ANSI /MEDBIQ VP.10.1-2010
MedBiquitous Virtual Patient Player
Specifications and Description Document

Version: 1.0
Date: April 6, 2010
Authors: Ben Azan, Valerie Smothers
Author email:benjamin.azan@mssm.edu, vsmothers@medbiq.org

Version History

Version No.	Date	Changed By	Changes Made
1.0	6 Apr 2010		

MedBiquitous Consortium XML Public License and Terms of Use

MedBiquitous XML (including schemas, specifications, sample documents, Web services description files, and related items) is provided by the copyright holders under the following license. By obtaining, using, and or copying this work, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions.

The Consortium hereby grants a perpetual, non-exclusive, non-transferable, license to copy, use, display, perform, modify, make derivative works of, and develop the MedBiquitous XML for any use and without any fee or royalty, provided that you include the following on ALL copies of the MedBiquitous XML or portions thereof, including modifications, that you make.

1. Any pre-existing intellectual property disclaimers, notices, or terms and conditions. If none exist, the following notice should be used: "Copyright © [date of XML release] MedBiquitous Consortium. All Rights Reserved. http://www.medbiq.org"
2. Notice of any changes or modification to the MedBiquitous XML files.
3. Notice that any user is bound by the terms of this license and reference to the full text of this license in a location viewable to users of the redistributed or derivative work.

In the event that the licensee modifies any part of the MedBiquitous XML, it will not then represent to the public, through any act or omission, that the resulting modification is an official specification of the MedBiquitous Consortium unless and until such modification is officially adopted.

THE CONSORTIUM MAKES NO WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, WITH RESPECT TO ANY COMPUTER CODE, INCLUDING SCHEMAS, SPECIFICATIONS, SAMPLE DOCUMENTS, WEB SERVICES DESCRIPTION FILES, AND RELATED ITEMS. WITHOUT LIMITING THE FOREGOING, THE CONSORTIUM DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY, EXPRESS OR IMPLIED, AGAINST INFRINGEMENT BY THE MEDBIQUITOUS XML OF ANY THIRD PARTY PATENTS, TRADEMARKS, COPYRIGHTS OR OTHER RIGHTS. THE LICENSEE AGREES THAT ALL COMPUTER CODES OR RELATED ITEMS PROVIDED SHALL BE ACCEPTED BY LICENSEE "AS IS". THUS, THE ENTIRE RISK OF NON-PERFORMANCE OF THE MEDBIQUITOUS XML RESTS WITH THE LICENSEE WHO SHALL BEAR ALL COSTS OF ANY SERVICE, REPAIR OR CORRECTION.

IN NO EVENT SHALL THE CONSORTIUM OR ITS MEMBERS BE LIABLE TO THE LICENSEE OR ANY OTHER USER FOR DAMAGES OF ANY NATURE, INCLUDING, WITHOUT LIMITATION, ANY GENERAL, DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL, OR SPECIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF ANY USE OF MEDBIQUITOUS XML.

LICENSEE SHALL INDEMNIFY THE CONSORTIUM AND EACH OF ITS MEMBERS FROM ANY LOSS, CLAIM, DAMAGE OR LIABILITY (INCLUDING, WITHOUT LIMITATION, PAYMENT OF ATTORNEYS' FEES AND COURT COSTS) ARISING OUT OF MODIFICATION OR USE OF THE MEDBIQUITOUS XML OR ANY RELATED CONTENT OR MATERIAL BY LICENSEE.

LICENSEE SHALL NOT OBTAIN OR ATTEMPT TO OBTAIN ANY PATENTS, COPYRIGHTS OR OTHER PROPRIETARY RIGHTS WITH RESPECT TO THE MEDBIQUITOUS XML.

THIS LICENSE SHALL TERMINATE AUTOMATICALLY IF LICENSEE VIOLATES ANY OF ITS TERMS AND CONDITIONS.

The name and trademarks of the MedBiquitous Consortium and its members may NOT be used in advertising or publicity pertaining to MedBiquitous XML without specific, prior written permission. Title to copyright in MedBiquitous XML and any associated documentation will at all times remain with the copyright holders.

Table of Contents

1 Acknowledgements

The MedBiquitous Consortium wishes to acknowledge the help of the MedBiquitous Consortium Virtual Patient Working Group members and other individuals that contributed to the creation of this document, including:

The MedBiquitous Consortium wishes to acknowledge the help of the MedBiquitous Consortium Virtual Patient Working Group members, staff, and other individuals that contributed to the creation of this document, including:

- Rachel Ellaway, Ph.D., Northern Ontario School of Medicine, Co-Chair
- J.B. McGee, M.D., University of Pittsburgh, Co-Chair
- Chris Candler, Association of American Medical Colleges, past Co-Chair

- Spenser Aden, Healthstream
- Susan Albright, Tufts University
- Ben Azan, MedBiquitous
- Dmitry Babinchenko, University of Pittsburgh
- Chara Balabubramaniam, St. George's, University of London
- Linda Bell, American Association of Critical-Care Nurses
- Emily Conradi, St. George's, University of London
- David Davies, Ph.D., IVIMEDS
- Parvati Dev, Stanford University
- Shona Dippie, HEAL
- Jeroen Donkers, University of Maastricht
- Uno Fors, D.D.S., Ph.D., Karolinska Institute
- Robert Galbraith, M.D., National Board of Medical Examiners
- Dennis Glenn, American Board of Surgery
- Peter Greene., M.D., MedBiquitous
- Michael Hagen, M.D., American Board of Family Medicine
- Frank Hess, University of Heidelberg
- Jörn Heid, University of Heidelberg
- Matthias Holzer, University of Munich
- Grace Huang, M.D., Harvard University
- Soren Huwendiek, M.D., University of Heidelberg
- Philip Jenkins, St. George’s University of London
- Patrik Jonsson, Karolinska Institute
- Carol Kamin, University of Colorado
- Peter Kant, University of Pittsburgh
- Joy Leffler, WE MOVE
- Ross Martin, M.D., Deloitte Consulting
- Sandra McIntyre, M.Ed., HEAL
- Yanko Michea, M.D., University of Connecticut

- Dick Moberg, Moberg Research
- Nancy Posel, McGill University
- Beth Powell, Centers for Disease Control
- Narain Ramluchumun, St. George's University of London
- Dan Rehak, Ph.D.
- Kathie Rose, National Board of Medical Examiners
- Deborah Sher, Department of Veterans Affairs
- Gurjeet Shokar, University of Texas Medical Branch
- Arnold Somasunderam, St. George's University of London
- Kevin Souza, University of California, San Francisco
- Valerie Smothers, MedBiquitous
- Dave Taylor, Imperial College London
- Hemal Thakore, M.D., University College, Dublin
- Greg Thompson, M.D., Medantic
- Chris Toth, University of Pittsburgh
- Marc Triola, M.D., New York University
- Dan Walker, Tufts University
- Pat Youngblood, Stanford
- Nabil Zary, Karolinska Institute

Joel Farrell, IBM, Chair of the MedBiquitous Technical Steering Committee, and Scott Hinkelman, IBM, past Web Services Technology Architect for MedBiquitous, and Dan Rehak, Learning Technologies Architect for MedBiquitous, have contributed their expertise to ensure that this specification is well-designed and interoperable with related industry standards.

We would also like to extend a particular note of thanks to Coco Ruiz and Marilyn Cheung of the Stein Gerontological Institute at the Miami Home and Hospital for the Aged for generously hosting the workshop which allowed this specification to be drawn together and this document to be created.

2 Introduction

This document describes a player specification complementary to the MedBiquitous Virtual Patient Specifications and Description Document. . It is intended for use by anyone who wants to create interoperable tools for virtual patients based on this specification. The status of the document is indicated at the bottom of the page; draft documents are subject to review and approval through the *MedBiquitous Consortium Standards Program Operating Procedures* (see http://www.medbiq.org/working_groups/consortium_process/MedBiquitousANSIProcess.pdf).

The objective of this specification is to describe functional requirements for a conformant player application, which uses data described in the Virtual Patient Data Specifications and Descriptions document to render a virtual patient activity. Together, the data and player specifications comprise the MedBiquitous Virtual Patient (MVP) specification and are used for the exchange and reuse of virtual patients.

For the purposes of this specification, a virtual patient is defined as:

> A*n interactive computer simulation of real-life clinical scenarios for the purpose of medical training, education, or assessment. Users may be learners, teachers, or examiners.*

Virtual patients are notoriously difficult and costly to author, adapt and exchange. Historically this has limited their uptake and utility, despite their being able to provide high quality learning opportunities. The development of virtual patient players with a core set of functionality ensures that much of the data and models comprising the virtual patient activity will be interpreted in a consistent way. Provisions for a player also allow virtual patients to be used outside of virtual patient authoring and delivery systems. Virtual patient activities can be delivered as stand-alone activities or as SCORM-conformant courses capable of being tracked within conformant learning management systems.

2.1 General Principles

There are many different ways that virtual patients can be created and employed. The MVP has been designed to be sufficiently abstract and adaptable so that it can accommodate a number of forms and uses.

The MVP architecture consists of five components, including a player functional specification. The data components and models can be accessed and assembled in a number of different ways. The *MedBiquitous Virtual Patient Specification and Description Document* provides a separate documentation resource focusing on the data components of the standard. These components may be rendered for use through different kinds of players, depending on the activity at hand and other local choices and requirements. A functional player specification that describes core functionality is part of the MVP architecture. This document comprises the player specification.

MVP packages may contain a player that conforms to the player specification to enable widespread use of virtual patients. Systems importing virtual patients may ignore the player in the package if they have virtual patient player capabilities. These five along with other related (but out of scope) components are shown in figure 1.

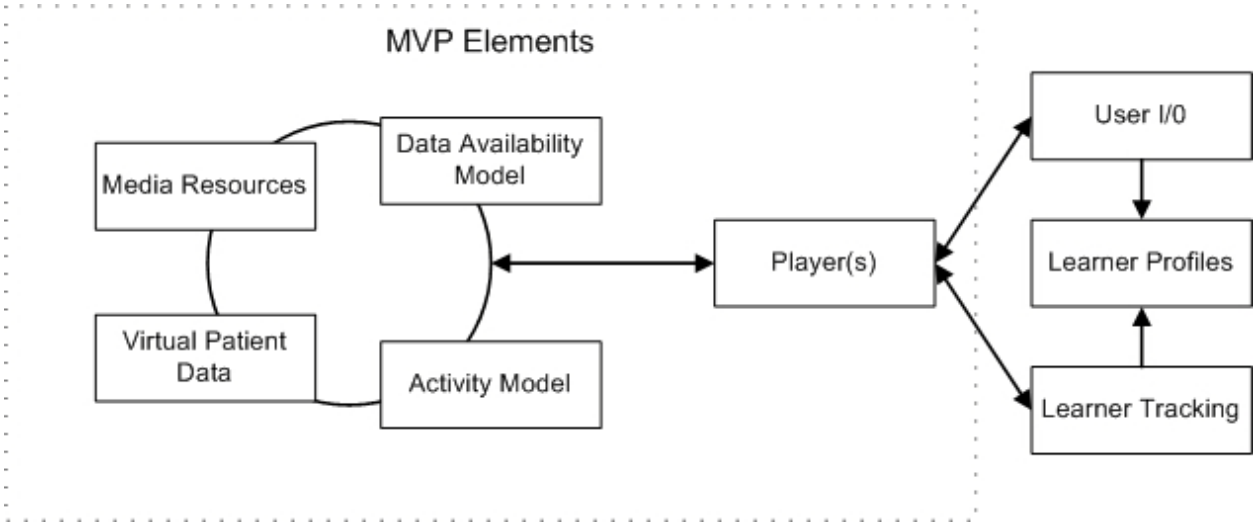

Figure 1: Components of the MVP Architecture

2.2 MVP Components

There are five core MVP components:

1. **Virtual Patient Data (VPD)**:

   The VPD provides the personal and clinical data that is relevant to the clinical scenario being simulated. The VPD is a bit like a clinical chart, containing data elements and some structure that corresponds to the medical history, physical examination, laboratory and radiology data, and procedure and outcome data. The VPD architecture has been designed to enable a flexible approach to how this data is expressed and managed.

