentPayloadWeight**

type	**xs:float**
properties	isRef 0 use optional
annotation	appInfo All weight not attributable to the vehicle or internal fuel.

source
```
<xs:attribute name="currentPayloadWeight" type="xs:float" use="optional">
  <xs:annotation>
    <xs:appinfo>All weight not attributable to the vehicle or internal fuel.</xs:appinfo>
  </xs:annotation>
</xs:attribute>
```

element **DirectionOfMotion**

diagram

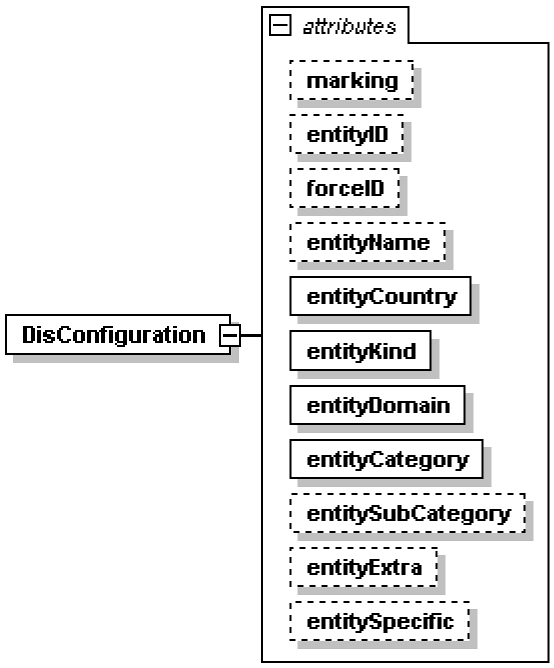


properties	abstract	true
used by	element	[LocationOrientation]()

source
```
<xs:element name="DirectionOfMotion" abstract="true"/>
```

element **DisConfiguration**

diagram



type extension of **FullyExtensibleSMALElementType**

properties	content	complex
used by	element	**NetworkedCommunicationParameterSet**

attributes

Name	Type	Use	Default	Fixed	Annotation
marking	**xs:string**	optional			
entityID	**xs:nonNegativeInteger**	optional			
forceID	**xs:nonNegativeInteger**	optional			
entityName	**xs:NMTOKEN**	optional			
entityCountry	**xs:nonNegativeInteger**	required			
entityKind	**xs:nonNegativeInteger**	required			
entityDomain	**xs:nonNegativeInteger**	required			
entityCategory	**xs:nonNegativeInteger**	required			
entitySubCategory	**xs:nonNegativeInteger**	optional			
entityExtra	**xs:nonNegativeInteger**	optional			
entitySpecific	**xs:nonNegativeInteger**	optional			

source
```
<xs:element name="DisConfiguration">
 <xs:complexType>
  <xs:complexContent>
```

```
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:attribute name="marking" type="xs:string" use="optional"/>
        <xs:attribute name="entityID" type="xs:nonNegativeInteger" use="optional"/>
        <xs:attribute name="forceID" type="xs:nonNegativeInteger" use="optional"/>
        <xs:attribute name="entityName" type="xs:NMTOKEN" use="optional"/>
        <xs:attribute name="entityCountry" type="xs:nonNegativeInteger" use="required"/>
        <xs:attribute name="entityKind" type="xs:nonNegativeInteger" use="required"/>
        <xs:attribute name="entityDomain" type="xs:nonNegativeInteger" use="required"/>
        <xs:attribute name="entityCategory" type="xs:nonNegativeInteger" use="required"/>
        <xs:attribute name="entitySubCategory" type="xs:nonNegativeInteger" use="optional"/>
        <xs:attribute name="entityExtra" type="xs:nonNegativeInteger" use="optional"/>
        <xs:attribute name="entitySpecific" type="xs:nonNegativeInteger" use="optional"/>
      </xs:extension>
      <!-- marking is limited to 11 characters -->
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

attribute **DisConfiguration/@marking**

type	**xs:string**
properties	isRef 0 use optional
source	`<xs:attribute name="marking" type="xs:string" use="optional"/>`

attribute **DisConfiguration/@entityID**

type	**xs:nonNegativeInteger**
properties	isRef 0 use optional
source	`<xs:attribute name="entityID" type="xs:nonNegativeInteger" use="optional"/>`

attribute **DisConfiguration/@forceID**

type	**xs:nonNegativeInteger**
properties	isRef 0 use optional
source	`<xs:attribute name="forceID" type="xs:nonNegativeInteger" use="optional"/>`

attribute **DisConfiguration/@entityName**

type	**xs:NMTOKEN**
properties	isRef 0 use optional
source	`<xs:attribute name="entityName" type="xs:NMTOKEN" use="optional"/>`

attribute **DisConfiguration/@entityCountry**

type	**xs:nonNegativeInteger**
properties	isRef 0 use required
source	`<xs:attribute name="entityCountry" type="xs:nonNegativeInteger" use="required"/>`

attribute **DisConfiguration/@entityKind**

type	**xs:nonNegativeInteger**

properties	isRef	0
	use	required

source `<xs:attribute name="entityKind" type="xs:nonNegativeInteger" use="required"/>`

attribute **DisConfiguration/@entityDomain**

type **xs:nonNegativeInteger**

properties	isRef	0
	use	required

source `<xs:attribute name="entityDomain" type="xs:nonNegativeInteger" use="required"/>`

attribute **DisConfiguration/@entityCategory**

type **xs:nonNegativeInteger**

properties	isRef	0
	use	required

source `<xs:attribute name="entityCategory" type="xs:nonNegativeInteger" use="required"/>`

attribute **DisConfiguration/@entitySubCategory**

type **xs:nonNegativeInteger**

properties	isRef	0
	use	optional

source `<xs:attribute name="entitySubCategory" type="xs:nonNegativeInteger" use="optional"/>`

attribute **DisConfiguration/@entityExtra**

type **xs:nonNegativeInteger**

properties	isRef	0
	use	optional

source `<xs:attribute name="entityExtra" type="xs:nonNegativeInteger" use="optional"/>`

attribute **DisConfiguration/@entitySpecific**

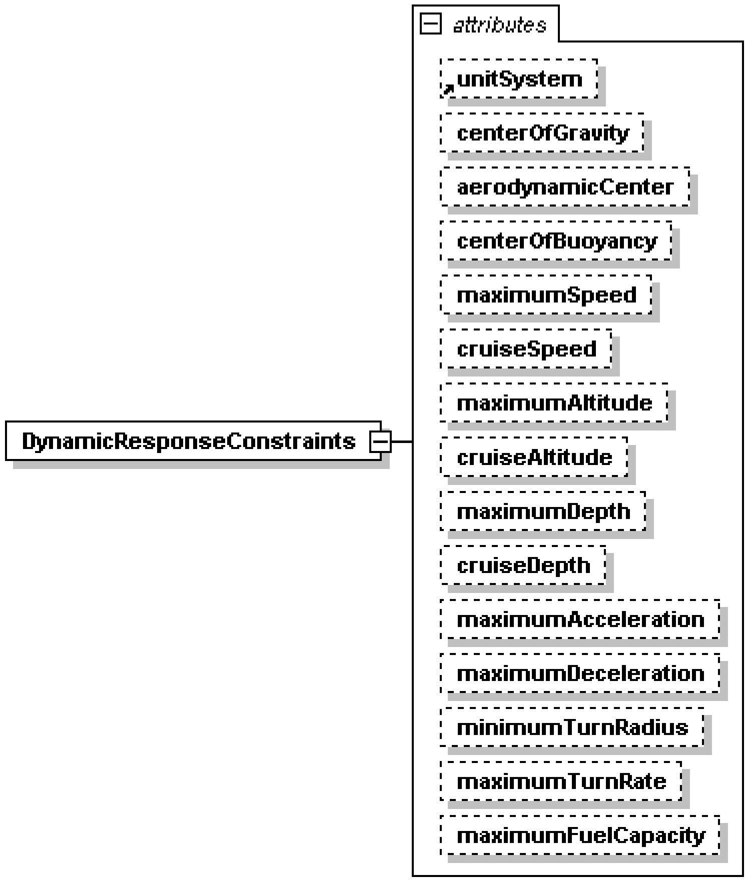
type **xs:nonNegativeInteger**

properties	isRef	0
	use	optional

source `<xs:attribute name="entitySpecific" type="xs:nonNegativeInteger" use="optional"/>`

element **DynamicResponseConstraints**

diagram



type extension of **FullyExtensibleSMALElementType**

properties content complex

used by element **PhysicalParameters**

attributes

Name	Type	Use	Default	Fixed	Annotation
unitSystem			Metric		
centerOfGravity	**threeTuplePattern**		0 0 0		
aerodynamicCenter	**threeTuplePattern**	optional			
centerOfBuoyancy	**threeTuplePattern**	optional			
maximumSpeed	**xs:float**				
cruiseSpeed	**xs:float**				
maximumAltitude	**xs:float**	optional			
cruiseAltitude	**xs:float**	optional			
maximumDepth	**xs:float**	optional			
cruiseDepth	**xs:float**	optional			
maximumAcceleration	**xs:float**	optional			
maximumDeceleration	**xs:float**	optional			
minimumTurnRadius	**xs:float**	optional			appInfo Tactical Radius for ships or Best Cornering Speed turn radius for aircraft.
maximumTurnRate	**xs:float**	optional			
maximumFuelCapacity	**xs:float**	optional			

annotation appInfo
Performance characteristics including speed, acceleration, and turn rate and radius.

source
```
<xs:element name="DynamicResponseConstraints">
 <xs:annotation>
  <xs:appinfo>Performance characteristics including speed, acceleration, and turn rate and radius.</xs:appinfo>
 </xs:annotation>
 <xs:complexType>
```

```
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:attribute ref="unitSystem"/>
        <xs:attribute name="centerOfGravity" type="threeTuplePattern" default="0 0 0"/>
        <xs:attribute name="aerodynamicCenter" type="threeTuplePattern" use="optional"/>
        <xs:attribute name="centerOfBuoyancy" type="threeTuplePattern" use="optional"/>
        <xs:attribute name="maximumSpeed" type="xs:float"/>
        <xs:attribute name="cruiseSpeed" type="xs:float"/>
        <xs:attribute name="maximumAltitude" type="xs:float" use="optional"/>
        <xs:attribute name="cruiseAltitude" type="xs:float" use="optional"/>
        <xs:attribute name="maximumDepth" type="xs:float" use="optional"/>
        <xs:attribute name="cruiseDepth" type="xs:float" use="optional"/>
        <xs:attribute name="maximumAcceleration" type="xs:float" use="optional"/>
        <xs:attribute name="maximumDeceleration" type="xs:float" use="optional"/>
        <xs:attribute name="minimumTurnRadius" type="xs:float" use="optional">
          <xs:annotation>
            <xs:appinfo>Tactical Radius for ships or Best Cornering Speed turn radius for aircraft.</xs:appinfo>
          </xs:annotation>
        </xs:attribute>
        <xs:attribute name="maximumTurnRate" type="xs:float" use="optional"/>
        <xs:attribute name="maximumFuelCapacity" type="xs:float" use="optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

attribute **DynamicResponseConstraints/@centerOfGravity**

type	**threeTuplePattern**
properties	isRef 0 default 0 0 0
facets	pattern (\+|\-)?([0-9]*.[0-9]*)((\+|\-)?([0-9]*.[0-9]*)){2}
source	`<xs:attribute name="centerOfGravity" type="threeTuplePattern" default="0 0 0"/>`

attribute **DynamicResponseConstraints/@aerodynamicCenter**

type	**threeTuplePattern**
properties	isRef 0 use optional
facets	pattern (\+|\-)?([0-9]*.[0-9]*)((\+|\-)?([0-9]*.[0-9]*)){2}
source	`<xs:attribute name="aerodynamicCenter" type="threeTuplePattern" use="optional"/>`

attribute **DynamicResponseConstraints/@centerOfBuoyancy**

type	**threeTuplePattern**
properties	isRef 0 use optional
facets	pattern (\+|\-)?([0-9]*.[0-9]*)((\+|\-)?([0-9]*.[0-9]*)){2}
source	`<xs:attribute name="centerOfBuoyancy" type="threeTuplePattern" use="optional"/>`

attribute **DynamicResponseConstraints/@maximumSpeed**

type	**xs:float**
properties	isRef 0
source	`<xs:attribute name="maximumSpeed" type="xs:float"/>`

attribute **DynamicResponseConstraints/@cruiseSpeed**

type	**xs:float**
properties	isRef 0
source	`<xs:attribute name="cruiseSpeed" type="xs:float"/>`

attribute **DynamicResponseConstraints/@maximumAltitude**

type	**xs:float**
properties	isRef 0 use optional
source	`<xs:attribute name="maximumAltitude" type="xs:float" use="optional"/>`

attribute **DynamicResponseConstraints/@cruiseAltitude**

type	**xs:float**
properties	isRef 0 use optional
source	`<xs:attribute name="cruiseAltitude" type="xs:float" use="optional"/>`

attribute **DynamicResponseConstraints/@maximumDepth**

type	**xs:float**
properties	isRef 0 use optional
source	`<xs:attribute name="maximumDepth" type="xs:float" use="optional"/>`

attribute **DynamicResponseConstraints/@cruiseDepth**

type	**xs:float**
properties	isRef 0 use optional
source	`<xs:attribute name="cruiseDepth" type="xs:float" use="optional"/>`

attribute **DynamicResponseConstraints/@maximumAcceleration**

type	**xs:float**
properties	isRef 0 use optional
source	`<xs:attribute name="maximumAcceleration" type="xs:float" use="optional"/>`

attribute **DynamicResponseConstraints/@maximumDeceleration**

type	**xs:float**
properties	isRef 0 use optional
source	`<xs:attribute name="maximumDeceleration" type="xs:float" use="optional"/>`

attribute **DynamicResponseConstraints/@minimumTurnRadius**

type	**xs:float**
properties	isRef 0

	use optional
annotation	appInfo Tactical Radius for ships or Best Cornering Speed turn radius for aircraft.
source	<xs:attribute name="minimumTurnRadius" type="xs:float" use="optional"> <xs:annotation> <xs:appinfo>Tactical Radius for ships or Best Cornering Speed turn radius for aircraft.</xs:appinfo> </xs:annotation> </xs:attribute>

attribute **DynamicResponseConstraints/@maximumTurnRate**

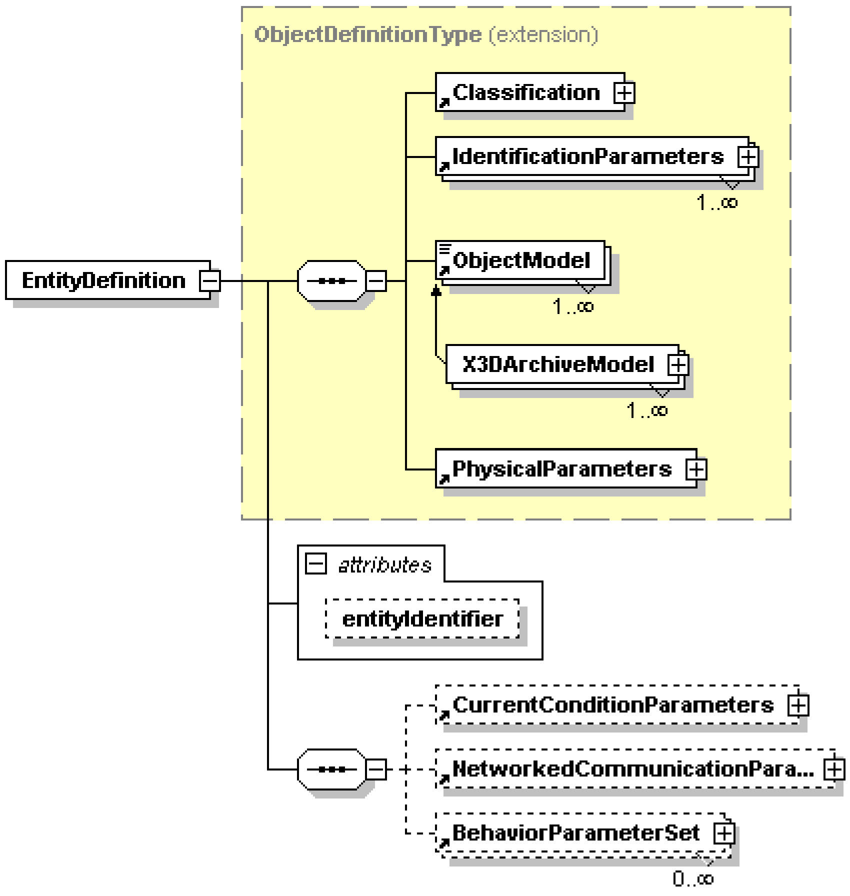
type	**xs:float**
properties	isRef 0 use optional
source	<xs:attribute name="maximumTurnRate" type="xs:float" use="optional"/>

attribute **DynamicResponseConstraints/@maximumFuelCapacity**

type	**xs:float**
properties	isRef 0 use optional
source	<xs:attribute name="maximumFuelCapacity" type="xs:float" use="optional"/>

element **EntityDefinition**

diagram



type extension of **ObjectDefinitionType**

properties	content	complex

children **Classification IdentificationParameters ObjectModel PhysicalParameters CurrentConditionParameters NetworkedCommunicationParameterSet BehaviorParameterSet**

used by	elements	**EntitySet SMAL**

attributes

Name	Type	Use	Default	Fixed	Annotation
entityIdentifier	**xs:ID**				

annotation appInfo
Defines a single entity down to network identification. Also includes default physical parameters and behavioral data for that particular entity type

source
```
<xs:element name="EntityDefinition">
  <xs:annotation>
    <xs:appinfo>Defines a single entity down to network identification. Also includes default physical parameters and
behavioral data for that particular entity type</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="ObjectDefinitionType">
        <xs:sequence>
          <xs:element ref="CurrentConditionParameters" minOccurs="0"/>
          <xs:element ref="NetworkedCommunicationParameterSet" minOccurs="0"/>
          <xs:element ref="BehaviorParameterSet" minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
              <xs:appinfo>Multiple BehaviorParameters sets allow for sensitivity analysis using different parameter settings or to
provide unclassified and classified parameters for the purpose of creating unclassified products from otherwise classified
datasets.</xs:appinfo>
            </xs:annotation>
          </xs:element>
        </xs:sequence>
        <xs:attribute name="entityIdentifier" type="xs:ID"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

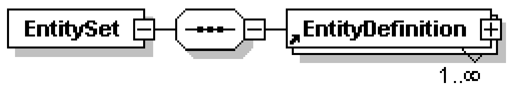
attribute **EntityDefinition/@entityIdentifier**

type **xs:ID**

properties	isRef	0

source
```
<xs:attribute name="entityIdentifier" type="xs:ID"/>
```

element **EntitySet**

diagram


type extension of **FullyExtensibleSMALElementType**

properties	content	complex

children **EntityDefinition**

used by	element	**Simulation**

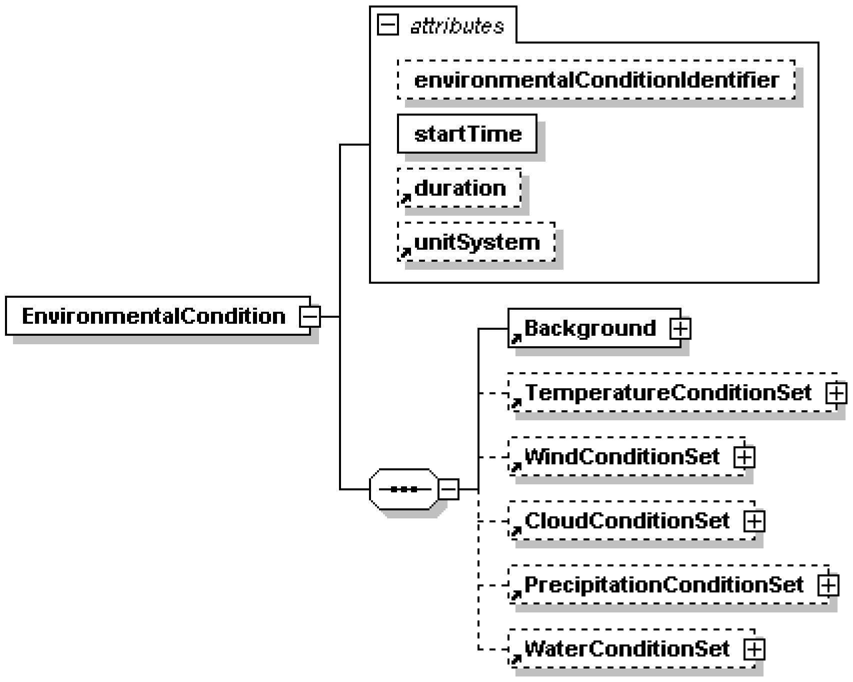
annotation appInfo
Contains the active entities which operate in the virtual environment.

source
```
<xs:element name="EntitySet">
  <xs:annotation>
    <xs:appinfo>Contains the active entities which operate in the virtual environment.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
```

```
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:sequence>
          <xs:element ref="EntityDefinition" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

element **EnvironmentalCondition**

diagram



type	extension of **FullyExtensibleSMALElementType**
properties	content complex
children	**Background TemperatureConditionSet WindConditionSet CloudConditionSet PrecipitationConditionSet WaterConditionSet**
used by	element **EnvironmentalConditionSet**

attributes

Name	Type	Use	Default	Fixed	Annotation
environmentalConditionIdentifier	**xs:ID**				
startTime	**xs:time**	required			
duration		optional	01:00:00		
unitSystem		optional	Metric		

annotation

appInfo
Contains a list of weather conditions such as cloud layers, temperature, precipitation, winds, and water currents, as well as the start time and duration of this set of conditions. Multiple sets of conditions can be established to create complex weather patterns.

