Grant Agreement Number ECP-2006-EDU-410030

www.virtualpatients.eu

VP Profile implementation and conformance testing

Deliverable number/name	*D2.2*
Dissemination level	*Public*
Delivery date	*09/01/09*
Status	*Draft/Final*
Author(s)	*eViP Technical Reference Group*

e*Content*plus

This project is funded under the *e*Content*plus* programme[1],
a multiannual Community programme to make digital content in Europe more accessible, usable and exploitable.

[1] OJ L 79, 24.3.2005, p. 1.

1 OVERALL DESCRIPTION 4

1.1 STRUCTURE OF THE DOCUMENT 4
1.2 INTRODUCTION 4
1.3 BACKGROUND 4
1.4 THE EVIP APPLICATION PROFILE 5
1.5 VIRTUAL PATIENT SYSTEMS 5
1.6 COMPONENTS OF THE EVIP APPLICATION PROFILE 6
1.6.1 The MedBiquitous Virtual Patient Specification 6
1.6.2 The eViP Metadata 10
1.6.3 The SCORM package 15
1.6.4 Conformance metrics 17

2 DESCRIPTION OF THE PROFILE IMPLEMENTATION IN EACH VP SYSTEM 19

2.1 CAMPUS 19
2.1.1 Description of the system and the VP model used by the system 19
2.1.2 Mapping of the VP XML MedBiquitous Standard to the Different Elements/Modules of the Player 20
2.1.3 Export module 23
2.1.4 Import Module 27
2.2 CASUS 32
2.2.1 Description of the system and the VP model used by the system 32
2.2.2 Mapping of the VP xml MedBiquitous standard to the different elements/modules of the player. 33
2.2.3 Export module 34
2.2.4 Import module 37
2.3 OPENLABYRINTH 40
2.3.1 Description of the system and the VP model used by the system 40
2.3.2 Mapping of the VP xml MedBiquitous standard to the different elements/modules of the system 41
2.3.3 OpenLabyrinth and VUE 41
2.3.4 OpenLabyrinth and QTI 42
2.3.5 Export module 42
2.3.6 Import module 45
2.3.7 Static aspects (e.g. class diagrams) 47
2.3.8 Dynamic aspects (e.g. activity/state diagrams) 50
2.4 WEB-SP 51
2.4.1 Description of the system and the VP model used by the system 51
2.4.2 Export module 53
2.4.3 Import module 54

3 TESTING FOR CONFORMANCE 55

3.1.1 Levels of conformance 55
3.1.2 eViP compliancy of a VP system 57
3.1.3 Conformance application 57
3.2 EVIP CONFORMANCE TESTING SUITES 58
3.2.1 Introduction 58
3.2.2 Description of eViP conformance testing application developed by KI 59
3.2.3 Description of XSLT-based eViP conformance testing application developed by HD 60
3.3 OTHER TOOLS AND RESOURCES USEFUL IN EVIP PROFILE CONFORMANCE TESTING 63
3.3.1 Description of conformance tests in Altova XML Spy 63
3.4 TEST CASES 64
3.5 VALIDATION OF IMPORTED CASES IN TARGET SYSTEMS - PILOT STUDY 65
3.6 STATISTICS OF MVP USAGE IN EVIP APPLICATION PROFILE 69
3.6.1 Introduction 69
3.6.2 Results 70
3.6.3 Conclusions 70

4 SUMMARY 70

4.1 SUCCESSES & CHALLENGES 71
4.2 FUTURE WORK 71

5 REFERENCES 71

6 ANNEX 1 STATISTICS OF MVP USAGE IN EVIP APPLICATION PROFILE 72

6.1 FREQUENCY OF OCCURRENCE OF MVP ELEMENTS IN ALL 6 TEST PACKAGES 72
6.2 RELATIVE TEXT WEIGHT OF MVP ELEMENTS IN ALL 6 TEST PACKAGES 74

7 ANNEX 2 MVP XML AND EVIP LOM SPECIFICATIONS 77

7.1 THE MVP SPECIFICATION 77
7.2 EVIP LOM 77

1 Overall description

1.1 Structure of the document

The goal of this document is to report on the implementation outcomes of the eViP Application Profile and the conformance testing process in all partner systems. At the start of the project there were as many as 5 virtual patients systems in operation. Since then, one of these systems has been withdrawn from the project and eViP has continued with the 4 main systems: WebSP, Casus, CAMPUS and OpenLabyrinth. It is a general viewpoint within eViP that the reduction in VP systems will be an advantage in the medium to long term as it reduces the number of different interfaces that users may need to become familiar with. This report takes into account the learnings and experiences from these 4 systems. The text is divided into four chapters:

1. Overall description - gives an overview of the eViP Application Profile in the current version and its elements
2. The VP systems - describes in detail the implementation process of the profile in each of the partner VP systems.
3. Testing for conformance - characterises in detail the conformance testing applications as well as the conformance protocol. Presents the import and export results of all systems.
4. Summary - Recapitulates the work done in this deliverable and gives and outlook for future work.

1.2 Introduction

One of the major goals of the eViP project is to enable the sharing of virtual patients across medical education centres. This addresses the frequent problem in development of e-learning resources, which is the lack of time and resources while in the same time lot of effort is wasted on duplicative work. The foundation of a successful and efficient bank of educational resources is a common interoperability standard that would facilitate the migration of data between diverse systems. Rather than reinventing the wheel, the eViP consortium has decided to analyse and adapt the already existent technical specifications.

1.3 Background

A significant milestone in standardisation of e-learning content was definitely the ADL SCORM (Sharable Content Object Reference Model) specification [21]. Despite its broad technical capabilities and growing user acceptance the specification could not cover completely all expectations of the community. Especially in the case of specialised learning objects, as e.g. virtual patients, the general scope of the specification does not seem to take advantage of all potentials of the shared content

The MedBiquitous Virtual Patient Working Group has developed a virtual patient data specification (MVP) defining a common format for exchanging the content of virtual patients [17] [19]. The specification utilises the advantages of SCORM reusable learning objects (like IMS content packaging or IEEE LOM metadata) while adding new features which are characteristic of virtual patients. This includes the definition of a common framework for storing static data of virtual patients (VPD), activity (AM) and data availability (DAM)

models describing the dynamic aspects of virtual patients, as well as an extended version of the IEEE LOM metadata specification (MedBiquitous Healthcare LOM) enabling fast discovery of relevant content. The requirements for a run-time environment are specified in a separate document entitled MVP Player Specifications and Description [18].

1.4 The eViP application profile

In practice it is not always feasible to implement whole specifications. Many of the features provided by the standards may be outside the areas of interest for a particular target-user community. On the other hand, some aspects of a general purpose specification may be defined too broad and need to be constrained. To solve these problems and accelerate the implementation of standards the concept of Application Profiles has been introduced. An Application Profile is defined as an assemblage of "data elements drawn from one or more namespaces, combined together by implementers, and optimised for a particular local application" [8] [22]. The aim of application profiles is not the declaration of new terms and definitions, but the selection and re-use of existing elements to tailor to the needs of a given group of users.

The eViP Application Profile 1.0 has been defined in the deliverable D2.1 as a profile of the MedBiquitous Virtual Patient in the version 0.46. The version 2.0.1 defined in April 2008 adapted the version 0.48 of MVP and included in the profile the IEEE 1484.12.1-2002 Standard for Learning Object Metadata (LOM) and MedBiquitous MEDBIQ LO.10.1-2008 Healthcare Learning Object Metadata Specification (Healthcare LOM). In addition it was decided to move up from IMS Content Packaging method to SCORM packages. This version was signed off for implementation.

In the course of the implementation further specifications proved to be useful in the realisation of the profile by some of the partners. Among them were a profiled version of the IMS Question and Test Interoperability specification (QTI) implemented by the CASUS and CAMPUS system and W3C Timed Text (TT) Authoring Format 1.0 – Distribution Format Exchange Profile (DFXP) implemented in the CAMPUS system. However, both specifications are still regarded as external to the profile in the current version of the eViP AP (v. 2.0.1).

The application profile includes additionally a conformance testing suite, which is a set of software tools for testing the degree to which the interchangeable virtual patient packages and the partner systems conform to the requirements of the application profile. There are four different levels of eViP compliance, which can be tested by two independent applications.

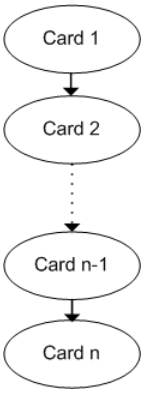
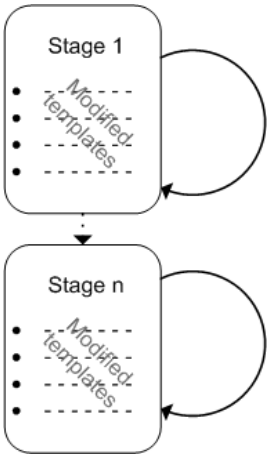
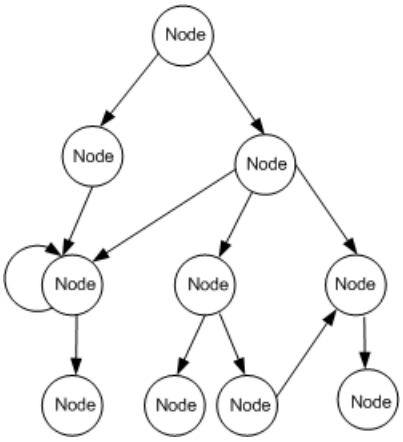
1.5 Virtual patient systems

The eViP application profile is developed to meet the needs of exchanging virtual patient packages between four systems

- CAMPUS (with two variants CAMPUS VP and CAMPUS Key Feature)
- CASUS
- OpenLabyrinth
- Web-SP

1.5.1.1 Virtual patient models

The navigation model enables the division of the systems in three major classes with different properties (table1):

Model	Illustration	Description
Linear		Virtual patients following this model are comprised of linearly ordered cards. A card consists of a text field, may contain media resources (as images and videos), hyperlinks to external resources, and (optionally) a set of one or more questions of diverse types. The transition from one card to another is often conditional upon answering a question. Navigation in this model is sometimes restricted to moves by one card forward (backward movements are usually unrestricted). Examples: CASUS, CAMPUS Key Feature
Semi-Linear		Semi-linear cases enable the choice from long lists of options. They are usually build of templates containing many standard interview answers, typical physical examination results or laboratory values. The default entries are modified to show symptoms of the given disease. The navigation between different stages of patient treatment is often linear. Examples: CAMPUS VP, Web-SP
Branched		Models of branched cases are graphs consisting of nodes (similar to cards in the linear model) interconnected by edges representing the potential decisions of the learner. Solving a case consist in finding a path through the graph. Examples: OpenLabyrinth.

Tab. 1 Virtual patient models in eViP profile

A more detailed insight into the systems implementing the eViP application profile is given in the next chapter.

1.6 Components of the eViP Application Profile

1.6.1 The MedBiquitous Virtual Patient Specification

The MedBiquitous Virtual Patient Specification consists of five main components:

- Virtual Patient Data (VPD)
- Media Resources (MR)
- Data Availability Model (DAM)
- Activity Model (AM)
- Virtual Patient Player (VPP)

XML Schemas in the versions adopted by the eViP Application Profile 2.0.1 are attached in Appendix I.

1.6.1.1 Virtual Patient Data (VPD)

Files: *virtualpatients.xml, virtualpatientdata.xsd*

This section contains the demographic and clinical data of the virtual patient as well as the narrative of the simulation.

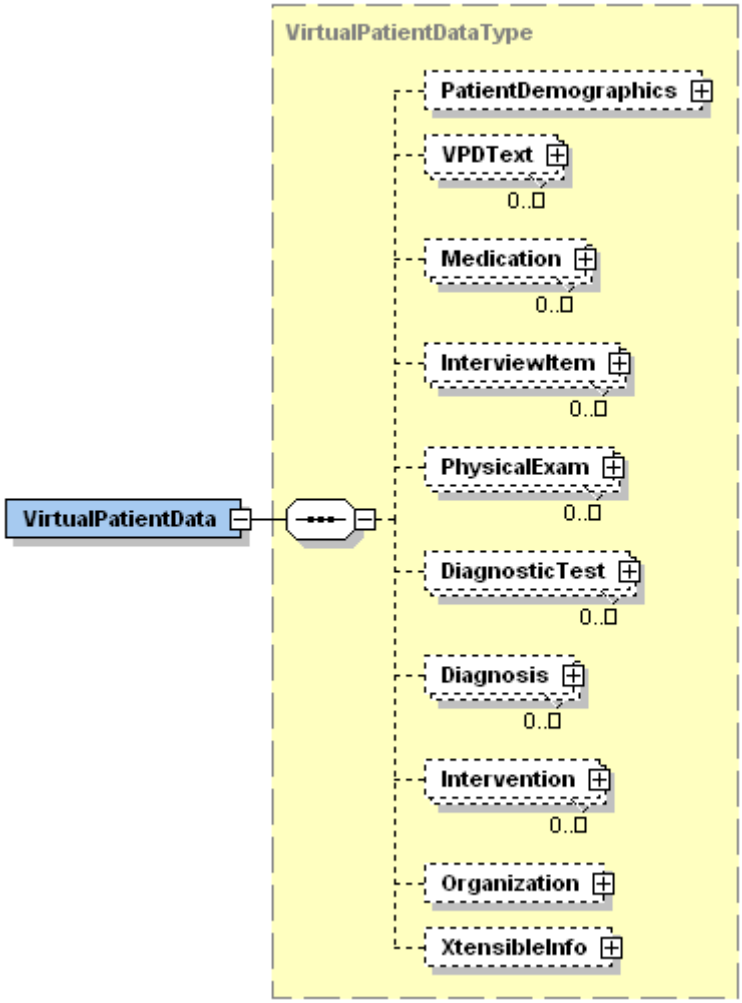


Fig. 1 Child elements of the MVP's VirtualPatientData

The clinical data is divided into six segments Medication, InterviewItem, PhysicalExam, DiagnosticTest, Diagnosis, Intervention (Fig. 1). Each of these parts split further into more detailed elements. The section VPDText provides general purpose fragments like narratives or

text descriptions. VPDText is frequently used by systems with models in which the patient data is not highly structured. The text may be XHTML formatted, however not all XHTML tags are permitted. The Organization element establishes a hierarchical structure of elements that can be used by authoring systems while importing the data. The last section XtensibleInfo provides a method for including into the packages elements from external specifications or content which is characteristic for a particular VP system. In the eViP profile it is used e.g. for adding QTI or Timed Text code. The XtensibleInfo is also permitted in DAM and AM sections.

1.6.1.2 Media Resources (MR)

Files: *imsmanifest.xml, imscp_v1p1.xsd, adlcp_v1p3.xsd*

Contains references to media files like images, video and audio clips, animations related to the simulated clinical scenario. All resources need to be declared in the manifest file described in more detail in the SCORM subchapter. Other elements from the MVP specification (Data Availability Model) refer to the resources by XPath expressions.

1.6.1.3 Data Availability Model (DAM)

Files: *dataavailabilitymodel.xml, dataavailabilitymodel.xsd*

The role of the Data Availability Model is to aggregate the VPD, MR and subordinated DAM elements for display in the player. The elements can be displayed immediately upon reaching a given activity node or at a later point after a trigger has been activated (e.g. after user interaction or a period of time). All elements in the DAM Nodes are referenced by XPath expressions. In the figure below a snippet of code is presented showing a DAM node aggregating a VPD, two MR and one smaller DAM node. The VPD and one of the MR nodes is displayed immediately after reaching an activity node, whereas the second MR and the DAM node are presented after releasing a trigger.

```
<DAMNode id="CARD_12977">
    <DAMNodeLabel>radiographic signs of atelectasis</DAMNodeLabel>
    <DAMNodeItem display="immediately">
        <itemPath>/VirtualPatientData/VPDText[@id='INFO_12978']</itemPath>
        <itemOrder>0</itemOrder>
    </DAMNodeItem>
    <DAMNodeItem display="immediately">
        <itemPath>/manifest/resources/resource[@identifier='MM_12983']</itemPath>
        <itemOrder>1</itemOrder>
    </DAMNodeItem>
    <DAMNodeItem display="ontrigger">
        <itemPath>/VirtualPatientData/VPDText[@id='TEXT_HL_12979']</itemPath>
        <itemOrder>0</itemOrder>
    </DAMNodeItem>
    <DAMNodeItem display="ontrigger">
        <itemPath>/DataAvailabilityModel/DAMNode[@id='COMMENT_12975']</itemPath>
        <itemOrder>1</itemOrder>
    </DAMNodeItem>
</DAMNode>
```

1.6.1.4 Activity Model (AM)

Files: *activitymodel.xml, activitymodel.xsd*

The Activity Model defines the way a learner may navigate through the content of the virtual patient. Many different types of activity models can be implemented including all three eViP navigation models (linear, semi-linear and branched). In addition to expressing the topology of the navigation graph this part of the MVP specification defines user scoring criteria, timers, conditional rules of navigation and elements for aggregating activity nodes.

To define the activity model the list of available states needs to be defined. Each state references to DAM nodes that can be displayed in the given activity.

The following example shows a nodesection representing a card in a linear VP system. This card is comprised of two activity nodes and the main text entitled “Patient presentation” with a comment. This nodesection is entitled “Patient history and physical examination”.

```
<ActivityNodes>
  <NodeSection id="DIUT_12942" label="Patient history and physical examination">
    <ActivityNode id="CARD_12944" label="Patient Presentation">
      <Content>/DataAvailabilityModel/DAMNode[@id = 'CARD_12944']</Content>
    </ActivityNode>
    <ActivityNode id="ANSWERCOMMENT_12952" label="ANSWERCOMMENT">
      <Content>/DataAvailabilityModel/DAMNode[@id = 'ANSWERCOMMENT_12952']</Content>
    </ActivityNode>
  </NodeSection>
</ActivityNodes>
```

In the next fragment of MVP code three activity nodes are interconnected forming a linear path. The link entitled “Next” joins the CARD_12944 node to the CARD_12952 node, and the link entitled “Solution” joins the CARD_12952 node to the CARD_12960 node.

```
<Links>
  <Link label="Next" display="On">
    <ActivityNodeA>/ActivityModel/ActivityNodes/NodeSection/ActivityNode[@id='CARD_12944']</ActivityNodeA>
    <ActivityNodeB>/ActivityModel/ActivityNodes/NodeSection/ActivityNode[@id='ANSWERCOMMENT_12952']</ActivityNodeB>
  </Link>
  <Link label="Solution" display="On">
    <ActivityNodeA>/ActivityModel/ActivityNodes/NodeSection/ActivityNode[@id='ANSWERCOMMENT_12952']</ActivityNodeA>
    <ActivityNodeB>/ActivityModel/ActivityNodes/NodeSection/ActivityNode[@id='CARD_12960']</ActivityNodeB>
  </Link>
</Links>
```

1.6.1.5 Virtual Patient Player (VPP)

The requirements for a MVP conformant Virtual Patient Player are described in detail in the Virtual Patient Player Specification and Description Document. The implementation of this specification is out of the scope of the eViP application profile. However, all eViP partner systems (CAMPUS, CASUS, OpenLabyrinth and Web-SP) by being conformant to the eViP profile fulfil many of the requirements for a MVP VPP player.

1.6.1.6 Structure of the MVP package

The MVP virtual patients are packaged as folders and distributed as a file-archive in a standard zip-format.