2. **Media Resources (MR):** Media resources are all of the images, animations, videos, audio files and any other discrete digital objects that are associated with the virtual patient at any point during the simulated patient scenario. As with the specific portions of the VPD data, the media resources are tagged with identifiers so that they can be made available at the right time to the learner. IMS Content Packaging is used to structure media resources within the MVP specification and provide unique identifiers for each media resource.

3. **Data Availability Model (DAM)**: This component expresses the aggregation of VPD and MR elements for exposure through the Activity Model. VPD and MR elements can be reused in this context in multiple DAM nodes controlling the way that data aggregations such as patient histories or test results are displayed.

4. **Activity Model (AM)**: The AM encodes what the learner can do and how they engage with the virtual patient. By creating available paths through the content using interconnected nodes and controlling how the user can follow them using a simple rule

system, a very great variety of virtual patient activities are possible. Note that all content exists either in the VPD or as MR elements; the AM provides the contexts in which they are exposed to the learner.

5. **Virtual Patient Player (VPP) Functional Specification**: The Virtual Patient Player presents the virtual patient to the learner and gathers and parses learner input. . The VPP describes the functionality necessary in a conformant player. The player must track which activity model nodes have been visited in the current user session and which VPD elements have been triggered.

In addition to the five MVP components, there are several applications and data sets that are related but external to the MVP standard:

- **Authoring systems:** in addition to players the MVP specification also relates to the systems and tools used to create or edit virtual patients. These tools have a very similar range of required affordances from the specification to those required by players and as such a number of structures (in particular VPD structure) have been incorporated into the specification. In developing the MVP equal attention was given to player and authoring system requirements.
- **User Tools:** there are various tools found in a number of VP systems such as making a differential diagnosis (DDX) or note taking. These are external to the MVP specification.
- **External LMS/VLE services**: The virtual patient activity is designed to run within a learning management system (LMS) or virtual learning environment (VLE). This may be an LMS or VLE specifically designed to deliver virtual patient activities, The LMS or VLE may track the following types of data:
    - General user information: user identity data (such as their name), as well as their current courses (or program of study), personal user preferences such as choice of language, required disability support features or choice of stylesheets, and academic data such as assessment data.
    - Virtual patient usage information: tracking data, records of which virtual patients have previously been accessed (including progress markers) and their relation to study markers such as learning objectives and outcomes. SCORM tracking, assessment metrics etc.

3 Documentation Conventions

This document uses the following conventions.

Documentation Conventions

Convention	Description
`monospaced type`	Sample XML tags, code, schema, or portion thereof
BoldText	When used with an XML tag name, indicates that the element contains sub-elements
Italicized Text	When used in an XML tag description, an attribute of the XML tag.
Tag description	Shading indicated that the tag is further described elsewhere in the document

The following graphical standards are used for the XML diagrams in this document.

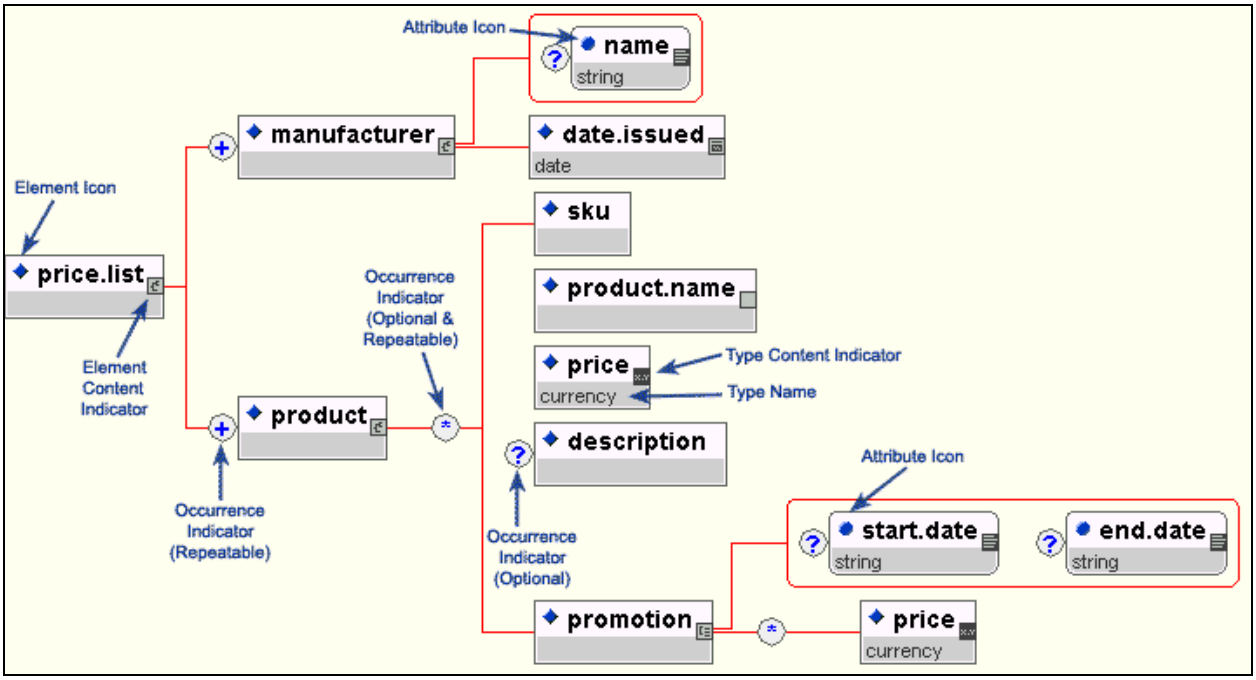

Figure 2: Graphical Standards from Tibco's Turbo XML, Copyright Tibco Software

4 General Player Comments

For more detailed information about the XML schemas for the MVP, please see the *MedBiquitous Virtual Patient Specification and Description Document.*

4.1 Interoperability and Display

To assure virtual patient interoperability, VP players must behave similarly when presented with identical VP XML packages. However, issues of presentation style, element location on screen and external services are left entirely to the individual players. This document specifies player behavior.

4.2 Launch the Player

The player is always launched by an external source. This could be an LMS invoking the player as the launchable resource in a package through an html wrapper page or a user opening the player application in a browser. In either case, the player must be launched in order for any part of the activity to be played. Note that, nevertheless, in cases where the MVP specification is being used as an import/export medium, there is no need for the player to reside in the content package. The import/export mechanism can read the XML from the package directly.

Note also that when the player is included in the content package, by convention, the player must be located at the root of the Virtual Patient Package, and that the activitymodel.xml, dataavailabilitymodel.xml, virtualpatientdata.xml and manifest.xml files must also be located at the root for the player to function properly upon launch.

4.3 The XtensibleInfo element

The virtual patient XML may contain extensions in the XtensibleInfo element. This element can exist as a child element of the root in the virtualpatientdata.xml, the dataavailabilitymodel.xml and the activitymodel.xml. Players that don't support the namespaces found in the XtensibleInfo should ignore XPath references to those elements. While specific authoring systems may make use of this element to supplement virtual patient functionality, generic players in a package can ignore elements in XtensibleInfo. The virtual patient decision path will not be dependent on data within XtensibleInfo. Thus, the virtual patient can be run without data found in XtensibleInfo. The player may display data in an AlternativePath if one is specified.

5 General Algorithmic Steps of Player Functionality

5.1 Introduction

In this section, the algorithmic steps of a conformant player are outlined, as shown in the following diagram.

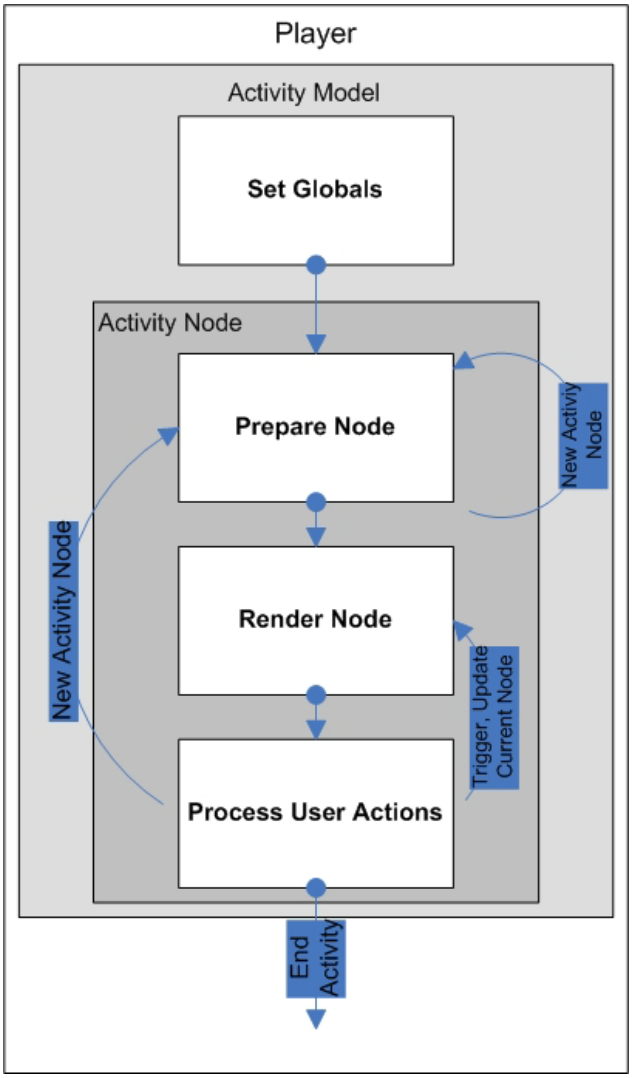

Figure 3: Algorithmic Steps of Player Functionality

There are five main steps. The first is global initialization. In this step, the player is launched. It finds the necessary files, parses them (partially or completely depending on implementation) and creates global variables, timers and counters necessary to run the activity. Once this is done, the player will cycle through each Activity Model node. For each node it goes through phases: prepare, render, process. In prepare, the player checks the node entrance requirements and updates timers, counters, etc. At this stage, the player can either move on to the render stage or redirect the user to a different Activity Node. During the render stage, the player retrieves all the data to be displayed in the Node. This includes content (text, video, picture, audio, etc) as well as

navigational elements (links to other nodes). Once the player has gathered all necessary data, it is displayed. This moves the player into the process stage. Here the player waits for user input. This input can be of two types: selection and navigation events. The main purpose of a selection event is to trigger the display of new data. Navigation events indicate that the user wants to move to a new Activity Node. This process continues until a leaf Activity node is reached. A leaf node indicates the fifth and final step, termination. In this case, the player will know that the activity has ended and may display data (collected throughout the activity) about the user's performance.

5.2 Set Globals

The following diagram illustrates the process for launching the activity and setting global properties.

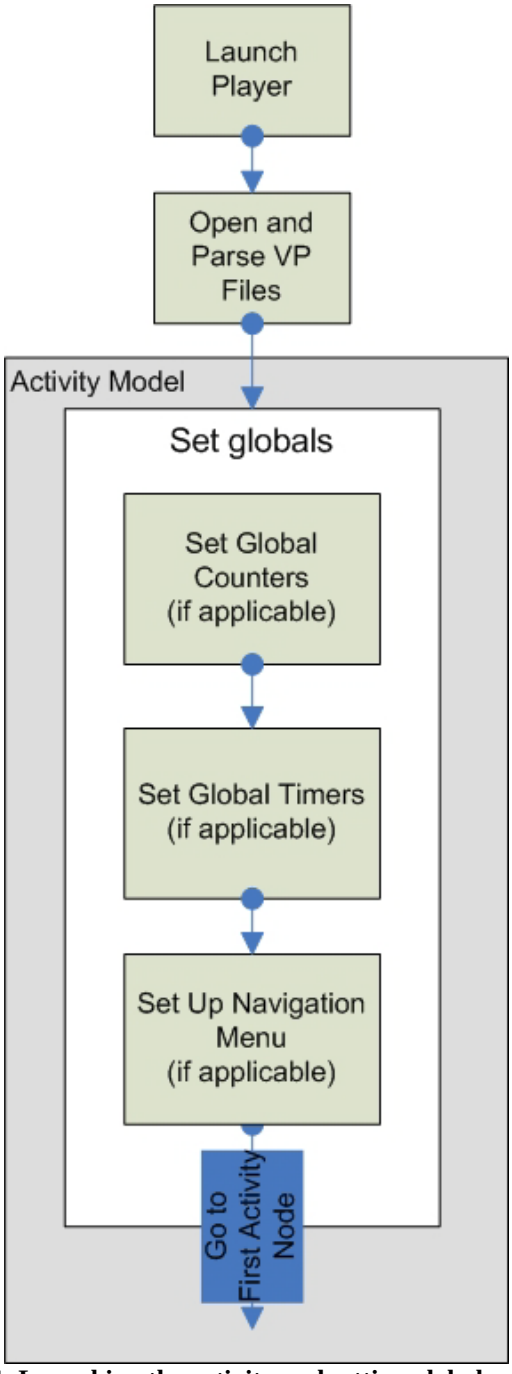

Figure 4: Launching the activity and setting global properties

5.2.1 Open Files

All virtual patient files must be within the virtual patient package. The following table describes the mandatory virtual patient data files, all of which are located in the root of the virtual patient package. Media resources are not required to be in the root directory of the virtual patient package.