source

```
<xs:element name="EnvironmentalCondition">
  <xs:annotation>
    <xs:appinfo>Contains a list of weather conditions such as cloud layers, temperature, precipitation, winds, and water
currents, as well as the start time and duration of this set of conditions. Multiple sets of conditions can be established to
create complex weather patterns.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:sequence>
          <xs:element ref="Background"/>
          <xs:element ref="TemperatureConditionSet" minOccurs="0"/>
          <xs:element ref="WindConditionSet" minOccurs="0"/>
          <xs:element ref="CloudConditionSet" minOccurs="0"/>
          <xs:element ref="PrecipitationConditionSet" minOccurs="0"/>
          <xs:element ref="WaterConditionSet" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="environmentalConditionIdentifier" type="xs:ID"/>
        <xs:attribute name="startTime" type="xs:time" use="required"/>
        <xs:attribute ref="duration" use="optional"/>
        <xs:attribute ref="unitSystem" use="optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

attribute **EnvironmentalCondition/@environmentalConditionIdentifier**

type **xs:ID**

properties isRef 0

source
```
<xs:attribute name="environmentalConditionIdentifier" type="xs:ID"/>
```

attribute **EnvironmentalCondition/@startTime**

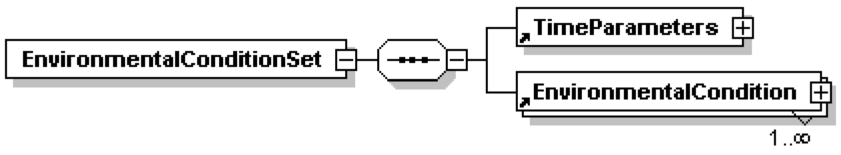
type **xs:time**

properties isRef 0
use required

source
```
<xs:attribute name="startTime" type="xs:time" use="required"/>
```

element **EnvironmentalConditionSet**

diagram



type extension of **FullyExtensibleSMALElementType**

properties content complex

children **TimeParameters** **EnvironmentalCondition**

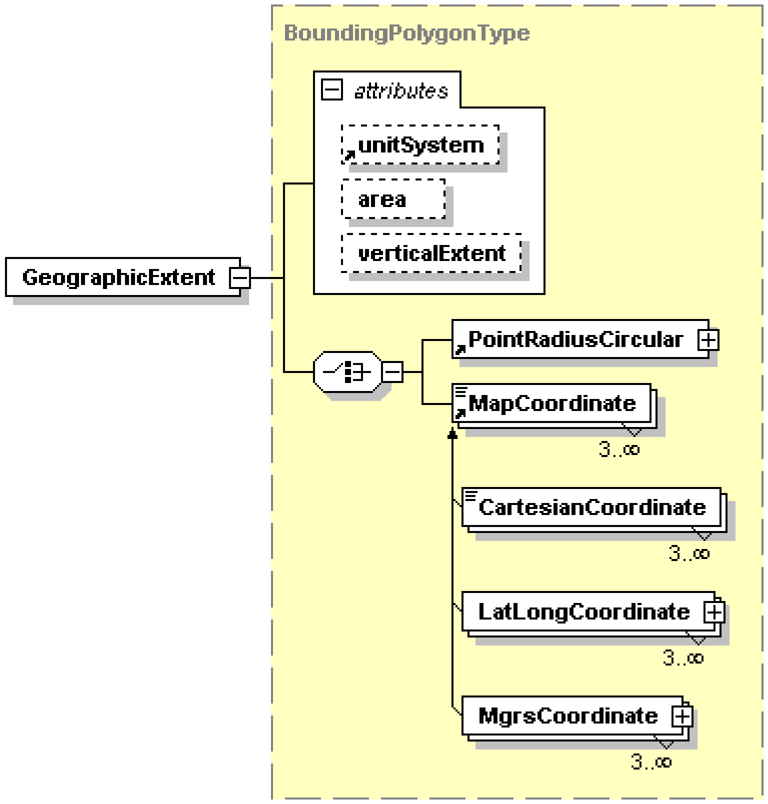
used by element **Simulation**

annotation appInfo
Holds information about changeable world conditions such as time of day and weather.

source
```
<xs:element name="EnvironmentalConditionSet">
  <xs:annotation>
    <xs:appinfo>Holds information about changeable world conditions such as time of day and weather.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:sequence>
          <xs:element ref="TimeParameters"/>
          <xs:element ref="EnvironmentalCondition" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

element **GeographicExtent**

diagram



type **BoundingPolygonType**

properties content complex

children **PointRadiusCircular** **MapCoordinate**

used by elements **CloudLayer** **LocalPrecipitation** **TerrainTile**

attributes

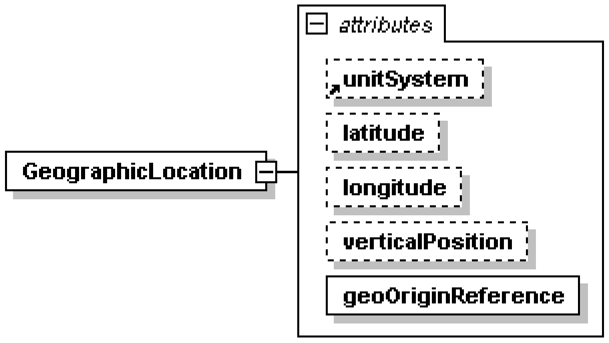
Name	Type	Use	Default	Fixed	Annotation
unitSystem			Metric		
area	**xs:float**	optional	0.0		
verticalExtent	**xs:float**	optional	0.0		

annotation appInfo
A BoundingPolygonType element used to describe the boundaries of a geographic element such as a TerrainTile.

source
```
<xs:element name="GeographicExtent" type="BoundingPolygonType">
  <xs:annotation>
    <xs:appinfo>A BoundingPolygonType element used to describe the boundaries of a geographic element such as a
TerrainTile.</xs:appinfo>
  </xs:annotation>
</xs:element>
```

element **GeographicLocation**

diagram



type extension of **FullyExtensibleSMALElementType**

properties

content	complex
substGrp	Location

attributes

Name	Type	Use	Default	Fixed	Annotation
unitSystem			Metric		
latitude	**latitudeOrdinatePattern**				
longitude	**longitudeOrdinatePattern**				
verticalPosition	**xs:float**				
geoOriginReference	**xs:IDREF**	required			

annotation appInfo
Global geographic reference based geographic location element. References a GeoOrigin element.

source
```
<xs:element name="GeographicLocation" substitutionGroup="Location">
  <xs:annotation>
    <xs:appinfo>Global geographic reference based geographic location element. References a GeoOrigin
element.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:attribute ref="unitSystem"/>
        <xs:attribute name="latitude" type="latitudeOrdinatePattern"/>
        <xs:attribute name="longitude" type="longitudeOrdinatePattern"/>
        <xs:attribute name="verticalPosition" type="xs:float"/>
        <xs:attribute name="geoOriginReference" type="xs:IDREF" use="required"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

attribute **GeographicLocation/@latitude**

type **latitudeOrdinatePattern**

properties isRef 0

facets pattern (((N|n)|(S|s))(([0-8]?[0-9])|90) ([0-5]?[0-9])?(\.[0-9]*))

source
```
<xs:attribute name="latitude" type="latitudeOrdinatePattern"/>
```

attribute **GeographicLocation/@longitude**

type **longitudeOrdinatePattern**

properties isRef 0

facets pattern (((E|e)|(W|w))([0-9]?[0-9]|1[0-8][0-9]) ([0-5]?[0-9])?(\.[0-9]*))

source
```
<xs:attribute name="longitude" type="longitudeOrdinatePattern"/>
```

attribute **GeographicLocation/@verticalPosition**

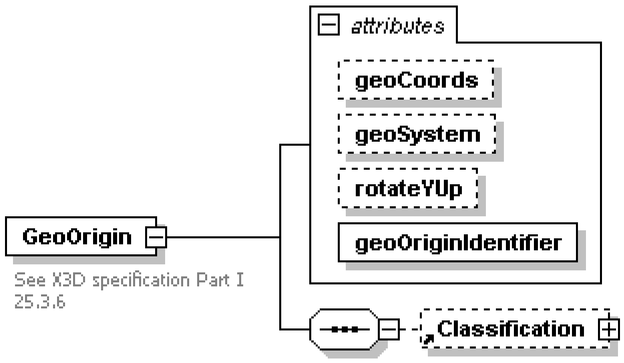
type	**xs:float**
properties	isRef 0
source	`<xs:attribute name="verticalPosition" type="xs:float"/>`

attribute **GeographicLocation/@geoOriginReference**

type	**xs:IDREF**
properties	isRef 0 use required
source	`<xs:attribute name="geoOriginReference" type="xs:IDREF" use="required"/>`

element **GeoOrigin**

diagram



type extension of **FullyExtensibleSMALElementType**

properties content complex

children **Classification**

used by elements **TerrainTile** **World**

attributes

Name	Type	Use	Default	Fixed	Annotation
geoCoords	**xs:string**	optional	N00 0.0 W00 0.0		
geoSystem	**xs:string**	optional	'GD', 'WE'		appInfo Two enumerations, one for spatial reference frames, the other for earth ellipsoid i.e. ["GD", "WE"] See X3D specification 25.2.3.
rotateYUp	**xs:boolean**	optional	true		
geoOriginIdentifier	**xs:ID**	required			

annotation

appInfo
The GeoOrigin node defines an absolute geographic location and an implicit local coordinate frame against which geometry is referenced. This node is used to translate from geographical coordinates into a local Cartesian coordinate system which can be managed by the browser.
documentation
See X3D specification Part I 25.3.6

source

```
<xs:element name="GeoOrigin">
 <xs:annotation>
  <xs:appinfo>The GeoOrigin node defines an absolute geographic location and an implicit local coordinate frame against
which geometry is referenced. This node is used to translate from geographical coordinates into a local Cartesian coordinate
system which can be managed by the browser.</xs:appinfo>
  <xs:documentation source="http://www.web3d.org/x3d/specifications/ISO-IEC-19775-FDIS-
X3dAbstractSpecification/Part01/components/geodata.html">See X3D specification Part I 25.3.6</xs:documentation>
 </xs:annotation>
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="FullyExtensibleSMALElementType">
```

```
        <xs:sequence>
          <xs:element ref="Classification" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="geoCoords" type="xs:string" use="optional" default="N00 0.0 W00 0.0"/>
        <xs:attribute name="geoSystem" type="xs:string" use="optional" default="'GD', 'WE'">
          <xs:annotation>
            <xs:appinfo>Two enumerations, one for spatial reference frames, the other for earth ellipsoid i.e. ["GD", "WE"] See
X3D specification 25.2.3.</xs:appinfo>
          </xs:annotation>
        </xs:attribute>
        <xs:attribute name="rotateYUp" type="xs:boolean" use="optional" default="true"/>
        <xs:attribute name="geoOriginIdentifier" type="xs:ID" use="required"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

attribute **GeoOrigin/@geoCoords**

type	**xs:string**
properties	isRef 0 default N00 0.0 W00 0.0 use optional
source	`<xs:attribute name="geoCoords" type="xs:string" use="optional" default="N00 0.0 W00 0.0"/>`

attribute **GeoOrigin/@geoSystem**

type	**xs:string**
properties	isRef 0 default 'GD', 'WE' use optional
annotation	appInfo Two enumerations, one for spatial reference frames, the other for earth ellipsoid i.e. ["GD", "WE"] See X3D specification 25.2.3.

source
```
<xs:attribute name="geoSystem" type="xs:string" use="optional" default="'GD', 'WE'">
  <xs:annotation>
    <xs:appinfo>Two enumerations, one for spatial reference frames, the other for earth ellipsoid i.e. ["GD", "WE"] See X3D
specification 25.2.3.</xs:appinfo>
  </xs:annotation>
</xs:attribute>
```

attribute **GeoOrigin/@rotateYUp**

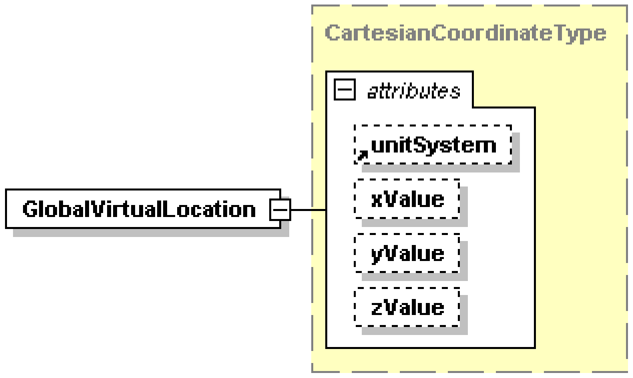
type	**xs:boolean**
properties	isRef 0 default true use optional
source	`<xs:attribute name="rotateYUp" type="xs:boolean" use="optional" default="true"/>`

attribute **GeoOrigin/@geoOriginIdentifier**

type	**xs:ID**
properties	isRef 0 use required
source	`<xs:attribute name="geoOriginIdentifier" type="xs:ID" use="required"/>`

element **GlobalVirtualLocation**

diagram



type	**CartesianCoordinateType**
properties	content complex substGrp Location
used by	element **AttachmentPoint**

attributes

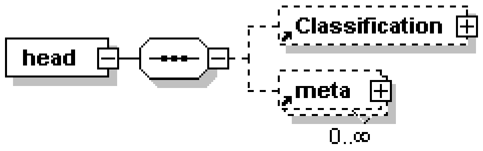
Name	Type	Use	Default	Fixed	Annotation
unitSystem			Metric		
xValue	**xs:float**		0		
yValue	**xs:float**		0		
zValue	**xs:float**		0		

annotation appInfo
Current point of origin based virtual location element.

source
```
<xs:element name="GlobalVirtualLocation" type="CartesianCoordinateType" substitutionGroup="Location">
  <xs:annotation>
    <xs:appinfo>Current point of origin based virtual location element.</xs:appinfo>
  </xs:annotation>
</xs:element>
```

element **head**

diagram

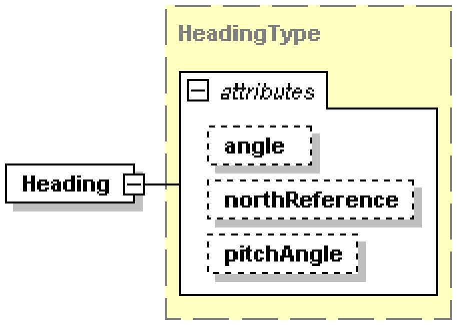


properties	content complex
children	**Classification** **meta**
used by	element **SMAL**

source
```
<xs:element name="head">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Classification" minOccurs="0"/>
      <xs:element ref="meta" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

element **Heading**

diagram



type **HeadingType**

properties

content	complex
substGrp	Orientation

attributes

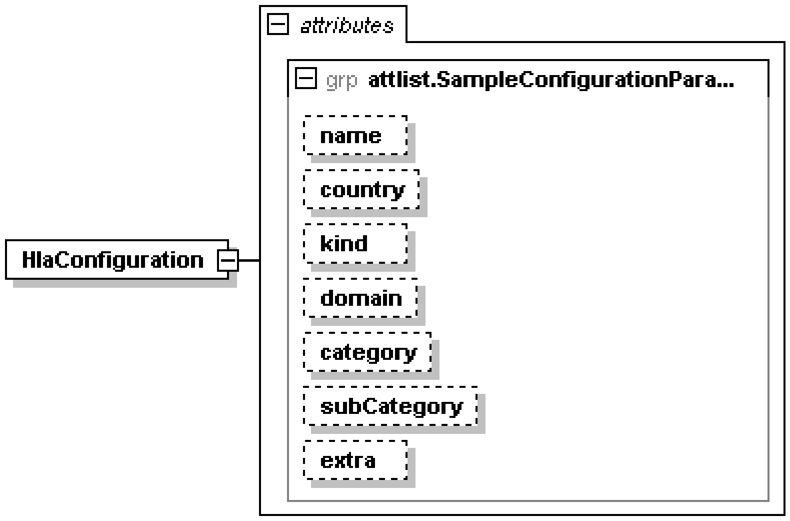
Name	Type	Use	Default	Fixed	Annotation
angle	**xs:float**		0		
northReference	**northReferenceEnumeration**		MAGNETIC		
pitchAngle	**xs:float**		0		

source

```
<xs:element name="Heading" type="HeadingType" substitutionGroup="Orientation"/>
```

element **HlaConfiguration**

diagram



properties

content	complex

used by

element	**NetworkedCommunicationParameterSet**

attributes

Name	Type	Use	Default	Fixed	Annotation
name	**xs:NMTOKEN**				
country	**xs:NMTOKEN**				
kind	**xs:NMTOKEN**				
domain	**xs:NMTOKEN**				
category	**xs:NMTOKEN**				
subCategory	**xs:NMTOKEN**				
extra	**xs:NMTOKEN**				

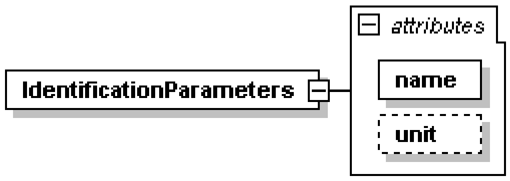
source

```
<xs:element name="HlaConfiguration">
 <!-- Hypothetical example only, not yet developed. -->
 <xs:complexType>
  <xs:attributeGroup ref="attlist.SampleConfigurationParameters"/>
 </xs:complexType>
```

```
</xs:element>
```

element **IdentificationParameters**

diagram



type extension of **FullyExtensibleSMALElementType**

properties

content	complex

used by

complexType	**ObjectDefinitionType**

attributes

Name	Type	Use	Default	Fixed	Annotation
name	**xs:string**	required			
unit	**xs:string**	optional			

annotation

appInfo
Human-readable identifying data for model. NOTE: Identification parameters are distinct from DIS enumerations.

source

```
<xs:element name="IdentificationParameters">
  <xs:annotation>
    <xs:appinfo>Human-readable identifying data for model. NOTE: Identification parameters are distinct from DIS
enumerations.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="unit" type="xs:string" use="optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

attribute **IdentificationParameters/@name**

type **xs:string**

properties

isRef	0
use	required

source

```
<xs:attribute name="name" type="xs:string" use="required"/>
```

attribute **IdentificationParameters/@unit**

type **xs:string**

properties

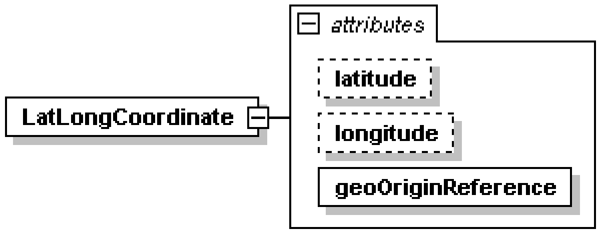
isRef	0
use	optional

source

```
<xs:attribute name="unit" type="xs:string" use="optional"/>
```

element **LatLongCoordinate**

diagram



type: extension of **AttributeExtensibleSMALElementType**

properties:

content	complex
substGrp	MapCoordinate

attributes:

Name	Type	Use	Default	Fixed	Annotation
latitude	**latitudeOrdinatePattern**		N00 00.0		
longitude	**longitudeOrdinatePattern**		W00 00.0		
geoOriginReference	**xs:IDREF**	required			

annotation: appInfo
Latitude-Longitude coordinate system coordinate-value location element. References a GeoOrigin element.

source
```
<xs:element name="LatLongCoordinate" substitutionGroup="MapCoordinate">
  <xs:annotation>
    <xs:appinfo>Latitude-Longitude coordinate system coordinate-value location element.  References a GeoOrigin
element.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="AttributeExtensibleSMALElementType">
        <xs:attribute name="latitude" type="latitudeOrdinatePattern" default="N00 00.0"/>
        <xs:attribute name="longitude" type="longitudeOrdinatePattern" default="W00 00.0"/>
        <xs:attribute name="geoOriginReference" type="xs:IDREF" use="required"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

attribute **LatLongCoordinate/@latitude**

type: **latitudeOrdinatePattern**

properties:

isRef	0
default	N00 00.0

facets:

pattern	(((N|n)|(S|s))(([0-8]?[0-9])|90) ([0-5]?[0-9])?(\.[0-9]*))

source
```
<xs:attribute name="latitude" type="latitudeOrdinatePattern" default="N00 00.0"/>
```

attribute **LatLongCoordinate/@longitude**

type: **longitudeOrdinatePattern**

properties:

isRef	0
default	W00 00.0

facets:

pattern	(((E|e)|(W|w))([0-9]?[0-9]|1[0-8][0-9]) ([0-5]?[0-9])?(\.[0-9]*))

source
```
<xs:attribute name="longitude" type="longitudeOrdinatePattern" default="W00 00.0"/>
```

attribute **LatLongCoordinate/@geoOriginReference**

type: **xs:IDREF**

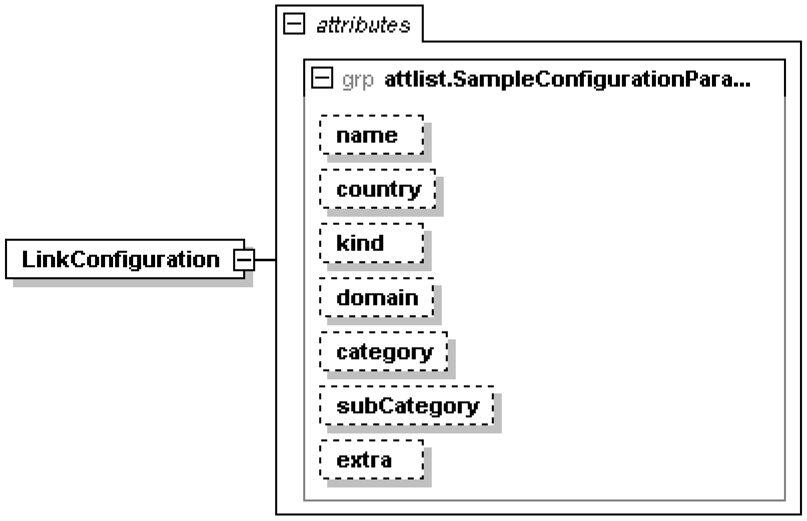
properties:

isRef	0
use	required

source
```
<xs:attribute name="geoOriginReference" type="xs:IDREF" use="required"/>
```

element **LinkConfiguration**

diagram



properties	content	complex
used by	element	**NetworkedCommunicationParameterSet**

attributes

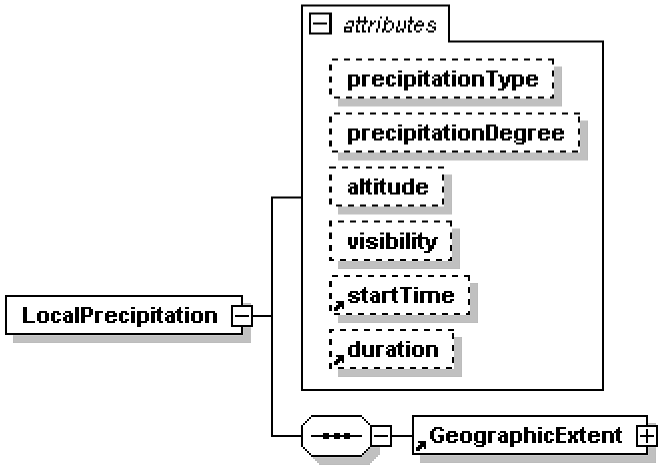
Name	Type	Use	Default	Fixed	Annotation
name	**xs:NMTOKEN**				
country	**xs:NMTOKEN**				
kind	**xs:NMTOKEN**				
domain	**xs:NMTOKEN**				
category	**xs:NMTOKEN**				
subCategory	**xs:NMTOKEN**				
extra	**xs:NMTOKEN**				

source