Folder structure should be:

ROOT

MVP

```
+ - activitymodel.xsd
+ - activitymodel.xml
+ - dataavailabilitymodel.xsd
+ - dataavailabilitymodel.xml
+ - virtualpatientdata.xsd
+ - virtualpatientdata.xml
+ /media
  + - image1.jpg
  + - image2.jpg
```

Scorm

```
+ index.html
+ player.swf
+ adlcp_v1p3.xsd
+ imscp_v1p1.xsd
+ imsmanifest.xml
+ datatypes.dtd
+ xml.xsd
+ XMLSchema.dtd
+ /address
  + address.xsd
+ /common
  + anyElement.xsd
  + dataTypes.xsd
  + elementNames.xsd
  + elementTypes.xsd
  + rootElement.xsd
  + vocabTypes.xsd
  + vocabValues.xsd
+ /extend
  + custom.xsd
+ /unique
  + strict.xsd
+ /vocab
  + custom.xsd
```

Metadata

```
+ metadata.xml
+ healthcarelom.xsd
+ /healthcare
  + healthcaremetadata.xsd
  + healthcarevocabularies.xsd
```

XHTML

```
+ xhtml1-strict.xsd
```

1.6.2 The eViP Metadata

Files: *metadata.xml, healthcarelom.xsd*

Metadata (in other words data about data) are descriptive labels used to index (learning) resources to make them easier to find and use [23]. In the eViP application profile the role of the metadata specification is based upon a subset of the IEEE 1484.12.1-2002 Standard for Learning Object Metadata and MedBiquitous Healthcare Learning Object Metadata Specification. The primary storage location for metadata in the eViP project is the eViP referatory. However, eViP profiled packages may convey that information in metadata files as well. The list of recommended metadata fields can be found in the table below:

Field	Description	XML Binding	Content type	Example
LOM 1.1 repository id	Unique ID generated for the package by the repository. Can be replaced by local id if eViP repository id is not available.	/lom/general/identifier	lom:identifier	<entry>evip:vp:1000263</entry>
LOM 1.2 title	Title of the virtual patient (in English, if available)	/lom/general/title	lom:LangString	<string>Infant with fever</string>
LOM 1.3 language of resource	Language of the narratives in the virtual patient package in the format: required language code followed by multiple, optional, hyphen-prefixed subcodes (ISO 639–1, ISO 3166–1) (e.g en-gb, en-us etc)	/lom/general/language	lom:CharacterString (eViP recommendation [ISO 639–1]-[ISO 3166–1])	de-DE
LOM 1.4 description	Brief description of the virtual patient (in English, if available)	/lom/general/description	lom:LangString	<string>5-month-old Katrin is brought to the pediatric outpatient clinic, she has been having fever for 2 days is increasingly floppy.</string>
LOM 2.3.1 role of contributor	Author of VP, owner, etc - not the same as the person creating the inventory record	/lom/lifeCycle/contribute/role	lom:Vocabulary	<value>author</value>
LOM 2.3.2 author	Name of VP author or owner. Recommended (but not mandatory) is vCard-format.	/lom/lifeCycle/contribute/entity	lom:CharacterString (recommended vCard)	BEGIN:VCARD VERSION:4.0 FN: Benjamin Hanebeck ORG: University of Heidelberg EMAIL;PREF;INTERNET: benjamin.hanebeck@med.uni-heidelberg.de END:VCARD
LOM 2.3.3 date	Date of contribution (when was the VP content finally signed off)	/lom/lifeCycle/contribute/date	lom:DateTime	<dateTime>2007-09-01</dateTime>
LOM 3.2.1 role of metadata contributor	e.g creator or validator (of metadata record – not the VP!)	/lom/metaMetadata/contribute/role	lom:Vocabulary	<value>creator</value>
LOM 3.2.2 author	Name of metadata creator. Recommended (but not mandatory) is vCard-format.	/lom/metaMetadata/contribute/entity	lom:CharacterString (recommended vCard)	BEGIN:VCARD VERSION:4.0 FN: Soeren Huwendiek ORG: University of Heidelberg EMAIL;PREF;INTERNET: soeren.huwendiek@med.uni-heidelberg.de

				END:VCARD
LOM 3.2.3 date	Date of metadata record creation	/lom/metaMetadata/contribute/date/dateTime	lom:DateTime	<dateTime>2008-10-01</dateTime>
LOM 4.1 technical format	MIME type value of the VP content package in accordance with RFC 2048. Recommended value is application/zip	/lom/technical/format	lom:CharacterString (MIME type)	application/zip
LOM 4.2 object size in bytes	Size of the virtual patients in bytes Estimate if not known exactly. The value has to be a non-negative integer. Insert 0 if no estimation is possible.	/lom/technical/size	xs:nonNegativeInteger	3902842
LOM 4.3 location of object	URL/URI to the VP in original system or repository (if available) - may be constant for all VPs.	/lom/technical/location	xs:anyURI	http://galaxy.mi.hs-heilbronn.de:3333/myzms
LOM 5.2 type of resource	Defines type of the resource in the VP package: Defaults to 'Virtual Patient'	/lom/educational/learningResourceType	hx:Vocabulary (Healthcare LOM Vocabulary) Constant for eViP profiles: virtual patient	<value>virtual patient</value>
LOM 6.1 payment required	Is payment for this virtual patient required? yes/no only answers only	/lom/rights/cost	lom:Vocabulary (yes/no)	<value>no</value>
LOM 6.2 subject to copyright	Is virtual patient cleared of copyright? yes/no only	/lom/rights/copyrightAndOtherRestrictions	lom:Vocabulary (yes/no)	<value>no</value>
LOM 6.3 statement of copyright	Free text describing copyright statement (in English). Can be replaced by a static reference to general eViP licence.	/lom/rights/copyrightAndOtherRestrictions	lom:LangString	<string>VP content approved for use with following Creative Common restrictions: attribution, noncommercial and share alike. Patient consent obtained from parents in written form.</string>
LOM 9.2 classification purpose	Purpose of the VP classification. Recommended is 'discipline'. For the ‘discipline’ purpose two classifications are recommended	/lom/classification/purpose	lom:Vocabulary	<value>no</value>

	ICD10 or MeSH.			
LOM 9.2.1 classification source	Source of the classification of VP. Recommended are ‘ICD10’ and/or 'MeSH'. See discussion in rationale.	/lom/classification/taxonPath/source	lom:LangString	<value>ICD-10</value>
LOM 9.2.2.1 classification node identifier	If available: Taxon identifier in the selected classification.	/lom/classification/taxonPath/source/taxon/id	lom:CharacterString	G00.9
LOM 9.2.2.2 classification label	If available: Taxon label in the selected classification.	/lom/classification/taxonPath/source/taxon/entry	lom:LangString	Bacterial meningitis

XML Schemas of metadata specification in the versions adopted by the eViP Application Profile 2.0.1 are attached in Appendix II.

1.6.2.1 Classifications in the eViP Metadata

There are currently two taxonomies recommended by the profile for classifying the resources: ICD-10 and MeSH.

There are many controlled terminologies available in the biomedical sector. New terminologies are published annually. Among well-established terminologies are (just to name a few): ICD, ICPC, CPT, DSM, Read Clinical Codes, SNOMED, Galen, LOINC, MeSH, WHO Drug Dictionary and UMLS [16]. For the description of VP in eViP partner repositories two classification systems has been used: ICD and MeSH. Following the IMS Meta-data Best Practice Guide for IEEE LOM recommendations we examined both classifications in respect to factors listed in the guideline.

The following table shows the strong position of both classifications.

IMS MD BPG Factor	ICD – International Statistical Classification of Diseases and Related Health Problems	MeSH – Medical Subject Headings
1. Authority	World Health Organization	United States National Library of Medicine
2. Stability	First published in 1893 (currently 10th edition)	Derives from Index Medicus published for the first time in 1879 (currently version 2009)
3. Maintenance	Revised at roughly 10-years intervals.	Updated annually.
4. Currency	worldwide	worldwide
5. Coverage	155000 different codes (ICD-10)	25186 subject headings (v. 2009)
6. Multilinguality	Available in many languages including all eViP partner languages	Available in many languages including almost all eViP partner languages (exception: Romanian)
7. Extend of current use	The ICD is used worldwide for morbidity and mortality statistics, reimbursement systems and automated decision support in medicine.	The MeSH is used worldwide for indexing medical literature (journal articles and books)
8. Applicability to user communities requirements	yes	yes
9. Conformance to standards and specifications	guaranteed by the authority	guaranteed by the authority

It is widely acknowledged that it is difficult to find one controlled vocabulary that is accepted by all communities. For that reasons the eViP profile does not enforce the selection of any of these two classifications.

1.6.3 The SCORM package

The MedBiquitous Virtual Patient specification leverages SCORM 2004 3rd edition for content packaging and exchange of files related to a single virtual patient activity. SCORM 2004 is a suite of specifications and standards, including the IMS Content packaging specification, that enables learning management systems (LMS) to import and deliver conformant web-based, self-directed learning. Each SCORM package is a zip file containing an XML manifest file and one or more files, each of which are referenced in the manifest. The virtual patient activity is packaged as a single Shareable Content Object (SCO) within the package, allowing the activity to be launched from conformant LMS.

Each package contains:

- Virtual patient data files
- Media files
- Virtual patient XML schemas
- Manifest document that details the files contained in the package and how the activity can be launched within a learning management system.
- Manifest XML schemas
- Metadata files
- Metadata XML schemas
- Index.html file to launch the activity from an LMS
- Javascript files to enable communications with the LMS (usually separate files)
- Virtual patient player
- XML schema and DTD documents
- XHTML schemas customised for virtual patients

eViP systems importing the virtual patient activity for delivery within their own virtual patient system may choose to ignore the following components of the SCORM package:

- Index.html file to launch the activity from an LMS
- Javascript files to enable communications with the LMS (usually separate files)
- Virtual patient player
- XML schema and DTD documents

Ultimately these components of the virtual patient package may be useful for sharing content with the broader European community, including those institutions that do not have a virtual patient specific system but do have a SCORM-conformant LMS.

The Manifest – Resource References

The manifest file, *imsmanifest.xml*, should contain a reference to each media file and launchable HTML document contained within the content package. These files are listed as resources, as shown within the following example:

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

Every resource must have:

- An identifier attribute to uniquely identify the resource
- A type attribute set to webcontent
- An adlcpscormType set to asset
- An href attribute indicating the name and location of the file associated with the resource
- A file sub-element that list the file associated with the resource

When referencing the index.html file, all media and other data files, including *virtualpatientdata.xml*, *dataavailabilitymodel.xml*, and *activitymodel.xml* should be listed using the dependency subelement. Dependency files are referenced though the unique identifier provided in the resource definition as in the following example:

```
<resource identifier="R_A1" type="webcontent" adlcp:scormType="sco" href="launchVP.html">
  <file href="launchVP.html" />
  <dependency identifierref="R_A2" />
  <dependency identifierref="R_A3" />
  <dependency identifierref="R_A4" />
  <dependency identifierref="R_A5" />
  <dependency identifierref="R_A6" />
  <dependency identifierref="R_A7" />
  <dependency identifierref="R_A8" />
  <dependency identifierref="R_A9" />
</resource>
<resource identifier="R_A2" type="webcontent" adlcp:scormType="asset" href="FakeVirtualPatientPlayer.swf">
  <file href="FakeVirtualPatientPlayer.swf" />
</resource>
<resource identifier="R_A3" type="webcontent" adlcp:scormType="asset" href="scripts/SCOFunctions.js">
  <file href="scripts/SCOFunctions.js" />
</resource>
<resource identifier="R_A4" type="webcontent" adlcp:scormType="asset" href="media/xray.jpg">
  <file href="media/xray.jpg" />
</resource>
<resource identifier="R_A5" type="webcontent" adlcp:scormType="asset" href="media/patientpicture.jpg">
  <file href="media/patientpicture.jpg" />
</resource>
<resource identifier="R_A6" type="webcontent" adlcp:scormType="asset" href="activitymodel.xml">
  <file href="activitymodel.xml" />
</resource>
<resource identifier="R_A7" type="webcontent" adlcp:scormType="asset" href="dataavailabilitymodel.xml">
  <file href="dataavailabilitymodel.xml" />
</resource>
<resource identifier="R_A8" type="webcontent" adlcp:scormType="asset" href="virtualpatientdata.xml">
  <file href="virtualpatientdata.xml" />
</resource>
<resource identifier="R_A9" type="webcontent" adlcp:scormType="asset" href="imsmanifest.xml">
  <file href="imsmanifest.xml" />
</resource>
```

The Manifest – Metadata Reference

The metadata subelement of manifest contains information about the package, not the virtual patient. The metadata subelement should be set to the following:

```
<metadata>
  <schema>ADL SCORM</schema>
  <schemaversion>2004 3rd Edition</schemaversion>
</metadata>
```

The manifest has an organizations subelement. This in turn will have one organization subelement to describe and reference the virtual patient activity. The organization element will have the following attributes and subelements:

- An identifier attribute that provides a unique identifier for this activity
- A title subelement that indicates the title of the virtual patient activity
- An item element that references the index.html resource
- A metadata element that uses the adlcp:location element to reference a metadata file contained within the content package

Example:

```
<organizations default="defaultorg">
  <organization identifier="defaultorg" structure="hierarchical" >
    <title>The org title</title>
    <item identifier="I1" identifierref="R_A1"/>
    <metadata>
      <adlcp:location>metadata.xml</adlcp:location>
    </metadata>
  </organization>
</organizations>
```

The Player

The index.html file should contain javascript code to communicate with the LMS. The following list shows the different functions and API calls.

- Initialise the activity within the LMS: Initialise
- Terminate the activity within the LMS: Terminate
- Set the status to completed if the learner reaches a leaf node: SetValue("cmi.completion_status", "complete")
- Report a counter score to the LMS if one counter and only one counter exists: SetValue("cmi.objectives.0.score.raw", "75")

1.6.4 Conformance metrics

Conformance testing is the process of evaluating the adherence of an implementation to the premises of the specification. The IMS Application Profile Guideline [22] distinguishes between the:

- Conformance of the application profile to the base specification(s)
- Conformance of vendor tools, products, and services to the application profile.

The first kind of conformance is guaranteed by the fact that the eViP application profile is a proper sub-set of the basic specifications by MedBiquitous (MVP and Healthcare LOM) and IEEE (LOM).

The second kind of conformance is graded by the degree of difficulty that divides it up into four conformance levels:

- Level 1 - Package validation
- Level 2 - XML/XSD validation
- Level 3 - Import validation
- Level 4 - Runtime validation

Testing of conformance is carried out by two test suites developed by the eViP partners. At least one package sample from each partner system has been tested for conformance by the suites.

A more detailed report on the conformance levels, application suites and test cases is described in chapter 3.

2 Description of the profile implementation in each VP System

2.1 CAMPUS

2.1.1 Description of the system and the VP model used by the system

CAMPUS consists of several “modules” (Fig. 2). These are:

1. Authoring System (Java-based application for all modules)
2. Classic-Player (a simulative, Java-based player)
3. Card-Player (card-based, (D)HTML-based player, generated from the authoring system)
4. Secure summative assessment software (Java-based application)
5. A special eViP-conformant (D)HTML-based player

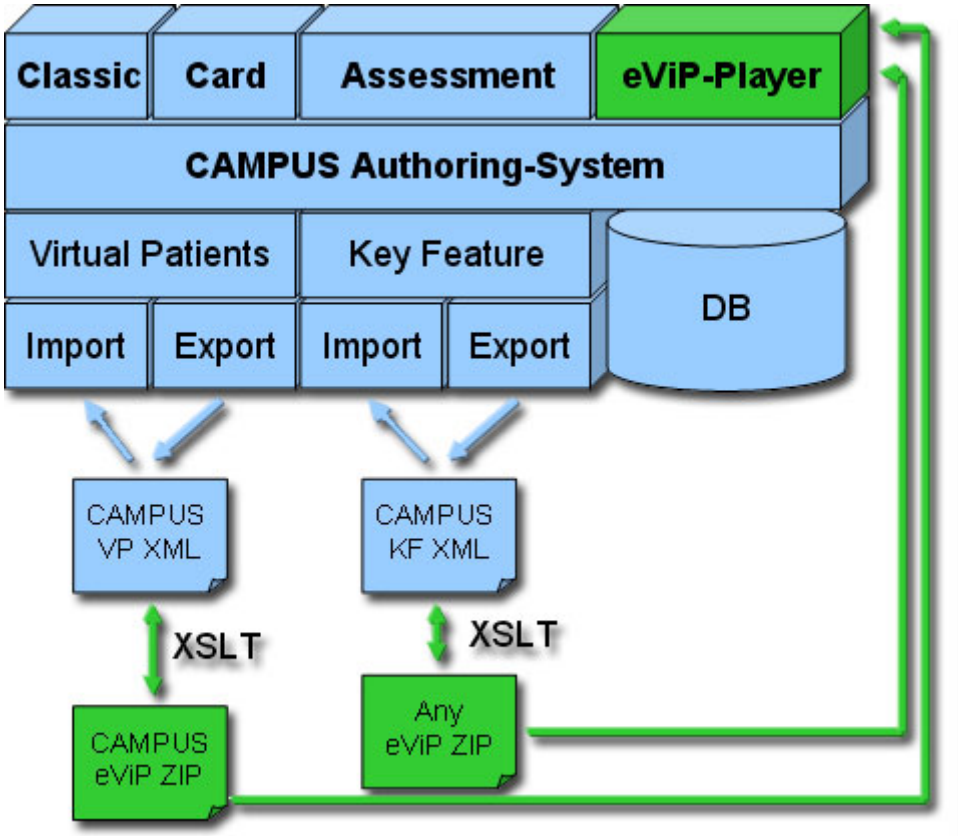


Fig. 2 Structure of the CAMPUS system

Everything can be created using the authoring system. The Virtual Patient data (for the Classic- and Card-Player) can be edited in a special editor while the data for summative assessments, supporting the key feature approach of Page and Bordage [11], are edited in another editor component.

One of the main principles of the CAMPUS Virtual Patient software are:

1. Everything is **vocabulary-based** (medical history, physical and opparative exams, diagnosis, laboratory tests, treatments) with standard answers whenever possible
2. The **semi-linear approach** of the Classic-Player and the **linear approach** of the Card-Player ask the student which tests he wants to do and gives feedback about consonances and differences

For displaying any eViP VP a new web-based player was developed to replace the old Card-Player and to display any MVP content.

For that CAMPUS supports the following modes:

1. **Import and export CAMPUS VP** data according to the eViP standard using the CAMPUS VP authoring system
2. **Import and export any other eViP VP data** and edit the content as text, media and question slides in the Key Feature authoring system
3. **Display any MVP VPs** standard-based with possible system based optimisations

2.1.2 Mapping of the VP XML MedBiquitous Standard to the Different Elements/Modules of the Player

2.1.2.1 CAMPUS VP

2.1.2.1.1 Export

For the CAMPUS VP data the data can be mapped quite easily as CAMPUS VPs uses all the different MVP Virtual Patient elements:

1. *PatientDemographics* for patient informations
2. *VPDText* for
    - summary of medical history
    - summary of physical examinations
    - progression
    - knowledge questions (until support for QTI)
    - links
    - kinds of treatments
    - epicrisis
    - prognosis
3. *InterviewItem* for medical history
4. *PhysicalExam* for physical and technical exams

5. *DiagnosticTest* for laboratory tests
6. *Intervention* for therapies
7. *Diagnosis* for diagnosis

2.1.2.1.1.1 CAMPUS XtensibleInfo

In the *virtualpatientdata.xml* CAMPUS specific data is stored in the *XtensibleInfo* region. The schema file is located inside the generated ZIP file.

2.1.2.1.1.2 QTI

CAMPUS supports six types of questions:

- Single choice
- Multiple choice
- Long menu
- Freetext
- Interval
- Hotspot
- Region of interest

Especially for long menu and region of interest there is no direct support in QTI. A self-definition won't help as currently there is no other software supporting it. Secondly, there has to be an extension definition for the MVP so that systems like CASUS can interpret the references. On this topic working has begun.