Table of data files located at the root of Virtual Patient package

File Name	Description
activitymodel.xml	Contains the structure and logic of the activity
dataavailabilitymodel.xml	Aggregates data and controls triggering
virtualpatientdata.xml	Raw patient data
manifest.xml	References files in package and determines location of media resources in file system.

5.2.2 Initialize Global Counters and Timers

Parameters necessary for global initialization are located in the activitymodel.xml file within the ActivityModel →Properties element. Data needed to instantiate global counters and timers are located there.

5.2.2.1 Initialize Global Counters

If the Counters element under Properties exists, then there is at least one global counter in the activity. Each counter must have the ability to be updated as the user moves through the activity. For now it is only important to create and initialize the counter. These counters can either be for display or invisible to the user. This is determined by the attribute isVisible of Counter. Counters are visible by default. CounterLabel is the label that goes does with the counter (eg. Score). Counter UnitsSuffix is text that goes after the value of the counter (eg. %). Counter UnitPrefix is text that goes before the value of the counter (eg $). CounterInitValue is the value with which the counter initializes.

CounterRules holds a set of rules that are to be checked every time the counter is updated (at the node entrance). Each rule will check the counter against a fixed value. If the relation between the counter value and the rule value is violated, the player should have the ability to display a message to the user and/or redirect the user to a new node. Note, this does not happen here while initializing globals, but, depending on the programming languages used, it may be convenient to establish a rule evaluator at this point that can be called upon later during execution.

See section Counters in *MedBiquitous Virtual Patient Specification and Description Document* for more information on the role of counters.

5.2.2.2 Initialize Global Timers

If the Timer element under Properties exists, then there is at least one global timer in the activity. Each global timer keeps track of the time spent in the activity as a whole. For now it is only important to create and initialize timers (there will usually only be one). These counters can either be for display or invisible to the user. This is determined by the attribute isVisible of timer. Timers are visible by default. Note that even if there is no Timer element under properties, the player must still keep track of how long the activity has taken.

TimerRules holds a set of rules that are to be checked every time the timer is updated (this can be once a second or at the entrance to each node, depending on the programming environment and individual player). Each rule will check the timer against a fixed value. If the relation between the timer value and the rule value is violated, the player should have the ability to display a message to the user and/or redirect the activity to a new node. Note this does not happen here in while initializing globals but, depending on the programming languages used, it may be convenient to establish a timers checker at this point that can be called upon later during execution.

See section Timers in MedBiquitous Virtual Patient Specification and Description Document for more information on the role of Global Timers.

5.2.3 Set up the Navigation Menu

Before starting to navigate through each Activity Node one at a time, it may be necessary to parse and run through all node sections and activity nodes in order to create an internal representation of the NodeSections and ActivityNodes. This is necessary in the event that global navigation is enabled. Although the rendering of this global navigation menu does not necessarily happen at this point in execution, it is discussed here because it does not pertain to specific nodes and will not change in structure when moving through various activity nodes.

Each node has a Navigate Global element, under Rules. If that property is on, then this node should be accessible from anywhere in the activity. This is done by displaying a link to the node in a navigation panel. Further, the NodeSection in which the node is located should also be displayed. A NodeSection may be nested within another NodeSection, in which case all parent NodeSections must be displayed. Displayed ActivityNodes within NodeSections in the navigation panel can be clicked, and these events are treated as navigation events. Note: because the there may be many visible activity nodes in a NodeSection, it may be good practice, for example, to display NodeSection and AcitvityNodes navigation as collapsible inset menus. The NodeSection label attribute is used as a title for the section to display in the menu. The ActivityNode label attribute is used as a title for the node to display in the menu. Whether the NodeSection label in the navigation menu is clickable, thus allowing users to jump between node sections, is left up to individual players.

NodeSections are displayed in the navigation menu in the same sequence and hierarchy as they appear in the activity model file. Similarly, Activity Nodes are displayed in the navigation menu in the same order in which they appear within each NodeSection in the activity model file. The first Activity Node for the entire activity is the first activity node in the activity model file (going from top to bottom). The first activity node of each section is the first activity node its respective NodeSection in the activity model file. Similarly, the first NodeSection is the first node section in the activity model file (going from top to bottom).

Even if NavigateGlobal is set to false, some players may decide to display the NodeSections and ActivityNodes without making them clickable. If NodeSections and ActivityNodes are visible,

players can decide, for example, to highlight the NodeSection and ActivityNode in which the user is currently located.

5.3 Prepare Node

In the prepare stage, the player checks to see if the user has satisfied all the requirements for entering the node. This includes checking global timers (if they are not already being checked in real time), conditional rules and counters. If some condition in not satisfied, the player should have the ability to display a message to the user and/or redirect the user to a different activity node. If all conditions are satisfied, then the user is allowed entry into the activity node. Once entry is guaranteed, the player has the possibility to create timers local to the activity node. The player then goes to the Render stage. The following diagram illustrates this process.

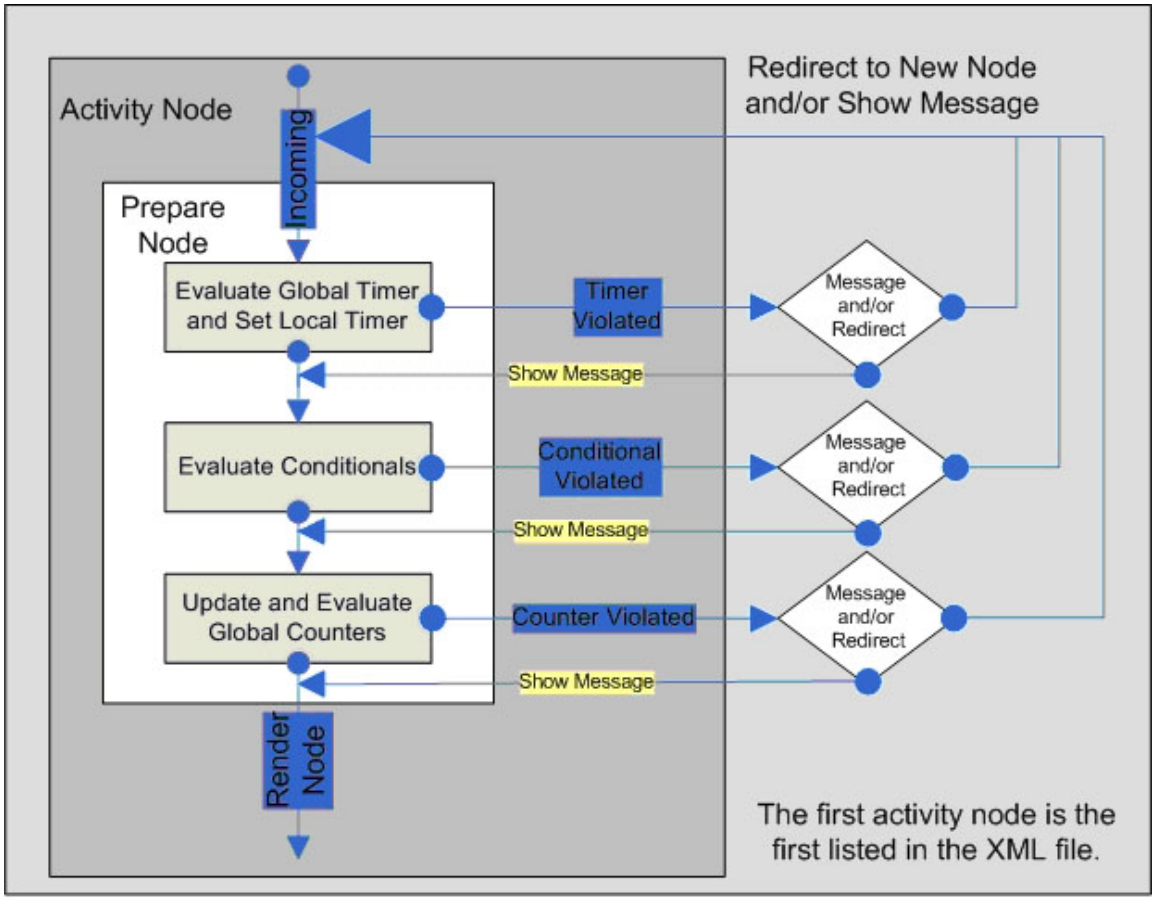

Figure 5: Preparing the Activity Node

5.3.1 Check Global Timer

Two paradigms are possible for timers. One in which the timer is checked at all times. This means that timer expiration within a node will create an asynchronous event (occurring independent of the main program flow) and cause the player to execute a Timer Rule. The other is to check the timer only on entry to nodes. If the later is in use, then the player must check the state of times on entry, during the Prepare phase. This is done by going to the global timer and checking its timer rules. If any have been violated, then the player either shows a message

and/or redirects the user to the specified node. Which action is taken is determined by which elements are present in the XML. If RuleRedirect is present, then the violation of the rule should redirect a user to another activity node. If RuleMessage is present, then a violation of the rule should give the user the message. If both are present, the player should do both, give the message and redirect. Note: the player must execute these two actions in a specific order. It must show the message and then redirect.

The state of a global timer should at minimum be checked on entry to each node.

5.3.2 Check Conditionals

Conditionals define a Boolean expression used to determine if entrance to the node is possible based on the nodes previously visited and virtual patient data previously triggered. After checking global timers, the player must verify that all conditionals have been met. These conditionals can block or allow entry based on the nodes previously visited as well as the VP data previously triggered (InterviewItem, PhysicalExam, DiagnosticTest, Intervention, DifferentialDiagnosis etc.). If the conditionals are met, the player moves on to checking the counters. If conditionals are not met, the player can show a message and/or redirects the user to a new activity node. Conditional expressions always evaluate to true or false.

The conditional expression is found in ActivityNode → Rules → ConditionalRule → Operator. It consists of a logical expression with operators such as And, Or, Nand and Nor, as well as operands elements which hold XPath expressions to Activity Node or to VPD data elements. An operator containing an Activity Node XPath evaluates to true if the activity node which it denotes has been previously visited, otherwise it evaluates to false. An operator containing only a VPD XPath evaluates to true if the VP data it denotes has been previously triggered, otherwise it evaluates to false. Conditionals are met if the entire logical expression evaluates to true. They are **not** met if the entire logical expression evaluates to false.

If the expression evaluates to false, the player must take action. If RuleRedirect is present then the player should redirect the user. If RuleMessage is present then player should give the user the message. If both are present, the player should do both, give the message and redirect the user. Note: the player must execute these two actions in a specific order. It must show message and then redirect.

See ConditionalRule section in MedBiquitous Virtual Patient Specification and Description Document for more information on the role of conditionals.

5.3.3 Check Counters

The player then updates all the counters which are specified in the CounterActionRule element within ActivityNode → Rules. For each such element, the player updates the referenced global counter. CounterPath gives an XPath pointing to the counter that needs to be updated. CounterOperator (+,-,or =) is applied to the counter with the value CounterRuleValue. The counter is increased, decreased, or set to a predetermined value accordingly.

Counter Rules may be enabled at activity nodes on a selective basis. If CounterRuleEnabled is set to On, or if the CounterRuleEnabled element is empty or missing, the player must check all CounterRules of the updated counters (located in the global properties) after all the counters have been updated.

CounterRules holds a set of rules that are to be checked every time the counter is updated. Each rule will check the counter against a relation and a fixed value, for example, less than 10. If the new counter value violates the rule value and relation and the counter rule is enabled, the player should have the ability to display a message to the user and/or redirect the user to a new activity node.