```
<xs:element name="LinkConfiguration">
  <!-- Hypothetical example only, projected for use with Link11 and Link16 protocols -->
  <xs:complexType>
    <xs:attributeGroup ref="attlist.SampleConfigurationParameters"/>
  </xs:complexType>
</xs:element>
```

element **LocalPrecipitation**

diagram



type	extension of **FullyExtensibleSMALElementType**	
properties	content	complex

children	**GeographicExtent**
used by	element **PrecipitationConditionSet**

attributes

Name	Type	Use	Default	Fixed	Annotation
precipitationType	**precipitationTypeEnumeration**				
precipitationDegree	**precipitationDegreeEnumeration**				
altitude	**xs:float**				
visibility	**nonNegativeFloat**				
startTime			00:00:00		
duration			01:00:00		

annotation appInfo
Describes a local precipitation event with Location, GeographicExtent, type, degree, starting altitued, and associated visibility. Precipitation may change over the course of an EnvironmentalCondition, so there is a duration attribute.

source
```
<xs:element name="LocalPrecipitation">
 <xs:annotation>
  <xs:appinfo>Describes a local precipitation event with Location, GeographicExtent, type, degree, starting altitued, and associated visibility.  Precipitation may change over the course of an EnvironmentalCondition, so there is a duration attribute.</xs:appinfo>
 </xs:annotation>
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="FullyExtensibleSMALElementType">
    <xs:sequence>
     <xs:element ref="GeographicExtent"/>
    </xs:sequence>
    <xs:attribute name="precipitationType" type="precipitationTypeEnumeration"/>
    <xs:attribute name="precipitationDegree" type="precipitationDegreeEnumeration"/>
    <xs:attribute name="altitude" type="xs:float"/>
    <xs:attribute name="visibility" type="nonNegativeFloat"/>
    <xs:attribute ref="startTime"/>
    <xs:attribute ref="duration"/>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
```

attribute **LocalPrecipitation/@precipitationType**

type **precipitationTypeEnumeration**

properties isRef 0

facets

enumeration	RAIN
enumeration	SNOW
enumeration	SLEET
enumeration	HAIL

source
```
<xs:attribute name="precipitationType" type="precipitationTypeEnumeration"/>
```

attribute **LocalPrecipitation/@precipitationDegree**

type **precipitationDegreeEnumeration**

properties isRef 0

facets

enumeration	HEAVY
enumeration	LIGHT

source
```
<xs:attribute name="precipitationDegree" type="precipitationDegreeEnumeration"/>
```

attribute **LocalPrecipitation/@altitude**

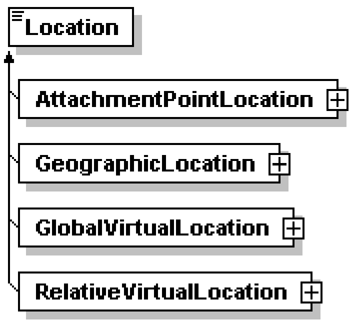
type	**xs:float**
properties	isRef 0
source	`<xs:attribute name="altitude" type="xs:float"/>`

attribute **LocalPrecipitation/@visibility**

type	**nonNegativeFloat**
properties	isRef 0
facets	minInclusive 0.0
source	`<xs:attribute name="visibility" type="nonNegativeFloat"/>`

element **Location**

diagram



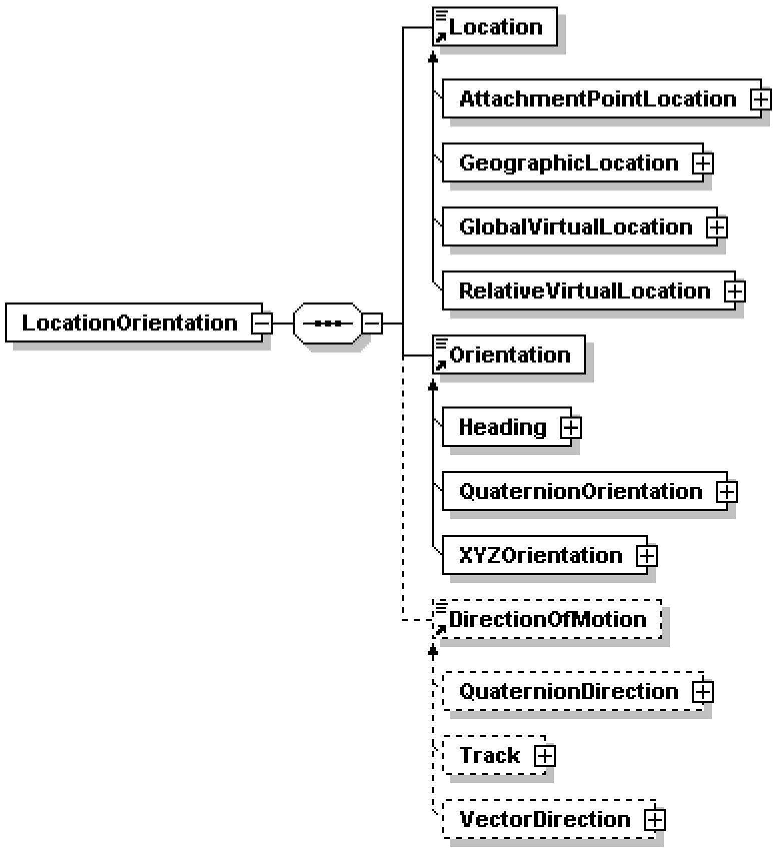
properties	abstract true
used by	elements **LocationOrientation** **TerrainTile**
annotation	appInfo Abstract placeholder for specific locator element.

source

```
<xs:element name="Location" abstract="true">
  <xs:annotation>
    <xs:appinfo>Abstract placeholder for specific locator element.</xs:appinfo>
  </xs:annotation>
</xs:element>
```

element **LocationOrientation**

diagram



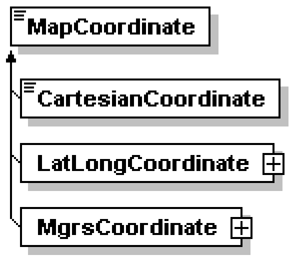
type	extension of **FullyExtensibleSMALElementType**
properties	content complex
children	**Location** **Orientation** **DirectionOfMotion**
used by	elements **CloudLayer** **CurrentConditionParameters** **StaticModelDefinition**
annotation	appInfo Contains data on position, orientation, and (optionally) direction of motion for the object it is describing.

source

```
<xs:element name="LocationOrientation">
  <xs:annotation>
    <xs:appinfo>Contains data on position, orientation, and (optionally) direction of motion for the object it is
describing.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:sequence>
          <xs:element ref="Location"/>
          <xs:element ref="Orientation"/>
          <xs:element ref="DirectionOfMotion" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

element **MapCoordinate**

diagram



properties

abstract	true

used by

element	**PointRadiusCircular**
complexType	**BoundingPolygonType**

annotation

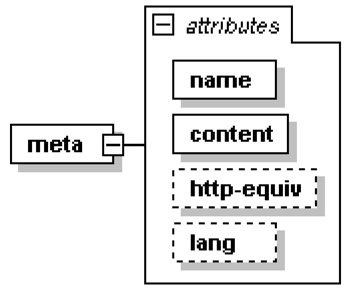
appInfo
Abstract placeholder for coordinate-reference-system-specific location value element.

source

```
<xs:element name="MapCoordinate" abstract="true">
  <xs:annotation>
    <xs:appinfo>Abstract placeholder for coordinate-reference-system-specific location value element.</xs:appinfo>
  </xs:annotation>
</xs:element>
```

element **meta**

diagram



properties

content	complex
mixed	false

used by

element	**head**

attributes

Name	Type	Use	Default	Fixed	Annotation
name	**xs:string**	required			
content	**xs:string**	required			
http-equiv	**xs:string**				
lang	**xs:string**				

source

```
<xs:element name="meta">
  <xs:annotation>
    <xs:appinfo/>
    <xs:appinfo source="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"/>
  </xs:annotation>
  <xs:complexType mixed="false">
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="content" type="xs:string" use="required"/>
    <xs:attribute name="http-equiv" type="xs:string"/>
    <xs:attribute name="lang" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

attribute **meta/@name**

type **xs:string**

properties

isRef	0

	use required
source	`<xs:attribute name="name" type="xs:string" use="required"/>`

attribute **meta/@content**

type	**xs:string**
properties	isRef 0 use required
source	`<xs:attribute name="content" type="xs:string" use="required"/>`

attribute **meta/@http-equiv**

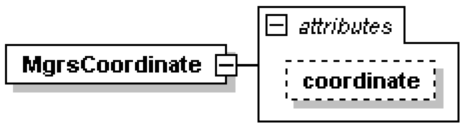
type	**xs:string**
properties	isRef 0
source	`<xs:attribute name="http-equiv" type="xs:string"/>`

attribute **meta/@lang**

type	**xs:string**
properties	isRef 0
source	`<xs:attribute name="lang" type="xs:string"/>`

element **MgrsCoordinate**

diagram



type extension of **AttributeExtensibleSMALElementType**

properties

content	complex
substGrp	MapCoordinate

attributes

Name	Type	Use	Default	Fixed	Annotation
coordinate	**mgrsCoordinatePattern**				

annotation

appInfo
Military Grid Reference System coordinate location value element.

source
```
<xs:element name="MgrsCoordinate" substitutionGroup="MapCoordinate">
 <xs:annotation>
  <xs:appinfo>Military Grid Reference System coordinate location value element.</xs:appinfo>
 </xs:annotation>
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="AttributeExtensibleSMALElementType">
    <xs:attribute name="coordinate" type="mgrsCoordinatePattern"/>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
```

attribute **MgrsCoordinate/@coordinate**

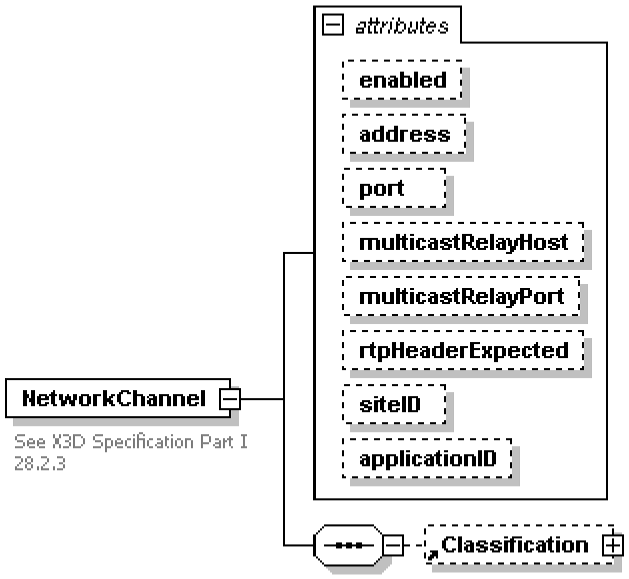
type	**mgrsCoordinatePattern**
properties	isRef 0

facets pattern (([0-9]?[0-9]([A-H,J-N,P-Z])?)|([0-9]?[0-9][A-H,J-N,P-Z]([A-H,J-N,P-Z][A-H,J-N,P-Z])(([0-9][0-9]){0,5})))

source
```
<xs:attribute name="coordinate" type="mgrsCoordinatePattern"/>
```

element **NetworkChannel**

diagram



type extension of **FullyExtensibleSMALElementType**

properties content complex

children **Classification**

used by element **Simulation**

attributes

Name	Type	Use	Default	Fixed	Annotation
enabled	**xs:boolean**	optional	true		
address	**ipAddressPattern**	optional	127.0.0.1		
port	**xs:nonNegativeInteger**	optional	62040		
multicastRelayHost	**ipAddressPattern**	optional	127.0.0.1		
multicastRelayPort	**xs:nonNegativeInteger**	optional	62040		
rtpHeaderExpected	**xs:boolean**		false		
siteID	**xs:nonNegativeInteger**	optional			
applicationID	**xs:nonNegativeInteger**	optional			

annotation

appInfo

Contains all data necessary to define the multicast port address, simulation, and local computer ID. A single Simulation has only one network channel.

documentation

See X3D Specification Part I 28.2.3

source
```
<xs:element name="NetworkChannel">
  <xs:annotation>
    <xs:appinfo>Contains all data necessary to define the multicast port address, simulation, and local computer ID. A single
Simulation has only one network channel.</xs:appinfo>
    <xs:documentation source="http://www.web3d.org/x3d/specifications/ISO-IEC-19775-FDIS-
X3dAbstractSpecification/Part01/components/dis.html">See X3D Specification Part I 28.2.3</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:sequence>
          <xs:element ref="Classification" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="enabled" type="xs:boolean" use="optional" default="true"/>
        <xs:attribute name="address" type="ipAddressPattern" use="optional" default="127.0.0.1"/>
        <xs:attribute name="port" type="xs:nonNegativeInteger" use="optional" default="62040"/>
        <xs:attribute name="multicastRelayHost" type="ipAddressPattern" use="optional" default="127.0.0.1"/>
        <xs:attribute name="multicastRelayPort" type="xs:nonNegativeInteger" use="optional" default="62040"/>
        <xs:attribute name="rtpHeaderExpected" type="xs:boolean" default="false"/>
        <xs:attribute name="siteID" type="xs:nonNegativeInteger" use="optional"/>
        <xs:attribute name="applicationID" type="xs:nonNegativeInteger" use="optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

attribute **NetworkChannel/@enabled**

type	**xs:boolean**	
properties	isRef	0
	default	true
	use	optional
source	`<xs:attribute name="enabled" type="xs:boolean" use="optional" default="true"/>`	

attribute **NetworkChannel/@address**

type	**ipAddressPattern**	
properties	isRef	0
	default	127.0.0.1
	use	optional
facets	pattern	(([0-1]?[0-9]?[0-9]|2[0-4][0-9])|25[0-5])((.(([0-1]?[0-9]?[0-9]|2[0-4][0-9])|25[0-5])){3})
source	`<xs:attribute name="address" type="ipAddressPattern" use="optional" default="127.0.0.1"/>`	

attribute **NetworkChannel/@port**

type	**xs:nonNegativeInteger**	
properties	isRef	0
	default	62040
	use	optional
source	`<xs:attribute name="port" type="xs:nonNegativeInteger" use="optional" default="62040"/>`	

attribute **NetworkChannel/@multicastRelayHost**

type	**ipAddressPattern**	
properties	isRef	0
	default	127.0.0.1
	use	optional
facets	pattern	(([0-1]?[0-9]?[0-9]|2[0-4][0-9])|25[0-5])((.(([0-1]?[0-9]?[0-9]|2[0-4][0-9])|25[0-5])){3})
source	`<xs:attribute name="multicastRelayHost" type="ipAddressPattern" use="optional" default="127.0.0.1"/>`	

attribute **NetworkChannel/@multicastRelayPort**

type	**xs:nonNegativeInteger**	
properties	isRef	0
	default	62040
	use	optional
source	`<xs:attribute name="multicastRelayPort" type="xs:nonNegativeInteger" use="optional" default="62040"/>`	

attribute **NetworkChannel/@rtpHeaderExpected**

type **xs:boolean**

properties	isRef	0
	default	false

source `<xs:attribute name="rtpHeaderExpected" type="xs:boolean" default="false"/>`

attribute **NetworkChannel/@siteID**

type **xs:nonNegativeInteger**

properties	isRef	0
	use	optional

source `<xs:attribute name="siteID" type="xs:nonNegativeInteger" use="optional"/>`

attribute **NetworkChannel/@applicationID**

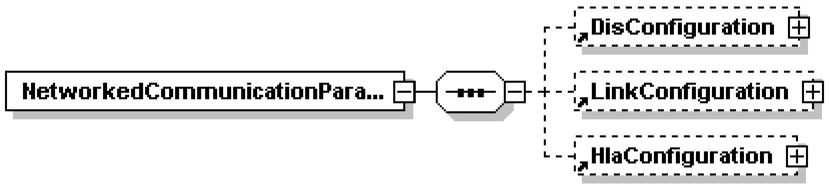
type **xs:nonNegativeInteger**

properties	isRef	0
	use	optional

source `<xs:attribute name="applicationID" type="xs:nonNegativeInteger" use="optional"/>`

element **NetworkedCommunicationParameterSet**

diagram



type extension of **FullyExtensibleSMALElementType**

properties	content	complex

children **DisConfiguration** **LinkConfiguration** **HlaConfiguration**

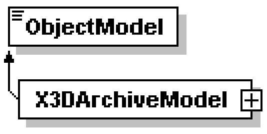
used by	element	**EntityDefinition**

annotation appInfo
Entity-specific network communication parameters for use in message construction. Parameters common to all entities are held in the ancestor NetworkChannel element. The protocol-specific tags will eventually be replaced by namespace references.

source
```
<xs:element name="NetworkedCommunicationParameterSet">
  <xs:annotation>
    <xs:appinfo>Entity-specific network communication parameters for use in message construction. Parameters common to
all entities are held in the ancestor NetworkChannel element. The protocol-specific tags will eventually be replaced by
namespace references.
    </xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:sequence>
          <xs:element ref="DisConfiguration" minOccurs="0"/>
          <xs:element ref="LinkConfiguration" minOccurs="0"/>
          <xs:element ref="HlaConfiguration" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

element **ObjectModel**

diagram



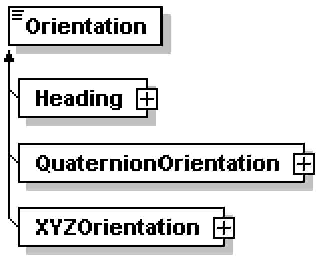
properties	abstract	true
used by	element	TerrainTile
	complexType	ObjectDefinitionType

source

```
<xs:element name="ObjectModel" abstract="true"/>
```

element **Orientation**

diagram



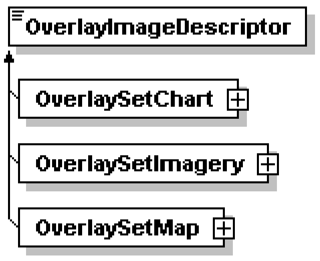
properties	abstract	true
used by	elements	AttachmentPoint LocationOrientation

source

```
<xs:element name="Orientation" abstract="true">
  <xs:annotation>
    <xs:appinfo/>
  </xs:annotation>
</xs:element>
```

element **OverlayImageDescriptor**

diagram



properties	abstract	true
used by	element	OverlaySet

annotation

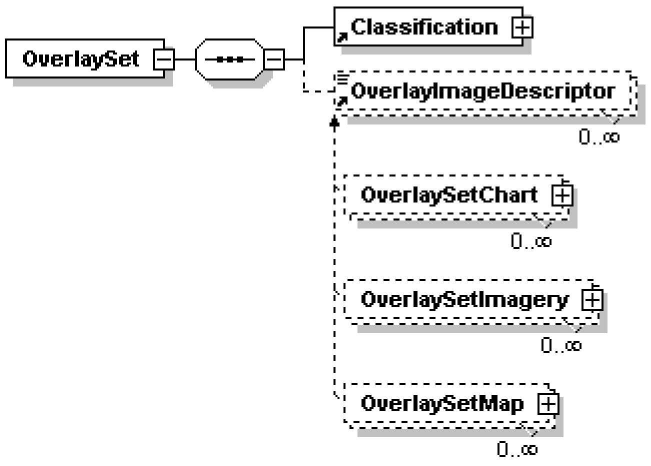
appInfo
Abstract placeholder for specific overlay elements in an OverlaySet element.

source