2.1.2.1.2 Import

For the import the already existing CAMPUS XML files are used. The XML file is gziped and base 64 encoded put in the *XtensibleInfo* tag of the *virtualpatientdata.xml* document:

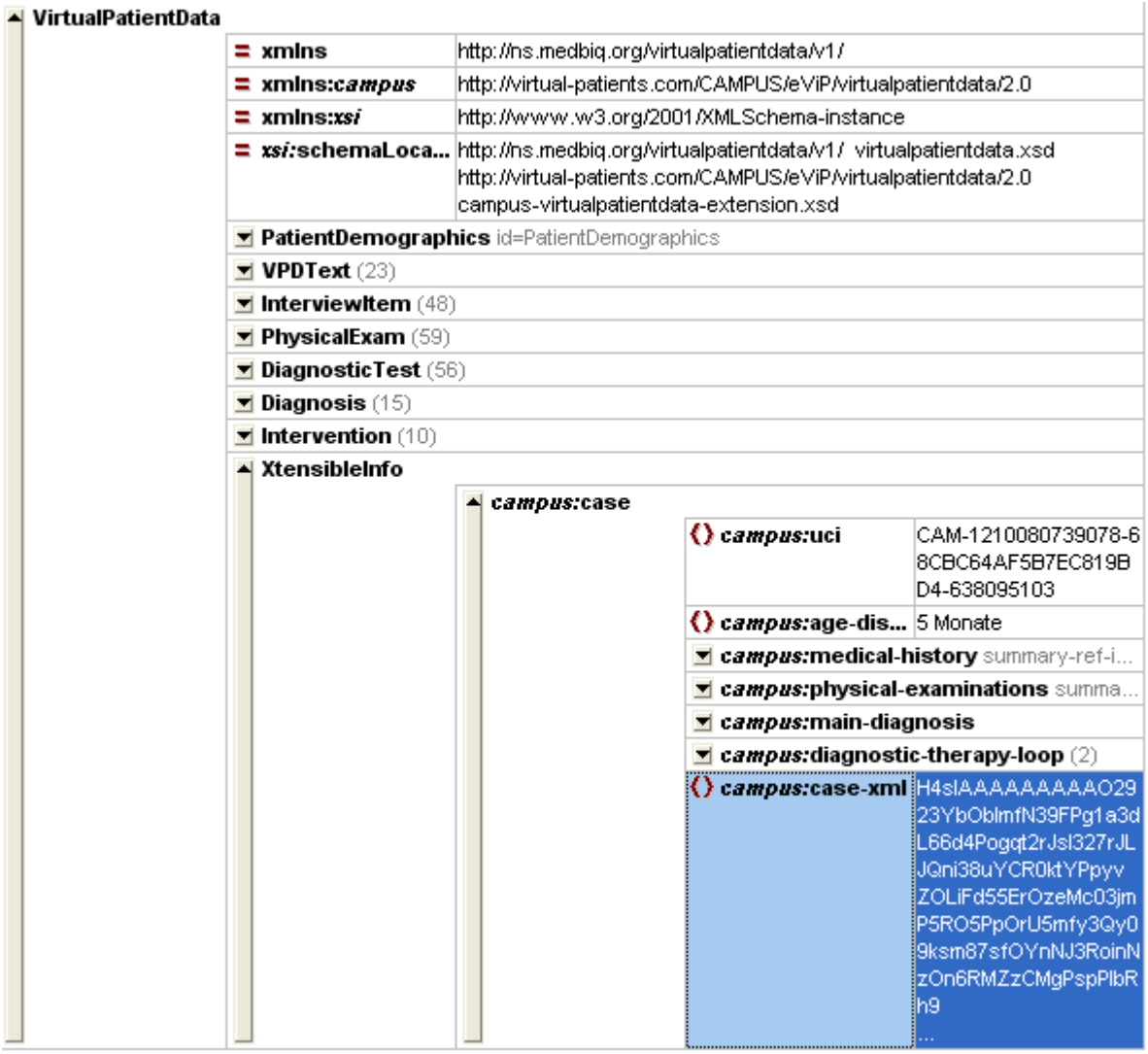


Fig. 3 4 CAMPUS extension of the eViP profile via the XtensibleInfo element

For that only an import of a CAMPUS generated eViP exports are possible with the VP system module.

Image media files are imported directly while videos are automatically transformed into the Flash FLV format and audio files in the MP3 format. It uses ffmpeg for converting. So, most formats are supported. Integrated text data (e.g. in the Quicktime format) will be extracted automatically.

2.1.2.2 Key Feature Sections

2.1.2.2.1 Export

The export serialises the XHTML, media and knowledge questions from each slide into one *ActivityNode* with one text *VPDText*, media references inside the DAM and *VPDText* for each question.In future, a QTI export for the knowledge questions will be implemented. The generated file consists of n *ActivityNode*s (= n Slides).

2.1.2.2.2 Import

The import takes any eViP files and serialises down each *ActivityNode* in XHTML. For more complex types like *PhysicalExam*s XHTML tables are built and imported as plain XHTML. Triggers, rules and meta data are ignored.

Image media files are imported directly while videos are transformed into the Flash FLV format and audio files in the MP3 format.

The import supports two different additional modules (as Xtensible namespaces):

2.1.2.2.2.1 Media Captions using W3C's Timed Text

In order to support captions in media files an extension to the MVP standard has been written (more on that in D2.3b). It separates the content from the media file for independence of the format and in order to easily translate and modify those captions in a standard way. The import can read this extension namespace and will give the author the ability to see and modify those texts.

2.1.2.2.2.2 CASUS QTI Export

CASUS uses QTI to export their knowledge question. For the question types published so far there are translations to the CAMPUS questions types as good as possible. As the export of CASUS is very special (for instance the question itself is not defined in the QTI block and different question types uses the same QTI question type) and the QTI standard is very complex, the transformation using XSLT is made especially for the CASUS export.

Tab. 2 presents the transformation matrix:

CASUS Type	QTI Type	CAMPUS Type
Multiple Choice (MC)	ChoiceInteraction	MC
Freetext	TextEntryInteraction	Freetext
Sorting	OrderInteraction	Cloze
Network	MatchInteraction	Cloze
Mapping	MatchInteraction	Cloze
Lab values	MatchInteraction	Cloze
Long menu	CustomInteraction	Long menu
Cloze	TextEntryInteraction	Cloze
Underline	HotTextInteraction	MC
Non evaluated Freetext	ExtendedTextInteraction	Freetext

Tab. 2 Transformation matrix of QTI elements between CASUS and CAMPUS

2.1.3 Export module

2.1.3.1 Functions Related to Export of VPs

2.1.3.1.1 CAMPUS VP

To export a CAMPUS VP you have to open the case and select *Learning Object → eViP-Export*:

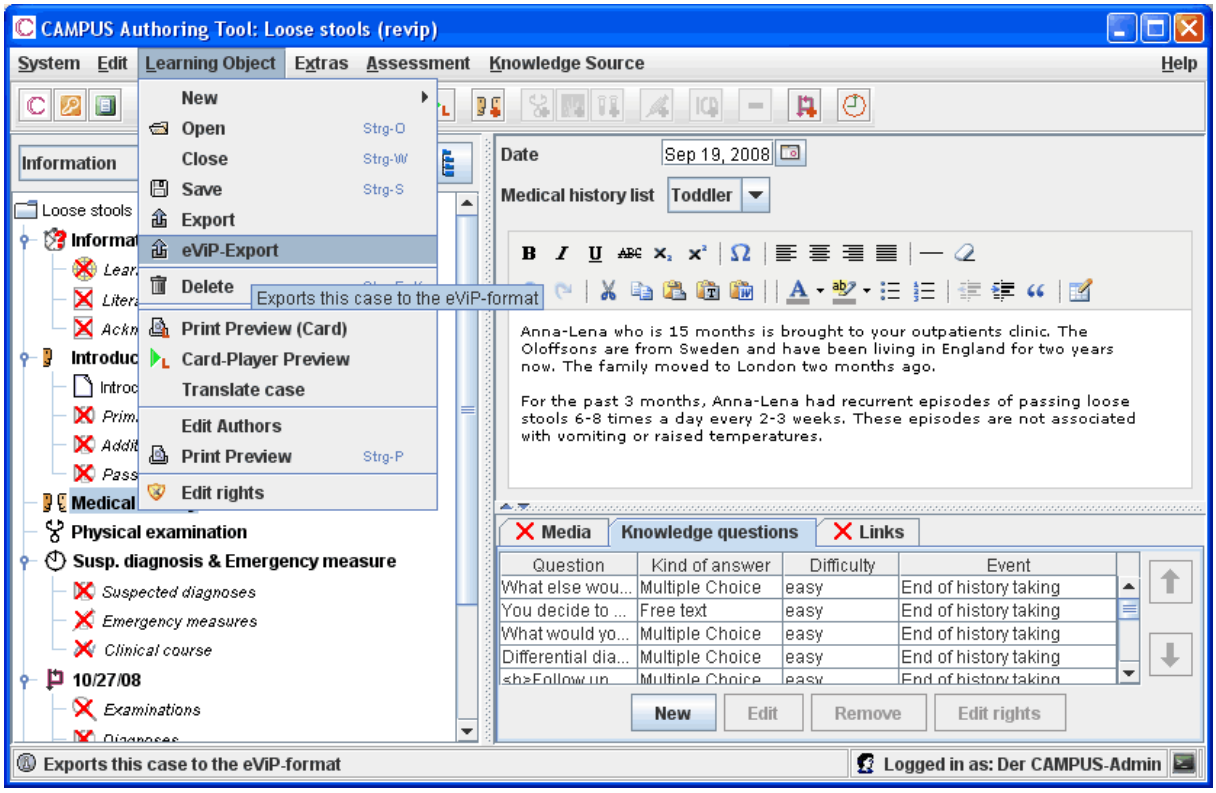


Fig. 5 Export of a CAMPUS VP case from the CAMPUS authoring tool to the eViP format – step 1

After that you can choose where the exported ZIP file should be stored:

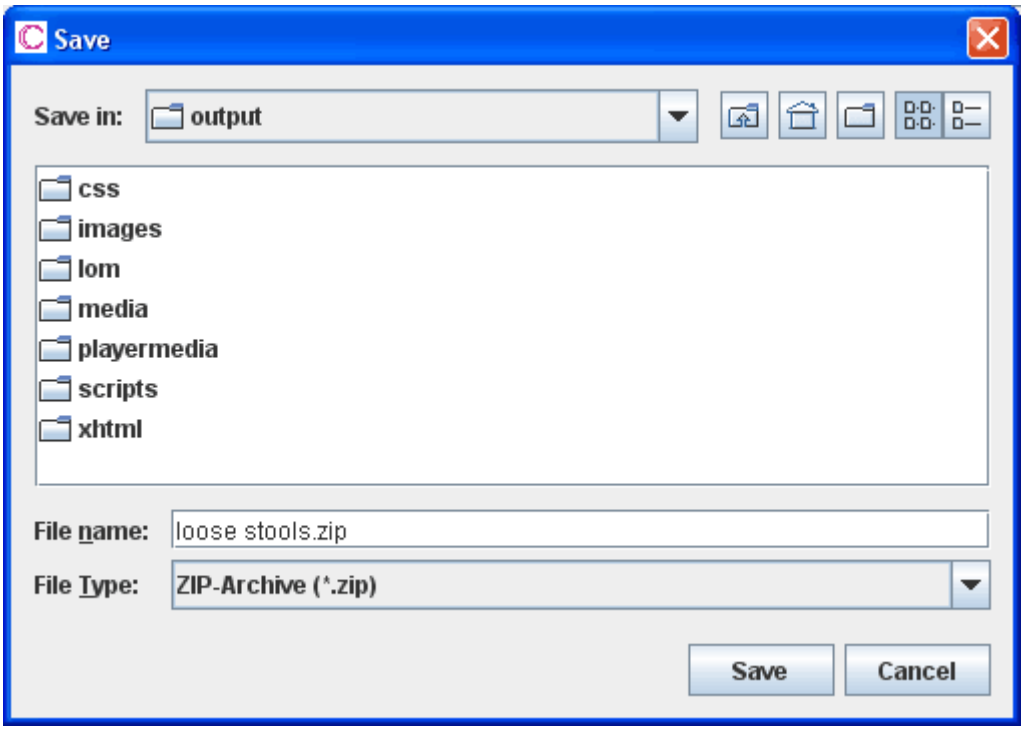


Fig. 6 Export of a case CAMPUS VP from the CAMPUS authoring tool to the eViP format – step 2

2.1.3.1.2 CAMPUS Key Feature

To export a CAMPUS key feature case you have to open the case and select *Learning Object → eViP-Export*:

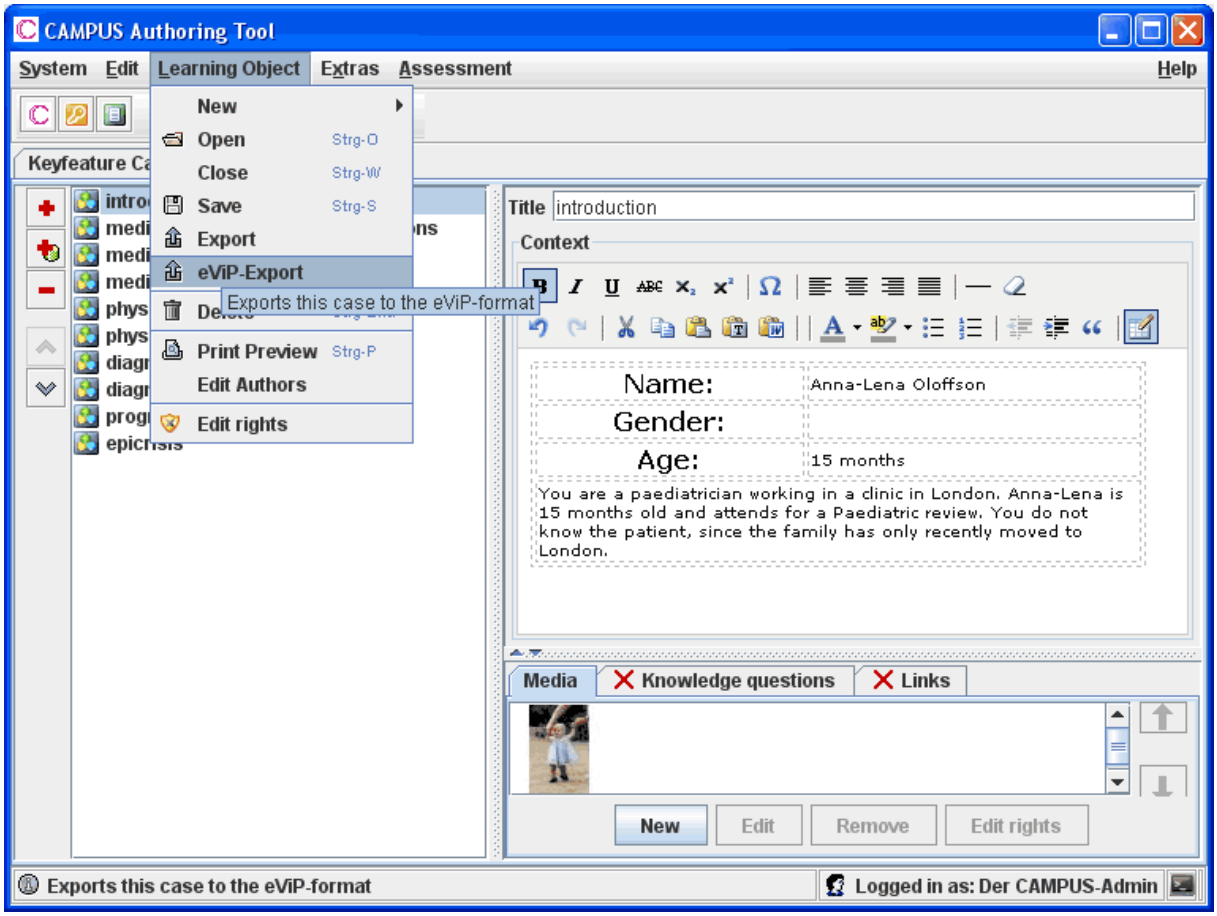


Fig. 7 Export of a CAMPUS Key Feature to the eViP format – step 1

After that you can choose where the exported ZIP file should be stored:

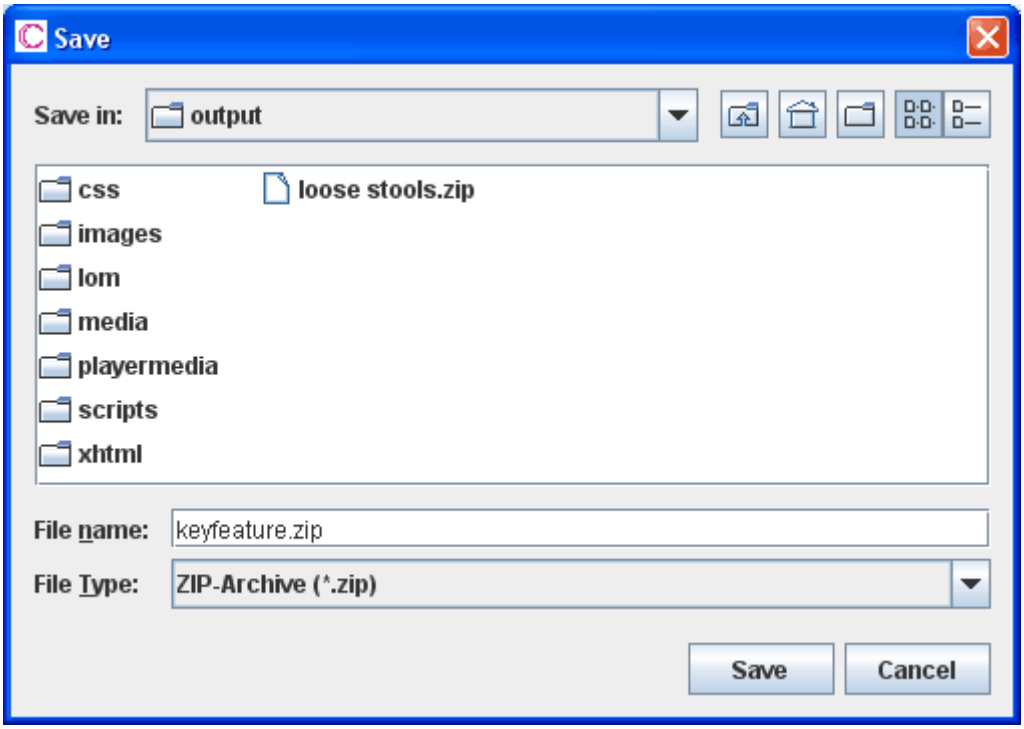


Fig. 8 Export of a CAMPUS Key Feature to the eViP format – step 2

2.1.3.2 Development of the Export Module

2.1.3.2.1 CAMPUS VP

The CAMPUS *fall.xml* has had to be transformed via XSLT files to the VP XML files. This was made possible with http://xml.apache.org/xalan-j/ Xalan-J. For direct importing in CAMPUS the original CAMPUS XML file is GZIPed (for smaller space requirements) and base64 encoded put inside the *XtensibleInfo*.

2.1.3.2.2 CAMPUS Key Feature

The CAMPUS *keyfeature.xml* has had to be transformed via XSLT files to the VP XML files. This was made possible with http://xml.apache.org/xalan-j/ Xalan-J.

2.1.3.3 Static Aspects

2.1.3.3.1 CAMPUS VP

For the export a CAMPUS XML export file is generated and a XSLT transformation is started. It consists of a file called *start.xslt* which includes the files

- *virtualpatientdata.xslt*, which includes *vpd-fragen.xslt* for knowledge questions
- *dataavailabilitymodel.xslt*
- *activitymodel.xslt*
- *lom_metadata.xslt*
- *imsmanifest.xslt*

Those files generate the different files for the MVP package. For those HTML fragments that are not XML conform, the library HTML Cleaner (http://htmlcleaner.sourceforge.net) is used. The already exported media files and the newly generated XML files are stored together with static files (e.g. the schema files) in one ZIP file.

2.1.3.3.2 CAMPUS Key Feature

For the export the CAMPUS KF export module is used. The generated XML file is transformed into the different files using the *export.xslt* (which includes the *vpd-fragen.xslt* sheet). After adding the media files and static files, everything is packed into a ZIP file.

2.1.3.4 Dynamic Aspects

2.1.3.4.1 CAMPUS VP

The export is arranged like a normal CAMPUS VP in the Classic- or Card-Player (DTL = diagnostic therapy loop).

ActivityNode	***DAMNode***	***DAMNodeItem*(s)**
Introduction	Introduction	Reference to *VPDText*
Medical history	Medical history	n references to *InterviewItem*s
Medical history summary	Medical history summary	Reference to a *VPDText*

Physical exam	Physical exam	n references to *PhysicalExam*
Physical exam summary	Physical exam summary	Reference to *VPDText*
Suspicious diagnosis	Suspicious diagnosis	n reference to *Diagnosis*
Start DTL x		
Treatment DTL x	Treatment DTL x	
Physical exam	Physical exam	n references to *PhysicalExam*
Diagnostic test	Diagnostic test	n references to *DiagnosticTest*
Technical exam	Technical exam	n references to *PhysicalExam*
Working diagnosis	Working diagnosis	n references to *Diagnosis*
Check-up	Technical exam, diagnostic test and/or physical exam	n references to *PhysicalExam* or *DiagnosticTest*
End DTL x		
Prognosis	Prognosis	Reference to *VPDText*
Epicrisis	Epicrisis	Reference to *VPDText*
Can be present anywhere:		
Knowledge question	Knowledge question	Reference to *VPDText*
*	*	Additional reference to media file

Tab. 3: CAMPUS-model from ActivityNodes down to VPD eleements

2.1.3.4.2 CAMPUS Key Feature

Every section (= slide) is transformed to one *VPDText*, one *DAMNode* and one *ActivityNode*. The *ActivityNodes* are just sequentially arranged then.