5.3.4 Initialize Local Timers

If after checking the timers, conditionals, and counters there has been no redirections, the user will now enter the current node. At this point, if a Timer element exists under the Services element in the ActivityNode, then the player must instantiate a local timer. These timers work much like global timers except that they only exist and have effect within one ActivityNode. If the user leaves an ActivityNode before the local timer expires, then the timer has no effect and does not carry over to subsequent ActivityNodes.

5.3.5 Leave the Prepare Stage

The player must record the ActivityNode ID (for use in conditional rules) before leaving the prepare stage. Once this is done the player moves into the render stage.

5.4 Render Node

The render process involves finding and displaying the information in a given node. Further, the player needs to be able to determine which elements in a node support user interaction (eg. navigation events as well as possible User selection events in InterviewItem, PhysicalExam, DiagnosticTest, Intervention, DifferentialDiagnosis, etc.). The following diagram illustrates the rendering process.

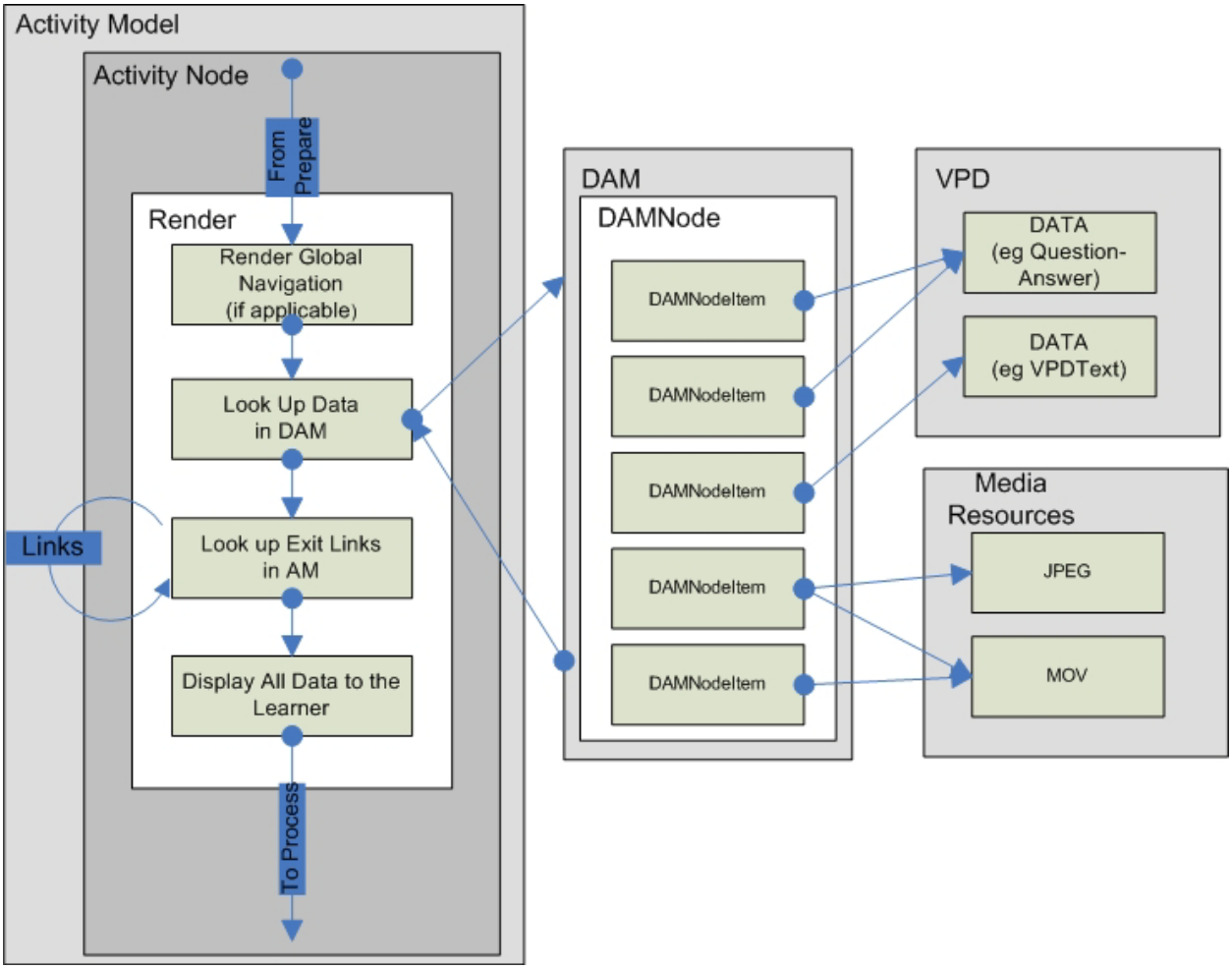

Figure 6: Rendering the Activity Node

5.4.1 Activity Model

Note: If this is the first activity node, the navigation menu needs to be rendered in this stage. This needs to be done if certain ActivityNodes have NavigateGlobal set to on or if a player decides to show and/or let learners navigate NodeSections. Also, the player might want to show user progress by highlighting the current NodeSection and/or ActivityNode (if they are shown).

The player then displays the ActivityNode label attribute. This is the title of the node. The player must then to go to ActivityNode → Content. In this element is an XPath for the DAM node that aggregates all the data found in this activity node. The player must find the correct DAM node and display its content.

5.4.2 Data Availability Model and Virtual Patient Data

Once the player has found the correct DAMNode, it must scan through all the DAMNodeItems in the DAMNode and display their content. In each DAMNodeItem, there is an ItemPath which

specifies the XPath referencing virtual patient data or a media resource. This is the data that is being encapsulated in DAMNodeItem.

The player cycles through DAMNodeItems in the DAMNode referenced and checks their display attributes. This attribute will dictate whether user interaction is required to display certain data referenced in ItemPath, ItemComments and DAMNodePath and, if so, whether the triggered data should be displayed directly after user interaction or in a later ActivityNode. The possible values of the display attribute are: immediately, ontrigger, delayed, and ifrequested. The default value is immediately.

Note: the display attribute does not have an effect on ItemPath if the path points to a media resource. Media resources referenced from a DAMNodeItem are always shown in full no matter the value of the display attribute.

- **immediately:** In this case all the data referenced in this DAMNodeItem is displayed immediately. No data is held back. Further, if the DAMNodeItem in question contains an ItemComment, the data referenced in this element should also be displayed along with the data from DAMNodeItem. Similarly, all DAM nodes in DAMNodePath elements are displayed, but according to their own display rules dictated by their own display attributes. Immediately is the default value.

- **ontrigger:** In this case, only certain elements from the referenced virtual patient data are displayed at first. The player enables the user to interact with the data (eg, clicking, checkbox, submit button etc.) Upon interaction, the player will display the remaining data elements. An example is the Interview Item scenario. The user is presented with a set of questions. The user may click on the question and the answer is displayed. Note that the identifiers of all items triggered in this way are recorded by the player for use in conditional rules as well as end of activity reports. Further, if the DAMNodeItem in question contains an ItemComment or DAMNodePath element, a triggering of the rest of the data will also trigger the display of data referenced by ItemComment and DAMNodePath. Elements within the DAM nodes referenced from ItemComment or DAMNodePath are displayed according to their own display attributes. See section Handling DAM Nodes for more information.

- **delayed:** In this case, only certain elements from the referenced virtual patient data are displayed. The player enables the user to interact with the data (eg, clicking, checkbox, submit button etc.). Upon interaction, the player simply records that this VPD Identifier has been triggered. It does not display the remaining data to the user in that node. However, the player could inform the learner that his action has been recorded and that he will receive the results in a subsequent node. The triggered data as well as other displayable data in the DAMNode will be displayed if, in a subsequent ActivityNode, this same DAM Node in which the virtual patient data was triggered is referenced once again. Data triggered on delay will also be displayed if it is included in separate DAMNode after being triggered. The player will know (by comparing the ids of previously triggered elements and VPD Identifier of data in a given node) that this is triggered data, and should display it in its entirety (and could optionally give some indication that results are

from a previous node). Further, if a DAMNode has data elements that have been triggered on delay and is referenced a second time, then DAM nodes pointed to by ItemComment and DAMNodePath elements will be displayed at the time of this second calling, according to their own display rules. See section Handling DAM Nodes for more information.

- **ifrequested:** This display element enables case designers to return data only if it has been previously triggered. If a VPD element in ItemPath of a DAMNodeItem element has been previously triggered and the display attribute is ifrequested, then the player has the same behavior as when the display attribute is immediately. All the data referenced in this DAMNodeItem is displayed immediately, and DAMNodes in ItemComment and DAMNodePath are also displayed. If a VPD element in ItemPath of a DAMNodeItem element has not been previously triggered and the display attribute is ifrequested, then the player should completely ignore the DAMNodeItem and display nothing for this DAMNodeItem.

VPD elements that are affected by the ontrigger, delayed, and if requested flags are InterviewItem, PhysicalExam, DiagnosticTest, Diagnosis and Intervention. Only part of the data is displayed before user interaction for these elements. The remaining data is either displayed directly OnTrigger or Delayed in some later node.

The other VPD elements can have ontrigger, delayed, or if requested display attributes associated with them. In the case of ontrigger, while all the data for these elements is displayed in the first pass, the data and media referenced through ItemComment and DAMNodePath elements will only be displayed after user interaction. In the case of delayed, the data referenced through ItemComment and DAMNodePath elements will only be displayed if the data in the DAMNodeItem is triggered and, in a subsequent node, the DAMNode from which the data was triggered is called again. In otherwords, if data is triggered with the display flag set to delayed, calling the DAMNode from which it was first triggered for a second time should display the results of any triggered elements. Calling an individual piece of data previously triggered as delayed will also cause the player to return this data in full as well as to display the ItemComment and DAMNodePath elements of the DAMNodeItem from which this data is referenced. See section Handling Virtual Patient Data for more information.

Once the player has taken note of the display attribute, it retrieves the data referenced in ItemPath and readies it for display. ItemPath holds an XPath. Here the XPath can reference the VPD or a media resource. In the case the XPath is referencing a media resource, the identifier of the resource is looked up in the manifest. The manifest holds a bijection of media resource IDs to media resource file locations. In this way, the player will be able to locate the file for the media resource that is to be displayed. In the case the XPath is referencing VPD, the player simply looks up the corresponding element in virtualpatientdata.xml.

Each time a VPD element is referenced from a DAM node, its id must be checked against the ids of VPD items that have been triggered but not yet returned to determine whether that virtual patient data item has been previously triggered with the display attribute set to delayed or if requested. If a match is found, this indicates that either the DAMNode from which this data was

triggered is included for a second time or that the triggered piece of data itself is included again but in a different DAMNode. In this case, the previously triggered VPD data should now be displayed in its entirety. If desired the data can be marked as a 'returned' piece of data.

If the ItemPath points to data within the XtensibleInfo section of VPD and the player does not support the functionality indicated, the player may make use data within an AlternativePath if one is indicated.

The player then handles all DAMNodePath elements found inside this DAMNodeItem. These are handled differently based on the value of the display attribute in DAMNodeItem. See section DAMNodePath for more information on the effects of the display attribute. DAMNodePath references should be handled like any other DAMNode, with the exception that the data inside them is displayed as a sub-element of the data referenced in DAMNodeItem. It is left up to the individual player to determine how this hierarchy is represented (indentation, html link, etc.). Elements within the DAM nodes referenced from DAMNodePath are displayed according to the display attributes within the referenced DAMNode.

Finally the player takes note of ItemOrder for each DAMNodeItem and displays the data in each item in the order dictated by ItemOrder ('1' first, '2' second, etc.).

5.4.3 Links

Once the player has gathered all the patient data that needs to be rendered in the node, it must gather the navigation links (or buttons). This is done by cross-referencing the current node ID with the list of links in ActivityModel → Links. The player finds all the links that leave the current node. Links are directed. Looking at the XML they go from ActivityNodeA to ActivityNodeB. It then checks the current activity node's Probability element.

If Probability is set to on, the player will only display one of the outgoing links (not all of them). The player will chose which to display based on the "Weighting" element of the individual links. The weighting is the percent chance that a link will be chosen for display. Since only one link is displayed the user does not have a choice of which link to follow to continue the activity, but still needs to click on the link to be taken to the next node.