```
<xs:element name="OverlayImageDescriptor" abstract="true">
  <xs:annotation>
    <xs:appinfo>Abstract placeholder for specific overlay elements in an OverlaySet element.</xs:appinfo>
  </xs:annotation>
</xs:element>
```

element **OverlaySet**

diagram



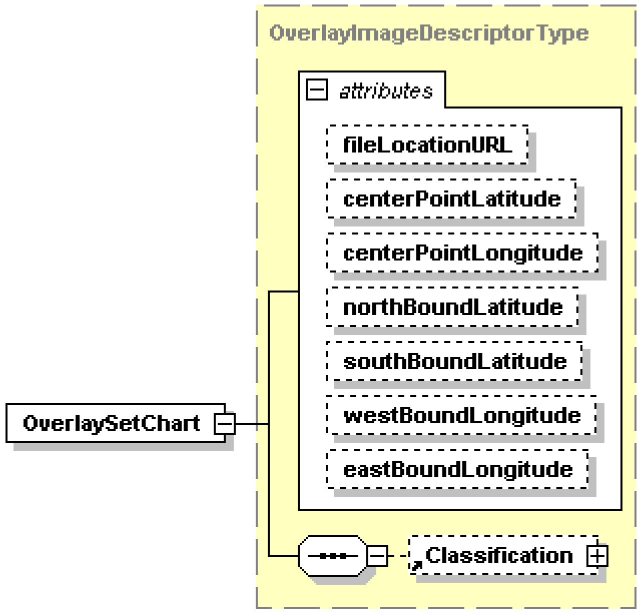
type	extension of **FullyExtensibleSMALElementType**
properties	content complex
children	**Classification** **OverlayImageDescriptor**
used by	element **TerrainTile**
annotation	appInfo Collects all imagery overlays associated with a TerrainTile.

source

```
<xs:element name="OverlaySet">
  <xs:annotation>
    <xs:appinfo>Collects all imagery overlays associated with a TerrainTile.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:sequence>
          <xs:element ref="Classification"/>
          <xs:element ref="OverlayImageDescriptor" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

element **OverlaySetChart**

diagram



type **OverlayImageDescriptorType**

properties

content	complex
substGrp	OverlayImageDescriptor

children **Classification**

attributes

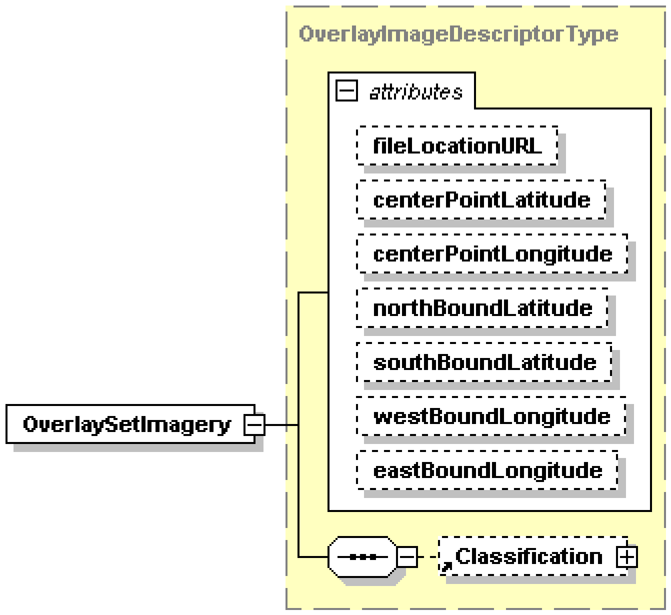
Name	Type	Use	Default	Fixed	Annotation
fileLocationURL	**xs:anyURI**		http://www.web3d.org/x3d/content/examples/		
centerPointLatitude	**latitudeOrdinatePattern**	optional	N00 00.0		
centerPointLongitude	**longitudeOrdinatePattern**	optional	W00 00.0		
northBoundLatitude	**latitudeOrdinatePattern**	optional	N00 00.0		
southBoundLatitude	**latitudeOrdinatePattern**	optional	N00 00.0		
westBoundLongitude	**longitudeOrdinatePattern**	optional	W00 00.0		
eastBoundLongitude	**longitudeOrdinatePattern**	optional	W00 00.0		

annotation appInfo
A chart image used as a TerrainTile overlay.

source
```
<xs:element name="OverlaySetChart" type="OverlayImageDescriptorType" substitutionGroup="OverlayImageDescriptor">
  <xs:annotation>
    <xs:appinfo>A chart image used as a TerrainTile overlay.</xs:appinfo>
  </xs:annotation>
</xs:element>
```

element **OverlaySetImagery**

diagram



type **OverlayImageDescriptorType**

properties

content	complex
substGrp	OverlayImageDescriptor

children **Classification**

attributes

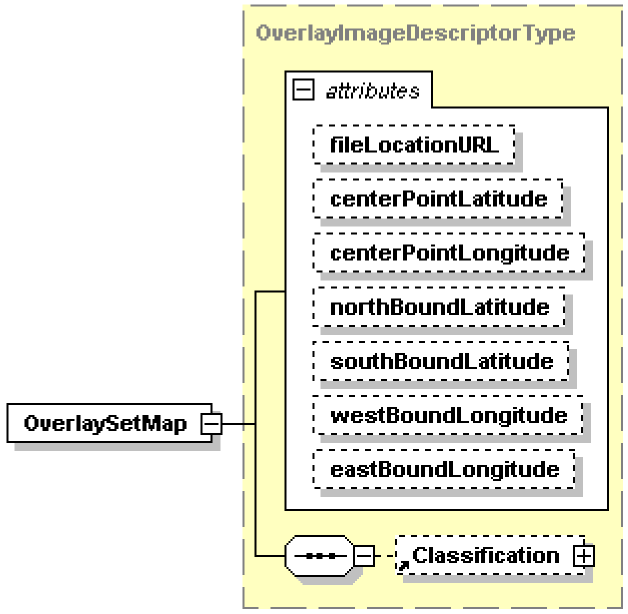
Name	Type	Use	Default	Fixed	Annotation
fileLocationURL	**xs:anyURI**		http://www.web3d.org/x3d/content/examples/		
centerPointLatitude	**latitudeOrdinatePattern**	optional	N00 00.0		
centerPointLongitude	**longitudeOrdinatePattern**	optional	W00 00.0		
northBoundLatitude	**latitudeOrdinatePattern**	optional	N00 00.0		
southBoundLatitude	**latitudeOrdinatePattern**	optional	N00 00.0		
westBoundLongitude	**longitudeOrdinatePattern**	optional	W00 00.0		
eastBoundLongitude	**longitudeOrdinatePattern**	optional	W00 00.0		

annotation appInfo
A satellite image or graphic used as a TerrainTile overlay.

source
```
<xs:element name="OverlaySetImagery" type="OverlayImageDescriptorType" substitutionGroup="OverlayImageDescriptor">
  <xs:annotation>
    <xs:appinfo>A satellite image or graphic used as a TerrainTile overlay.</xs:appinfo>
  </xs:annotation>
</xs:element>
```

element **OverlaySetMap**

diagram



type **OverlayImageDescriptorType**

properties

content	complex
substGrp	OverlayImageDescriptor

children **Classification**

attributes

Name	Type	Use	Default	Fixed	Annotation
fileLocationURL	**xs:anyURI**		http://www.web3d.org/x3d/content/examples/		
centerPointLatitude	**latitudeOrdinatePattern**	optional	N00 00.0		
centerPointLongitude	**longitudeOrdinatePattern**	optional	W00 00.0		
northBoundLatitude	**latitudeOrdinatePattern**	optional	N00 00.0		
southBoundLatitude	**latitudeOrdinatePattern**	optional	N00 00.0		
westBoundLongitude	**longitudeOrdinatePattern**	optional	W00 00.0		
eastBoundLongitude	**longitudeOrdinatePattern**	optional	W00 00.0		

annotation

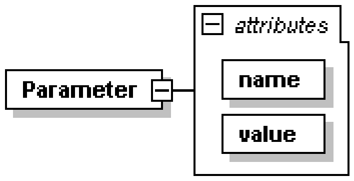
appInfo
A map image used as a TerrainTile overlay.

source

```
<xs:element name="OverlaySetMap" type="OverlayImageDescriptorType" substitutionGroup="OverlayImageDescriptor">
  <xs:annotation>
    <xs:appinfo>A map image used as a TerrainTile overlay.</xs:appinfo>
  </xs:annotation>
</xs:element>
```

element **Parameter**

diagram



type extension of **FullyExtensibleSMALElementType**

properties

content	complex

used by

element	**SimulationAgent**

attributes

Name	Type	Use	Default	Fixed	Annotation
name	**xs:NMTOKEN**	required			
value	**xs:string**	required			

source
```
<xs:element name="Parameter">
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="FullyExtensibleSMALElementType">
    <xs:attribute name="name" type="xs:NMTOKEN" use="required"/>
    <xs:attribute name="value" type="xs:string" use="required"/>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
```

attribute **Parameter/@name**

type **xs:NMTOKEN**

properties

isRef	0
use	required

source `<xs:attribute name="name" type="xs:NMTOKEN" use="required"/>`

attribute **Parameter/@value**

type **xs:string**

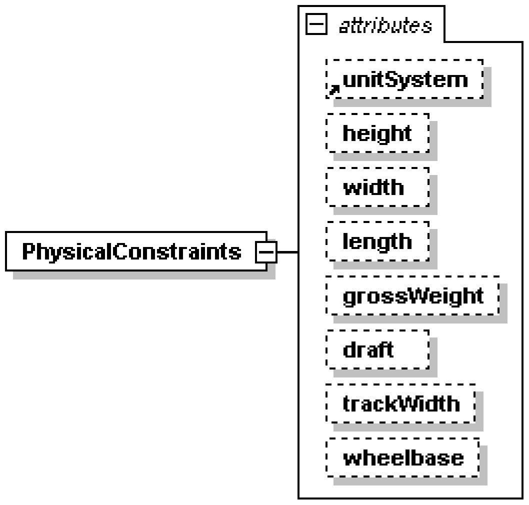
properties

isRef	0
use	required

source `<xs:attribute name="value" type="xs:string" use="required"/>`

element **PhysicalConstraints**

diagram



type extension of **FullyExtensibleSMALElementType**

properties

content	complex

used by

element	**PhysicalParameters**

attributes

Name	Type	Use	Default	Fixed	Annotation
unitSystem			Metric		
height	**xs:float**				
width	**xs:float**				
length	**xs:float**				
grossWeight	**xs:float**				
draft	**xs:float**	optional			
trackWidth	**xs:float**	optional			
wheelbase	**xs:float**	optional			

annotation	appInfo Physics modeling data for collision detection and physics engine use. NOTE: AVCL parameters can be included optionally here.

source
```
<xs:element name="PhysicalConstraints">
  <xs:annotation>
    <xs:appinfo>Physics modeling data for collision detection and physics engine use. NOTE: AVCL parameters can be
included optionally here. </xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:attribute ref="unitSystem"/>
        <xs:attribute name="height" type="xs:float"/>
        <xs:attribute name="width" type="xs:float"/>
        <xs:attribute name="length" type="xs:float"/>
        <xs:attribute name="grossWeight" type="xs:float"/>
        <xs:attribute name="draft" type="xs:float" use="optional"/>
        <xs:attribute name="trackWidth" type="xs:float" use="optional"/>
        <xs:attribute name="wheelbase" type="xs:float" use="optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

attribute **PhysicalConstraints/@height**

type **xs:float**

properties isRef 0

source
```
<xs:attribute name="height" type="xs:float"/>
```

attribute **PhysicalConstraints/@width**

type **xs:float**

properties isRef 0

source
```
<xs:attribute name="width" type="xs:float"/>
```

attribute **PhysicalConstraints/@length**

type **xs:float**

properties isRef 0

source
```
<xs:attribute name="length" type="xs:float"/>
```

attribute **PhysicalConstraints/@grossWeight**

type **xs:float**

properties isRef 0

source
```
<xs:attribute name="grossWeight" type="xs:float"/>
```

attribute **PhysicalConstraints/@draft**

type **xs:float**

properties isRef 0
use optional

source
```
<xs:attribute name="draft" type="xs:float" use="optional"/>
```

attribute **PhysicalConstraints/@trackWidth**

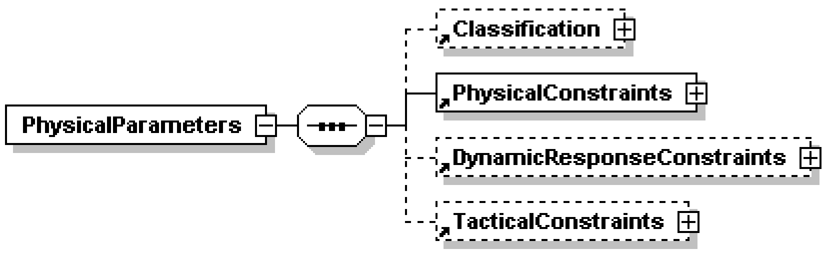
type	**xs:float**
properties	isRef 0 use optional
source	`<xs:attribute name="trackWidth" type="xs:float" use="optional"/>`

attribute **PhysicalConstraints/@wheelbase**

type	**xs:float**
properties	isRef 0 use optional
source	`<xs:attribute name="wheelbase" type="xs:float" use="optional"/>`

element **PhysicalParameters**

diagram



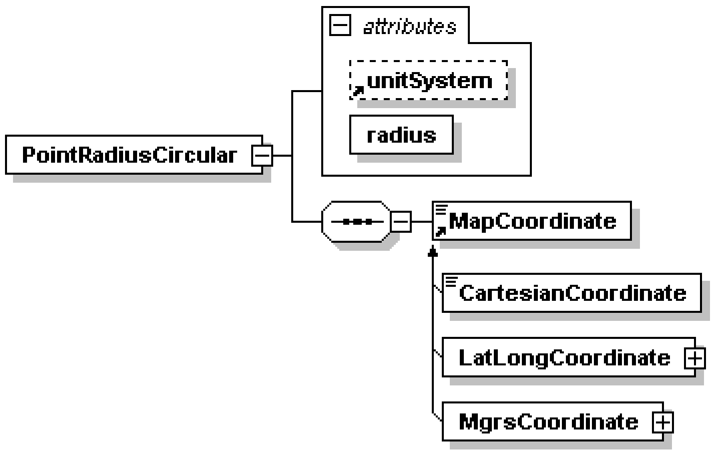
type	extension of **FullyExtensibleSMALElementType**
properties	content complex
children	**Classification** **PhysicalConstraints** **DynamicResponseConstraints** **TacticalConstraints**
used by	complexType **ObjectDefinitionType**
annotation	appInfo Contains all parameters which are dependent on the physical characteristics and performance of the model.

source

```
<xs:element name="PhysicalParameters">
  <xs:annotation>
    <xs:appinfo>Contains all parameters which are dependent on the physical characteristics and performance of the
model.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:sequence>
          <xs:element ref="Classification" minOccurs="0"/>
          <xs:element ref="PhysicalConstraints"/>
          <xs:element ref="DynamicResponseConstraints" minOccurs="0"/>
          <xs:element ref="TacticalConstraints" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

element **PointRadiusCircular**

diagram



type	extension of **FullyExtensibleSMALElementType**
properties	content complex
children	**MapCoordinate**
used by	complexType **BoundingPolygonType**

attributes

Name	Type	Use	Default	Fixed	Annotation
unitSystem			Metric		
radius	**nonNegativeFloat**	required			

annotation appInfo
A shape descriptor used to specify circular shapes in BoundingPolygonType elements.

source
```
<xs:element name="PointRadiusCircular">
  <xs:annotation>
    <xs:appinfo>A shape descriptor used to specify circular shapes in BoundingPolygonType elements.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:sequence>
          <xs:element ref="MapCoordinate"/>
        </xs:sequence>
        <xs:attribute ref="unitSystem"/>
        <xs:attribute name="radius" type="nonNegativeFloat" use="required"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

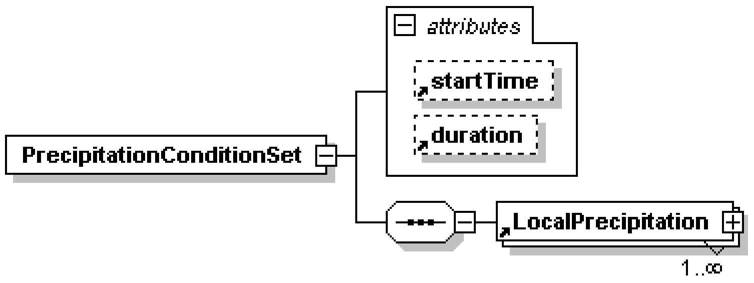
attribute **PointRadiusCircular/@radius**

type	**nonNegativeFloat**
properties	isRef 0 use required
facets	minInclusive 0.0

source
```
<xs:attribute name="radius" type="nonNegativeFloat" use="required"/>
```

element **PrecipitationConditionSet**

diagram



type	extension of **FullyExtensibleSMALElementType**
properties	content complex
children	**LocalPrecipitation**
used by	element **EnvironmentalCondition**

attributes

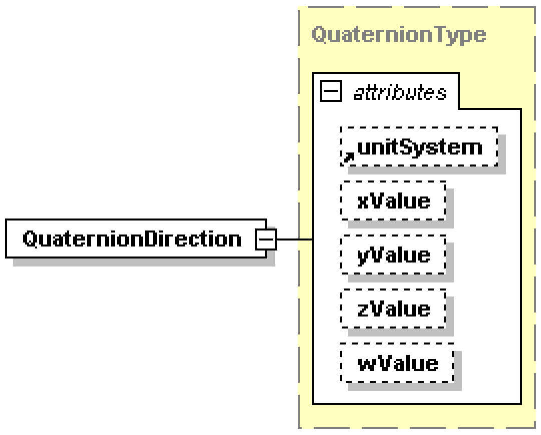
Name	Type	Use	Default	Fixed	Annotation
startTime			00:00:00		
duration			01:00:00		

annotation
appInfo
Collects the precipitation elements for a single EnvironmentalCondition element.

source
```
<xs:element name="PrecipitationConditionSet">
  <xs:annotation>
    <xs:appinfo>Collects the precipitation elements for a single EnvironmentalCondition element.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:sequence>
          <xs:element ref="LocalPrecipitation" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute ref="startTime"/>
        <xs:attribute ref="duration"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

element **QuaternionDirection**

diagram



type **QuaternionType**

properties

content	complex
substGrp	DirectionOfMotion

attributes

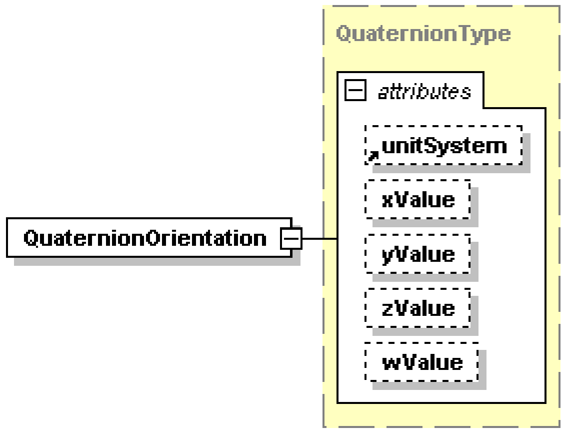
Name	Type	Use	Default	Fixed	Annotation
unitSystem			Metric		
xValue	**unitValueFloat**		0		
yValue	**unitValueFloat**		0		
zValue	**unitValueFloat**		0		
wValue	**nonNegativeFloat**		0		

source

```
<xs:element name="QuaternionDirection" type="QuaternionType" substitutionGroup="DirectionOfMotion"/>
```

element **QuaternionOrientation**

diagram



type **QuaternionType**

properties

content	complex
substGrp	Orientation

attributes

Name	Type	Use	Default	Fixed	Annotation
unitSystem			Metric		
xValue	**unitValueFloat**		0		
yValue	**unitValueFloat**		0		
zValue	**unitValueFloat**		0		
wValue	**nonNegativeFloat**		0		

annotation

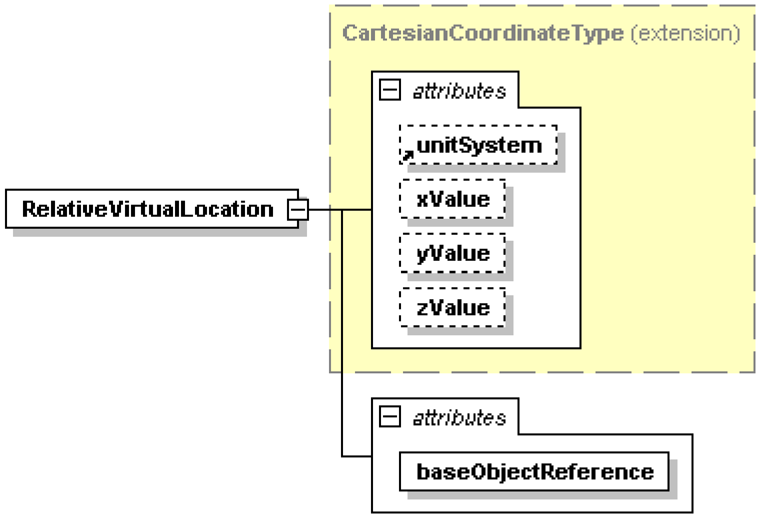
appInfo
A set of four values based on the current reference coordinate axes. Use fractional values -1 to 1 for the first three numbers and a radian value for the fourth.