2.1.4 Import Module

2.1.4.1 Functions Related to Import of VPs

2.1.4.1.1 CAMPUS VP

To import a CAMPUS VP eViP export file click on *System* → *Import case*:

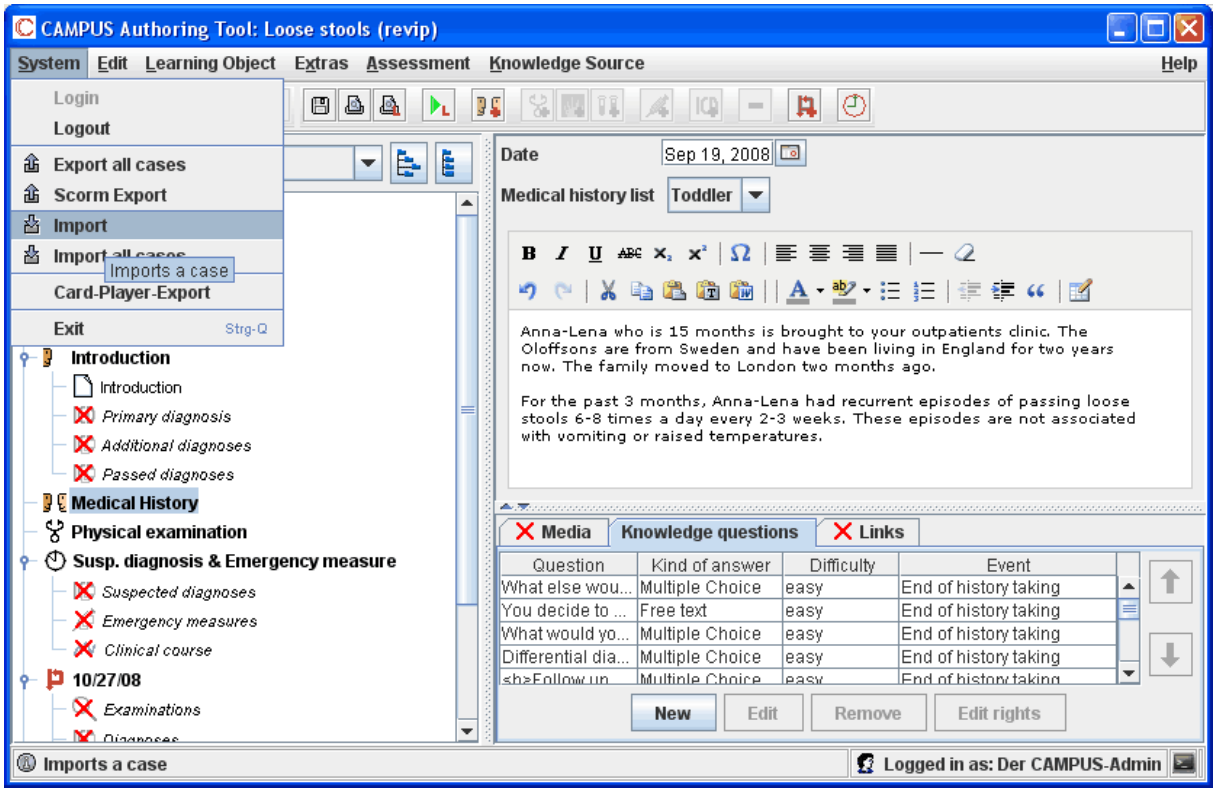


Fig. 9 Importing a case into the CAMPUS VP system – step 1

Select the right import filter and choose the file:

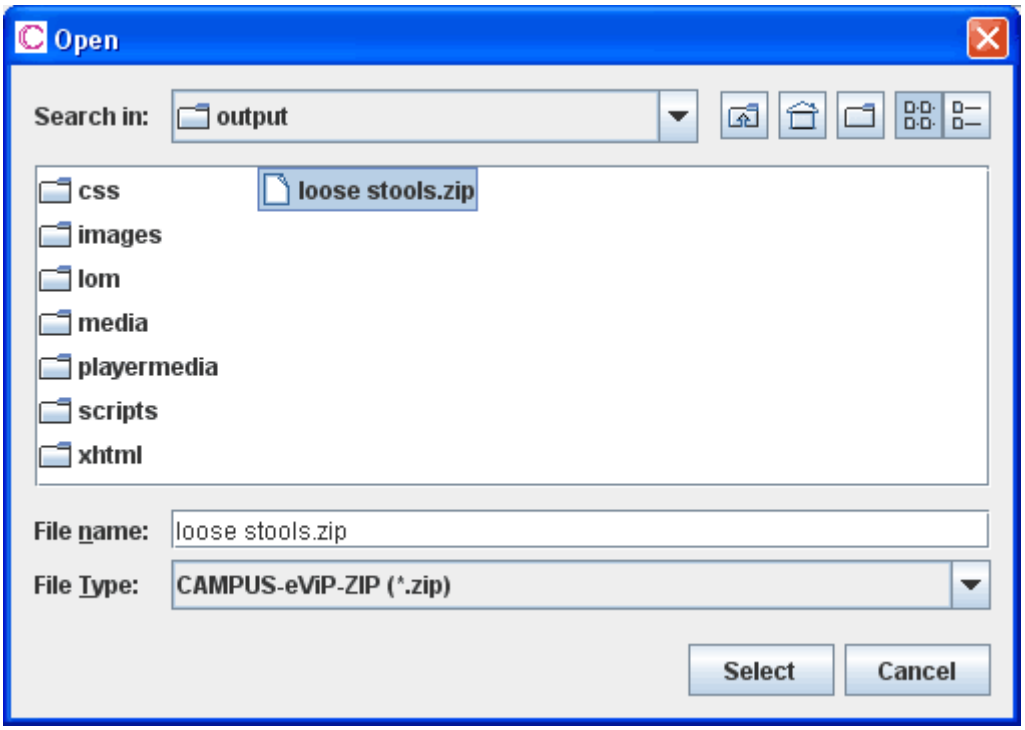


Fig. 10 Importing a case into the CAMPUS VP system – step 2

2.1.4.1.2 CAMPUS Key Feature

To import an eViP file as a key feature case click on *System* → *Import case*:

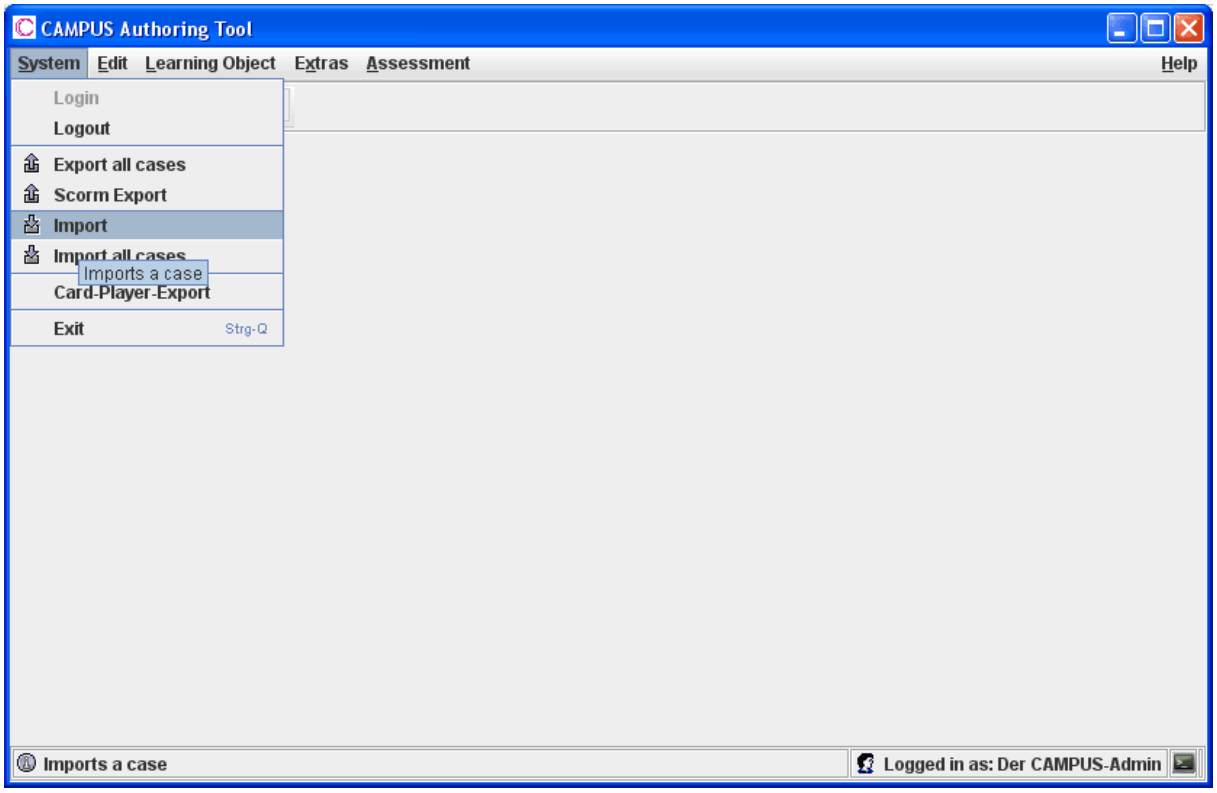


Fig. 11 Importing a case into the CAMPUS Key Feature system – step 1

Select the right import filter and choose the file:

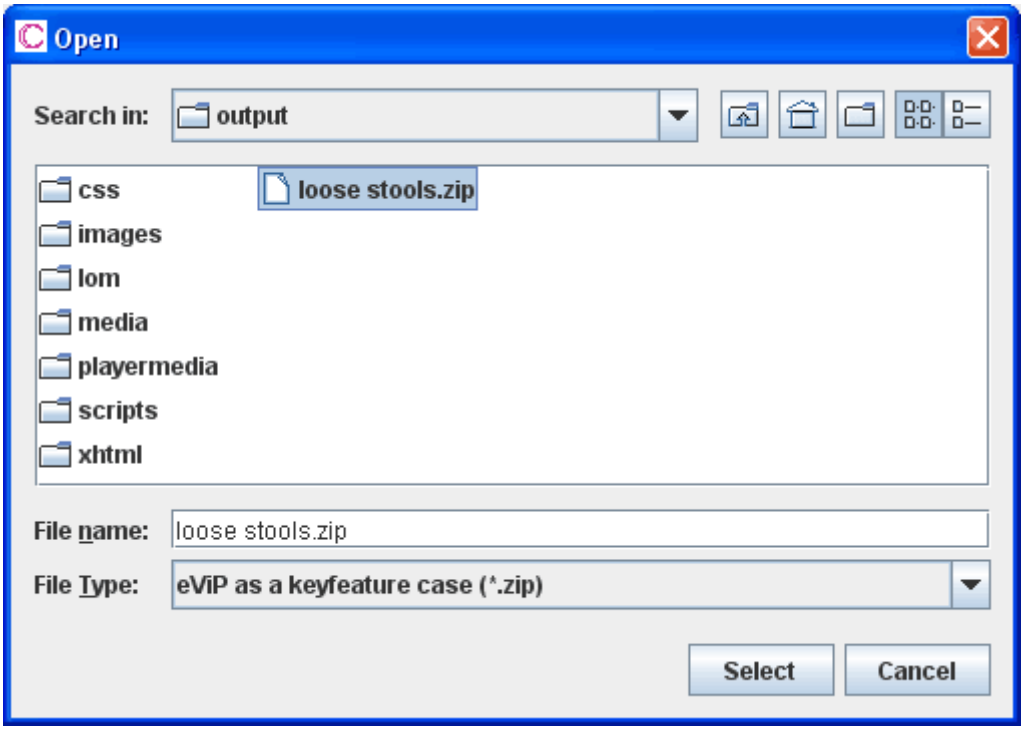


Fig. 12 Importing a case into the CAMPUS Key Feature system – step 2

You can now open the newly imported key feature case and look through the slides:

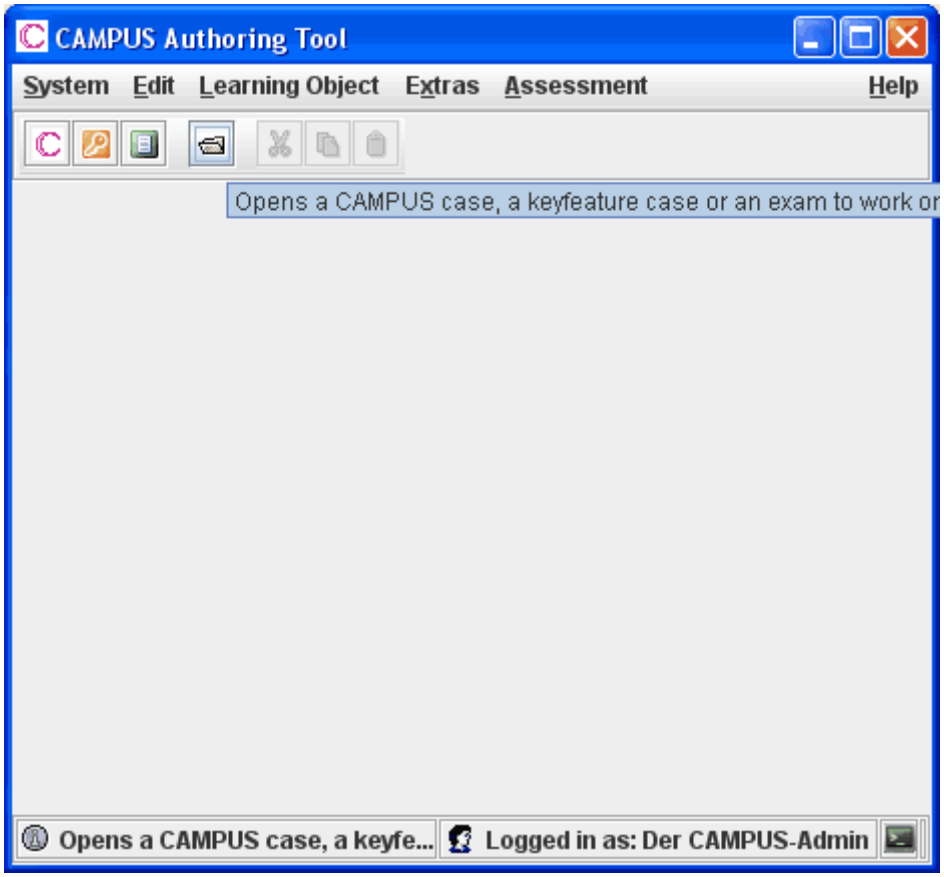


Fig. 13 Browsing through the newly imported case in CAMPUS Key Feature – step 1

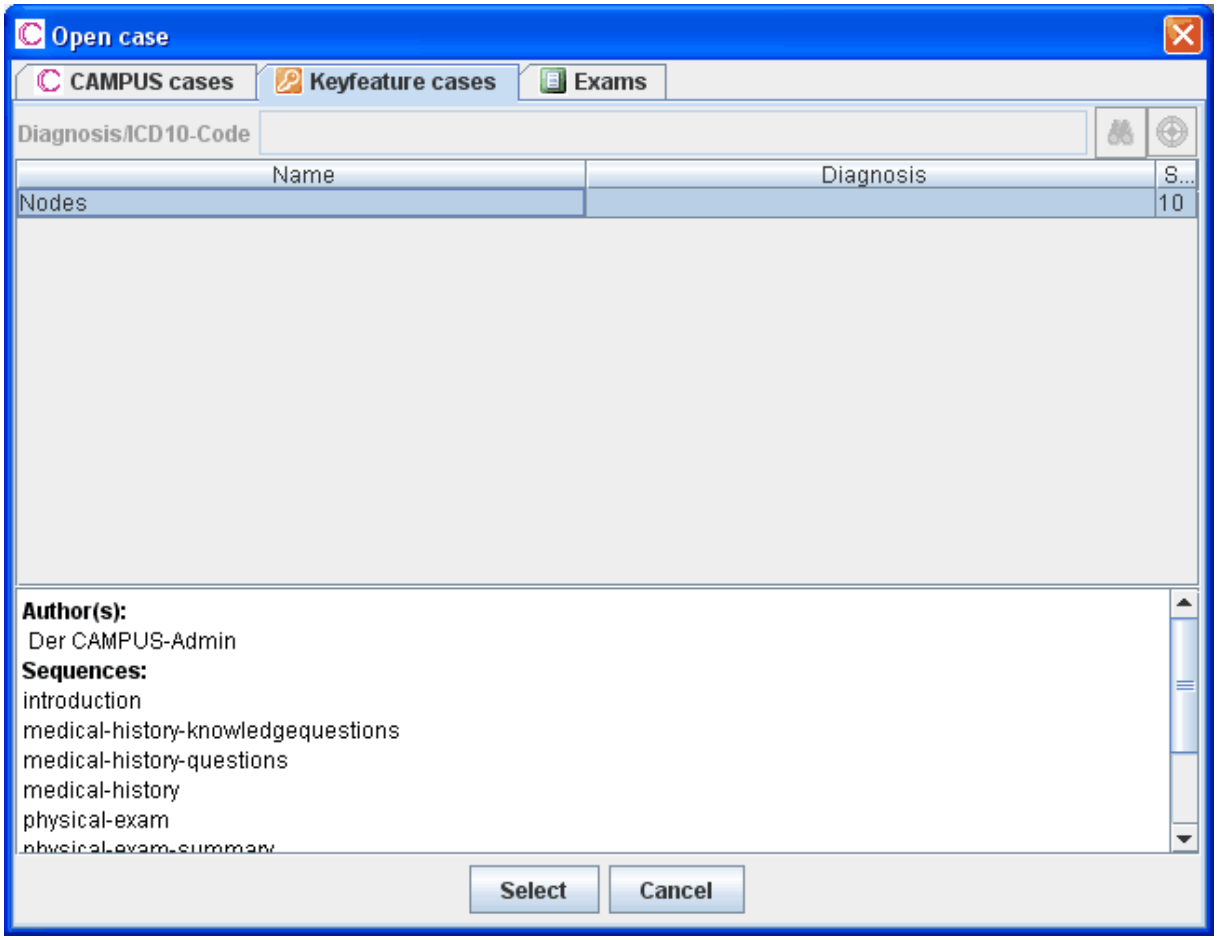


Fig. 14 Browsing through the newly imported case in CAMPUS Key Feature - step 2

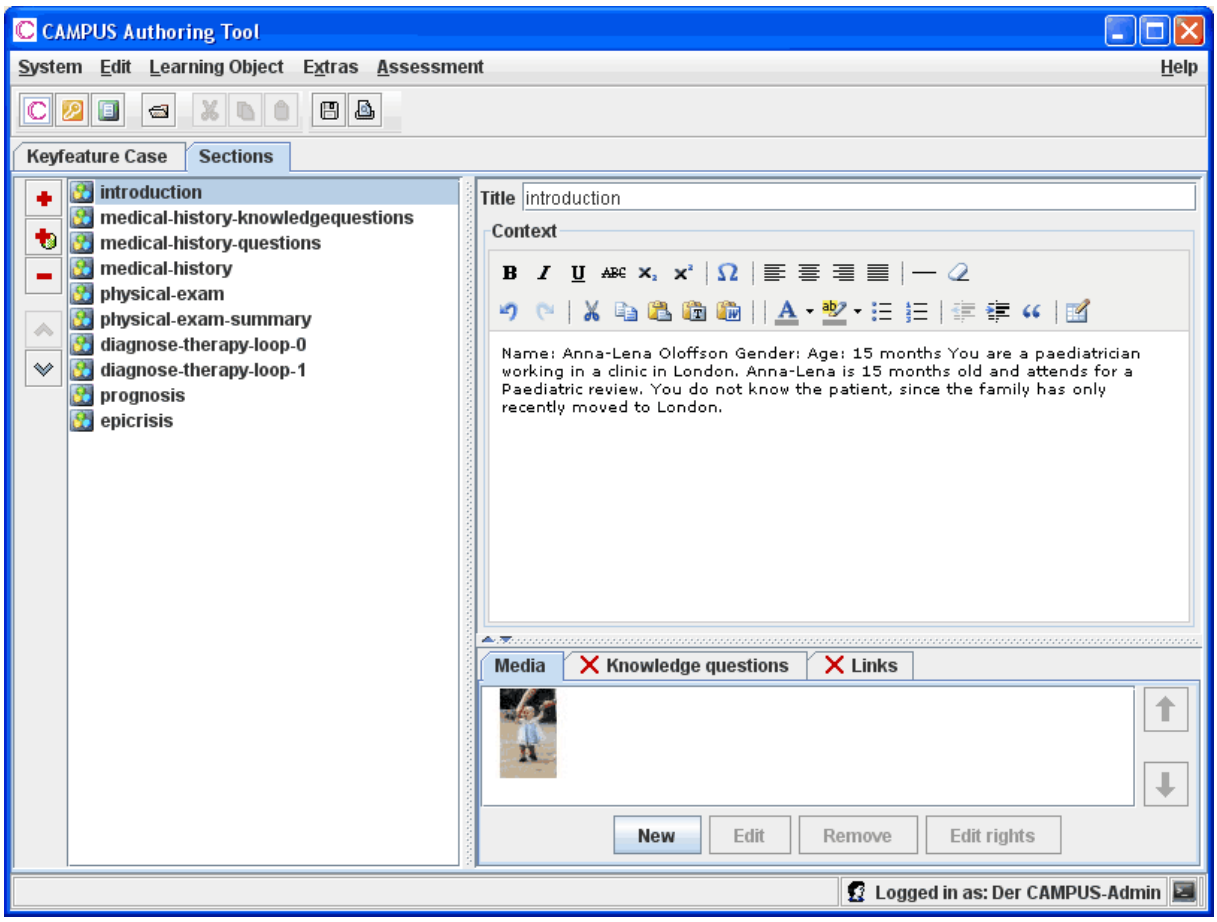


Fig. 15 Browsing through the newly imported case in CAMPUS Key Feature - step 3

2.1.4.2 Development of the Import Module

2.1.4.2.1 CAMPUS VP

For inner CAMPUS VP import the *virtualpatient.xml* file is used because it contains the whole CAMPUS VP import format in the *XtensibleInfo* section. This section is interpreted and can be easily imported.