If Probability is set to off or does not exist, the player will display all the links away from the current node. It will be up to the user to choose which link to follow.

The label attribute is used as the link (or button) text. If this label is missing the player can also use the ActivityNode label of the node to which the link leads. Labels are displayed to the learner by default.

Links may have CounterActionRules associated with them, but these rules have no impact on counter scores unless the learner selects the link.

5.5 Process User Actions

In the process stage the player is in a wait state. It is ready to receive and process user events such as navigation and selection events. The following diagram illustrates the process stage.

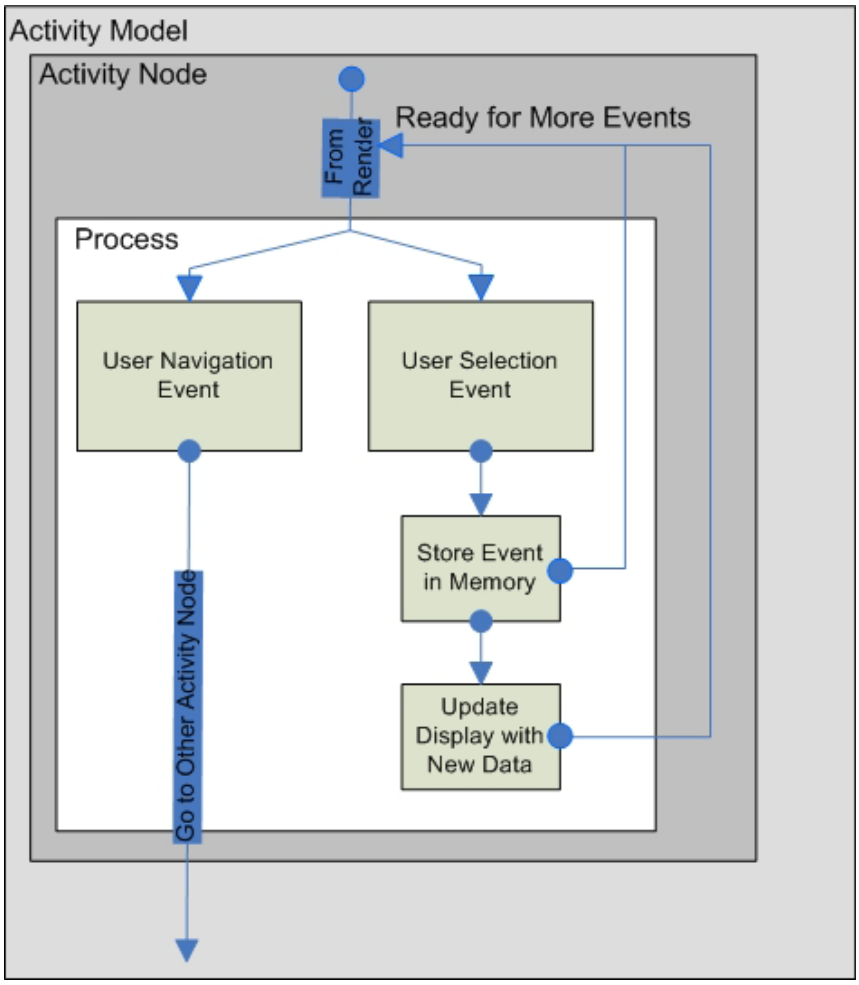

Figure 7: Processing User Selection and Navigation Events

5.5.1 Selection events

Selection events happen when the user interacts with a piece of data (eg. clicks, checkbox, submit button). Depending on the display attribute in the DAMNodeItem referencing the data, this can have different effects:

- **immediately:** No effect, it should not be possible for the user to interact with the data.

- **ontrigger:** Upon interaction with a piece of data, the player will display the remaining data elements from this data that have not yet been displayed. An example is InterviewItemdata. The user is presented with a set of interview questions. They click on the question they wish to ask and the patient's answer is displayed. Note that the identifiers of all items triggered in this way are recorded by the player for use in conditional rules as well as end of activity reports. Further, if the triggered

DAMNodeItem contains an ItemComment or DAMNodePath, then data referenced through ItemComment and DAMNodePath should also be displayed at trigger time.

- **delayed:** Upon interaction, the player simply records that the VPD identifier of the data element that has been triggered. It does not display the remaining data to the user in the current node. Optionally, the player may inform the user that his action has been recorded and that he should receive the results in a subsequent node. The results of the trigger will be displayed if, in a subsequent node, this same VPD identifier is referenced once again.

- **ifrequested:** No effect, it should not be possible for the user to interact with the data (it is either show as in immediately or not displayed at all).

VPD elements that are affected by the ontrigger ,delayed, and if requested flags are InterviewItem, PhysicalExam, DiagnosticTest, Diagnosis and Intervention. Only part of the data is displayed before user interaction for these elements. The remaining data is either displayed directly OnTrigger or Delayed in some later node. Other VPD elements are not affected by ontrigger ,delayed, and if requested flags associated with them. With these other VPD elements, all the data is displayed in the first pass, but data referenced through ItemComment and DAMNodePath elements within the DAMNodeItem are still affected by the display attribute. See section Virtual Patient Data Display Rules for more information.

5.5.2 Navigation events

These types of events can be caused by using the global navigation menu or clicking a navigation link. In both cases the action is the same. The player takes the user to a new node and begins this algorithm again, starting from Prepare Node.

In the case a player makes node sections clickable, it is up to the player to decide where to take the learner when this navigation event occurs.

Navigation links may have associated CounterActionRules that impact Counter scores and potentially navigation results. These CounterActionRules should be processed when a learner selects the navigation link. CounterRules, part of the Activity Model's Properties, holds a set of rules that are to be checked every time the counter is updated. CounterPath gives an XPath pointing to the counter that needs to be updated. CounterOperator (+,-,or =) is applied to the counter with the value CounterRuleValue. The counter is increased, decreased, or set to a predetermined value accordingly.

Each rule will check the counter against a relation and a fixed value, for example, less than 10. If the new counter value violates the rule value and relation and the counter rule is enabled, the player should have the ability to display a message to the user and/or redirect the activity to a new node.

5.6 Terminate the activity

A terminal node (or leaf node) is one which has no links away from it or only has a single link to the first activity node listed in the Activity Model file. This signifies an end of the activity. In

this node, feedback on the learner's performance in the activity may be shown. This could include time taken to complete the activity, current counter values, path taken through the nodes and triggers triggered (ie, list of all questions asked, diagnostic test ordered, physical exams, etc.) and more. The amount of feedback given to the user is left up to the individual player. If a link to the first activity node is present, the user is given the option to restart the activity. Restart sets the virtual patient back to its initial state and starts the activity from scratch. At this point, the user should also have the ability to quit and/or return to the LMS if one is being used. Quit closes the player.

6 Handling Activity Nodes

6.1 Ordering

The first Activity Node for the entire activity is the first activity node in the activity model file (going from top to bottom). The first activity node of each section is the first activity node its respective NodeSection in the activity model file. Similarly, the first NodeSection is the first node section in the activity model file (going from top to bottom). NodeSections may be nested, and their display order follows the display order in the activity model file.

6.2 Evaluating Conditionals

Conditionals are located in ConditionalRule under Rules in an Activity Node. They serve to determine the conditions that must be met in order to have access to a Node.

When an ActivityNode Identifier (in the form of an XPath) is found in the operand element of a conditional rule, it evaluates to true if the node has already been visited. If the node that has not been visited, it will evaluate to false. In order to determine if a node has been visited, the player needs to compare ActivityNode identifier in the operand with the ids of nodes previously visited. The player is thus required to record the identifiers of the nodes that have been visited.

The same applies if a VPD identifier (again as an XPath) is found in the operand element of a conditional rule. It evaluates to true if the data has already been triggered. If the data has never been triggered, it will evaluate to false. In order to determine if data has been triggered, the player needs to compare VPD identifiers in the operand with the list of data items previously triggered. The player is thus required to record the identifiers of the VP elements that have been trigged. This includes triggers in which the display attribute in the DAMNodeItem was set toontrigger, or delayed, .

In the schema, Conditional Rules are set up as Boolean expressions. The outer element is the operator. All items within an element are evaluated according to the evaluation rule for the outer operator.

Example#1:

```
<ConditionalRule>
   <Operator>
      <And>
         <Operand>
        /ActivityModel/ActivityNodes/NodeSection/ActivityNode[@id = '1']
     </Operand>
         <Operand>
        /VirtualPatientData/InterviewItem[@id = '6']
         </Operand>
      </And>
   </Operator>
</ConditionalRule>
```

The learner must have visited ActivityNode with id 1 and have triggered vpd data with id 2 in order for this statement to evaluate to true (and thus enable to user to enter the node).

Example#2:

```
<ConditionalRule>
  <Operator>
    <Or>
       <Operand>
      /ActivityModel/ActivityNodes/NodeSection/ActivityNode[@id = '5']
       </Operand>
       <And>
          <Operand>
        /ActivityModel/ActivityNodes/NodeSection/ActivityNode[@id = '1']
    </Operand>
          <Operand>
        /ActivityModel/ActivityNodes/NodeSection/ActivityNode[@id = '2']
           </Operand>
        </And>
    </Or>
  </Operator>
</ConditionalRule>
```

Here learner must have either visited ActivityNode with id 5 or have visited both ActivityNode with id 1 and ActivityNode with id 2 for this statement to evaluate to true.

Because of the pattern of conditional rule, a good approach to conditional evaluation may be through recursion. An algorithm for such an evaluation is provided here for convenience. However, any algorithm which follows Boolean rules and correctly parses the XML structure of the expression is satisfactory.

```
evaluateConditional(ConditionalRule){

  if(top level element is an ActivityNodeID || top level element is a VPDID){
    if(ActivityNode is previously visited || VPDID is previously triggered){
      return true
    }
    else{
      return false
    }
  }

   if (top level element is an And){
     while(And has subelements, for each subelement){
        if evalutateConditional(subelement)==false return false
     }
     return true
  }

  if (top level element is an Or){
     while(Or has subelements, for each subelement){
        if evalutateConditional(subelement)==true return true
     }
     return false
  }
```

```
  if (top level element is an Nor){
     while(Nor has subelements, for each subelement){
        if evalutateConditional(subelement)==true return false
     }
     return true
  }

  if (top level element is an Nand){
     while(Nand has subelements, for each subelement){
        if evalutateConditional(subelement)==false return true
     }
     return false
  }
}
```

6.3 Resolving XPath Element Identifiers

XPath is a language for finding information in an XML document. XPath is used to navigate through elements and attributes in an XML document. It is a W3C Standard.

Most major programming languages have built in libraries that offer support for XPath. Given an XPath, these libraries will return either the referenced XML container or the reference text. Here are some examples.

Example #1:

```
/DataAvailabilityModel/DAMNode[@id = '321']
```

This will return the DAMNode with id 321.

Example #2:

```
/VirtualPatientData/InterviewItem[@id = '6']
```

This will return the InterviewItem with id 6.

Example #3:

```
/VirtualPatientData/PatientDemographics/CoreDemographics/Name/text()
```

This will return the Name of the patient.

Example #4:

```
/manifest/resources/resource[@identifier = '3424']/@href
```

This will return the href attribute (holds the file path to the media file) of the media resource with identifier 3424.

Note, because XPath needs to be run on the proper XML object, the player will first need to determine whether the XPath is referring to the Activity Model, the DAM or the VPD. A VPD query run on a DAM document will give back an empty results set. Determining the correct document can be easily done by checking the first token of the XPath.

- Activity Model XPaths always start with: /ActivityModel
- DAM XPaths always start with: /DataAvailabilityModel
- VPD XPaths always start with: /VirtualPatientData
- Media Resources XPaths always start with: /manifest/resources/

6.4 Handling User Interaction

6.4.1 Recording user actions

The player must track a set of core events as the user progresses through the activity. Players are free to record more information but must at minimum record the following:

- Nodes Visited – Each time a node is entered (once the prepare phase is done), the player must record that this node has been visited. This is used to evaluate entrance rules in subsequent nodes. To record that a node has been visited, the player should use the node's id as this is a unique identifier for each node.

- Action taken by the user – All PatientDemographics, CoreDemographics, DemographicCharacteristic, VPDText, InterviewItem, PhysicalExam, DiagnosticTest, Diagnosis and Intervention data which were presented through a DAMNode with the display attribute in DAMNodeItem set to ontrigger, or delayed, and with which the user interacted should be recorded. This amounts to the user having ordered one of these items. These items should be recorded with their VPD id as this is a unique identifier for the given VPD element.