source

```
<xs:element name="QuaternionOrientation" type="QuaternionType" substitutionGroup="Orientation">
  <xs:annotation>
    <xs:appinfo>A set of four values based on the current reference coordinate axes.  Use fractional values -1 to 1 for the first
three numbers and a radian value for the fourth.</xs:appinfo>
  </xs:annotation>
</xs:element>
```

element **RelativeVirtualLocation**

diagram



type extension of **CartesianCoordinateType**

properties

content	complex
substGrp	Location

attributes

Name	Type	Use	Default	Fixed	Annotation
unitSystem			Metric		
xValue	**xs:float**		0		
yValue	**xs:float**		0		
zValue	**xs:float**		0		
baseObjectReference	**xs:IDREF**	required			

annotation appInfo
Object-reference based location element used when there is no specific attachment point specified on that object. The @baseLocationObject IDREF value is used to reference the @entityIdentifier ID value of an EntityDefinition element.

source
```
<xs:element name="RelativeVirtualLocation" substitutionGroup="Location">
  <xs:annotation>
    <xs:appinfo>Object-reference based location element used when there is no specific attachment point specified on that
object. The @baseLocationObject IDREF value is used to reference the @entityIdentifier ID value of an EntityDefinition
element.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="CartesianCoordinateType">
        <xs:attribute name="baseObjectReference" type="xs:IDREF" use="required"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

attribute **RelativeVirtualLocation/@baseObjectReference**

type **xs:IDREF**

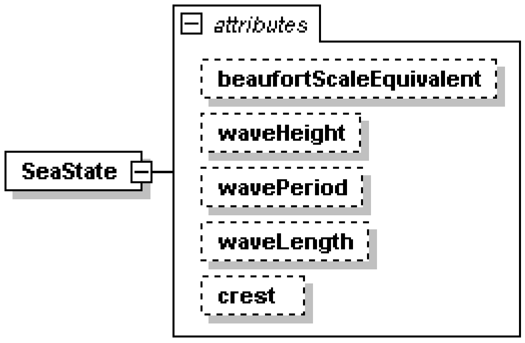
properties

isRef	0
use	required

source
```
<xs:attribute name="baseObjectReference" type="xs:IDREF" use="required"/>
```

element **SeaState**

diagram



type extension of **FullyExtensibleSMALElementType**

properties content complex

used by element **WaterConditionSet**

attributes

Name	Type	Use	Default	Fixed	Annotation
beaufortScaleEquivalent	**xs:nonNegativeInteger**	optional			
waveHeight	**nonNegativeFloat**	optional			
wavePeriod	**nonNegativeFloat**	optional			
waveLength	**nonNegativeFloat**	optional			
crest	**xs:NMTOKEN**	optional			

annotation appInfo
Describes water surface conditions to optionally include the Beaufort Scale equivalent condition for winds and wave action, and/or specific wave conditions such as crest conditions and wave height, length, and period. Sea state is considered persistent and for a given EnvironmentCondition so there is no duration attribute.

source
```
<xs:element name="SeaState">
 <xs:annotation>
  <xs:appinfo>Describes water surface conditions to optionally include the Beaufort Scale equivalent condition for winds and
wave action, and/or specific wave conditions such as crest conditions and wave height, length, and period.  Sea state is
considered persistent and for a given EnvironmentCondition so there is no duration attribute.</xs:appinfo>
 </xs:annotation>
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="FullyExtensibleSMALElementType">
    <xs:attribute name="beaufortScaleEquivalent" type="xs:nonNegativeInteger" use="optional"/>
    <xs:attribute name="waveHeight" type="nonNegativeFloat" use="optional"/>
    <xs:attribute name="wavePeriod" type="nonNegativeFloat" use="optional"/>
    <xs:attribute name="waveLength" type="nonNegativeFloat" use="optional"/>
    <xs:attribute name="crest" type="xs:NMTOKEN" use="optional"/>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
```

attribute **SeaState/@beaufortScaleEquivalent**

type **xs:nonNegativeInteger**

properties
isRef 0
use optional

source
```
<xs:attribute name="beaufortScaleEquivalent" type="xs:nonNegativeInteger" use="optional"/>
```

attribute **SeaState/@waveHeight**

type **nonNegativeFloat**

properties
isRef 0
use optional

facets minInclusive 0.0

source <xs:attribute name="waveHeight" type="nonNegativeFloat" use="optional"/>

attribute **SeaState/@wavePeriod**

type	**nonNegativeFloat**
properties	isRef 0 use optional
facets	minInclusive 0.0
source	<xs:attribute name="wavePeriod" type="nonNegativeFloat" use="optional"/>

attribute **SeaState/@waveLength**

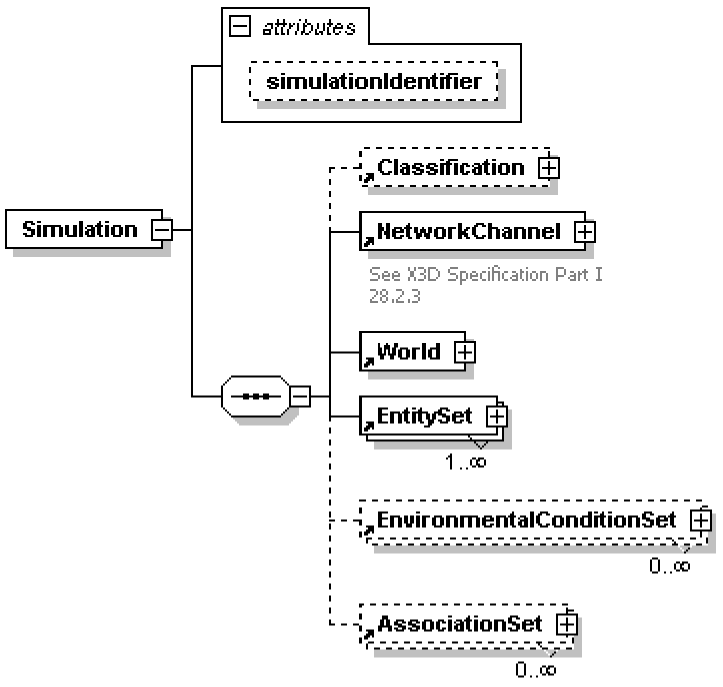
type	**nonNegativeFloat**
properties	isRef 0 use optional
facets	minInclusive 0.0
source	<xs:attribute name="waveLength" type="nonNegativeFloat" use="optional"/>

attribute **SeaState/@crest**

type	**xs:NMTOKEN**
properties	isRef 0 use optional
source	<xs:attribute name="crest" type="xs:NMTOKEN" use="optional"/>

element **Simulation**

diagram



type	extension of **FullyExtensibleSMALElementType**
properties	content complex

children	**Classification** **NetworkChannel** **World** **EntitySet** **EnvironmentalConditionSet** **AssociationSet**
used by	element **SMAL**

attributes

Name	Type	Use	Default	Fixed	Annotation
simulationIdentifier	**xs:ID**	optional			

annotation
appInfo
Top level tag for a single simulation. Allows multiple simulations to be run separately, but at the same time under the root SMAL tag.

source
```
<xs:element name="Simulation">
 <xs:annotation>
  <xs:appinfo>Top level tag for a single simulation. Allows multiple simulations to be run separately, but at the same time
under the root SMAL tag.</xs:appinfo>
 </xs:annotation>
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="FullyExtensibleSMALElementType">
    <xs:sequence>
     <xs:element ref="Classification" minOccurs="0"/>
     <xs:element ref="NetworkChannel"/>
     <xs:element ref="World"/>
     <xs:element ref="EntitySet" maxOccurs="unbounded"/>
     <xs:element ref="EnvironmentalConditionSet" minOccurs="0" maxOccurs="unbounded"/>
     <xs:element ref="AssociationSet" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="simulationIdentifier" type="xs:ID" use="optional"/>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
```

attribute **Simulation/@simulationIdentifier**

type **xs:ID**

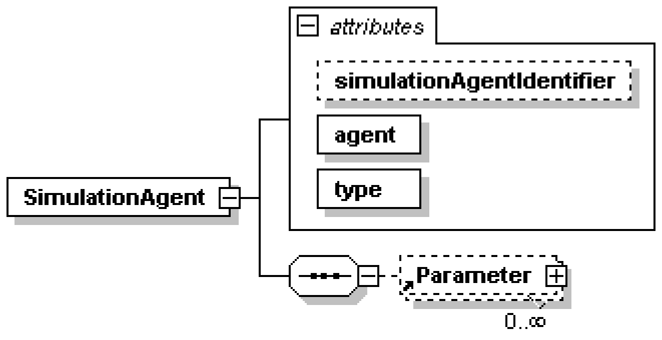
properties

isRef	0
use	optional

source
```
<xs:attribute name="simulationIdentifier" type="xs:ID" use="optional"/>
```

element **SimulationAgent**

diagram



type extension of **FullyExtensibleSMALElementType**

properties content complex

children **Parameter**

used by element **BehaviorParameterSet**

attributes

Name	Type	Use	Default	Fixed	Annotation
simulationAgentIdentifier	**xs:ID**	optional			
agent	**simulationEngineEnumeration**	required			

type	**xs:anyURI**	required	appInfo The filename or other identifier for the agent to be attached to this entity.

annotation appInfo
Agent-specific information required to run the simulation. Includes behavioral probability parameters and similar information not specifically enumerated in the tactical or dynamic response constraints.

source
```
<xs:element name="SimulationAgent">
 <xs:annotation>
  <xs:appinfo>Agent-specific information required to run the simulation. Includes behavioral probability parameters and
similar information not specifically enumerated in the tactical or dynamic response constraints.</xs:appinfo>
 </xs:annotation>
 <xs:complexType>
  <xs:annotation>
   <xs:appinfo>The entity name should be found in the parent EntityDefinition/@entityIdentifier attribute.</xs:appinfo>
  </xs:annotation>
  <xs:complexContent>
   <xs:extension base="FullyExtensibleSMALElementType">
    <xs:sequence>
     <xs:element ref="Parameter" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="simulationAgentIdentifier" type="xs:ID" use="optional"/>
    <xs:attribute name="agent" type="simulationEngineEnumeration" use="required"/>
    <xs:attribute name="type" type="xs:anyURI" use="required">
     <xs:annotation>
      <xs:appinfo>The filename or other identifier for the agent to be attached to this entity.</xs:appinfo>
     </xs:annotation>
    </xs:attribute>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
```

attribute **SimulationAgent/@simulationAgentIdentifier**

type **xs:ID**

properties

isRef	0
use	optional

source
```
<xs:attribute name="simulationAgentIdentifier" type="xs:ID" use="optional"/>
```

attribute **SimulationAgent/@agent**

type **simulationEngineEnumeration**

properties

isRef	0
use	required

facets

enumeration	Simkit
enumeration	Viskit
enumeration	HilesMultiAgent
enumeration	Other

source
```
<xs:attribute name="agent" type="simulationEngineEnumeration" use="required"/>
```

attribute **SimulationAgent/@type**

type **xs:anyURI**

properties

isRef	0
use	required

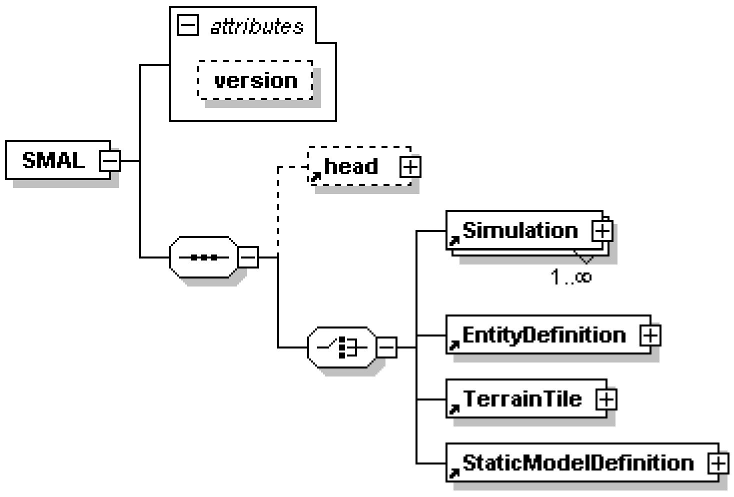
annotation appInfo
The filename or other identifier for the agent to be attached to this entity.

source
```
<xs:attribute name="type" type="xs:anyURI" use="required">
 <xs:annotation>
```

```
    <xs:appinfo>The filename or other identifier for the agent to be attached to this entity.</xs:appinfo>
  </xs:annotation>
</xs:attribute>
```

element **SMAL**

diagram



type extension of **FullyExtensibleSMALElementType**

properties content complex

children **head** **Simulation** **EntityDefinition** **TerrainTile** **StaticModelDefinition**

attributes

Name	Type	Use	Default	Fixed	Annotation
version	**xs:string**			1.0	

annotation appInfo
SAVAGE Model Analysis Language: collects data necessary to create a 3D scene out of disparate elements

source

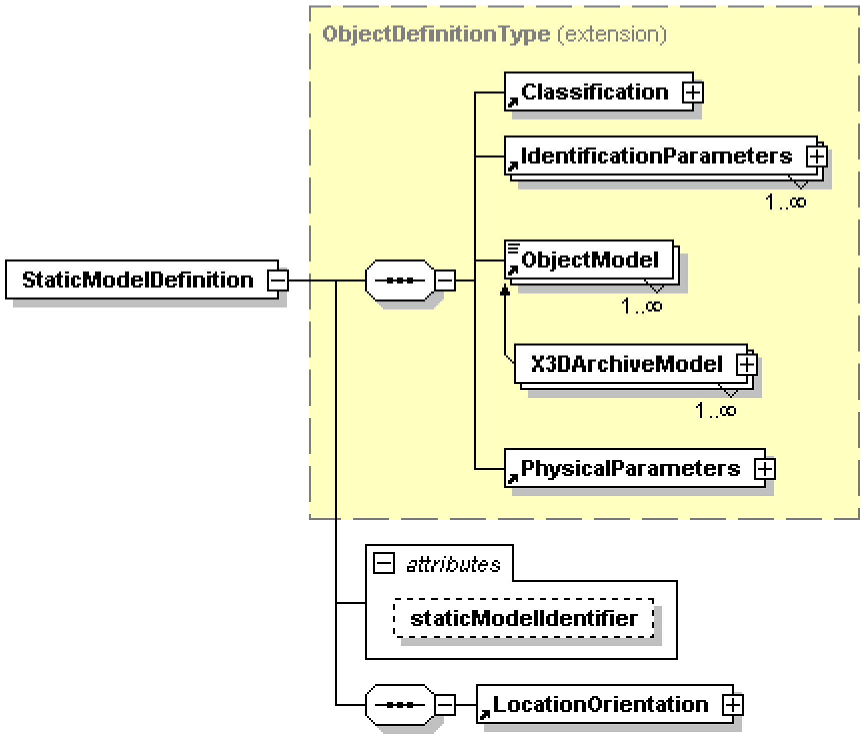
```
<xs:element name="SMAL">
  <xs:annotation>
    <xs:appinfo>SAVAGE Model Analysis Language: collects data necessary to create a 3D scene out of disparate
elements</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:sequence>
          <xs:element ref="head" minOccurs="0"/>
          <xs:choice>
            <xs:annotation>
              <xs:appinfo>A SMAL file will contain either a full-fledged Simulation or a partial/default metadata corresponding to a
simulation component file such as an entity, static model, or terrain tile.</xs:appinfo>
            </xs:annotation>
            <xs:element ref="Simulation" maxOccurs="unbounded"/>
            <xs:element ref="EntityDefinition"/>
            <xs:element ref="TerrainTile"/>
            <xs:element ref="StaticModelDefinition"/>
          </xs:choice>
        </xs:sequence>
        <xs:attribute name="version" type="xs:string" fixed="1.0"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

attribute **SMAL/@version**

type	**xs:string**
properties	isRef 0 fixed 1.0
source	<xs:attribute name="version" type="xs:string" fixed="1.0"/>

element **StaticModelDefinition**

diagram



type extension of **ObjectDefinitionType**

properties content complex

children **Classification IdentificationParameters ObjectModel PhysicalParameters LocationOrientation**

used by elements **SMAL StaticModelSet**

attributes

Name	Type	Use	Default	Fixed	Annotation
staticModelIdentifier	**xs:ID**				

annotation appInfo
Defines a single static model.

source
```
<xs:element name="StaticModelDefinition">
  <xs:annotation>
    <xs:appinfo>Defines a single static model.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="ObjectDefinitionType">
        <xs:sequence>
          <xs:element ref="LocationOrientation"/>
        </xs:sequence>
        <xs:attribute name="staticModelIdentifier" type="xs:ID"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

attribute **StaticModelDefinition/@staticModelIdentifier**

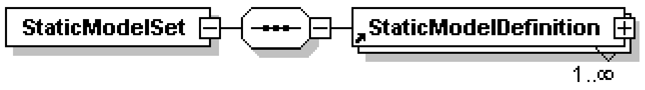
type **xs:ID**

properties isRef 0

source
```
<xs:attribute name="staticModelIdentifier" type="xs:ID"/>
```

element **StaticModelSet**

diagram



type extension of **FullyExtensibleSMALElementType**

properties content complex

children **StaticModelDefinition**

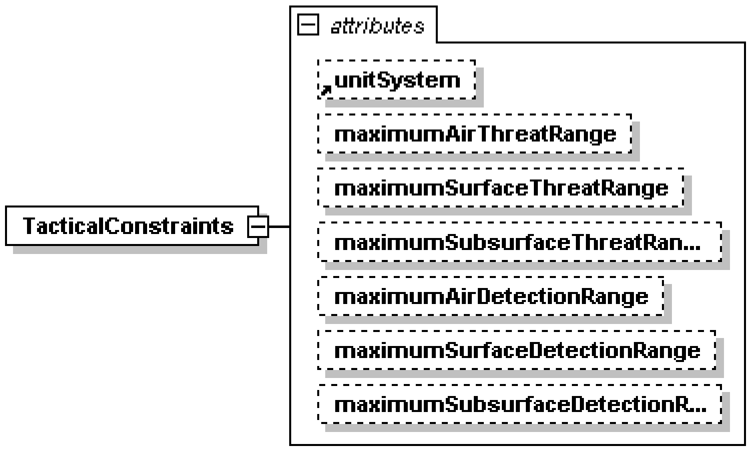
used by element **World**

annotation appInfo
These models are not entities, but rather unchanging objects in the X3D scene: buildings, signs, roads, bridges, etc.

source
```
<xs:element name="StaticModelSet">
  <xs:annotation>
    <xs:appinfo>These models are not entities, but rather unchanging objects in the X3D scene: buildings, signs, roads,
bridges, etc.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:sequence>
          <xs:element ref="StaticModelDefinition" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

element **TacticalConstraints**

diagram


attributes
unitSystem
maximumAirThreatRange
maximumSurfaceThreatRange
maximumSubsurfaceThreatRan...
maximumAirDetectionRange
maximumSurfaceDetectionRange
maximumSubsurfaceDetectionR...
TacticalConstraints

type extension of **FullyExtensibleSMALElementType**

properties content complex

used by element **PhysicalParameters**

attributes

Name	Type	Use	Default	Fixed	Annotation
unitSystem			Metric		

	maximumAirThreatRange	**xs:float**	optional
	maximumSurfaceThreatRange	**xs:float**	optional
	maximumSubsurfaceThreatRange	**xs:float**	optional
	maximumAirDetectionRange	**xs:float**	optional
	maximumSurfaceDetectionRange	**xs:float**	optional
	maximumSubsurfaceDetectionRange	**xs:float**	optional
annotation	appInfo Domain-specific threat and detection ranges for attack and defense capabilities.		

source
```
<xs:element name="TacticalConstraints">
 <xs:annotation>
  <xs:appinfo>Domain-specific threat and detection ranges for attack and defense capabilities.</xs:appinfo>
 </xs:annotation>
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="FullyExtensibleSMALElementType">
    <xs:attribute ref="unitSystem"/>
    <xs:attribute name="maximumAirThreatRange" type="xs:float" use="optional"/>
    <xs:attribute name="maximumSurfaceThreatRange" type="xs:float" use="optional"/>
    <xs:attribute name="maximumSubsurfaceThreatRange" type="xs:float" use="optional"/>
    <xs:attribute name="maximumAirDetectionRange" type="xs:float" use="optional"/>
    <xs:attribute name="maximumSurfaceDetectionRange" type="xs:float" use="optional"/>
    <xs:attribute name="maximumSubsurfaceDetectionRange" type="xs:float" use="optional"/>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
```

attribute **TacticalConstraints/@maximumAirThreatRange**

type **xs:float**

properties

isRef	0
use	optional

source
```
<xs:attribute name="maximumAirThreatRange" type="xs:float" use="optional"/>
```

attribute **TacticalConstraints/@maximumSurfaceThreatRange**

type **xs:float**

properties

isRef	0
use	optional

source
```
<xs:attribute name="maximumSurfaceThreatRange" type="xs:float" use="optional"/>
```

attribute **TacticalConstraints/@maximumSubsurfaceThreatRange**

type **xs:float**

properties

isRef	0
use	optional

source
```
<xs:attribute name="maximumSubsurfaceThreatRange" type="xs:float" use="optional"/>
```

attribute **TacticalConstraints/@maximumAirDetectionRange**

type **xs:float**

properties

isRef	0
use	optional

source
```
<xs:attribute name="maximumAirDetectionRange" type="xs:float" use="optional"/>
```

attribute **TacticalConstraints/@maximumSurfaceDetectionRange**

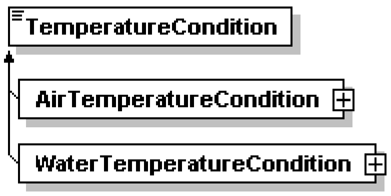
type	**xs:float**	
properties	isRef	0
	use	optional
source	`<xs:attribute name="maximumSurfaceDetectionRange" type="xs:float" use="optional"/>`	

attribute **TacticalConstraints/@maximumSubsurfaceDetectionRange**

type	**xs:float**	
properties	isRef	0
	use	optional
source	`<xs:attribute name="maximumSubsurfaceDetectionRange" type="xs:float" use="optional"/>`	

element **TemperatureCondition**

diagram



properties	abstract	true
used by	element	TemperatureConditionSet

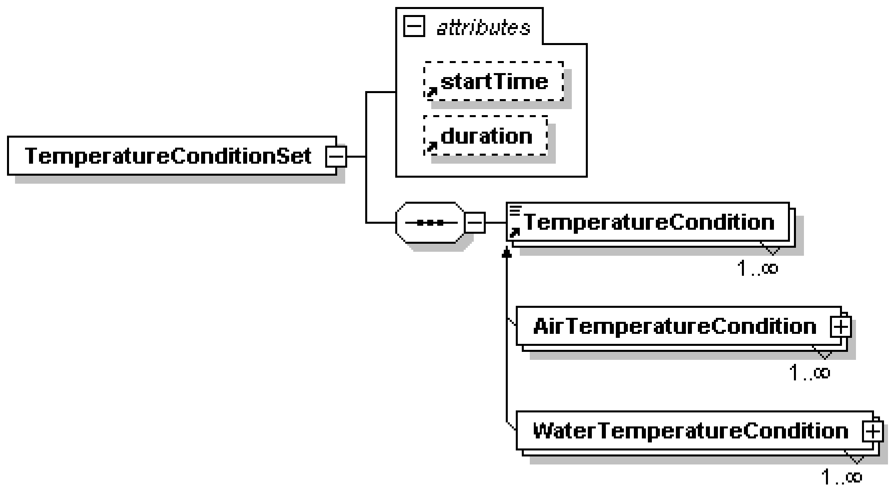
annotation
appInfo
Abstract placeholder for specific temperature elements in a TemperatureConditionSet element.