2.1.4.2.2 CAMPUS Key Feature

The eViP KF import translates the eViP XML files to CAMPUS KF XML import file using a XSLT style sheet. It takes every ActivityNode and looks for referencing VP elements. Those elements are put together in one big XHTML per Node while some VP elements are rendered into XHTML structures (e.g. *InterviewItem*s are rendered with one one *div* for the question and one *div* for the answer).

2.1.4.3 Static Aspects

2.1.4.3.1 CAMPUS VP

For the import the already existing import module could be used for CAMPUS VPs only. The original XML file is base 64 decoded and unzipped. After that it is imported together with the media files.

2.1.4.3.2 CAMPUS Key Feature

For the import the XSLT file *evip_to_keyfeature.xslt* is used. The style sheets loads the different MVP files and generates a *keyfeature.xml* file for the normal CAMPUS KF import module.

2.1.4.4 Dynamic aspects

2.1.4.4.1 CAMPUS VP

As the import is just possible for CAMPUS VP eViP exports and reacts just like the normal CAMPUS VP import no additional development took place.

2.1.4.4.2 CAMPUS Key Feature

For the import each activity node is sequentially used:

- Get the *ActivityNode*
- Get all the referenced *DAMNodeItem*s
- Get all the referenced media files and VP elements
- Serialise all VP elements to XHTML, probably with special templates
- Import the serialised XHTML and media data in one slide (key feature section)

2.2 CASUS

2.2.1 Description of the system and the VP model used by the system

The player is one of the 4 modules (authoring, player, coursemanagement+evaluation, administration) of Casus. The player is card-based and linear (although semi-linear is possible depending on the setting).

The educational approach of Casus VPs is based on hypothetical-deductive diagnosing. Interactive elements like different assessment items support this approach by encouraging the learner to take an active part. Casus VPs are mostly used in a self-directed learning scenario or as a learning by teaching tool in which the learner create the VPs themselves.

On the server-side the player (and all other modules) are implemented as Java servlets using Hibernate (www.hibnerate.org) and an Oracle database. The client side is implemented with HTML-templates using some Javascript and DHTML. All CASUS components works on all main browsers like Internet Explorer, Firefox or Safari.

A case in Casus contains of about 5 (short case) up to 30 cards (long case). A case is structured in chapters with (optional) subchapters and then the cards. Each card can contain several items (which can be combined as needed):

- text (all text elements can include hyperlinks to external or internal resources)
- multimedia material (images with optional hotspots or a movie)
- question with the following

- question types (MCQ, 2 types of Freetext, Long Menu, Sorting, Mapping, Differential diagnostic network, Underline, Lab values, Cloze)
- answercomment, which is availabe (including a quantitative feedback) after the user has given an answer and clicked on solution.
- expert comment, which is available via a button in the navigation bar

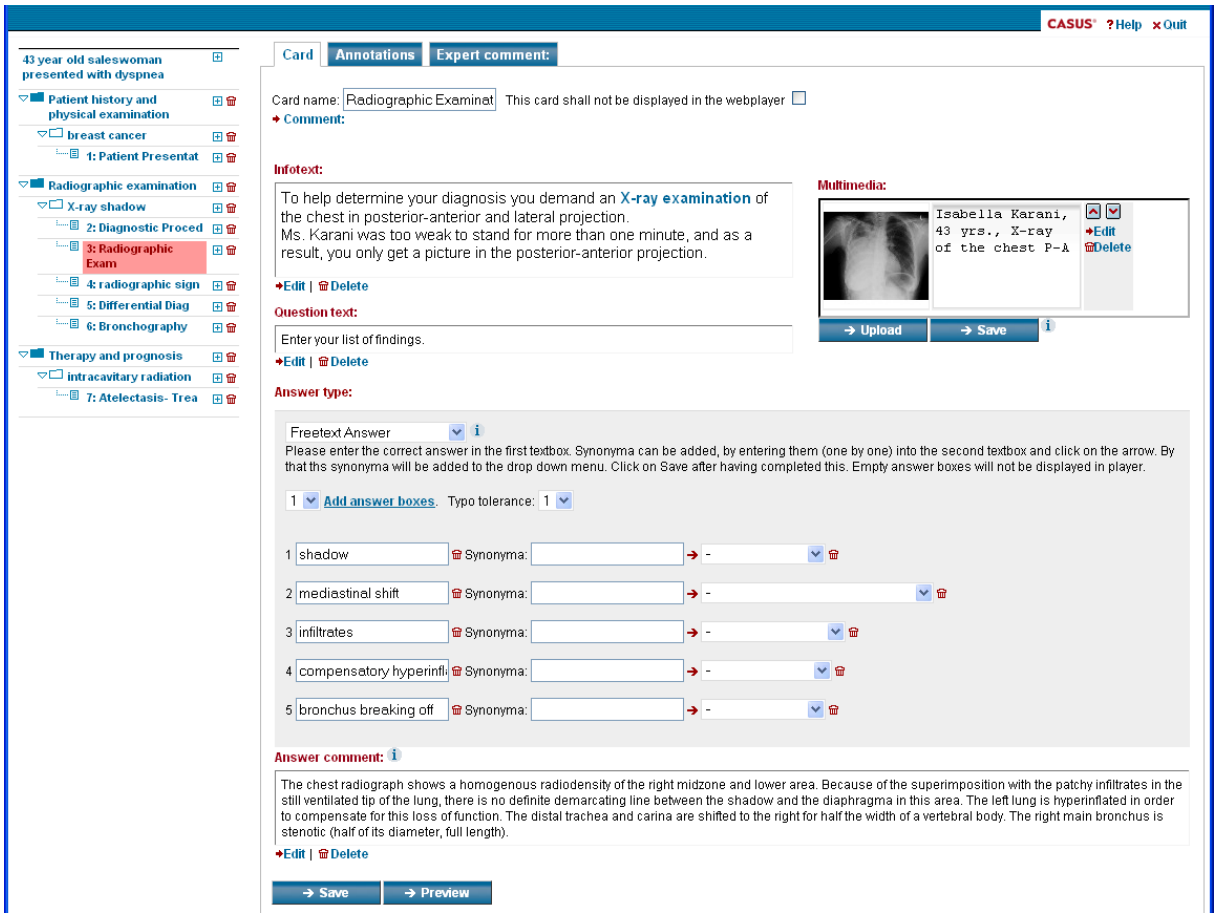


Fig. 16 Screenshot of a case (authoring view) showing the elements of a case

Example cases of evip in Casus are available http://evip.casus.net (click on demo course)

2.2.2 Mapping of the VP xml MedBiquitous standard to the different elements/modules of the player

The mapping of the Casus model classes (Java Beans) to the MedBiquitous model was realised with the castor lib (http://www.castor.org).

	AM	DAM	VPD	Manifest
Chapter	NodeSection	-	-	
Subchapter	NodeSection (conformant since MVP0.5)	-		
Card	ActivityNode	DAMNode	-	
Card-Info	-	DAMNodeItem	VPDText	
Card-Question	-	DAMNodeItem	VPDText	

Card-Expertcomment	-	DAMNodeItem	VPDText	
Card-Answer(s)	-	DAMNodeItem	XtensibleInfo (QTI) or VPDText (without QTI)	
Card-Answercomment	-	DAMNode (referenced through ItemComment of answer	VPDText	
Card-Hyperlinks (internal)	-	DAMNodeItem	VPDText	
Card-Multimedia	-	DAMNodeItem		resource
Card-Multimedia-comment	-	DAMNodeItem	VPDText	

Tab. 4 Mapping of CASUS controls to MVP elements

The 10 different question types available in Casus have been implemented as QTI (2.1) as follows:

Answer Type	QTI type
Multiple Choice (MC)	ChoiceInteraction
Freetext	TextEntryInteraction
Sorting	OrderInteraction
Network	MatchInteraction
Mapping	MatchInteraction
Lab values	MatchInteraction
Long Menu	CustomInteraction (see 1)
Cloze	TextEntryInteraction
Underline	HotTextInteraction
Non evaluated Freetext	ExtendedTextInteraction

Tab. 5 Mapping of CASUS questions to QTI assessment items

An example "case" export with different answer types is presented as the sixth test case scenario in 3.4 . This example contains the QTI implementation of MC, freetext, non evaluated freetext, sorting, network, lab values and underline. The original "case" can be accessed at http://evip.casus.net (login=password=evip) [12].

2.2.3 Export module

2.2.3.1.1 Functions related to export of VPs

The import and export of cases can currently be accessed by Casus course administrators via the main menu page in Casus (see attached screenshot). For export the id of the case has to be entered. In case an error/exception occurs during the export, the user will be given a detailed error report.

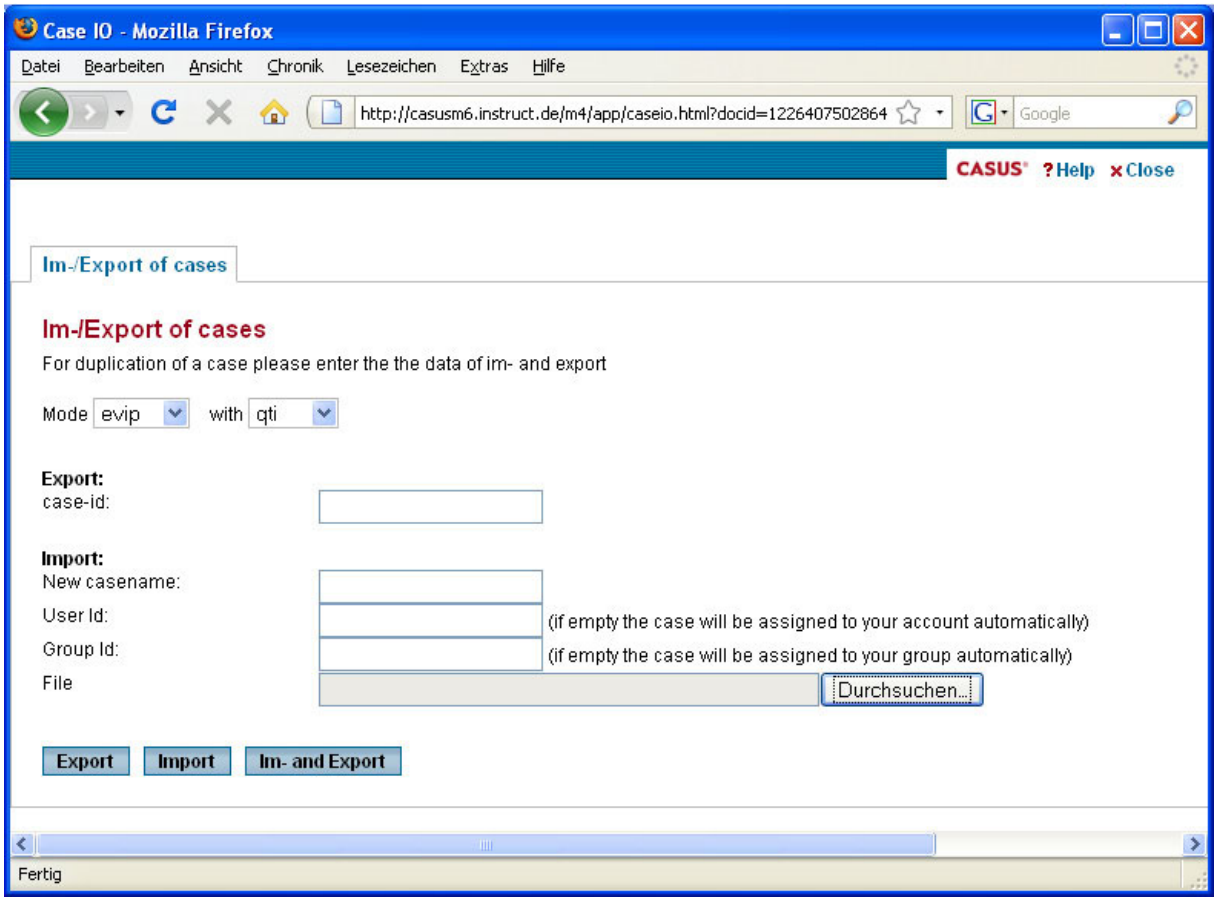


Fig. 17 Import/Export window in CASUS

A case can be exported in two different ways:

- Using QTI for modelling the answers of this case. This allows systems supporting QTI to import the answers automatically.
- Modelling the answers as VPDText elements. The importing system has to convert the answer into a format it can handle.

2.2.3.1.2 Development of the export module

All elements of the MedBiquitous profile are implemented as Java classes and castor mapping files. When exporting a Casus case is loaded from the database and the Casus case model is converted into the MedBiquitous model. This model is marshalled and the created xml files are zipped.

2.2.3.1.3 Static aspects (e.g. class diagrams)

The following class diagram displays the modelling of the MedBiquitous standard (For clarity reasons we did neither include any classes modelling a Casus case nor operations performed by the classes.

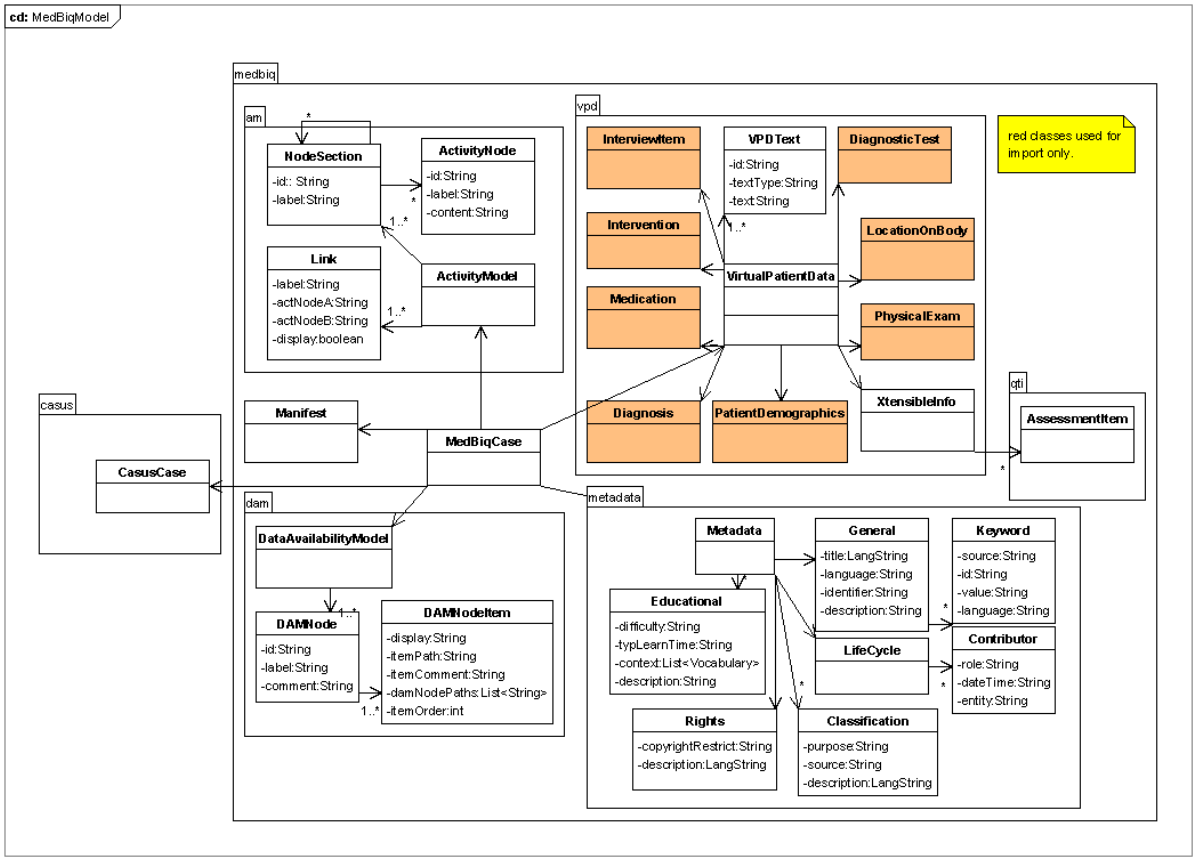


Fig. 18 UML class diagram of the export module in CASUS

2.2.3.1.4 Dynamic aspects (e.g. activity/state diagrams)

For exporting a Casus case the following steps are necessary:

- Creating the MedBiqModel (described under 2.2.3.1.3))
- Load the case from the database
- export the **metadata** and create the **manifest**
- create NodeSections for Chapters and Subchapters (ActivityModel)
- create the **VirtualPatientdata** elements (VPDText and QTI in XtensibleInfo for answers)
- create a DAMNode for each card and DAMNodeItems for each item created in the VirtualPatientData (=each item on a card) (**DataAvilabilityModel**)
- if a card contains multimedia, this resource is added to the manifest and the media file copied to the export folder
- Creating an ActivityNode for each card (**ActivityModel**)
- Loading the Castor mapping files and marshalling each component of the MedBiqCase

The following activity diagram displays the (simplified) process of exporting a Casus case:

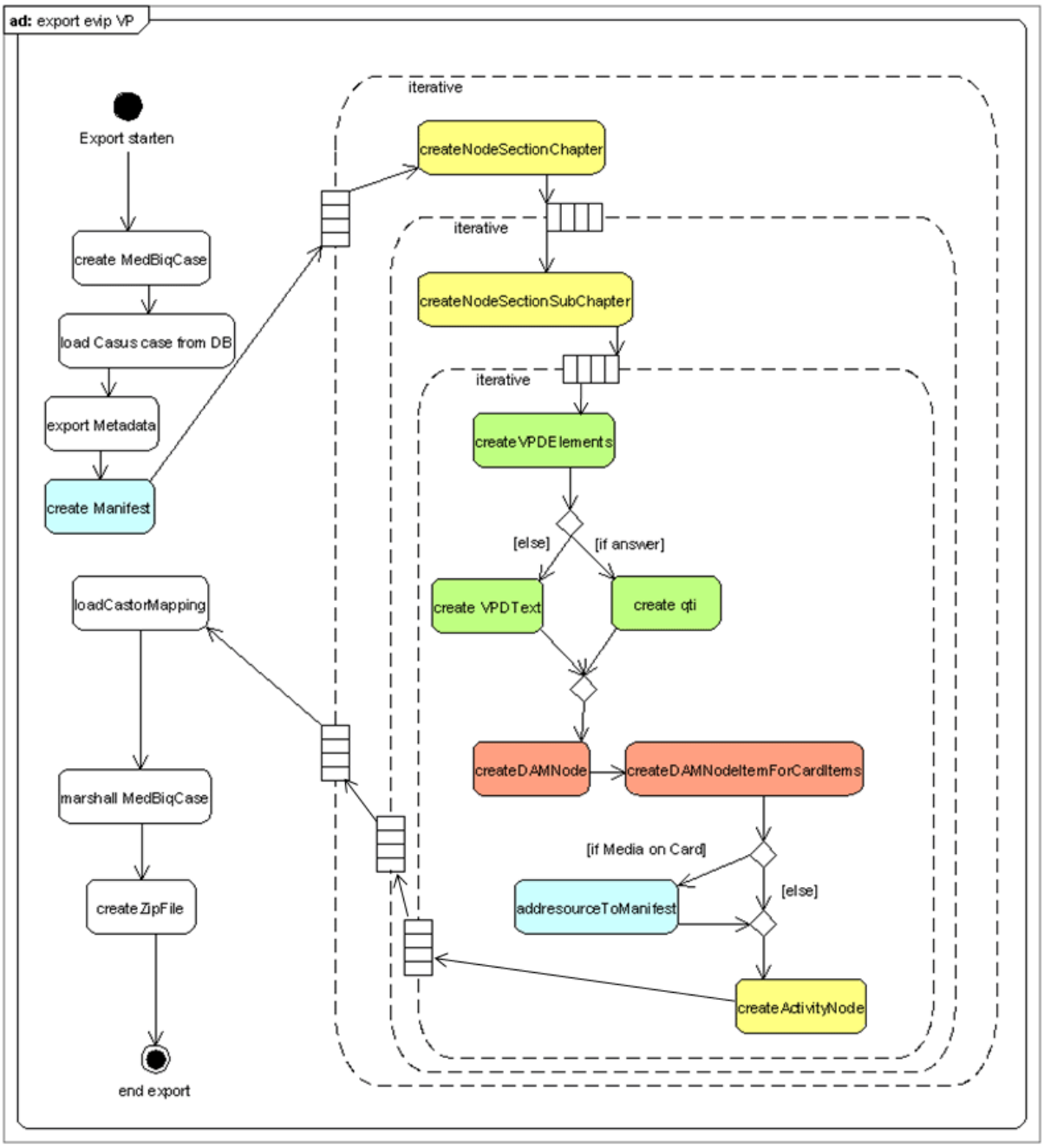


Fig. 19 UML activity diagram of the export function in CASUS (yellow: ActivityModel, blue: Manifest, green: VirtualPatientData, red: DataAvailabilityModel)

2.2.4 Import module

2.2.4.1.1 Functions related to import of VPs

see export. For importing a VP, the zip file has to be loaded. Optionally the userid (and groupid) of the author who should be given access to, can be entered. Per default the user triggering the import will be given access to.