- Timing – Whether timers are on or off, the player should record the time spent in each node and the activity as a whole. This can be used for activity analysis.

Notes:

- The player tracks only what is triggered. Child DAM nodes in a DAM hierarchy are not tracked by the player even if they are triggered by the parent. If data within a DAMNodeItem element is triggered, only the data in this DAMNodeItem element is tracked, not the child elements, such as DAMNodes in ItemComment or DAMNodePath. However, if VPD data in these triggered DAM nodes is itself triggered, the player will track this triggering.
- Media resources referenced in the DAM are not tracked. Media resources may be paired with an associated VPD element, which is tracked.

Data Stored by Player

Data Persistent Through the Activity (ie stored)	Purpose
Identifier of Activity Nodes Visited	Conditional Rule / Final reports / Data Collection / Evaluation
Identifier of Data Triggered in User Selection Events	Conditional Rule / Final reports / Data Collection / Evaluation
Counters Values (ie scores)	Counter Action Rule / Final reports / Data Collection / Evaluation
Total Time Spent in Activity	Timer Rule / Final reports / Data Collection / Evaluation
Time Spent in Each Node	Timer Rule / Final reports / Data Collection / Evaluation
User Navigation Events (path through nodes)	Final reports / Data Collection / Evaluation

Players can choose to track any additional data necessary for individualized functionality.

7 Handling DAM Nodes

DAM nodes can be called from Activity Nodes or other DAM nodes. They aggregate data to be displayed to the user.

When a DAM node is called, the player should look through the set of DAM nodes and find the node with the id corresponding to the DAMNode id called. Once the correct DAMNode has been located the player should follow the procedure outlined below.

DAMNodelabel is used for authoring purposes only and should be ignored by the player.

7.1 DAMNodeItem

A DAMNode can have one or more DAMNodeItems. For each DAMNodeItem in a DAMNode the player does the following (order may vary as long as final outcome is the same):

- Checks ItemOrder to make sure items are displayed in the order specified. Higher numbers come after lower numbers. (1-first, 2-second, 3-third, etc.)
- Checks attribute display in DAMNodeItem (see section Display Behavior Summary Table for a high-level summary):
    - If no display attribute is included, the player sets the display attribute to immediately.
    - If it display attribute is set to immediately, the player displays all data in the given DAMNodeItem at the same time. No user interaction is required or allowed.
    - If display attribute is set to ontrigger, the player may display only a subset of the data encapsulated in the VPD element referenced in DAMNodeItem. Whether or not the player displays a subset depends on the type of element (see section Virtual Patient Data Display Rules). User interaction on those elements that display a subset of data should be allowed. The remaining data should be displayed only after user interaction. This functionality can be used to order a diagnostic test, for example. A subset of data is shown for the following VPD elements: : InterviewItem, PhysicalExam, DiagnosticTest, Diagnosis and Intervention. The specifics of which fields are to be displayed before and after user interactions are specified in the section Data Display Rules.
    - If display attribute is set to delayed, the player may display only a subset of the data encapsulated int eh VPD element referenced. Whether or not the player displays a subset depends on the type of element (see section The Display Attribute and Player Display Behavior Summary Table). User interaction on those elements that display a subset of data should be allowed. Upon interaction the player stores the VPD Identifier of the element that has been triggered. The remaining data is not displayed immediately but will be displayed in a subsequent node. Only the following elements are affected by delayed: InterviewItem, PhysicalExam, DiagnosticTest, Diagnosis, and Intervention. The player can, if it

chooses to do so, give the user feedback, so they know that their interaction has been recorded.
    - If it is set to ifrequested the player displays all data in the given DAMNodeItem at the same time if the VPD data contained in ItemPath has been previously triggered. No user interaction is required or allowed. If the data has not been previously triggered the player ignores the element and nothing is displayed for the DAMNodeItem in question.
- Finds the VPD or MR. In correct file, the player does an XPath query using the XPath identifier string found in ItemPath. Data found through ItemPath of a DAMNodeItem is of the same hierarchical level as data from other DAMNodeItems. The different data types are handled as follows:
    - If MR - Go to content packaging. Find Media Resource. Display media. (it is recommended that video have play/pause Fast forward/Rewind capabilities)
    - If VPD - Go to VPD file. Find element referenced in the XPath. Display data according to rules for displaying Virtual Patient Data (See section Virtual Patient Data Display Rules).
- Checks the id of any VPD elements referenced against the ids of VPD items that have been triggered . If a match is found, the VPD data should be displayed in its entirety. Optionally, this data can be marked as a returned piece of data.
- If the item is being displayed as part of a returned piece of data caused by a display attribute in a previous Activity Node being set to delayed or in a current Activity Node being set to ontrigger, then ItemComment data and any data referenced through DAMNodePath should be displayed along with the returned data (ie, after user interaction). This means that the player should allow the user to interact with the data if the display attribute is set to ontrigger or delayed. If ItemComment is present in a DAMNode in which the display attribute is set to immediately, then ItemComment is displayed with the rest of the data in the Node. Note that ItemComment holds an XPath reference to another DAM node. This DAM node should be displayed according to its own display rules. See section Display Behavior Summary Table for a summary of this behavior.

7.2 Triggering Data Display

The DAMNodeItem display attribute modifies how referenced and embedded data are displayed. The display attribute also effects whether data referenced in embedded ItemComment and DAMNodePath elements are displayed. The effect on specific Virtual Patient Data elements is described in section Handling Virtual Patient Data. No matter which Virtual Patient Data is referenced from a DAMNodeItem, the display attribute will always have an effect on ItemComment and DAMNodePath, if they exist.

Effect on VPD Data:
If the display attribute is set to immediately, all referenced data are displayed immediately, i.e. when the learner reaches an activity model node that references the DAMNode containing the item.

If the display attribute is set to ontrigger, only certain sub-elements of InterviewItem, PhysicalExam, DiagnosticTest, DifferentialDiagnosis and Intervention are displayed initially.

The rest of the data in the element is displayed immediately after the user interacts with the displayed data. For example, in InterviewItem, only the question is displayed at first and the answer is only displayed when the learner clicks on the question (ie, asks the question).

If the display attribute is set to delayed, only certain sub-elements are displayed when the learner reaches an activity model node referencing the DAMNode that contains the DAMNodeItem. The learner must interact with the data to trigger it, but the data will only be displayed when the learner reaches a subsequent activity model node in which the VPD data in questions is referenced once again. For example, a learner may order a diagnostic test in Activity Model Node 1, but the results of the diagnostic test will not be visible until the learner reaches Activity Model Node 3. This feature can also enable the development of a patient chart feature that shows the learner up-to-date patient data based on the diagnostic tests and therapies ordered. The delayed data may be displayed because the same DAMNode is referenced for a second time or because the VPD data element is referenced a second time from a different DAMNode.

Effect on ItemComment and embedded DAMNodes:
If the display attribute is set to immediately, the data referenced by the ItemComment and by any DAMNodePath embedded within this DAMNodeItem are displayed in accordance to the display rules set by the DAMNodes referenced by ItemComment and DAMNodePath.

If the display attribute is set to ontrigger, the data referenced by ItemComment and by any DAMNodePath embedded within this DAMNodeItem are not displayed initially. If the user interacts with the data pointed to by DAMNodeItem (ie, triggers the data), then the DAM nodes referenced by ItemComment and DAMNodePath in this DAMNodeItem element are displayed according to their own display rules.

If the display attribute is set to delayed, the data referenced by ItemComment and by any DAMNodePath embedded within this DAMNodeItem are not displayed initially. If the user interacts with the data referenced by DAMNodeItem (ie, triggers the data) the player will record the VPDID of the data that was triggered. The data referenced by the ItemComment and DAMNodePaths embedded within the DAMNodeItem may be displayed only if the following conditions are met:

- The same DAMNode is called again.
- The DAMNodeItem containing the embedded references contains delayed triggered data.

Under these conditions, the referenced data are displayed according to their own display rules.

7.3 DAMNodePath

There is a hierarchical relationship between data referenced through DAMNodeItem and data referenced through DAMNodePath, which is considered a subelement in the hierarchy. The player determines whether to display the data referenced by DAMNodePath by looking at the display attribute of the DAMNodeItem in which the DAMNodePath is located.

The value of the display attribute in DAMNodeItem has the following effect on its DAMNodePath elements:

- **immediately:** The player displays all DAMNodePath nodes in the DAMNodeItem without user interaction.

- **ontrigger:** The player does not display the DAMNodePath nodes in DAMNodeItem at first. After user interaction, the player displays the DAMNodePath nodes in DAMNodeItem according to their own display rules.

- **delayed:** The player does not display the DAMNodePath nodes in DAMNodeItem at first. If data in the DAMNode is triggered then, when this same DAMNode is included again in a later activity node, the player displays the content of the DAMNodePath nodes belonging to the DAMNodeItem containing triggered data. DAMNodePath are always displayed according to their own display rules.

- **ifrequested:** The player displays all DAMNodePath nodes in the DAMNodeItem without user interaction if the data is ItemPath has been triggered in a previous node. The player displays nothing and ignores the entire DAMNodeItem if the data in ItemPath has not been previously triggered.

Important Note: Once the DAM Node referenced in DAMNodePath is displayed, data within it should be displayed according to its own display rules, dictated by the display attribute in its DAMNodeItem elements.

To display the data the player first finds the DAMNode XPath referenced in DAMNodePath. The player then recurses back into the algorithm to handle DAMNodes, but displays this new data as a "sub-element" of the current data. How data is graphically shown to be a sub-element of other data is left up to the player. Some examples could be: Indentation or links that expand upon clicking. The DAMNode label is used for authoring purposes only and should be ignored by the player.

7.4 Display Behavior Summary Table

The following table summarizes player display behavior for the various display options at different times or trigger states.

Legend:

Show all content: This means show all content referenced from ItemPath in the DAMNodeItemin question.

Show partial content: For the content referenced from ItemPath in the DAMNodeItem, show only the data specified by the data display rules described in section 8, Handling Virtual Patient Data. For InterviewItem, for example, this means showing the interview question and not the answer.

Show previously hidden content: For the content referenced from ItemPath in the DAMNodeItem, display all content referenced from ItemPath in the DAMNodeItem, including data previously hidden according to display rules in section 8, Handling Virtual Patient Data.

Show ItemComment and DAMNodePath: Show any content referenced from a DAMNodePath or contained within an ItemComment associated with the DAMNodeItem.

Hide ItemComment and DAMNodePath: Hide any content referenced from a DAMNodePath or contained within an ItemComment associated with the DAMNodeItem.

No effect on display in current activity node: There is no change from the content already displayed.

	immediately	ontrigger	delayed	ifrequested
non-triggered data	show all content show ItemComment and DAMNodePath data	show partial content Hide ItemComment and DAMNodePath data	show partial content Hide ItemComment and DAMNodePath data	**Display nothing** **Ignore element completely.**
Data triggered in the current activity node	No effect on display in current activity node.	Show previously hidden content Show ItemComment and DAMNodePath data	Show partial content Hide ItemComment and DAMNodePath data (system can choose to provide some feedback that the order was recorded)	No effect on display in current activity node. (system can choose to provide some feedback that the order was recorded)
Data Triggered at some previous point in the activity	show all content show ItemComment and DAMNodePath data	show all content show ItemComment and DAMNodePath data	show all content show ItemComment and DAMNodePath data	show all content show ItemComment a DAMNodePath data

8 Handling Virtual Patient Data

8.1 Modifier: The Display Attribute in the DAMNodeItem

When a VPD element is called from a DAMNodeItem (in DAMNode), the display attribute in DAMNodeItem will affect both how VPD data elements are presented and whether the user can interact with the data (ie click, checkbox). VPD elements that are affected by the ontrigger and delayed flags are InterviewItem, PhysicalExam, DiagnosticTest, Diagnosis and Intervention. The other VPD elements are not affected by ontrigger or delayed attributes associated with them. With these other VPD elements, all the data, including ItemComment, is displayed in the first pass.

If attribute display is **immediately** – All elements from the called VP data are displayed. This is used for example to present the history of diagnostic tests previously ordered for a patient. In this case it is not possible for the user to interact with the data. It is simply displayed.