source

```
<xs:element name="TemperatureCondition" abstract="true">
  <xs:annotation>
    <xs:appinfo>Abstract placeholder for specific temperature elements in a TemperatureConditionSet element.</xs:appinfo>
  </xs:annotation>
</xs:element>
```

element **TemperatureConditionSet**

diagram



type	extension of **FullyExtensibleSMALElementType**	
properties	content	complex

children	**TemperatureCondition**
used by	element **EnvironmentalCondition**

attributes

Name	Type	Use	Default	Fixed	Annotation
startTime			00:00:00		
duration			01:00:00		

annotation

appInfo

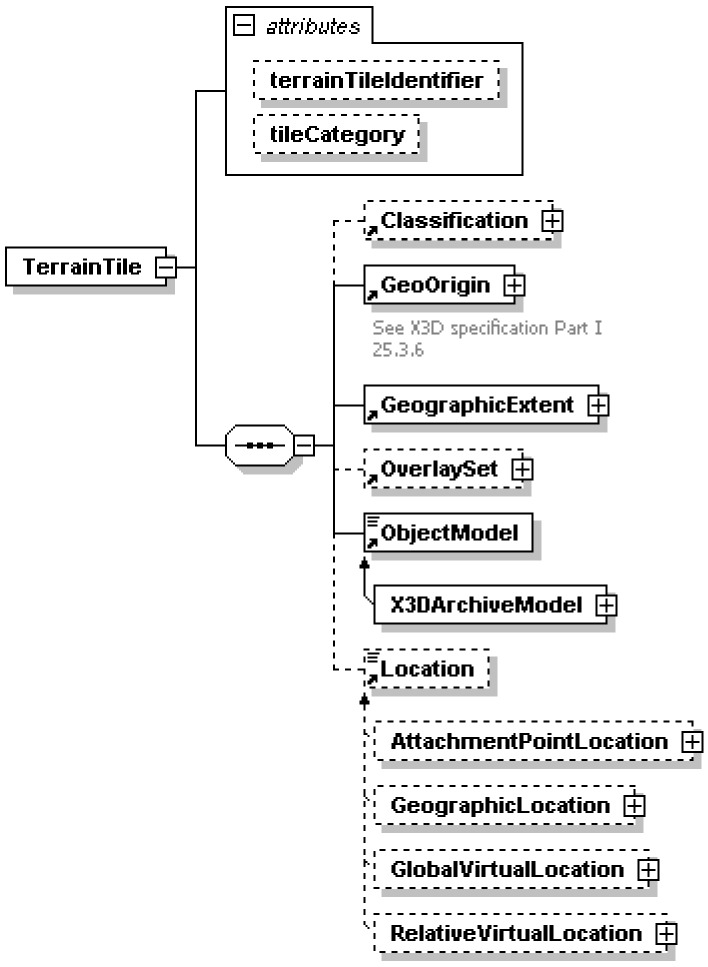
Air and water temperatures can be established based on altitude and depth. Includes a duration attribute to allow for changes over time.

source

```
<xs:element name="TemperatureConditionSet">
  <xs:annotation>
    <xs:appinfo>Air and water temperatures can be established based on altitude and depth. Includes a duration attribute to allow for changes over time.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:sequence>
          <xs:element ref="TemperatureCondition" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute ref="startTime"/>
        <xs:attribute ref="duration"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

element **TerrainTile**

diagram



type	extension of **FullyExtensibleSMALElementType**
properties	content complex
children	**Classification** **GeoOrigin** **GeographicExtent** **OverlaySet** **ObjectModel** **Location**
used by	elements **SMAL** **TerrainTileSet**

attributes

Name	Type	Use	Default	Fixed	Annotation
terrainTileIdentifier	**xs:ID**	optional			
tileCategory	**tileCategoryEnumeration**		landTerrain		

annotation

appInfo
A single terrain model containing size, vertical extent information, and references to the charts, maps, and/or image overlays associated with it.

source

```
<xs:element name="TerrainTile">
  <xs:annotation>
    <xs:appinfo>A single terrain model containing size, vertical extent information, and references to the charts, maps, and/or
image overlays associated with it.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:sequence>
          <xs:element ref="Classification" minOccurs="0"/>
          <xs:element ref="GeoOrigin"/>
          <xs:element ref="GeographicExtent"/>
          <xs:element ref="OverlaySet" minOccurs="0"/>
          <xs:element ref="ObjectModel"/>
          <xs:element ref="Location" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="terrainTileIdentifier" type="xs:ID" use="optional"/>
        <xs:attribute name="tileCategory" type="tileCategoryEnumeration" default="landTerrain"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

attribute **TerrainTile/@terrainTileIdentifier**

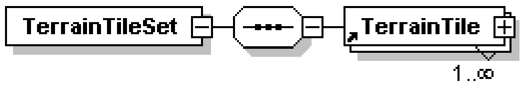
type	**xs:ID**
properties	isRef 0 use optional
source	`<xs:attribute name="terrainTileIdentifier" type="xs:ID" use="optional"/>`

attribute **TerrainTile/@tileCategory**

type	**tileCategoryEnumeration**
properties	isRef 0 default landTerrain
facets	enumeration landTerrain enumeration bathymetry enumeration planetarySurface
source	`<xs:attribute name="tileCategory" type="tileCategoryEnumeration" default="landTerrain"/>`

element **TerrainTileSet**

diagram



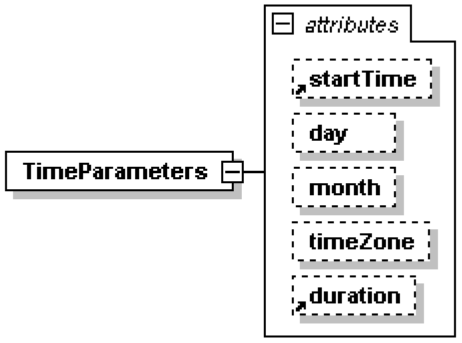
type	extension of **ElementExtensibleSMALElementType**
properties	content complex
children	**TerrainTile**
used by	element **World**
annotation	appInfo Contains the set of all Topography and Bathymetry models and their associated chart/imagery overlays.

source

```
<xs:element name="TerrainTileSet">
  <xs:annotation>
    <xs:appinfo>Contains the set of all Topography and Bathymetry models and their associated chart/imagery
overlays.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="ElementExtensibleSMALElementType">
        <xs:sequence>
          <xs:element ref="TerrainTile" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

element **TimeParameters**

diagram



type extension of **FullyExtensibleSMALElementType**

properties content complex

used by element **EnvironmentalConditionSet**

attributes

Name	Type	Use	Default	Fixed	Annotation
startTime		optional	00:00:00		
day	**dayType**	optional	1		
month	**monthType**	optional	1		
timeZone	**xs:int**	optional	-7		
duration		optional	01:00:00		

annotation appInfo
Holds temporal data about the scenario such as start time, time of year, time zone, and duration of the simulation.

source
```
<xs:element name="TimeParameters">
 <xs:annotation>
  <xs:appinfo>Holds temporal data about the scenario such as start time, time of year, time zone, and duration of the
simulation.</xs:appinfo>
 </xs:annotation>
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="FullyExtensibleSMALElementType">
    <xs:attribute ref="startTime" use="optional"/>
    <xs:attribute name="day" type="dayType" use="optional" default="1"/>
    <xs:attribute name="month" type="monthType" use="optional" default="1"/>
    <xs:attribute name="timeZone" type="xs:int" use="optional" default="-7"/>
    <xs:attribute ref="duration" use="optional"/>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
```

attribute **TimeParameters/@day**

type **dayType**

properties

isRef	0
default	1
use	optional

facets

minInclusive	1
maxInclusive	31

source
```
<xs:attribute name="day" type="dayType" use="optional" default="1"/>
```

attribute **TimeParameters/@month**

type **monthType**

properties

isRef	0
default	1
use	optional

facets

minInclusive	1
maxInclusive	12

source `<xs:attribute name="month" type="monthType" use="optional" default="1"/>`

attribute **TimeParameters/@timeZone**

type **xs:int**

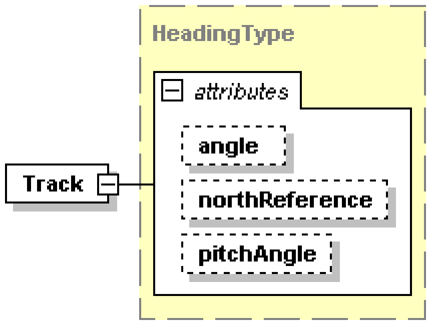
properties

isRef	0
default	-7
use	optional

source `<xs:attribute name="timeZone" type="xs:int" use="optional" default="-7"/>`

element **Track**

diagram



type **HeadingType**

properties

content	complex
substGrp	DirectionOfMotion

attributes

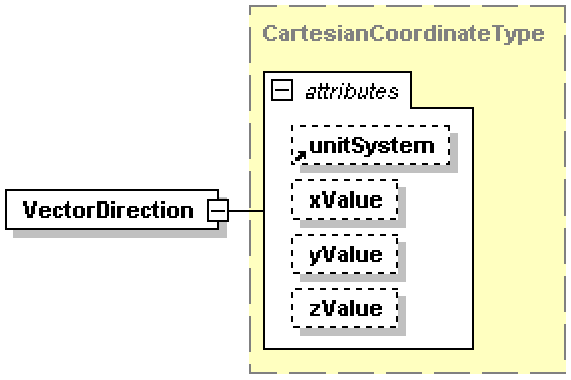
Name	Type	Use	Default	Fixed	Annotation
angle	**xs:float**		0		
northReference	**northReferenceEnumeration**		MAGNETIC		
pitchAngle	**xs:float**		0		

annotation appInfo
The current direction of travel of this object.

source
```
<xs:element name="Track" type="HeadingType" substitutionGroup="DirectionOfMotion">
  <xs:annotation>
    <xs:appinfo>The current direction of travel of this object.</xs:appinfo>
  </xs:annotation>
</xs:element>
```

element **VectorDirection**

diagram



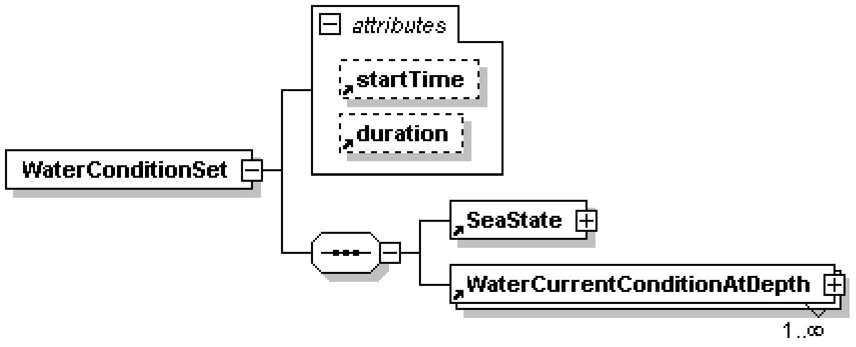
type **CartesianCoordinateType**

properties	content	complex				
	substGrp	DirectionOfMotion				
attributes	Name	Type	Use	Default	Fixed	Annotation
	unitSystem			Metric		
	xValue	**xs:float**		0		
	yValue	**xs:float**		0		
	zValue	**xs:float**		0		

source
```
<xs:element name="VectorDirection" type="CartesianCoordinateType" substitutionGroup="DirectionOfMotion"/>
```

element **WaterConditionSet**

diagram



type extension of **FullyExtensibleSMALElementType**

properties content complex

children **SeaState** **WaterCurrentConditionAtDepth**

used by element **EnvironmentalCondition**

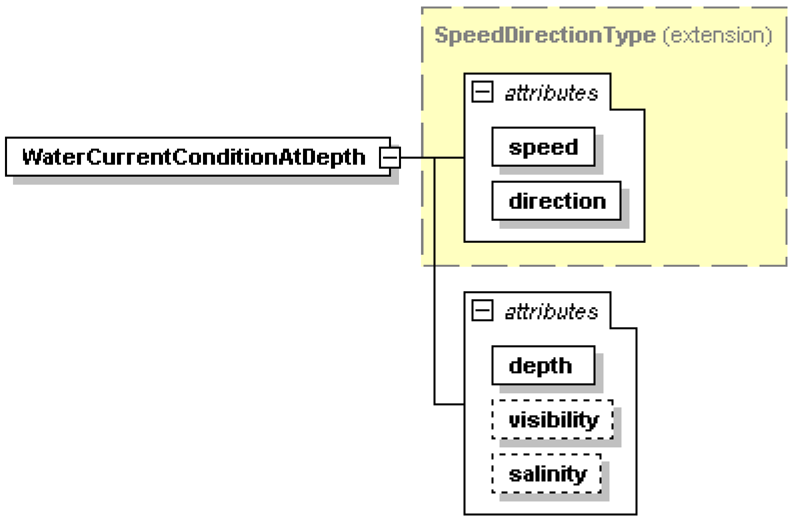
attributes	Name	Type	Use	Default	Fixed	Annotation
	startTime			00:00:00		
	duration			01:00:00		

annotation appInfo
Collects the sea state and all water current condition elements for a single EnvironmentalCondition element. Currents may change over the course of a single EnvironmentalCondition, so there is a duration attribute.

source
```
<xs:element name="WaterConditionSet">
 <xs:annotation>
  <xs:appinfo>Collects the sea state and all water current condition elements for a single EnvironmentalCondition element.
Currents may change over the course of a single EnvironmentalCondition, so there is a duration attribute.</xs:appinfo>
 </xs:annotation>
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="FullyExtensibleSMALElementType">
    <xs:sequence>
     <xs:element ref="SeaState"/>
     <xs:element ref="WaterCurrentConditionAtDepth" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute ref="startTime"/>
    <xs:attribute ref="duration"/>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
```

element **WaterCurrentConditionAtDepth**

diagram



type extension of **SpeedDirectionType**

properties content complex

used by element **WaterConditionSet**

attributes

Name	Type	Use	Default	Fixed	Annotation
speed	**nonNegativeFloat**	required			
direction	**degreeType**	required			
depth	**nonNegativeFloat**	required			
visibility	**nonNegativeFloat**	optional	0.0		
salinity	**nonNegativeFloat**	optional	0.0		

annotation appInfo
Water currents are defined by depth, speed, and direction. Additional data for salinity and visibility are included. The value for depth represents the shallowest depth at which this condition exists. Water currents, salinity, and visibility by depth are considered persistent over the course of an EnvironmentalCondition. Changes can be made by using multiple WaterCurrentConditionSets.

source
```
<xs:element name="WaterCurrentConditionAtDepth">
  <xs:annotation>
    <xs:appinfo>Water currents are defined by depth, speed, and direction.  Additional data for salinity and visibility are
included.  The value for depth represents the shallowest depth at which this condition exists. Water currents, salinity, and
visibility by depth are considered persistent over the course of an EnvironmentalCondition.  Changes can be made by using
multiple WaterCurrentConditionSets.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="SpeedDirectionType">
        <xs:attribute name="depth" type="nonNegativeFloat" use="required"/>
        <xs:attribute name="visibility" type="nonNegativeFloat" use="optional" default="0.0"/>
        <xs:attribute name="salinity" type="nonNegativeFloat" use="optional" default="0.0"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

attribute **WaterCurrentConditionAtDepth/@depth**

type **nonNegativeFloat**

properties isRef 0
use required

facets minInclusive 0.0

source
```
<xs:attribute name="depth" type="nonNegativeFloat" use="required"/>
```

attribute **WaterCurrentConditionAtDepth/@visibility**

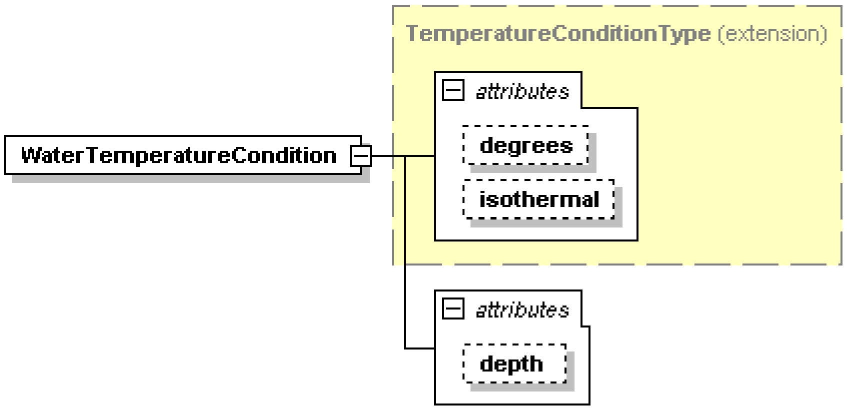
type	**nonNegativeFloat**
properties	isRef 0 default 0.0 use optional
facets	minInclusive 0.0
source	`<xs:attribute name="visibility" type="nonNegativeFloat" use="optional" default="0.0"/>`

attribute **WaterCurrentConditionAtDepth/@salinity**

type	**nonNegativeFloat**
properties	isRef 0 default 0.0 use optional
facets	minInclusive 0.0
source	`<xs:attribute name="salinity" type="nonNegativeFloat" use="optional" default="0.0"/>`

element **WaterTemperatureCondition**

diagram


TemperatureConditionType (extension)
attributes
degrees
isothermal
WaterTemperatureCondition
attributes
depth

type extension of **TemperatureConditionType**

properties

content	complex
substGrp	TemperatureCondition

attributes

Name	Type	Use	Default	Fixed	Annotation
degrees	**xs:float**		15		
isothermal	**xs:boolean**		false		
depth	**nonNegativeFloat**		0		

annotation

appInfo

Holds the water temperature and the depth at which this temperature occurs. Temperature is assumed to decrease as depth increases unless isothermal is set to true.