2.2.4.1.2 Development of the import module

see export. When importing a evip VP, the MedBiquitous files are unmarshalled into the MedBiquitous model. This model is converted into the Casus case model, which is finally stored in the database.

2.2.4.1.3 Static aspects (e.g. class diagrams)

see export. The model classes used for import and export are the same.

2.2.4.1.4 Dynamic aspects (e.g. activity/state diagrams)

The (simplified) import is structured as follows:

- Loading zip File
- Creating an empty casus case and the empty MedBiqModel
- Loading the Castor mappings
- Unmarshalling all elements (ActivityModel, DataAvailabilityModel, Metadata, VirtualPatientData, Manifest) "fills" the MedBiqModel
- Convert the **Metadata** into Casus format
- Create the chapters from the list of NodeSections (**ActivityModel**)
- Go through the DAMNodes (**DataAvailabilityModel**) and create a card for each DAMNode
- All **VirtualPatientData** elements referenced by the DAMNodeItems of this DAMNode are attached into the info text of the created card. (The Casus system does not support the VirtualPatientData elements, except VPDTexts. Therefore all content of elements (like DiagnosticTest, InterviewItem,...) referenced within one DAMNode, are attached together as the info text of the card representing this DAMNode.) Referenced multimedia resources are included on the card in the Casus format. The current version does not include the XtensibleInfo elements (except qti).
- Get the NodeSection the created card belongs to and add the card to the appropriate chapter created for this NodeSection.
- Save the completed case into the Casus database.

The following activity diagram displays this process:

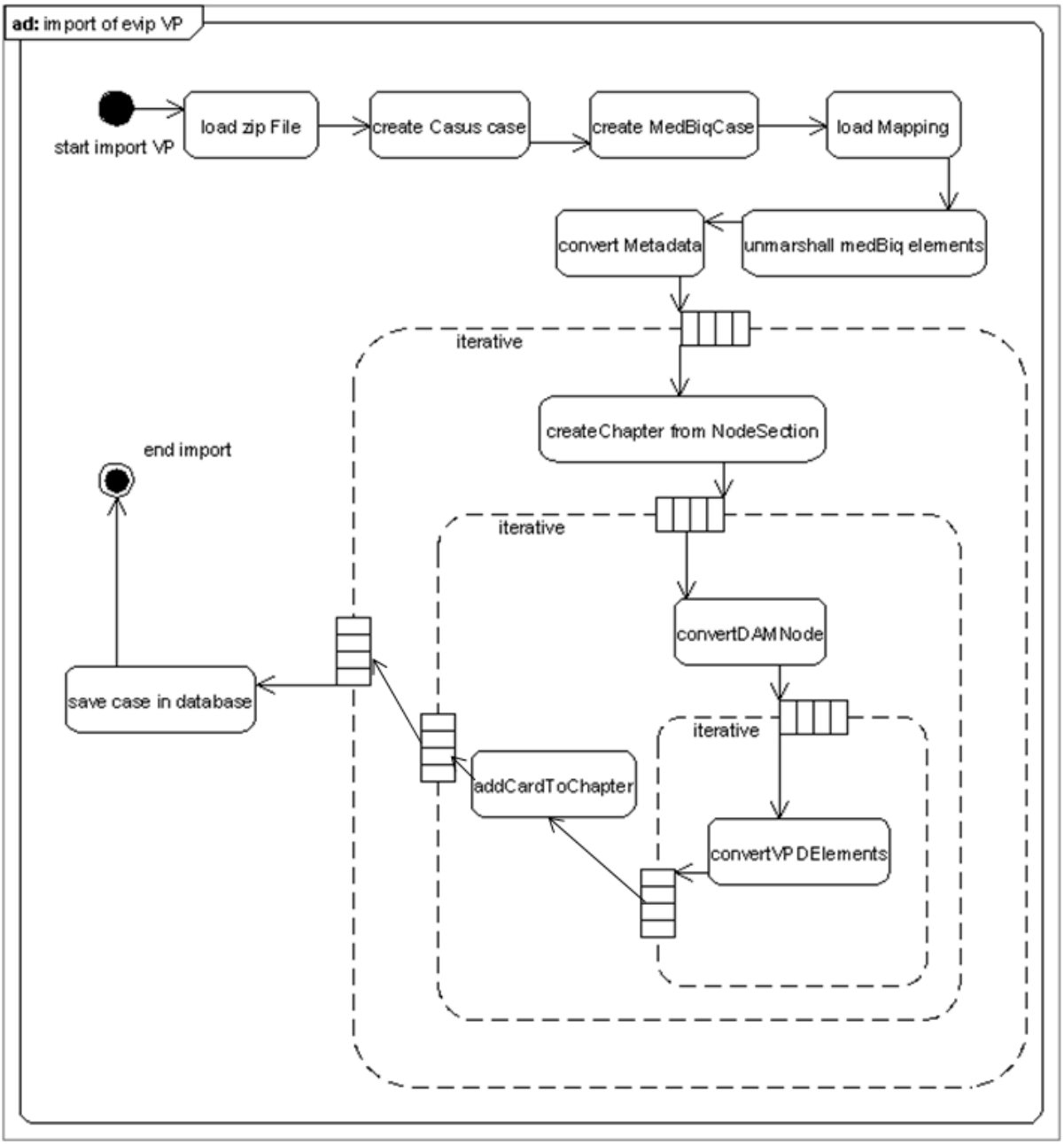


Fig. 20 UML activity diagram of the import function in CASUS (yellow: Activity model, blue: Manifest, green: VirtualPatientData, red: DataAvailabilityModel)

Import of branched VPs:

It is difficult to import a branched VP into a linear one without loosing information, which might be important, although not included in the main path. Moreover often the main path is not implemented in the branched system. Therefore we decided not to import only one branch, but all nodes of all branches. This allows the author to decide about the main path, which information will be kept or skipped. Our solution to import branched VPs into a linear system is, to implement the activityNodes in a directed graph model. This model enables multiple functionalities:

- Request of the start node
- Request of the end node(s) (list)
- Request of all previous nodes of a node (list or null if start node)

- Request of all next nodes of a node (list or null if end node)

As a first implementation step the list of potential next nodes/cards is displayed on each imported card. The authors then can decide about the order they want to display the cards. Unused nodes/cards can be converted into expert comments or deleted completely.

In addition a graphical display will be implemented in a future version.

The imported VPs of the partner systems (CAMPUS, OpenLabyrinth and Web-SP) are available at http://evip.casus.net (login=password=evip).

2.3 OpenLabyrinth

2.3.1 Description of the system and the VP model used by the system

OpenLabyrinth [28] is a web application for authoring and delivering virtual patient (VP) and other decision path and maze-like activities. It is written in Active Server Pages/Visual Basic Script (ASP/VBScript) and requires an Open Database Connectivity (ODBC) Structured Query Language (SQL) database. SGUL is currently using release v2.06 of OpenLabyrinth. All eViP application profile standardisation developments have been implemented on v2.06 with a view to make all changes easily on future releases of OpenLabyrinth [29].

SGUL uses OpenLabyrinth to create primarily branched VPs as part of its institutional e-learning strategy. OpenLabyrinth also has the ability to create linear VPs. A branched VP is one that offers students different clinical decision making choices at key scenario points where the story unfolds. The branched VP consists of a multitude of choices and key scenario points. The student is free to take whichever path they feel to complete a case. The emphasis is not always on making the correct decisions as it is felt that there is much to be learnt from making poor decisions too. Like many other VP systems, much of OpenLabyrinth's appeal is the originality and quality of the VP case narrative. There is a dependency on the Visual Understanding Environment tool (VUE) to author a VP case narrative. VUE is an Open Source project based at Tufts University in the United States (http://vue.tufts.edu/).

Below is a typical workflow for the creation of VPs from scratch:

1. Clinicians and other subject matter experts author the case, create choices and map branches using VUE. An example is provided in Fig. 21a

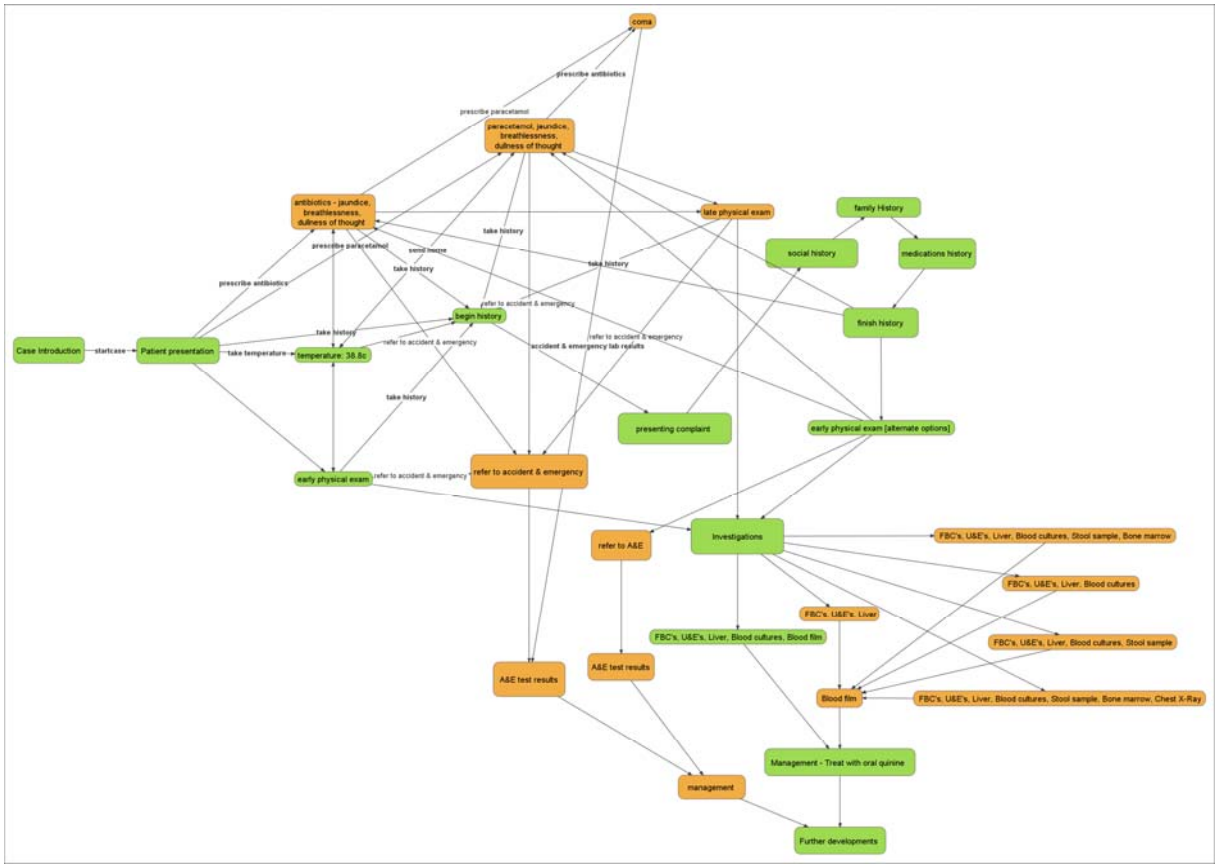


Fig. 21a VUE map for OpenLabyrinth case

2. Clinicians and other subject matter experts finalise the VUE file and this file is then imported into OpenLabyrinth by a learning technologist.

3. The learning technologist works with the clinician and other subject matter experts to content enrich the case with images, x-rays, laboratory reports, videos and animations.

4. All persons involved in the creation of the VP finalise the VP and it is realised to students.

For the purposes of eViP, the above workflow is modified to repurpose and enrich existing VPs as opposed to creating VPs from scratch.

2.3.2 Mapping of the VP xml MedBiquitous standard to the different elements/modules of the system

The Import/Export Module system was developed as a 3-Tier System (UserTier, BusinessTier/MiddleTier, DataTier Combination) and compilation based on **C#.Net Assemblies**. Each of the relevant VP xml and containing attributes, elements and subelements of MedBiquitous standard were designed and implemented as **serializable/deserializable class objects**. The business abstract class (called ***ContractBase*** defines the contract of serialization/deserialization usage of the base class. Each VP xml class, node class, childnode class must inherit from this ContractBase base class to satisfy serialization/deserialization capability of its constituent nodes/childnodes.

2.3.3 OpenLabyrinth and VUE

A distinctive feature of the OpenLabyrinth system is its branched activity model. The special strength of this approach is enabling students to select various medical examinations or

treatment methods and later on to deal with the consequences of their previous choices. Branched navigation can be easily encoded using the activity model elements from the MVP specification. Presenting virtual patient scenarios in multiple pathways, where the mistakes may not become immediately apparent contributes to achieving a higher grade of realism of educational materials.

The branched nature of VPs created for use in OpenLabyrinth is such that there is generally an optimum path (or paths) for navigating through the VP. However, while this is often described visually in the VUE diagram that would have been created when authoring the case, there are no indicators or references to this optimum path held within OpenLabyrinth or its database. As a consequence, cases exported from OpenLabyrinth do not offer any guidance to assist in identifying an optimum path through the VP. When importing an OpenLabyrinth export into another system which does not support a branched activity model it may be necessary to make value judgements about the content of certain nodes and their suitability for inclusion in the imported case. Without access to the information regarding the structure of the branches and the optimum path(s), these judgements will be difficult to make. There is the potential to wrongly exclude valuable information from the import or to include content from contradictory nodes that may break the narrative logic of the case. In order to provide the information to enable these decisions to be made, SGUL intend to make an export of the VUE map available for all the VPs selected for eViP. This export should provide a reference for the original structure of the VP and provide an indication of the optimum path(s) through the case.

2.3.4 OpenLabyrinth and QTI

The branched activity model used by OpenLabyrinth does not necessary go along well with the instant answer-feedback nature of common test items like MCQs, sorting or filling-in-the-gap. Consequentially, OpenLabyrinth does not support adding test items into virtual patients and does not need to export questions to external systems. A similar approach is also taken by the MVP specification which currently does not include elements for encoding questions, referring instead to the external specifications like QTI. If needed, it is recommended to encoded assessment items within the optional XtensibleInfo section. This method is used by VP systems as CASUS or CAMPUS in which question items play an important role. The OpenLabyrinth model is theoretically flexible enough to support the import of some question items from QTI blocks created by other systems. A single-answer MCQ question including feedback, for instance, can be converted into an n+2 node large subgraph, where n is the number of possible choices in the question.

2.3.5 Export module

2.3.5.1 Methods/Functions related to export of VPs

In Fig. 21b the UML use case diagram of the OpenLabyrinth's export function is presented.

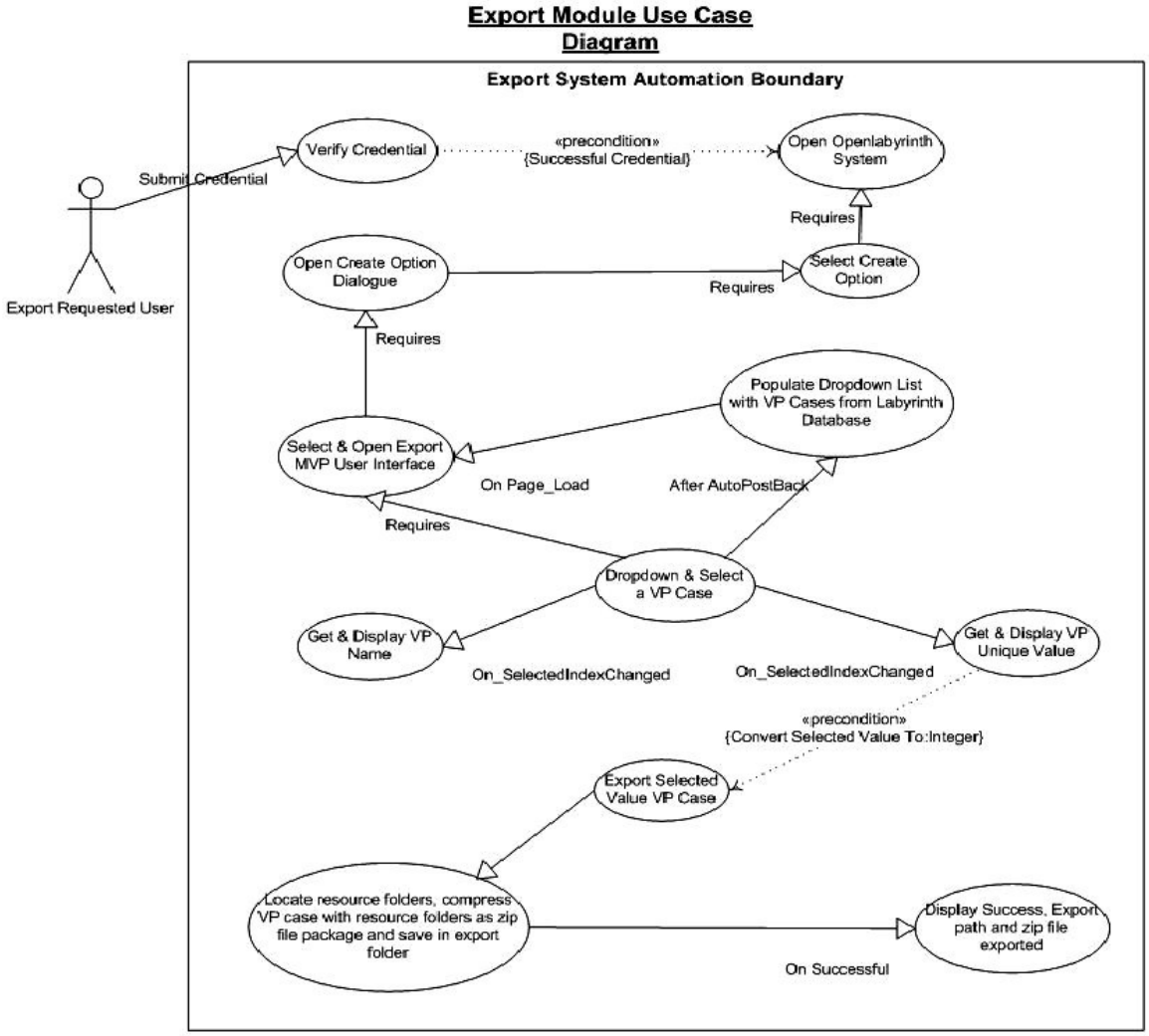


Fig. 21b UML Case diagram for the OpenLabyrinth's export module

A description of how to export an MVP package from OpenLabyrinth system (step by step screenshots with explanations) is presented in the figures below.