If attribute display is **ontrigger** – Only certain elements with the VP data are displayed to the user. Once the user interacts with the data, the player displays the remaining VP data elements. This can be used to order a physical exam, for example. The name of the exam is displayed. The user must click (ei do) the exam to get the results.

If attribute display is **delayed** – Only certain elements with the VP data are displayed to the user. Once the user interacts with the data, the player stores the identifier of the VPD elements that have been triggered. The results will be returned to the user in a later node when the VPD element is referenced again.

If attribute display is **ifrequested** – Only data that the user interacted with in a previous activity node are displayed to the user. Non-triggered data are not displayed at all. Note that ifrequested is used to return previously triggered data only. To allow users to interact with data to order tests or ask interview questions, use the delayed value. Once users have interacted with data in this manner, the ifrequested value may be used to return only triggered data, such as diagnostic tests ordered by the user.

See individual VP data entries below for handling specification based on attribute displayed.

8.2 Virtual Patient Data Display Rules

For element specific rules, see the table below. The table outlines which pieces of data the player needs to display in the case that the element in question is called viaXPath. Note that if a data element is optional in the VPD and left out of a particular piece of data, then it need not be displayed.

Data Display Rules

IF element is:	And IF display setting is:	DISPLAY this data immediately.	IF the learner:	THEN display these elements.
InterviewItem	immediately	All subelements	N/A	N/A
	ontrigger	Question	Interacts with the data	All subelements
	delayed	Question	Interacted with the data in a previous activity node	All subelements
	ifrequested	None	Interacted with the data in a previous activity node	All subelements
PhysicalExam	immediately	All subelements	N/A	N/A
	ontrigger	ExamName, LocationOnBody, Action	Interacts with the data	All subelements
	delayed	ExamName, LocationOnBody, Action	Interacted with the data in a previous activity node	All subelements
	ifrequested	None	Interacted with the data in a previous activity node	All subelements
DiagnosticTest	immediately	All subelements	N/A	N/A
	ontrigger	TestName	Interacts with the data	All subelements
	delayed	TestName	Interacted with the data in a previous activity node	All subelements
	ifrequested	None	Interacted with the data in a previous activity node	All subelements

IF element is:	And IF display setting is:	DISPLAY this data immediately.	IF the learner:	THEN display these elements.
Diagnosis	immediately	All subelements	N/A	N/A
	ontrigger	DiagnosisName	Interacts with the data	All subelements
	delayed	DiagnosisName	Interacted with the data in a previous activity node	All subelements
	ifrequested	None	Interacted with the data in a previous activity node	All subelements
Intervention	immediately	All subelements	N/A	N/A
	ontrigger	InterventionName, Medication	Interacts with the data	All subelements
	delayed	InterventionName, Medication	Interacted with the data in a previous activity node	All subelements
	ifrequested	None	Interacted with the data in a previous activity node	All subelements
Organization	Not used by the player			
PatientDemographics, CoreDemographics, DemographicCharacteristic, VPDText, Medication,	immediately, ontrigger, delayed, or ifrequested	All subelements	N/A	N/A

8.3 Data Display Notes

8.3.1 VPDText

If VPDText is called, the player simply displays the text in VPDText. The player can chose how to modify text properties based on the textType attribute. Different types of text may be displayed differently depending on the desired effect. Further, because VPDText contains an extended "div" element, which contains extended XHTML, the player needs to know how to interpret specific XHTML elements and attributes as well as the media element extension. As specified in the ***MedBiquitous Virtual Patient Specifications and Description Document***, only certain XHTML elements are allowed. The player should know how to render these specified tags. Since players included in the MVP package will be web-based, in many cases, the player may be able to rely on the browser functionality to interpret this XHTML. The XHTML img tag should never in be used to point to images external to the package. Such use would be non-conformant with the specification and thus need not be supported by conformant players.

The following XHTML tags can be found within the div element of VPDText and should be supported by players:

XHTML Tags Supported

Tag Use	Tag	Attributes
Bullet List	ul, li, ol	none
Headers	h1, h2, h3, h4, h5	none
Image	img	alt, src,
Line Break	br	none
Link	a	href,
Paragraph	p	none
Subscript/Superscript	sub, sup	none
Table	table, tbody, tr, td, th	border
strong emphasis	strong	none
emphasis	em	none
Division/section	div	class
Media	media	See section media in Virtual Patient Specifications and Description Document

XHTML contained in VPDText may contain a media reference extension. Media provides an attribute that uses an XPath reference to the manifest to reference media resources. The player must resolve the media reference or display the XHTML contained within the media element.

If the player is leveraging a Web browser to display XHTML content, the player must resolve the media references and translate them to an appropriate XHTML object reference that leverages any formatting data contained in the media width, height, and align attributes in order for the media to display. In this case, any XHTML elements contained within the media element are ignored.

If the player is unable to support XHTML extensions, the player may display the XHTML content of the media element.

8.3.2 PhysicalExam

LocationOnBody is a subelement of PhysicalExam but does not have its own id. It will always be present in the context of PhysicalExam. Displaying LocationOnBody simply involves displaying the data in the subelements of LocationOnBody. The contents of this element could also be displayed using a graphical technique.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<DataAvailabilityModel xmlns="http://ns.medbiq.org/dataavailabilitymodel/v1/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ns.medbiq.org/dataavailabilitymodel/v1/
dataavailabilitymodel.xsd">
   <DAMNode id="DAMNode1">
      <DAMNodeItem display="immediately">
         <ItemPath>/VirtualPatientData/VPDText[@id = 'VPD1']</ItemPath>
         <DAMNodePath>/DataAvailabilityModel/DAMNode[@id =
'DAMNode2']</DAMNodePath>
         <DAMNodePath>/DataAvailabilityModel/DAMNode[@id =
'DAMNode3']</DAMNodePath>
         <ItemOrder>1</ItemOrder>
      </DAMNodeItem>
   </DAMNode>
   <DAMNode id="DAMNode2">
   <DAMNodeItem display="immediately">
      <ItemPath>/manifest/resources/resource[@identifier = 'MR1a']</ItemPath>
      <ItemOrder>1</ItemOrder>
   </DAMNodeItem>
   <DAMNodeItem display="immediately">
      <ItemPath>/manifest/resources/resource[@identifier = 'MR1b']</ItemPath>
      <ItemOrder>2</ItemOrder>
   </DAMNodeItem>
   <DAMNodeItem display="ontrigger">
      <ItemPath>>/VirtualPatientData/InterviewItem[@id = 'VPD2']</ItemPath>
      <ItemOrder>3</ItemOrder>
   </DAMNodeItem>
   </DAMNode>
   <DAMNode id="DAMNode3">
   <DAMNodeItem display="immediately">
      <ItemPath>/manifest/resources/resource[@identifier = 'MR2']</ItemPath>
      <ItemOrder>1</ItemOrder>
   </DAMNodeItem>
   <DAMNodeItem display="ontrigger">
```

```
      <ItemPath>/VirtualPatientData/DiagnosticTest[@id = 'VPD3']</ItemPath>
      <ItemOrder>2</ItemOrder>
   </DAMNodeItem>
</DAMNode>
</DataAvailabilityModel>
```

In the example above DAMNode1 has the display attribute in DAMNodeItem set to immediately. This means that all data with id VPD1 will be displayed immediately. It also means that the contents of DAMNode2 and DAMNode3 will be displayed according to their own display rules. In other words, in DAMNode2, both MR1a and MR1b will be displayed immediately, because the display attribute in the DAMNodeItem in which they are located is set to immediately. However, VPD2 will be displayed according to the ontrigger rules, because the display attribute in the DAMNodeItem in which it is located is set to ontrigger. Similarly with DAMNode3, MR2 will be displayed immediately while VPD3 will be displayed according to the ontrigger rules.

Example 2:

```
   <DAMNode id="DAMNode4">
      <DAMNodeItem display="ontrigger">
         <ItemPath>/VirtualPatientData/VPDText[@id = 'VPD10']</ItemPath>
         <DAMNodePath>/DataAvailabilityModel/DAMNode[@id =
'DAMNode5']</DAMNodePath>
         <DAMNodePath>/DataAvailabilityModel/DAMNode[@id =
'DAMNode6']</DAMNodePath>
         <ItemOrder>1</ItemOrder>
      </DAMNodeItem>
```

In this example, the display attribute of the DAMNodeItem in which VP10 is located is set to ontrigger. The user must interact with VPD10 before DAMNode5 and DAMNode6 could be displayed. Data in DAMNode5 and DAMNode6 would then be displayed according to the display attributes found within their DAMNodeItem elements.

Example 3:

```
   <DAMNode id="DAMNode7">
      <DAMNodeItem display="ontrigger">
         <ItemPath>/VirtualPatientData/VPDText[@id = 'VPD15']</ItemPath>
         <DAMNodePath>/DataAvailabilityModel/DAMNode[@id =
'DAMNode8']</DAMNodePath>
         <DAMNodePath>/DataAvailabilityModel/DAMNode[@id =
'DAMNode9']</DAMNodePath>
         <ItemOrder>1</ItemOrder>
      </DAMNodeItem>
```

In this example, the DAMNodeItem in which VPD15 is located has been set to delayed. The user must first interact with VPD15. This would cause the player to record the VPDID of VPD15 as a piece of data that has been triggered as delayed. If, a subsequent DAMNode references DAMNode7, the player would recognize that VPD15 has been previously triggered. VPD1 as

well as the data in DAMNode2 and DAMNode3 would be displayed at this time, according to the display attribute setting found within DAMNode2 and DAMNode3. If the VPD1 element is called again from a different DAMNode (and thus a different DAMNodeItem) after being triggered as delayed, then the ItemComment and/or DAMNodePath elements from that DAMNodeItem would be displayed at the time of this second reference.

9 MVP as a SCORM Content Package

Each virtual patient package will be a SCORM 2004 4th edition package.
There are two application profiles of SCORM:

- Resource Package – a profile that bundles learning resources with no organization or sequencing instructions for the SCORM LMS. No player is included in the content package. The Resource Package is used to import virtual patient files to a virtual patient authoring or delivery system.
- Content Aggregation Package – a profile that bundles learning resources and their structure and sequencing. The player is included in the content package and the package may be used in a SCORM-conformant LMS.

For virtual patient packages using the Content Aggregation package, the virtual patient is a Sharable Content Object (SCO) within the package. In the case a virtual patient is launched within the context of a Learning Management System (LMS), the player in the package must allow for the exchange of information with the LMS. If a virtual patient is launched independent of an LMS, the player should be unaffected and should still be able to run the activity. A discussion of SCORM requirements as they relate to the player follows.

9.1 SCORM Basics

SCORM stands for Sharable Content Object Reference Model. It is a set of standards and specifications for sharable and reusable e-learning content objects that can be run in any conformant LMS. In order to be a SCO, an object must satisfy a minimum set of requirements. The simplest SCO simply gives the LMS an initialization notice when it starts running and gives a termination notice once it stops running. Additionally, SCOs have access to the SCORM Application Programming Interface (API) in order to report variables like scores, completion status, and success status to the LMS. All communication with the LMS is done with JavaScript (ECMAScript), thus all players included in MVP packages must be capable of making JavaScript calls.

9.2 Wrapping a Virtual Patient to be a SCO

All virtual patient content aggregation packages will contain an HTML wrapper file. In the running examples in this spec, the HTML wrapper is called launchVP.html. This is an HTML file that is to be called in order to launch the virtual patient in the context of an LMS. The file has the role of telling the LMS that the activity has started and notifying the LMS when the activity has ended. A diagram outlining this functionality is presented below:

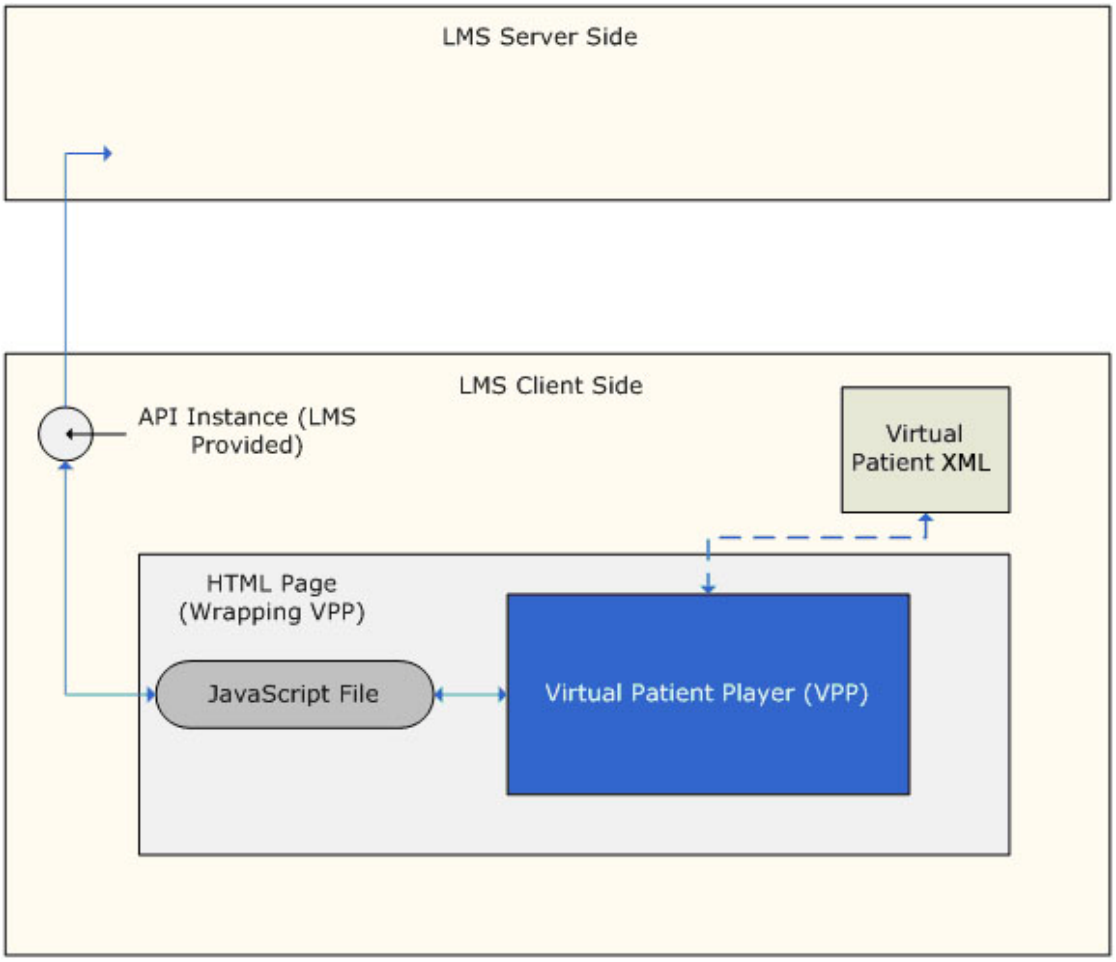

Figure 8: Virtual Patient Communications with an LMS, Modified from diagram made by Schawn Thropp, Concurrent Technologies Corporation

The HTML file contains JavaScript code, the use of which is required by the SCORM reference model. Our example file contains a frame, which in turn contains the virtual patient player. Thus, this HTML file is a container, wrapping the virtual patient to make it into a SCO.

An example launchVP.html follows:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title>
      A simple SCO wrapping a Virtual Patient Player
   </title>
   <script type="text/javascript" src=”SCOFunctions.js">
   </script>
</head>
  <frameset rows="100%,*">
   <frame id="StageFrame" src="VirtualPatientPlayer.swf" />
    //here replace src value with URL of local Virtual Patient Player
   <frame id="DummyFrame" src="about:blank" />
  </frameset>
</html>
```