source

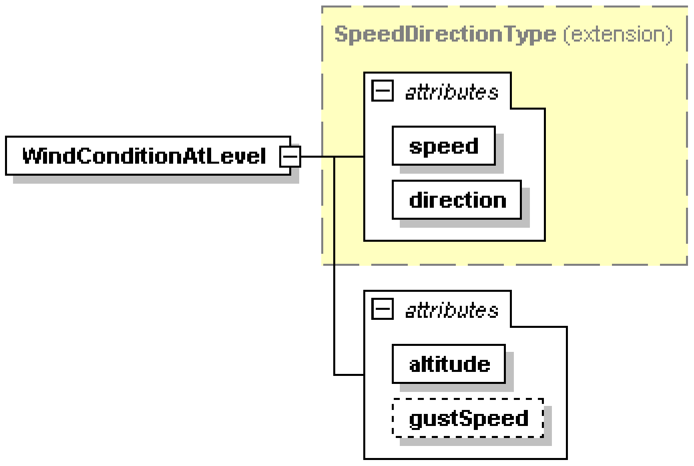
```
<xs:element name="WaterTemperatureCondition" substitutionGroup="TemperatureCondition">
  <xs:annotation>
    <xs:appinfo>Holds the water temperature and the depth at which this temperature occurs. Temperature is assumed to
decrease as depth increases unless isothermal is set to true.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="TemperatureConditionType">
        <xs:attribute name="depth" type="nonNegativeFloat" default="0"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

attribute **WaterTemperatureCondition/@depth**

type	**nonNegativeFloat**
properties	isRef 0 default 0
facets	minInclusive 0.0
source	`<xs:attribute name="depth" type="nonNegativeFloat" default="0"/>`

element **WindConditionAtLevel**

diagram



type extension of **SpeedDirectionType**

properties content complex

used by element **WindConditionSet**

attributes

Name	Type	Use	Default	Fixed	Annotation
speed	**nonNegativeFloat**	required			
direction	**degreeType**	required			
altitude	**nonNegativeFloat**	required			
gustSpeed	**nonNegativeFloat**	optional	0.0		

annotation appInfo
Winds are defined by altitude, speed, maximum gust speed, and direction. The value for altitude represents the lowest altitude at which this condition exists. Wind conditions at a given altitude are considered persistent over the course of an EnvironmentCondition. Changes are made using multiple WindConditionSets.

source
```
<xs:element name="WindConditionAtLevel">
 <xs:annotation>
  <xs:appinfo>Winds are defined by altitude, speed, maximum gust speed, and direction. The value for altitude represents the lowest altitude at which this condition exists.  Wind conditions at a given altitude are considered persistent over the course of an EnvironmentCondition.  Changes are made using multiple WindConditionSets.</xs:appinfo>
 </xs:annotation>
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="SpeedDirectionType">
    <xs:attribute name="altitude" type="nonNegativeFloat" use="required"/>
    <xs:attribute name="gustSpeed" type="nonNegativeFloat" use="optional" default="0.0"/>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
```

attribute **WindConditionAtLevel/@altitude**

type **nonNegativeFloat**

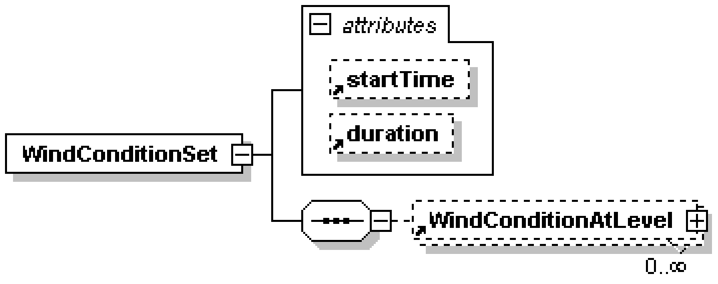
properties	isRef 0 use required
facets	minInclusive 0.0
source	`<xs:attribute name="altitude" type="nonNegativeFloat" use="required"/>`

attribute **WindConditionAtLevel/@gustSpeed**

type	**nonNegativeFloat**
properties	isRef 0 default 0.0 use optional
facets	minInclusive 0.0
source	`<xs:attribute name="gustSpeed" type="nonNegativeFloat" use="optional" default="0.0"/>`

element **WindConditionSet**

diagram



type	extension of **FullyExtensibleSMALElementType**
properties	content complex
children	**WindConditionAtLevel**
used by	element **EnvironmentalCondition**

attributes

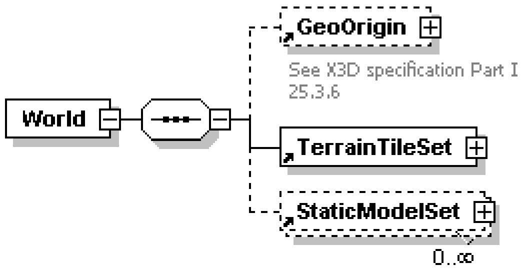
Name	Type	Use	Default	Fixed	Annotation
startTime			00:00:00		
duration			01:00:00		

source

```
<xs:element name="WindConditionSet">
 <xs:complexType>
  <xs:annotation>
   <xs:appinfo>Collects all wind condition elements for a single EnvironmentalCondition element.</xs:appinfo>
  </xs:annotation>
  <xs:complexContent>
   <xs:extension base="FullyExtensibleSMALElementType">
    <xs:sequence>
     <xs:element ref="WindConditionAtLevel" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute ref="startTime"/>
    <xs:attribute ref="duration"/>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
```

element **World**

diagram



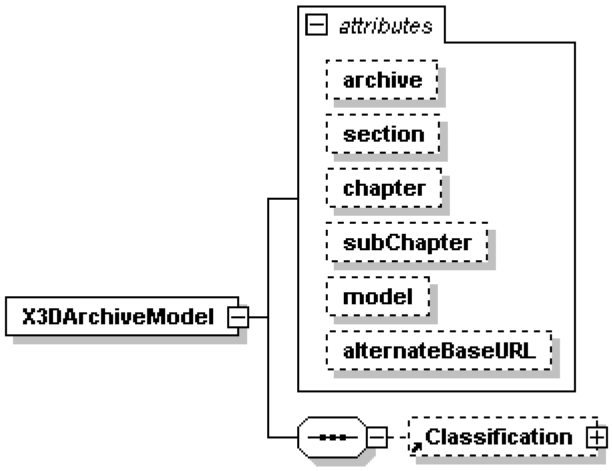
type	extension of **FullyExtensibleSMALElementType**
properties	content complex
children	**GeoOrigin** **TerrainTileSet** **StaticModelSet**
used by	element **Simulation**
annotation	appInfo Defines the largely static environment in which the simulation will run. It will contain all terrain features (terrain and bathymetry), associated charts or maps, and any buildings or other significant model data which will not be tracked for movement or other actions.

source

```
<xs:element name="World">
  <xs:annotation>
    <xs:appinfo>Defines the largely static environment in which the simulation will run. It will contain all terrain features
(terrain and bathymetry), associated charts or maps, and any buildings or other significant model data which will not be
tracked for movement or other actions.</xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:sequence>
          <xs:element ref="GeoOrigin" minOccurs="0"/>
          <xs:element ref="TerrainTileSet"/>
          <xs:element ref="StaticModelSet" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

element **X3DArchiveModel**

diagram



type	extension of **FullyExtensibleSMALElementType**
properties	content complex substGrp ObjectModel

children **Classification**

attributes

Name	Type	Use	Default	Fixed	Annotation
archive	**modelArchiveEnumeration**		Savage		
section	**xs:NMTOKEN**				
chapter	**xs:NMTOKEN**				
subChapter	**xs:NMTOKEN**				
model	**xs:NMTOKEN**				
alternateBaseURL	**xs:anyURI**		http://www.web3d.org/x3d/content/examples/		appInfo Required only when using non-SAVAGE archive x3d models.

source
```
<xs:element name="X3DArchiveModel" substitutionGroup="ObjectModel">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="FullyExtensibleSMALElementType">
        <xs:sequence>
          <xs:element ref="Classification" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="archive" type="modelArchiveEnumeration" default="Savage"/>
        <xs:attribute name="section" type="xs:NMTOKEN"/>
        <xs:attribute name="chapter" type="xs:NMTOKEN"/>
        <xs:attribute name="subChapter" type="xs:NMTOKEN"/>
        <xs:attribute name="model" type="xs:NMTOKEN"/>
        <xs:attribute name="alternateBaseURL" type="xs:anyURI" default="http://www.web3d.org/x3d/content/examples/">
          <xs:annotation>
            <xs:appinfo>Required only when using non-SAVAGE archive x3d models.</xs:appinfo>
          </xs:annotation>
        </xs:attribute>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

attribute **X3DArchiveModel/@archive**

type **modelArchiveEnumeration**

properties

isRef	0
default	Savage

facets

enumeration	Savage
enumeration	SavageDefense
enumeration	X3dExamples
enumeration	X3dConformanceNist
enumeration	Other

source
```
<xs:attribute name="archive" type="modelArchiveEnumeration" default="Savage"/>
```

attribute **X3DArchiveModel/@section**

type **xs:NMTOKEN**

properties isRef 0

source
```
<xs:attribute name="section" type="xs:NMTOKEN"/>
```

attribute **X3DArchiveModel/@chapter**

type **xs:NMTOKEN**

properties isRef 0

source
```
<xs:attribute name="chapter" type="xs:NMTOKEN"/>
```

attribute **X3DArchiveModel/@subChapter**

type **xs:NMTOKEN**

properties	isRef	0

source `<xs:attribute name="subChapter" type="xs:NMTOKEN"/>`

attribute **X3DArchiveModel/@model**

type **xs:NMTOKEN**

properties	isRef	0

source `<xs:attribute name="model" type="xs:NMTOKEN"/>`

attribute **X3DArchiveModel/@alternateBaseURL**

type **xs:anyURI**

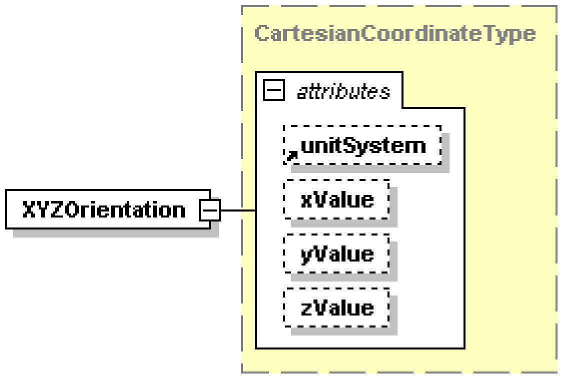
properties	isRef	0
	default	http://www.web3d.org/x3d/content/examples/

annotation appInfo
Required only when using non-SAVAGE archive x3d
models.

source
```
<xs:attribute name="alternateBaseURL" type="xs:anyURI" default="http://www.web3d.org/x3d/content/examples/">
  <xs:annotation>
    <xs:appinfo>Required only when using non-SAVAGE archive x3d models.</xs:appinfo>
  </xs:annotation>
</xs:attribute>
```

element **XYZOrientation**

diagram



type **CartesianCoordinateType**

properties	content	complex
	substGrp	Orientation

attributes

Name	Type	Use	Default	Fixed	Annotation
unitSystem			Metric		
xValue	**xs:float**		0		
yValue	**xs:float**		0		
zValue	**xs:float**		0		

annotation appInfo
A simple set of orthogonal angles based on the current reference coordinate axes. Use fractional degree measurements for the angles vice radians.

source
```
<xs:element name="XYZOrientation" type="CartesianCoordinateType" substitutionGroup="Orientation">
  <xs:annotation>
    <xs:appinfo>A simple set of orthogonal angles based on the current reference  coordinate axes. Use fractional degree
measurements for the angles vice radians.</xs:appinfo>
  </xs:annotation>
</xs:element>
```

GLOBAL DEFINITIONS

complexType **AttributeExtensibleSMALElementType**

diagram

properties   abstract   true

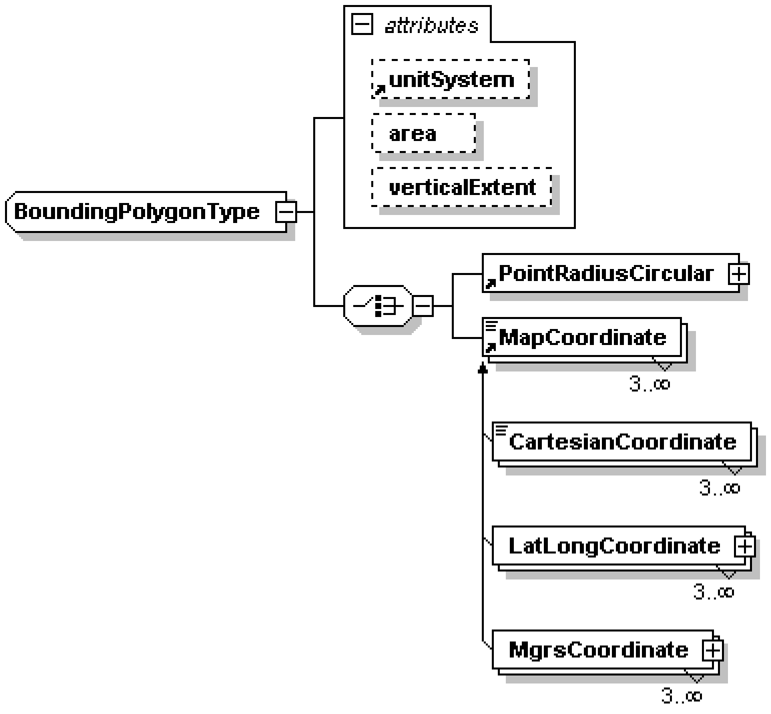
used by   elements   **AttachmentPointLocation** **LatLongCoordinate** **MgrsCoordinate**
complexTypes   **CartesianCoordinateType** **TemperatureConditionType**

source

```
<xs:complexType name="AttributeExtensibleSMALElementType" abstract="true">
  <!--xs:anyAttribute namespace="other" processContents="lax"/-->
</xs:complexType>
```

complexType **BoundingPolygonType**

diagram



type   extension of **FullyExtensibleSMALElementType**

properties   base   FullyExtensibleSMALElementType

children   **PointRadiusCircular** **MapCoordinate**

used by   element   **GeographicExtent**

attributes

Name	Type	Use	Default	Fixed	Annotation
unitSystem			Metric		
area	**xs:float**	optional	0.0		
verticalExtent	**xs:float**	optional	0.0		

annotation   appInfo
A base complexType used to describe a two-and-a-half dimensional shape for the purposes of establishing boundaries. Description of the shape is either by defining the points of a polygon or by establishing a centerpoint and radius for a circular boundary.

source

```
<xs:complexType name="BoundingPolygonType">
  <xs:annotation>
    <xs:appinfo>A base complexType used to describe a two-and-a-half dimensional shape for the purposes of establishing
boundaries.  Description of the shape is either by defining the points of a polygon or by establishing a centerpoint and radius
for a circular boundary.</xs:appinfo>
```

```
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="FullyExtensibleSMALElementType">
      <xs:choice>
        <xs:element ref="PointRadiusCircular"/>
        <xs:element ref="MapCoordinate" minOccurs="3" maxOccurs="unbounded"/>
      </xs:choice>
      <xs:attribute ref="unitSystem"/>
      <xs:attribute name="area" type="xs:float" use="optional" default="0.0"/>
      <xs:attribute name="verticalExtent" type="xs:float" use="optional" default="0.0"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

attribute **BoundingPolygonType/@area**

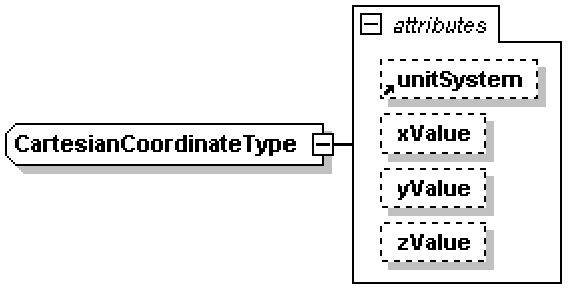
type	**xs:float**
properties	isRef 0 default 0.0 use optional
source	`<xs:attribute name="area" type="xs:float" use="optional" default="0.0"/>`

attribute **BoundingPolygonType/@verticalExtent**

type	**xs:float**
properties	isRef 0 default 0.0 use optional
source	`<xs:attribute name="verticalExtent" type="xs:float" use="optional" default="0.0"/>`

complexType **CartesianCoordinateType**

diagram



type extension of **AttributeExtensibleSMALElementType**

properties

base	AttributeExtensibleSMALElementType
final	restriction

used by elements **GlobalVirtualLocation** **RelativeVirtualLocation** **VectorDirection** **XYZOrientation**

attributes

Name	Type	Use	Default	Fixed	Annotation
unitSystem			Metric		
xValue	**xs:float**		0		
yValue	**xs:float**		0		
zValue	**xs:float**		0		

annotation appInfo
A complexType used to describe a three-value coordinate location.

source

```
<xs:complexType name="CartesianCoordinateType" final="restriction">
  <xs:annotation>
    <xs:appinfo>A complexType used to describe a three-value coordinate location.</xs:appinfo>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AttributeExtensibleSMALElementType">
      <xs:attribute ref="unitSystem"/>
```

```
        <xs:attribute name="xValue" type="xs:float" default="0"/>
        <xs:attribute name="yValue" type="xs:float" default="0"/>
        <xs:attribute name="zValue" type="xs:float" default="0"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

attribute **CartesianCoordinateType/@xValue**

type	**xs:float**	
properties	isRef	0
	default	0
source	`<xs:attribute name="xValue" type="xs:float" default="0"/>`	

attribute **CartesianCoordinateType/@yValue**

type	**xs:float**	
properties	isRef	0
	default	0
source	`<xs:attribute name="yValue" type="xs:float" default="0"/>`	

attribute **CartesianCoordinateType/@zValue**

type	**xs:float**	
properties	isRef	0
	default	0
source	`<xs:attribute name="zValue" type="xs:float" default="0"/>`	

complexType **ElementExtensibleSMALElementType**

diagram

properties abstract true

used by element **TerrainTileSet**

source
```
<xs:complexType name="ElementExtensibleSMALElementType" abstract="true">
  <!--
                                <xs:sequence>
                                        <xs:any namespace="other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                                </xs:sequence>
-->
</xs:complexType>
```

complexType **FullyExtensibleSMALElementType**

diagram

properties abstract true

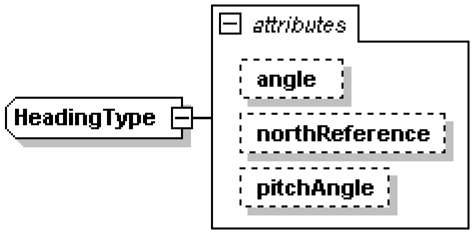
used by elements **AssociatedEntity Association AssociationSet AttachmentPoint AttachmentPointSet Background BehaviorParameterSet Classification CloudConditionSet CurrentConditionParameters DisConfiguration DynamicResponseConstraints EntitySet EnvironmentalCondition EnvironmentalConditionSet GeographicLocation GeoOrigin IdentificationParameters LocalPrecipitation LocationOrientation NetworkChannel NetworkedCommunicationParameterSet OverlaySet Parameter PhysicalConstraints**

complexTypes	**PhysicalParameters PointRadiusCircular PrecipitationConditionSet SeaState Simulation SimulationAgent SMAL StaticModelSet TacticalConstraints TemperatureConditionSet TerrainTile TimeParameters WaterConditionSet WindConditionSet World X3DArchiveModel BoundingPolygonType HeadingType ObjectDefinitionType OverlayImageDescriptorType QuaternionType SpeedDirectionType**

source
```
<xs:complexType name="FullyExtensibleSMALElementType" abstract="true">
 <!--xs:sequence>
        <xs:any maxOccurs="unbounded" minOccurs="0" namespace="##other" processContents="lax"/>
 </xs:sequence>
 <xs:anyAttribute namespace="other" processContents="lax"/-->
</xs:complexType>
```

complexType **HeadingType**

diagram



type: extension of **FullyExtensibleSMALElementType**

properties: base FullyExtensibleSMALElementType

used by: elements **Heading Track**

attributes

Name	Type	Use	Default	Fixed	Annotation
angle	**xs:float**		0		
northReference	**northReferenceEnumeration**		MAGNETIC		
pitchAngle	**xs:float**		0		

source
```
<xs:complexType name="HeadingType">
 <xs:complexContent>
  <xs:extension base="FullyExtensibleSMALElementType">
   <xs:attribute name="angle" type="xs:float" default="0"/>
   <xs:attribute name="northReference" type="northReferenceEnumeration" default="MAGNETIC"/>
   <xs:attribute name="pitchAngle" type="xs:float" default="0"/>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

attribute **HeadingType/@angle**

type: **xs:float**

properties

isRef	0
default	0

source
```
<xs:attribute name="angle" type="xs:float" default="0"/>
```

attribute **HeadingType/@northReference**

type: **northReferenceEnumeration**

properties

isRef	0
default	MAGNETIC

facets

enumeration	TRUE
enumeration	MAGNETIC

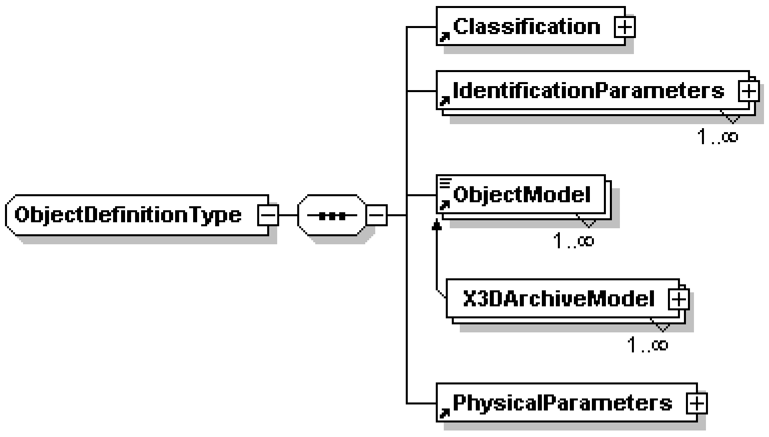
source
```
<xs:attribute name="northReference" type="northReferenceEnumeration" default="MAGNETIC"/>
```

attribute **HeadingType/@pitchAngle**

type	**xs:float**
properties	isRef 0 default 0
source	`<xs:attribute name="pitchAngle" type="xs:float" default="0"/>`

complexType **ObjectDefinitionType**

diagram



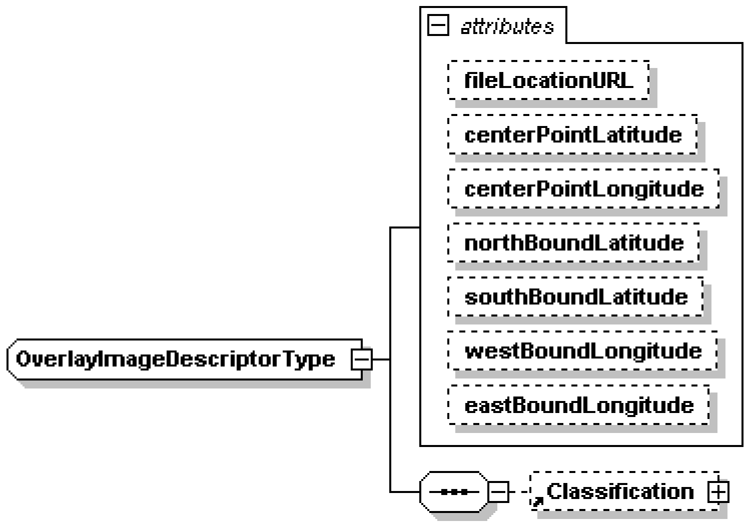
type	extension of **FullyExtensibleSMALElementType**
properties	base FullyExtensibleSMALElementType
children	**Classification** **IdentificationParameters** **ObjectModel** **PhysicalParameters**
used by	elements **EntityDefinition** **StaticModelDefinition**
annotation	appInfo Super-type which contains the base elements for all non-terrain models, including identification, 3D model, and physical parameters.