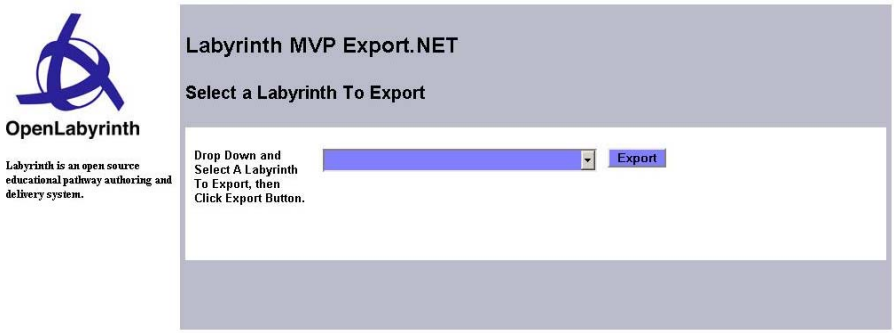


Fig. 22 Step 1: Export Graphical User Interface (GUI) Page is loaded.

Fig. 23 Step 2: Error Message, if user clicks Export button without making a selection.

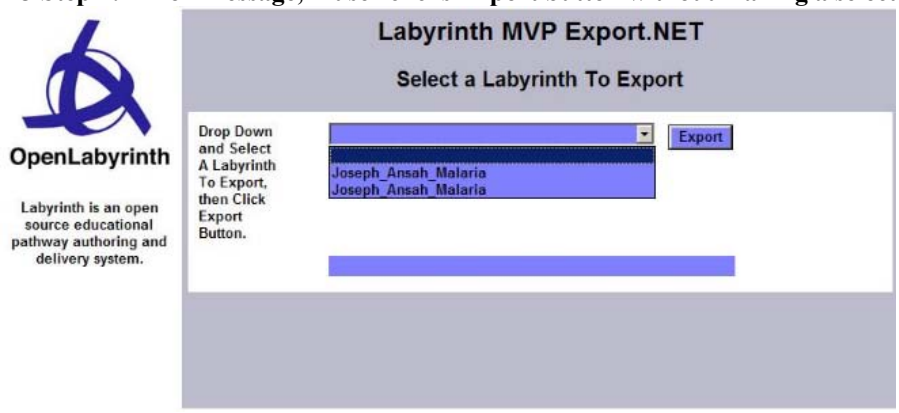


Fig. 24 Step 3: Drop down select button to select a VP case to export.

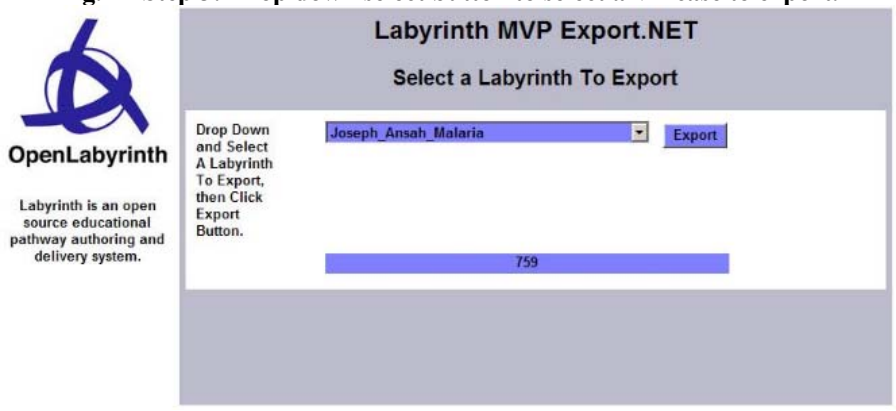


Fig. 25 Step 4: Select a VP case of choice to export from dropdown list. The unique id in OpenLabyrinth is displayed to provide a distinction between multiple instances of the same case.

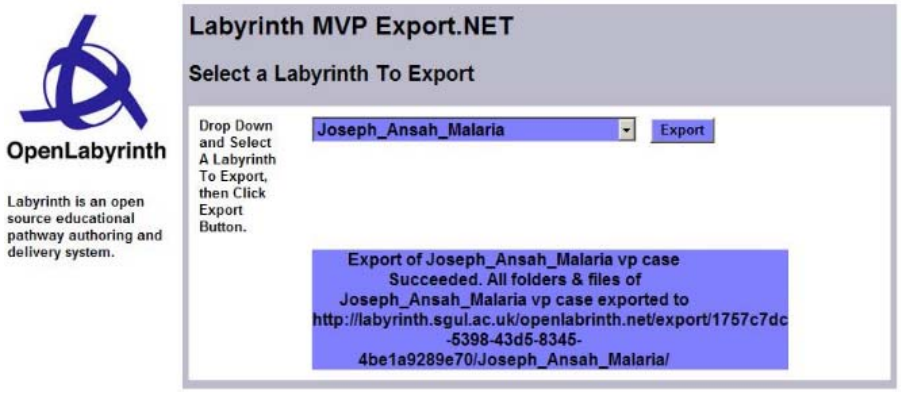


Fig. 26 Step 4: Select a VP case of choice to export from dropdown list. The unique id in OpenLabyrinth is displayed to provide a distinction between multiple instances of the same case.

2.3.5.2 Development of the export module

The development of the export module was carried out in the following stages:

- Restructuring and Normalisation of OpenLabyrinth Database

- Analysis of existing OpenLabyrinth MVP export methods
- Development of MVP Export Prototype
- Refining and finalisation of Export Requirement
- Object Orientated Coding with C#.NET
    - VirtualPatientData XML (All containing elements, subelements with attributes implemented as Serializable/Deserializable Class objects)
    - ImsManifest XML (All containing elements, subelements with attributes implemented as Serializable/Deserializable Class objects)
    - ActivityModel XML (All containing elements, subelements with attributes implemented as Serializable/Deserializable Class objects)
    - DataAvailabilityModel XML (All containing elements, subelements with attributes implemented as Serializable/Deserializable Class objects)
    - Metadata XML (All containing elements, subelements with attributes implemented as Serializable/Deserializable Class objects)
- Assembled MVP test Packages
- Unit and Integrated Testing of Export Module with the Assembled Test Package
- Deployment of Export Module

2.3.6 Import module

2.3.6.1 Methods/Functions related to import of VPs

In the and UML use case diagram of the import function is presented.

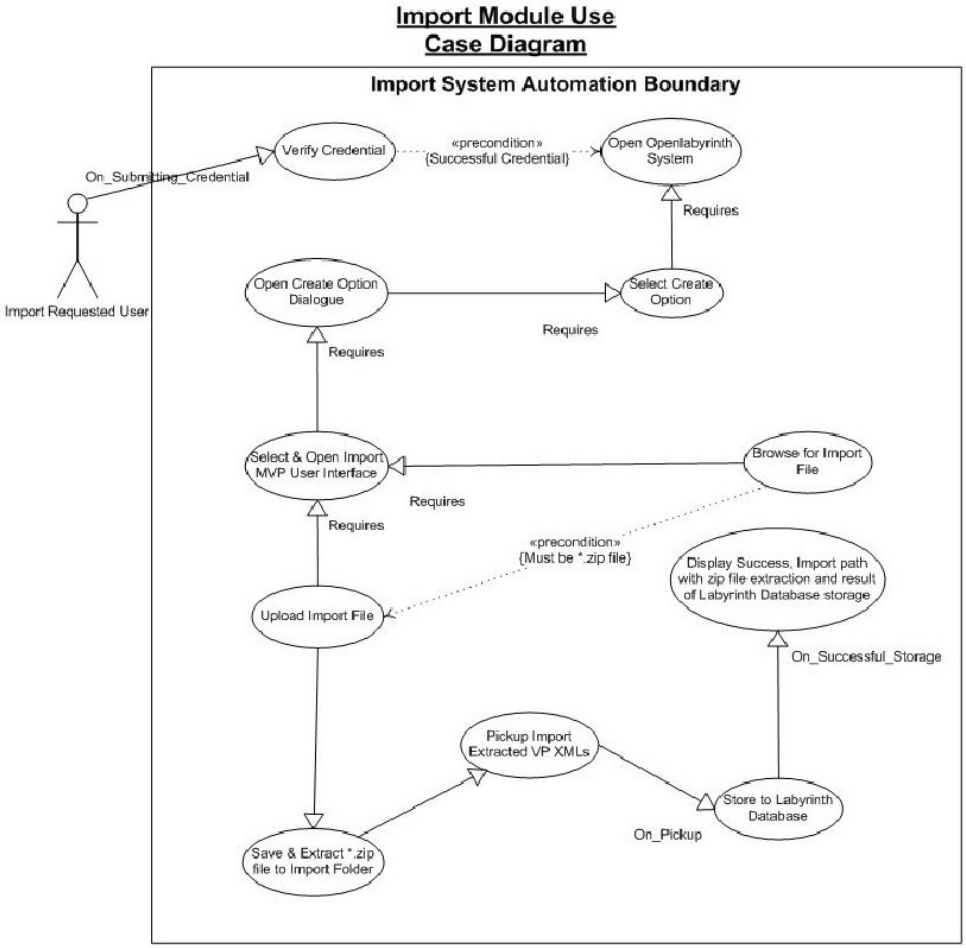


Fig. 27 UML Case diagram for the OpenLabyrinth's import module

A description of how to export an MVP package from OpenLabyrinth system (step by step screenshots with explanations) is presented in the figures below.

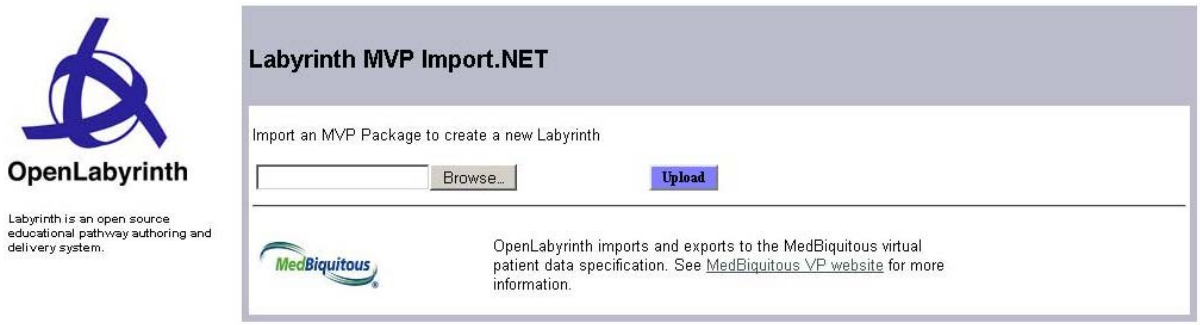


Fig. 28 Step 1 Import Graphical User Interface (GUI) Page is loaded.

Fig. 29 Step 2 Display error message, if the upload button is clicked without first browsing for file to import.

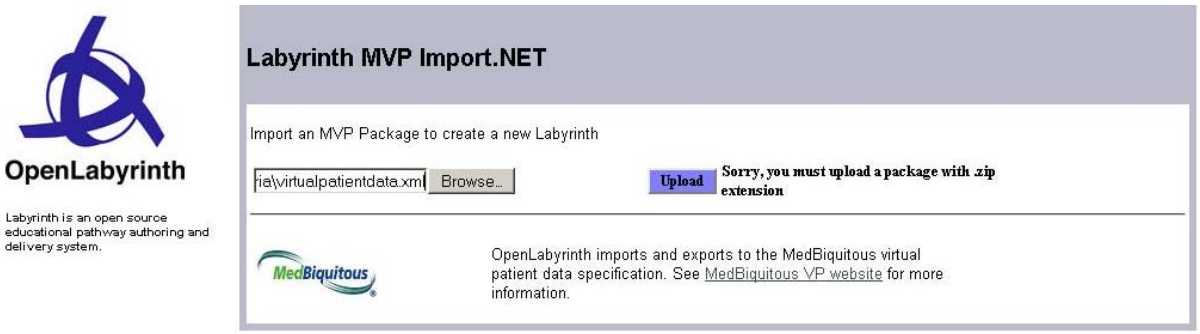


Fig. 30 Step 3 Display error message, if you try to upload a file with no .zip **extension.**

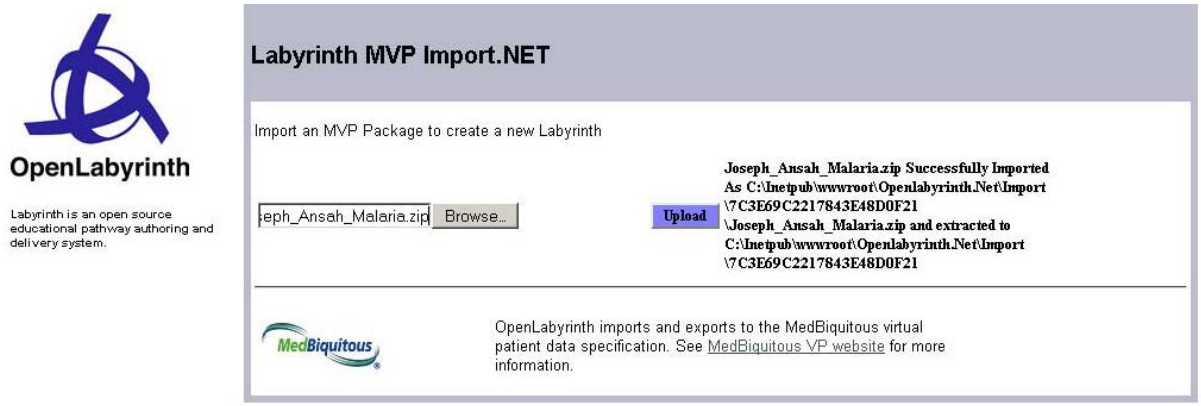


Fig. 31 Step 4 Display message, on successful import of Joseph_Ansah_Malaria.zip case package.

2.3.7 Static aspects (e.g. class diagrams)

The following section contains class diagrams of import/export components of OpenLabyrinth.