Modified from In the Eye of the SCORM An introduction to SCORM 2004 for Content Developers, by Claude Ostyn, *www.ostyn.com/standards/docs/Eye_Of_The_SCORM_draft.pdf*

In the example above, SCOFunctions.js is a javascript file that is placed in the same directory as this HTML file (note, could be in a sub-folder if src path is updated accordingly). This file contains the functions necessary to search and find the SCORM API provided that an LMS launched the launchVP.html file. Once found, the JavaScript code will take care of initializing and terminating communication with the LMS. Initialization should happen when launchVP.html is loaded. Termination should happen when launchVP.html is unloaded (ie, the window is closed). Further, the JavaScript code provides functions for the player to communicate with the LMS. These are the functions the player needs to use in order to report back counter values as well as completion status. They can also be used to report back any other type of data that is tracked by SCORM conformant LMSs.

Initialization and termination are the minimum requirements for a SCO.

9.3 Minimum reporting for Virtual Patient player in VP Package

During the LMS launched execution of a MedBiquitous Virtual Patient by the player included in a package, the following minimum conditions must be satisfied.

- The activity must be initialized and terminated with the LMS. This is usually taken care of by the JavaScript code included in the HTML wrapper page.
- If the learner reaches a leaf node, the player must set the completion status to completed. Otherwise, if the activity is started but a leaf node is not reached, the completion status must be set to incomplete.
- In the case that the Activity contains only one counter, this score is reported by the player using the main raw score variable of the LMS.

Individual players may choose to add additional functionality like reporting back time spent in the activity (in the session_time element), suspend data (to allow learner to suspend activities between sessions), success status (in the cmi.success_status element), objective completions status, and objective scores etc. Success status is not defined because different activities and different institutions may have different benchmarks for success.

Data that Player Must Report Back to LMS	LMS Element
Completion status (complete, incomplete, not attempted)	cmi.completion_status
Counter, if only one exists	cmi.score.raw

For more information of the SCORM elements available in SCORM conformant LMSs see the *SCORM® 2004 4th Edition Run-Time Environment (RTE) Version 1.0 Specification*.

9.3.1 Communicating with the LMS

There will be cases in which a player would like to either retrieve data from the LMS or pass data back to the LMS. In this case, the SCOFunctions.js provides helper functions to undertake this task. Note SCOFunctions.js is provided as a helpful example of how to communicate with the LMS. Player developers can choose to write their own JavaScript to communicate with the LMS. Players using a different JavaScript interface will be conformant as long as they follow the minimum SCORM requirements for players as stated above. These players should also be able to handle running in the absence of an LMS. Thus in the absence of an LMS, players should either recognize the absence of an LMS and refrain from making calls to LMS APIs or the JavaScript interface to the LMS (the JavaScript file included in the lauchVP.html file) must handle errors when calls are made by the player but no LMS is present.

Note: The following examples are applicable when SCOFunctions.js file is used to interface with the LMS. They are provided to illustrate the process of communication with the LMS. If a package developer writes their own JavaScript to interface with the LMS, these calls and function names may differ. The player included in the package should be compatible with the JavaScript file included in the HTML wrapper file (ie the player should know which function to call in order to communicate with the LMS). The SCOFunction.js is taken from Claude Ostyn. The name of the original file is ostyn2004sco.js. It is documented further in *In the Eye of the SCORM An Introduction to SCORM 2004 for Content Developers*, available at http://www.ostyn.com/standards/scorm/samples/ostyn2004sco.js and is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 2.5 License.

9.3.2 Passing Data Back to the LMS

In order to get data to the LMS the following function should be used, called from within the virtual patient in JavaScript:

```
window.parent.ScormSetValue(somevalue, settothis)
```

The virtual patient is running as a frame inside the launchVP.html file. The window.parent prefix will tell the web browser to look at the parent window of the virtual patient frame, which is the launchVP.html file. ScormSetValue is a function in the SCOFunctions.js file. Note that both 'somevalue' and 'settothis' are of type String. 'somevalue' is the name (in string form) of the SCORM variable that is to be updated. 'settothis' is the value (in string form) to which the SCORM variable should be set.

Eample of setting the completions status:

```
window.parent.ScormSetValue("cmi.completion_status", "complete")
```

This will set the variable cmi.completion_status in a SCORM conformant LMS to complete.

Example of setting the raw score:

```
window.parent.ScormSetValue("cmi.score.raw", "234")
```

This will set the variable cmi.score.raw in a SCORM conformant LMS to 234.

Note there are many other variables that can be accessed in this way. For a full list see *SCORM® 2004 4th Edition Run-Time Environment (RTE) Version 1.0 specification*.

For more information on the functions used in the SCOFunctions.js see *In the Eye of the SCORM An introduction to SCORM 2004 for Content Developers.*

9.4 Non-LMS Launch of Virtual Patient

MedBiquitous Virtual Patients do not need to be run in the context of an LMS. They can be used as independent learning resources or imported to a virtual patient system. One of the following conditions should be met to allow the virtual patient activity to be run outside of an LMS using the player in the package:

- The player recognizes the absence of an LMS and refrains from making calls to LMS APIs

  OR

- The JavaScript interface to the LMS handles errors when calls are made by the player but no LMS is present.

In the SCOFunctions.js file the former solution is implemented. If the player included in the package calls functions in SCOFunctions.js when the LMS was not found, these calls will return the string “false.” Is it up to the player to recognize this and continue with the activity. In this case, because no LMS is present, it will be impossible for the player to report any information pertaining to the activity.

To launch an activity outside of an LMS, the learner should open the virtual patient package and launch the wrapper HTML file manually in a web-browser. The HTML wrapper file will in turn launch the player within the browser.

10 Handling the Manifest

MedBiquitous Virtual Patients use the SCORM 2004 4th Edition profile of IMS Content packaging v1.1.4 in order to group together the resources of a virtual patient. Each virtual patient will contain a file named imsmanifest.xml. This file will contain a file path to all files (text and media) included in the virtual patient.

10.1 Virtual Patient Title

If the player makes use of the virtual patient title, it can be found through the metadata element defined inside the manifest. The metadata element contains an adlcp:location element which points to an external metadata file. This file uses Healthcare Learning Object Metadata to describe the virtual patient activity, including the title. The location of the title is shown in the example below:

```
<lom>
   <general>
      . . .
      <title>
         <string>  MVP Title Here </string>
      </title>
       . . .
   <general>
     . . .
</lom>
```

10.2 Anatomy of Resources in the Manifest File

The manifest matches each media resource file used by the Virtual Patient with a unique identifier and the media resource's location in the file system. The location of the media file is expressed using relative referencing and will usually be in a sub-folder of the root of the virtual patient package.

For example:

```
      <resource identifier="R_A2" type="webcontent" adlcp:scormType="asset"
href="xray.jpg">
         <file href="xray.jpg" />
      </resource>
```

Here R_A2 is the unique identifier for the media resource. Its location is in the root of the virtual patient folder and its file name is xray.jpg.

See the *MedBiquitous Virtual Patient Specification and Description Document* for more information on the how manifest is used in the context of virtual patients. Also see the *SCORM® 2004 4th Edition Content Aggregation Model (CAM) Version 1.0* specification for more information on how SCORM uses manifest files.

10.3 Looking up Media Resources in the Manifest

In order to find the file path to media resources referenced from the DAM or VPD, the player must look up these resources in the manifest.

Media references from the DAM will be in the form of an absolute XPath reference which is to be used to construct a query on the manifest file. They will be of the following form:

```
/manifest/resources/resource[@identifier = 'X']
```

X will be the identifier attached to the media resource. The file path to the media resource is located in the href attribute of the resource element. In order to retrieve the file path to the media file, the following XPath can be used:

```
/manifest/resources/resource[@identifier = 'X']/@href
```

This will retrieve the href attribute of the resources with identifier *X*.

Once the player holds the file path to a media file, the player can retrieve the resource from the local file system.

11 References

MedBiquitous Documents:

- MedBiquitous Virtual Patient Specifications and Description Document
  http://www.medbiq.org/working_groups/virtual_patient/VirtualPatientDataSpecification.pdf

- Healthcare Learning Object Metadata Specifications and Description Document
  http://www.medbiq.org/working_groups/learning_objects/HealthcareLOMSpecification.pdf

Advanced Distributed Learning (ADL) Documents:

- SCORM® 2004 4th Edition Content Aggregation Model (CAM) Version 1.0

- SCORM® 2004 4th Edition Run-Time Environment (RTE) Version 1.0

- Sharable Content Object Reference Model (SCORM)® 2004 4th Edition Overview Version 1.0

Others documents:

- In the Eye of the SCORM An introduction to SCORM 2004 for Content Developers by Claude Ostyn
  http://www.ostyn.com/standards/docs/Eye_Of_The_SCORM_draft.pdf

- XML Path Language (XPath) 2.0, W3C Recommendation
  http://www.w3.org/TR/xpath

- XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition)
  http://www.w3.org/TR/xhtml1/