source

```
<xs:complexType name="ObjectDefinitionType">
  <xs:annotation>
    <xs:appinfo>Super-type which contains the base elements for all non-terrain models, including identification, 3D model,
and physical parameters.</xs:appinfo>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="FullyExtensibleSMALElementType">
      <xs:sequence>
        <xs:element ref="Classification"/>
        <xs:element ref="IdentificationParameters" maxOccurs="unbounded"/>
        <xs:element ref="ObjectModel" maxOccurs="unbounded"/>
        <xs:element ref="PhysicalParameters"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

complexType **OverlayImageDescriptorType**

diagram



type extension of **FullyExtensibleSMALElementType**

properties base FullyExtensibleSMALElementType

children **Classification**

used by elements **OverlaySetChart** **OverlaySetImagery** **OverlaySetMap**

attributes

Name	Type	Use	Default	Fixed	Annotation
fileLocationURL	**xs:anyURI**		http://www.web3d.org/x3d/content/examples/		
centerPointLatitude	**latitudeOrdinatePattern**	optional	N00 00.0		
centerPointLongitude	**longitudeOrdinatePattern**	optional	W00 00.0		
northBoundLatitude	**latitudeOrdinatePattern**	optional	N00 00.0		
southBoundLatitude	**latitudeOrdinatePattern**	optional	N00 00.0		
westBoundLongitude	**longitudeOrdinatePattern**	optional	W00 00.0		
eastBoundLongitude	**longitudeOrdinatePattern**	optional	W00 00.0		

annotation appInfo
Super-type used to describe an image overlay for a TerrainTile. The image is referenced by a URI, and its geographic location and extent is described using centerpoint and bounding latitudes/longitudes.

source
```
<xs:complexType name="OverlayImageDescriptorType">
 <xs:annotation>
  <xs:appinfo>Super-type used to describe an image overlay for a TerrainTile.  The image is referenced by a URI, and its geographic
location and extent is described using centerpoint and bounding latitudes/longitudes.</xs:appinfo>
 </xs:annotation>
 <xs:complexContent>
  <xs:extension base="FullyExtensibleSMALElementType">
   <xs:sequence>
    <xs:element ref="Classification" minOccurs="0"/>
   </xs:sequence>
   <xs:attribute name="fileLocationURL" type="xs:anyURI" default="http://www.web3d.org/x3d/content/examples/"/>
   <xs:attribute name="centerPointLatitude" type="latitudeOrdinatePattern" use="optional" default="N00 00.0"/>
   <xs:attribute name="centerPointLongitude" type="longitudeOrdinatePattern" use="optional" default="W00 00.0"/>
   <xs:attribute name="northBoundLatitude" type="latitudeOrdinatePattern" use="optional" default="N00 00.0"/>
   <xs:attribute name="southBoundLatitude" type="latitudeOrdinatePattern" use="optional" default="N00 00.0"/>
   <xs:attribute name="westBoundLongitude" type="longitudeOrdinatePattern" use="optional" default="W00 00.0"/>
   <xs:attribute name="eastBoundLongitude" type="longitudeOrdinatePattern" use="optional" default="W00 00.0"/>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

attribute **OverlayImageDescriptorType/@fileLocationURL**

type **xs:anyURI**

properties isRef 0

default http://www.web3d.org/x3d/content/examples/

source
```
<xs:attribute name="fileLocationURL" type="xs:anyURI" default="http://www.web3d.org/x3d/content/examples/"/>
```

attribute **OverlayImageDescriptorType/@centerPointLatitude**

type **latitudeOrdinatePattern**

properties
isRef 0
default N00 00.0
use optional

facets
pattern (((N|n)|(S|s))(([0-8]?[0-9])|90) ([0-5]?[0-9])?(\.[0-9]*))

source
```
<xs:attribute name="centerPointLatitude" type="latitudeOrdinatePattern" use="optional" default="N00 00.0"/>
```

attribute **OverlayImageDescriptorType/@centerPointLongitude**

type **longitudeOrdinatePattern**

properties
isRef 0
default W00 00.0
use optional

facets
pattern (((E|e)|(W|w))([0-9]?[0-9]|1[0-8][0-9]) ([0-5]?[0-9])?(\.[0-9]*))

source
```
<xs:attribute name="centerPointLongitude" type="longitudeOrdinatePattern" use="optional" default="W00 00.0"/>
```

attribute **OverlayImageDescriptorType/@northBoundLatitude**

type **latitudeOrdinatePattern**

properties
isRef 0
default N00 00.0
use optional

facets
pattern (((N|n)|(S|s))(([0-8]?[0-9])|90) ([0-5]?[0-9])?(\.[0-9]*))

source
```
<xs:attribute name="northBoundLatitude" type="latitudeOrdinatePattern" use="optional" default="N00 00.0"/>
```

attribute **OverlayImageDescriptorType/@southBoundLatitude**

type **latitudeOrdinatePattern**

properties
isRef 0
default N00 00.0
use optional

facets
pattern (((N|n)|(S|s))(([0-8]?[0-9])|90) ([0-5]?[0-9])?(\.[0-9]*))

source
```
<xs:attribute name="southBoundLatitude" type="latitudeOrdinatePattern" use="optional" default="N00 00.0"/>
```

attribute **OverlayImageDescriptorType/@westBoundLongitude**

type **longitudeOrdinatePattern**

properties
isRef 0
default W00 00.0
use optional

facets
pattern (((E|e)|(W|w))([0-9]?[0-9]|1[0-8][0-9]) ([0-5]?[0-9])?(\.[0-9]*))

source
```
<xs:attribute name="westBoundLongitude" type="longitudeOrdinatePattern" use="optional" default="W00 00.0"/>
```

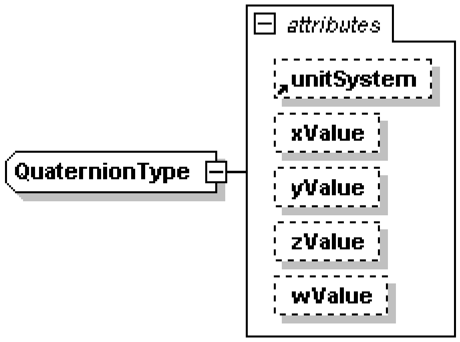
attribute **OverlayImageDescriptorType/@eastBoundLongitude**

type	**longitudeOrdinatePattern**	
properties	isRef	0
	default	W00 00.0
	use	optional
facets	pattern	(((E|e)|(W|w))([0-9]?[0-9]|1[0-8][0-9]) ([0-5]?[0-9])?(\.[0-9]*))

source
```
<xs:attribute name="eastBoundLongitude" type="longitudeOrdinatePattern" use="optional" default="W00 00.0"/>
```

complexType **QuaternionType**

diagram



type	extension of **FullyExtensibleSMALElementType**	
properties	base	FullyExtensibleSMALElementType
used by	elements	**QuaternionDirection** **QuaternionOrientation**

attributes

Name	Type	Use	Default	Fixed	Annotation
unitSystem			Metric		
xValue	**unitValueFloat**		0		
yValue	**unitValueFloat**		0		
zValue	**unitValueFloat**		0		
wValue	**nonNegativeFloat**		0		

source
```
<xs:complexType name="QuaternionType">
 <xs:complexContent>
  <xs:extension base="FullyExtensibleSMALElementType">
   <xs:attribute ref="unitSystem"/>
   <xs:attribute name="xValue" type="unitValueFloat" default="0"/>
   <xs:attribute name="yValue" type="unitValueFloat" default="0"/>
   <xs:attribute name="zValue" type="unitValueFloat" default="0"/>
   <xs:attribute name="wValue" type="nonNegativeFloat" default="0"/>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

attribute **QuaternionType/@xValue**

type	**unitValueFloat**	
properties	isRef	0
	default	0
facets	minInclusive	-1.0
	maxInclusive	1.0

source
```
<xs:attribute name="xValue" type="unitValueFloat" default="0"/>
```

attribute **QuaternionType/@yValue**

type	**unitValueFloat**	
properties	isRef	0
	default	0

facets	minInclusive	-1.0
	maxInclusive	1.0

source `<xs:attribute name="yValue" type="unitValueFloat" default="0"/>`

attribute **QuaternionType/@zValue**

type **unitValueFloat**

properties	isRef	0
	default	0
facets	minInclusive	-1.0
	maxInclusive	1.0

source `<xs:attribute name="zValue" type="unitValueFloat" default="0"/>`

attribute **QuaternionType/@wValue**

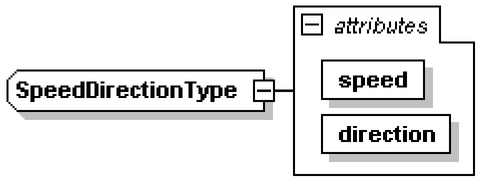
type **nonNegativeFloat**

properties	isRef	0
	default	0
facets	minInclusive	0.0

source `<xs:attribute name="wValue" type="nonNegativeFloat" default="0"/>`

complexType **SpeedDirectionType**

diagram



type extension of **FullyExtensibleSMALElementType**

properties base FullyExtensibleSMALElementType

used by elements **WaterCurrentConditionAtDepth** **WindConditionAtLevel**

attributes

Name	Type	Use	Default	Fixed	Annotation
speed	**nonNegativeFloat**	required			
direction	**degreeType**	required			

annotation appInfo
Super-type containing the base attributes for elements which describe the speed and direction of an object or phenomena using attribute values.

source
```
<xs:complexType name="SpeedDirectionType">
 <xs:annotation>
  <xs:appinfo>Super-type containing the base attributes for elements which describe the speed and direction of an object or
phenomena using attribute values.</xs:appinfo>
 </xs:annotation>
 <xs:complexContent>
  <xs:extension base="FullyExtensibleSMALElementType">
   <xs:attribute name="speed" type="nonNegativeFloat" use="required"/>
   <xs:attribute name="direction" type="degreeType" use="required"/>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

attribute **SpeedDirectionType/@speed**

type **nonNegativeFloat**

properties	isRef	0
	use	required

facets	minInclusive	0.0

source
```
<xs:attribute name="speed" type="nonNegativeFloat" use="required"/>
```

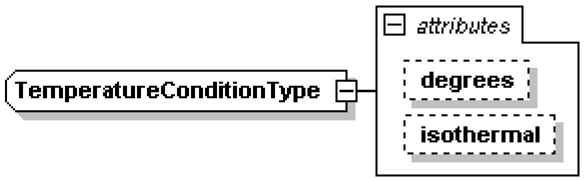
attribute **SpeedDirectionType/@direction**

type	[degreeType](#)	
properties	isRef	0
	use	required
facets	minInclusive	0
	maxExclusive	360

source
```
<xs:attribute name="direction" type="degreeType" use="required"/>
```

complexType **TemperatureConditionType**

diagram



type: extension of **[AttributeExtensibleSMALElementType](#)**

properties	base	AttributeExtensibleSMALElementType
used by	elements	**[AirTemperatureCondition](#)** **[WaterTemperatureCondition](#)**

attributes

Name	Type	Use	Default	Fixed	Annotation
[degrees](#)	**xs:float**		15		
[isothermal](#)	**xs:boolean**		false		

annotation: appInfo
Super-type describing the base attributes required for describing a temperature profile.

source
```
<xs:complexType name="TemperatureConditionType">
  <xs:annotation>
    <xs:appinfo>Super-type describing the base attributes required for describing a temperature profile.</xs:appinfo>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AttributeExtensibleSMALElementType">
      <xs:attribute name="degrees" type="xs:float" default="15"/>
      <xs:attribute name="isothermal" type="xs:boolean" default="false"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

attribute **TemperatureConditionType/@degrees**

type	**xs:float**	
properties	isRef	0
	default	15

source
```
<xs:attribute name="degrees" type="xs:float" default="15"/>
```

attribute **TemperatureConditionType/@isothermal**

type	**xs:boolean**	
properties	isRef	0
	default	false

source
```
<xs:attribute name="isothermal" type="xs:boolean" default="false"/>
```

attributeGroup **attlist.SampleConfigurationParameters**

used by	elements	**HlaConfiguration** **LinkConfiguration**

attributes

Name	Type	Use	Default	Fixed	Annotation
name	**xs:NMTOKEN**				
country	**xs:NMTOKEN**				
kind	**xs:NMTOKEN**				
domain	**xs:NMTOKEN**				
category	**xs:NMTOKEN**				
subCategory	**xs:NMTOKEN**				
extra	**xs:NMTOKEN**				

source
```
<xs:attributeGroup name="attlist.SampleConfigurationParameters">
  <xs:attribute name="name" type="xs:NMTOKEN"/>
  <xs:attribute name="country" type="xs:NMTOKEN"/>
  <xs:attribute name="kind" type="xs:NMTOKEN"/>
  <xs:attribute name="domain" type="xs:NMTOKEN"/>
  <xs:attribute name="category" type="xs:NMTOKEN"/>
  <xs:attribute name="subCategory" type="xs:NMTOKEN"/>
  <xs:attribute name="extra" type="xs:NMTOKEN"/>
</xs:attributeGroup>
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Bayer, Matthew E., *Analysis of Binary XML Suitability for NATO Tactical Messaging,* Master's Thesis, Naval Postgraduate School, Monterey, California, September 2005

Boyd, John R., *Organic Design for Command and Control*, Briefing, May 1987

Brutzman, Donald P., *Next-Generation USW Interoperability Using Extensible Markup Language (XML),* Undersea Warfare XML Working Group, January 2006

Brutzman, Donald P., *Web-Based 3D Graphics Rendering of Dynamic Deformation Structures in Large-Scale Distributed Simulations, Technical Report*, Naval Postgraduate School, Monterey, California, November 2003

Buss, Arnold H. *Component-Based Simulation Modeling*. Proceedings of the 2000 Winter Simulation Conference. 2000

Buss, Arnold H. *Simkit Analysis Workbench for Rapid Construction of Modeling and Simulation Components,* Proceedings of the Fall Simulation Interoperability Workshop, Orlando, Florida, September 2004

Davis, Eddie L. *Evaluation of the Extensible Markup Language (XML) as a Means of Establishing Interoperability Between Multiple Department of Defense (DoD) Databases*, Master's Thesis, Naval Postgraduate School, Monterey, California, June 2001

Defense Modeling and Simulation Office (DMSO), *Joint Data Base Elements for Modeling and Simulation Methodology Manual*, February 1995

Fitzpatrick, Stephen K., and Paul J. Hargaden, *Multimedia Communications in a Tactical Environment*, IEEE, Atlanta, Georgia, 1994

Hewett, Baecker, Card, et. al. *ACM SIGCHI Curricula for Human-Computer Interaction,* 1996

Hittner, Brian E., *Rendering Large-Scale Terrain Models and Positioning Objects in Relation to 3D Terrain, Master's Thesis*, Naval Postgraduate School, Monterey, California, December 2003

Hodges, Glenn A., *Designing a Common Interchange Format for Unit Data Using the Command and Control Information Exchange Data Model (C2IEDM) and XSLT,* Master's Thesis, Naval Postgraduate School, Monterey, California, September 2004

Hutton, Claude O., *3D Battlespace Visualization Using Operational Planning Data.* Master's Thesis, Naval Postgraduate School, Monterey, California, September 2003

International Standards Organization. ISO/IEC JTC1/SC29/WG11, *MPEG-7 Overview version 10*, http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm October 2004

International Standards Organization. ISO/FDIS 19115, *Geographic information-Metadata*, October 2004

Jaster, Jeffrey F. *Digitized Mapping: The Key to Achieving a Common Command and Control Picture*, IEEE AESS Systems Magazine, October 2002

Johnson, Thomas, *Extended C2IEDM Data Model to Include ATP-45 Message Sets and Data Elements: NATO NBC Communications Information System (CIS) Interoperability*, NBC, CIS and W&R SC6 Meeting, Skive, Denmark, June 2004

Kaye, Tom, and George Galdorisi. Achieving Information Dominance: Seven Imperatives for Success, San Diego, Califormia, May 2002

Kleiner, M. S., S. A. Carey, J. Beach, Communication Mission-Type Orders to Virtual Commanders. Proceedings of the 1998 Winter Simulation Conference, IEEE, Piscataway, New Jersey, 1998

Lacy, Lee W., Joel Pawloski, *Semantic Web Applications for Modeling and Simulation*, DMSO Technical Exchange Meeting, 11 July, 2001

Macklin, C. and H. Dudfield. *Campaign Assessment Visualization Techniques for Command Teams,* Defense Evaluation and Research Agency Report, Farnborough, United Kingdom, 2001

McGregor, D., Brutzman, D., Blais, C., and Armold, A., *DIS-XML: Moving DIS to Open Data Exchange Standards,* Spring Simulation Interoperability Workshop, Simulation Interoperability Standards Organization, Huntsville, Alabama, April 2006

Mejdal, Sig, Michael H. McCauley, Dennis B. Beringer. Human Factors Design Guidelines for Multifunction Displays. U.S. Department of Transportation, Federal Aviation Administration Final Report, October 2001

Murray, Mark W., and Jason M. Quigley, *Automatically Generating a Distributed 3D Battlespace Using USMTF and XML-MTF Air Tasking Order, Extensible Markup Language (XML) and Virtual Reality Modeling Language (VRML)*, Master's Thesis, Naval Postgraduate School, Monterey, California, September 2003

National Information Standards Organization. *Understanding Metadata*, Bethesda, Maryland, 2004

Neushul, James D. *Interoperability, Data Control, and Battlespace Visualization Using XML, XSLT, and X3D*, Master's Thesis, Naval Postgraduate School, Monterey, California, September 2003

Nicklaus, Shane D. *Scenario Authoring and Visualization for Advanced Graphical Environments*, Master's Thesis, Naval Postgraduate School, Monterey, California, September 2003

Norbraten, Terry D. *Utilization of Forward Error Correction (FEC) Techniques with Extensible Markup Language (XML) Schema-based Binary Compression (XSBC)*, Master's Thesis, Naval Postgraduate School, Monterey, California, December 2004

North Atlantic Treaty Organization. *NATO Standardization Agreement (STANAG) 7023 Air (Edition 3) Air Reconnaissance Primary Imagery Data Standard*, September 2004

Obasanjo, Dare. *Designing Extensible, Versionable XML Formats*. http://www.xml.com/pub/a/2004/07/21/design.html, Accessed: February 2006

Obasanjo, Dare. *W3C XML Schema Design Patterns: Avoiding Complexity*, http://www.xml.com/pub/a/2002/11/20/schemas.html, Accessed: February 2006

Office of the DoN Chief Information Officer, *DoN Policy on the Use of Extensible Markup Language (XML)*, 13 December 2002

Office of the DoN Chief Information Officer, *DoN XML Naming and Design Rules (NDR), Version 2.0*, January 2005

Open Geospatial Consortium, *OGC 00-029, Geography Markup Language (GML) v1.0*, May 2000

Orchard, David. *Extensibility, XML Vocabularies, and XML Schema*, http://www.xml.com/pub/a/2004/10/27/extend.html, Accessed: February 2006

Simulation Interoperability Standards Organization, SISO-REF-015-2006: Military Scenario Definition Language (MSDL) Study Group Final Report, http://www.sisostds.org, 28 February 2006, Accessed: March 2006

Sudnikovich, William P., J. Mark Pullen, et. al. *Extensible Battle Management Language as a Transformation Enabler*, SIMULATION Vol. 80 Issue 12, December 2004

United States Army Training and Doctrine Command, Threat Support Directorate, *Worldwide Equipment Guide (WEG)*, February 1999

usw-xml Working Group: *Undersea Warfare (USW) Interoperability via XML Interchange*, restricted, Accessed: March 2006

Web3D Consortium, ISO/IEC FDIS 19775:200x *Information technology—Computer graphics and image processing—Extensible 3D (X3D)*, http://www.web3d.org/x3d/specifications/ISO-IEC-19775-X3DAbstractSpecification, 2004

World Wide Web Consortium, *OWL Web Ontology Language Overview, W3C Recommendation*, http://www.w3.org/TR/owl-features, 10 February 2004

World Wide Web Consortium, *RDF Primer, W3C Recommendation*, http://www.w3.org/TR/rdf-primer, 10 Feburary 2004, Accessed: January 2006

World Wide Web Consortium, *XML in Ten Points*, http://www.w3.org/XML/1999/XML-in-10-points, 13 November 2001, Accesed: September 2005

World Wide Web Consortium, *Namespaces in XML 1.1, W3C Recommendation*, http://www.w3.org/TR/xml-names11/, 4 February 2004

Xj3D, http://www.xj3d.org, Accessed: August 2005

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Don Brutzman
   Naval Postgraduate School
   Monterey, California

4. Jeffrey Weekley
   Naval Postgraduate School
   Monterey, California

5. Curt Blais
   Naval Postgraduate School
   Monterey, California

6. Leonard Daly
   Daly Realism
   Valley Glen, California

7. Jerry Kniphfer
   Virtual Target Center
   Redstone Arsenal, Alabama

8. Alan Hudson
   Yumetech
   Seattle, Washington

9. Robert Taylor
   Naval Facilities Engineering Service Center
   Port Hueneme, California

10. Dallas Meggitt
    Sound and Sea Technology
    Edmonds, Washington

11. John Moore
    Navy Modeling and Simulation Office
    Washington, District of Columbia

12. Dr. Richard Puk
Intelligraphics
Carlsbad, California

13. Richard Lee
Office of the Secretary of Defense
Washington, District of Columbia