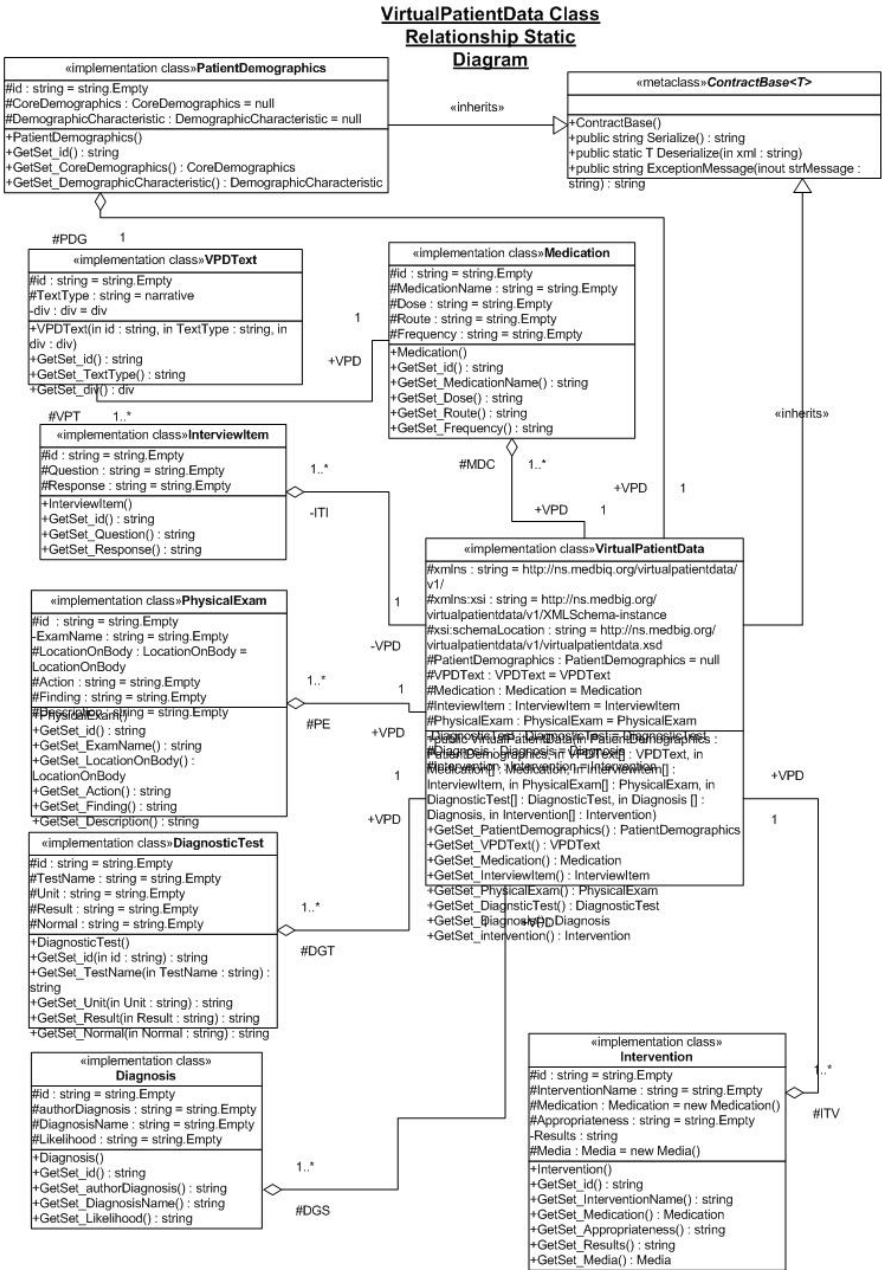


Fig. 32 UML Class Diagram for the MVP VPD model implemented in OpenLabyrinth

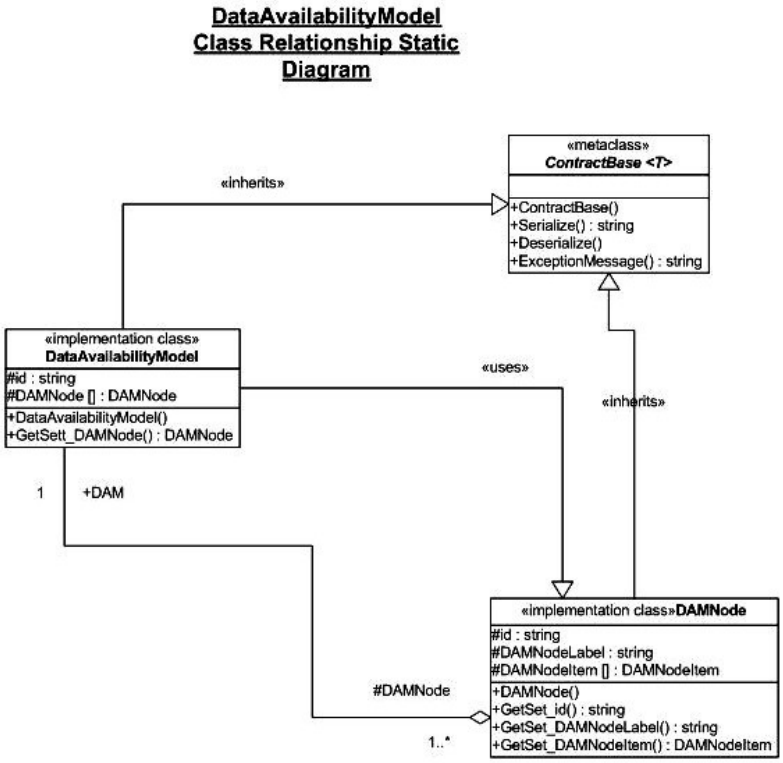


Fig. 33 UML Class Diagram for the MVP DAM model implemented in OpenLabyrinth

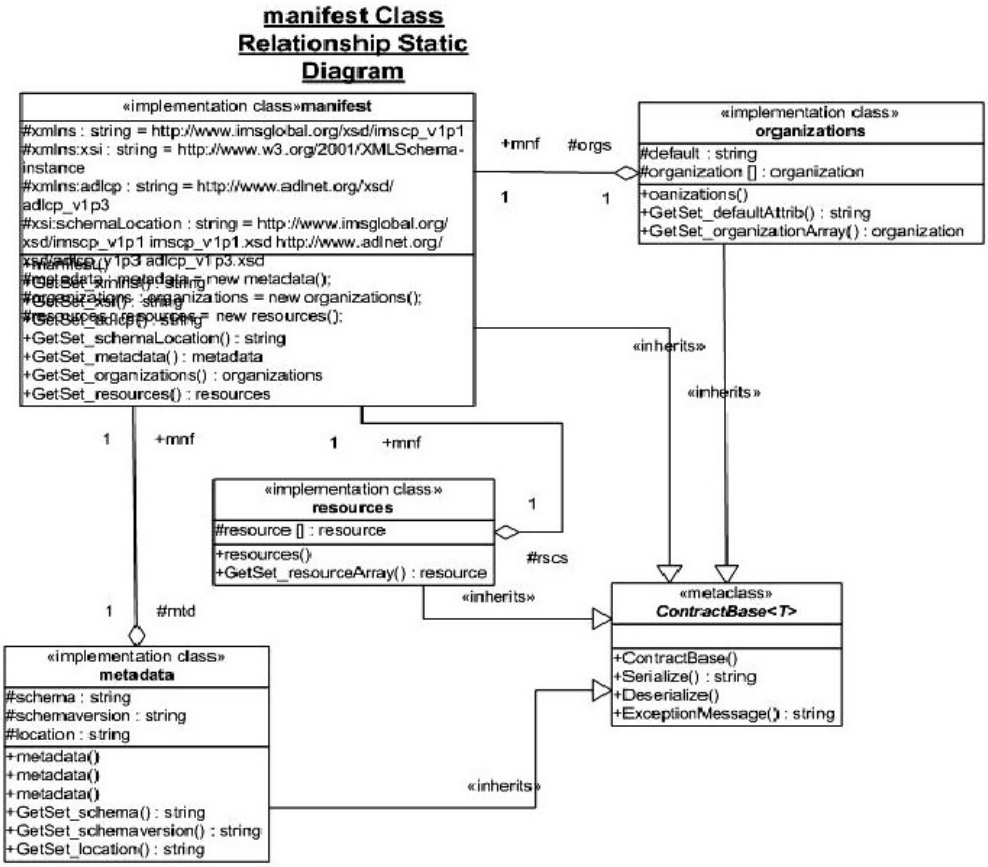


Fig. 34 UML Class Diagram for the IMS manifest implemented in OpenLabyrinth

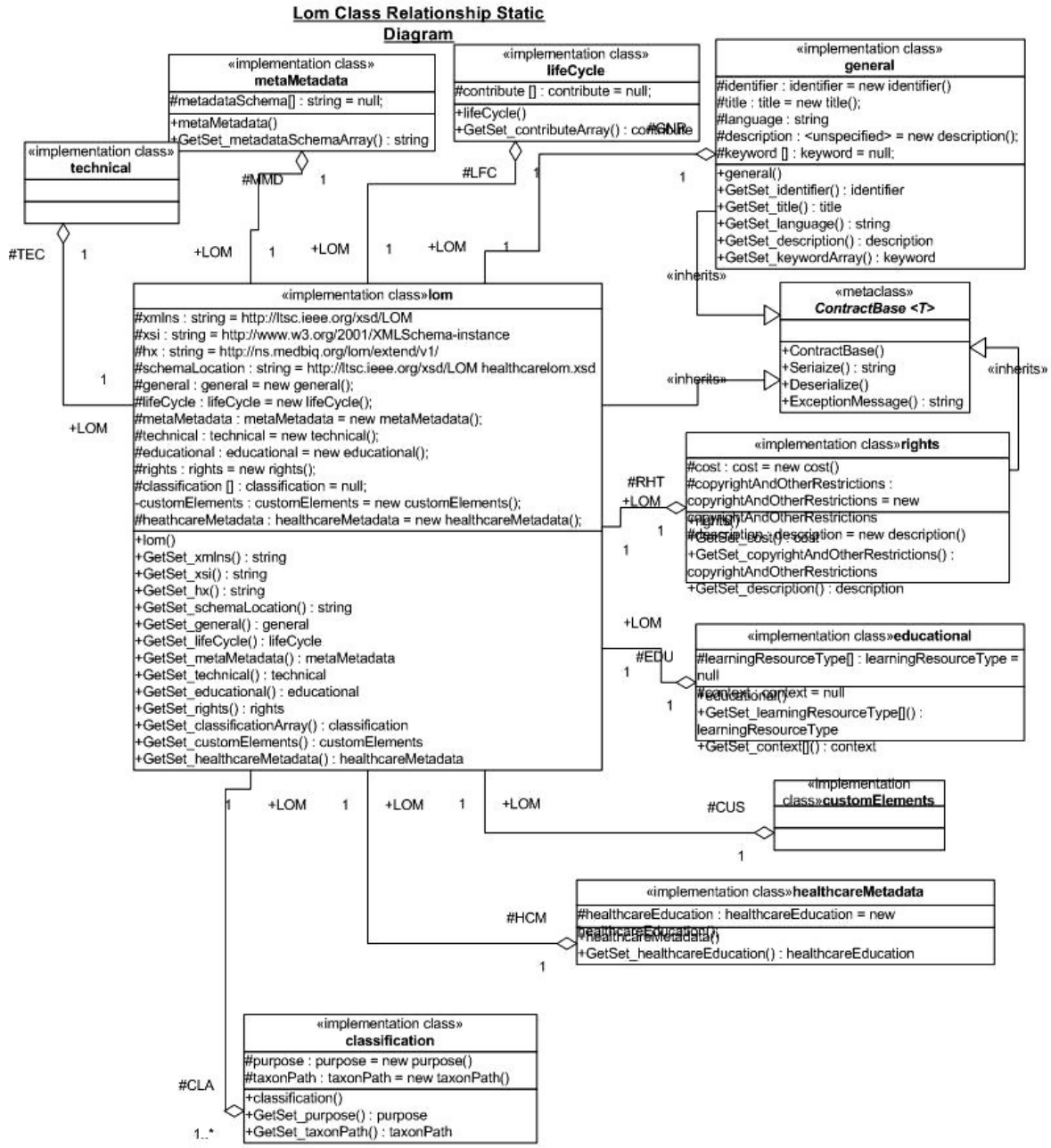


Fig. 35 UML Class Diagram for the IEEE LOM metadata implemented in OpenLabyrinth

2.3.8 Dynamic aspects (e.g. activity/state diagrams)

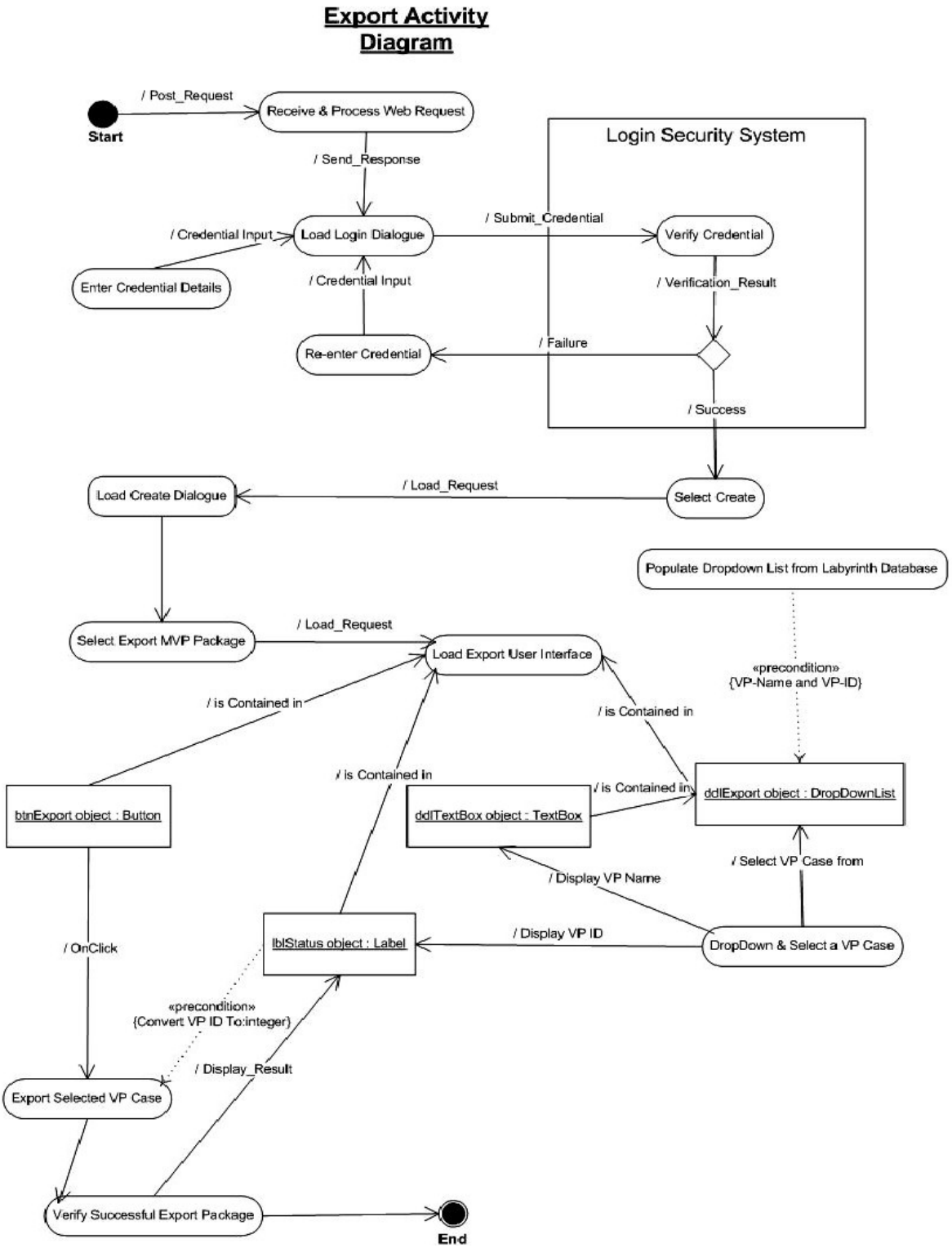


Fig. 36 UML Activity diagram for the export function in OpenLabyrinth

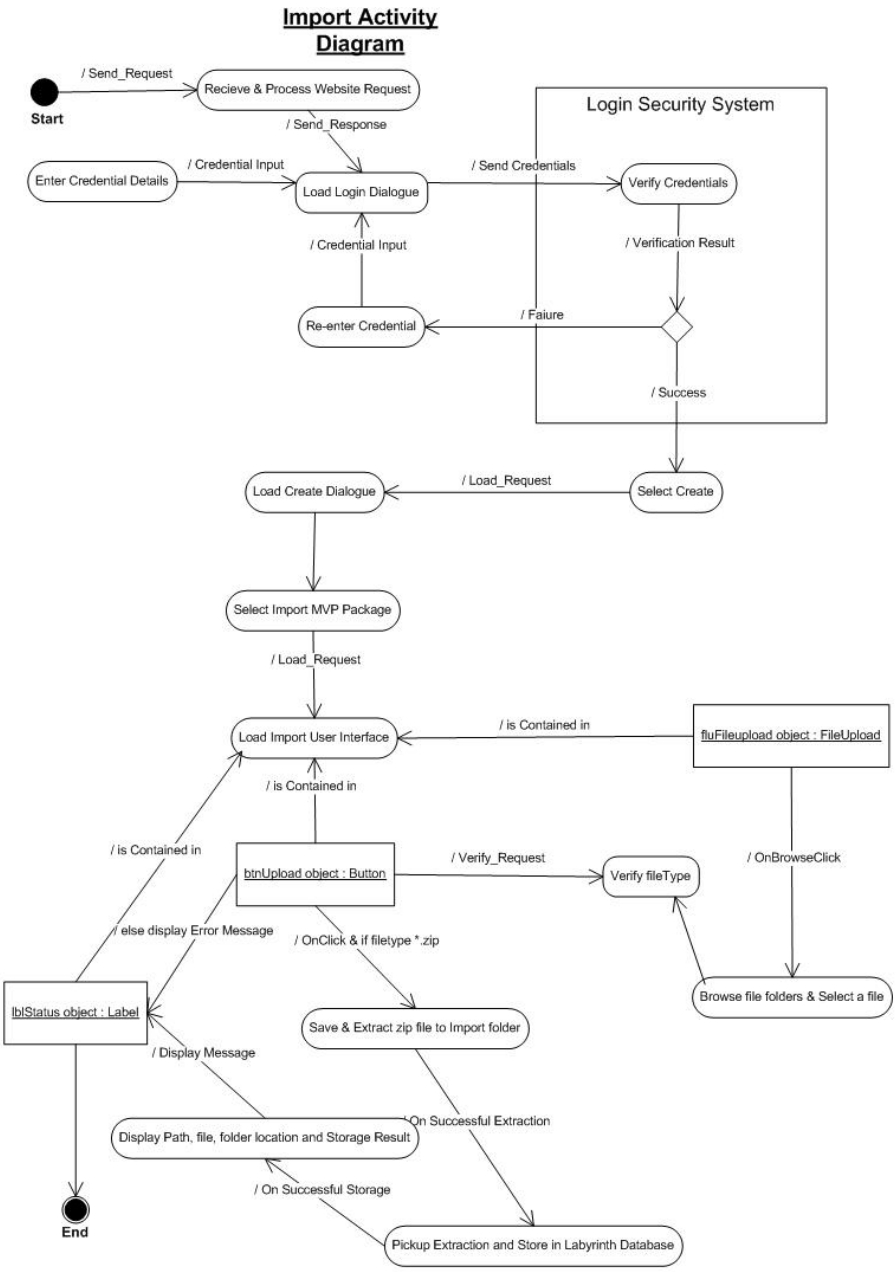


Fig. 37 UML Activity diagram for the import function in OpenLabyrinth

2.4 Web-SP

2.4.1 Description of the system and the VP model used by the system

Web-SP [20] is a web-based Virtual Patient system developed at Karolinska Institute with the aim to easily author, run and manage Virtual Patients (VP) for training and assessment purposes. Web-SP was successfully implemented at several universities world-wide.

Web-SP version 3.2 is a framework composed of VP players and supporting tools (authoring, administration, grading, reporting etc…)

- **VP players**

Two types of players are available. A single patient encounter player (Fig. 38) and multiple patient) encounter player which presents a set of single encounters in an integrated manner.

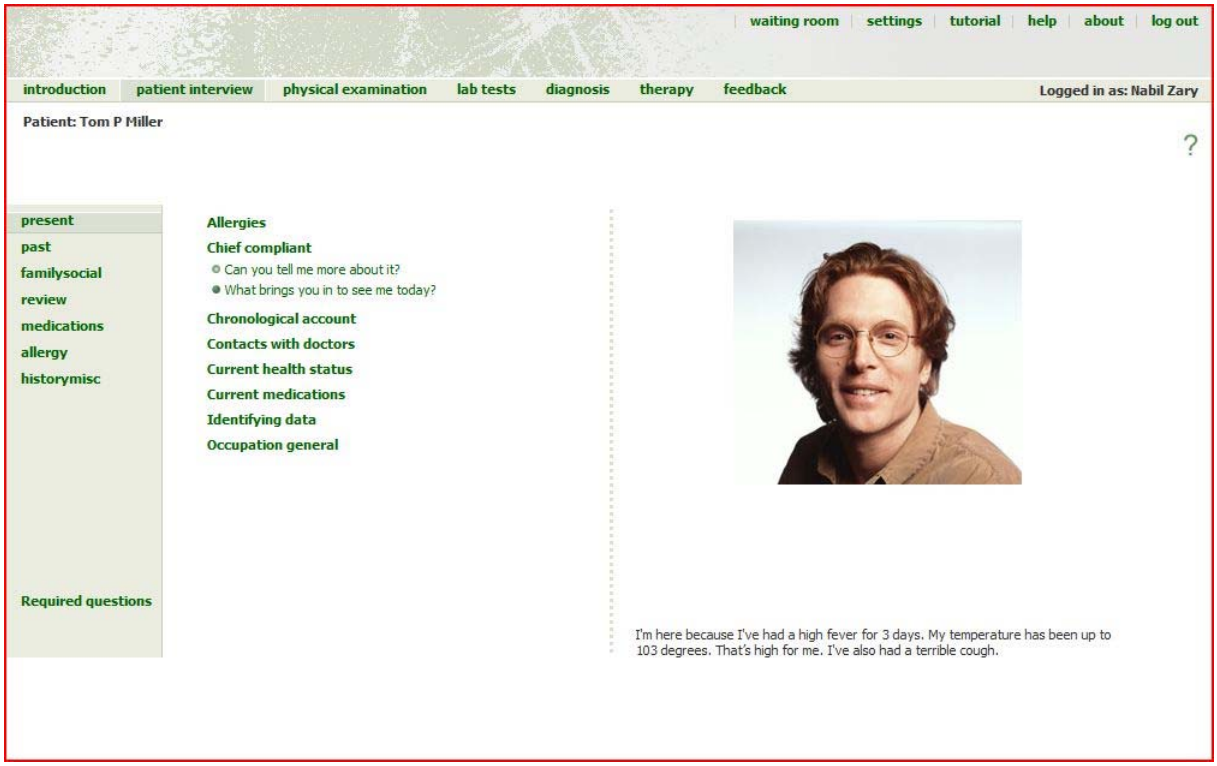


Fig. 38 The patient interview module of Web-SP 3.2

Eight modules are available in version 3.2 (introduction, patient interview, examination, investigation, diagnosis, patient management, feedback, assessment)

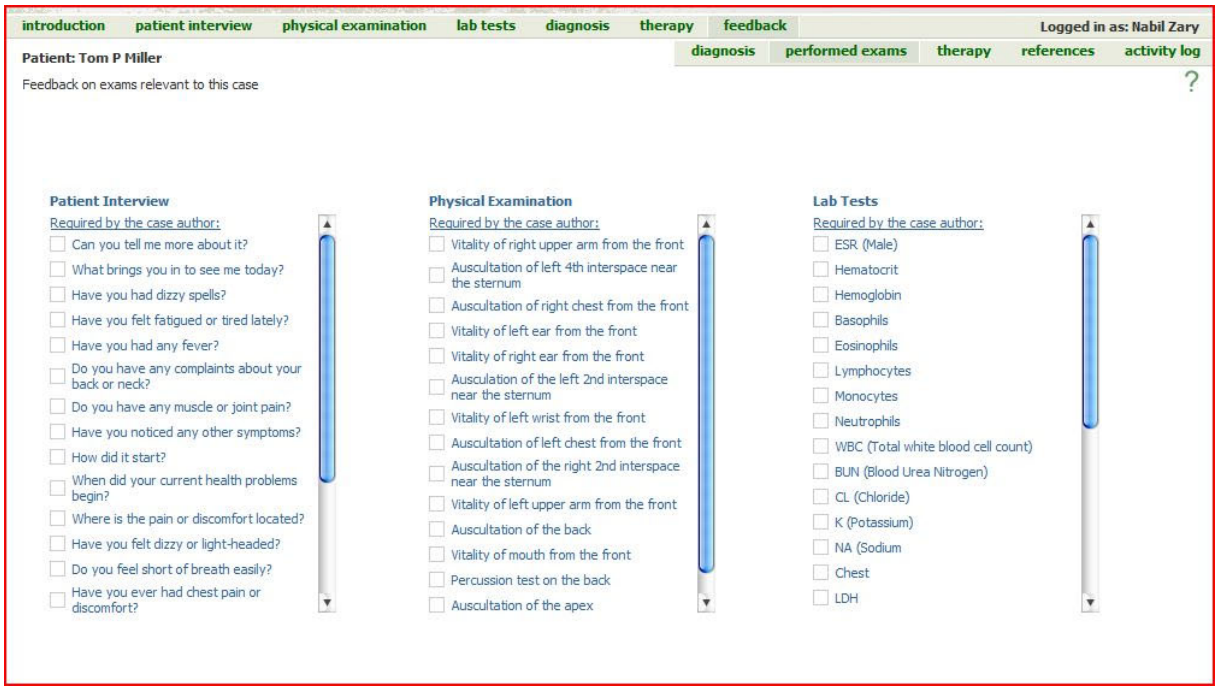


Fig. 39 Shows feedback on the examinations relevant to the case

The VP players are skeletons that can output different templates based on the need of a discipline, course or specific learning outcomes.

A template defines what modules are made available, the overall structure of each module (eg. What main medical history categories are displayed) and what extensions should be enabled (eg. dentistry extension or cost extension).

- **Authoring environments**

Two types of editors are currently used by the VP authors: built-in editor (Fig. 40) and a standalone form-based editor.

Both could be used separately or in combination based on the authoring to be performed.

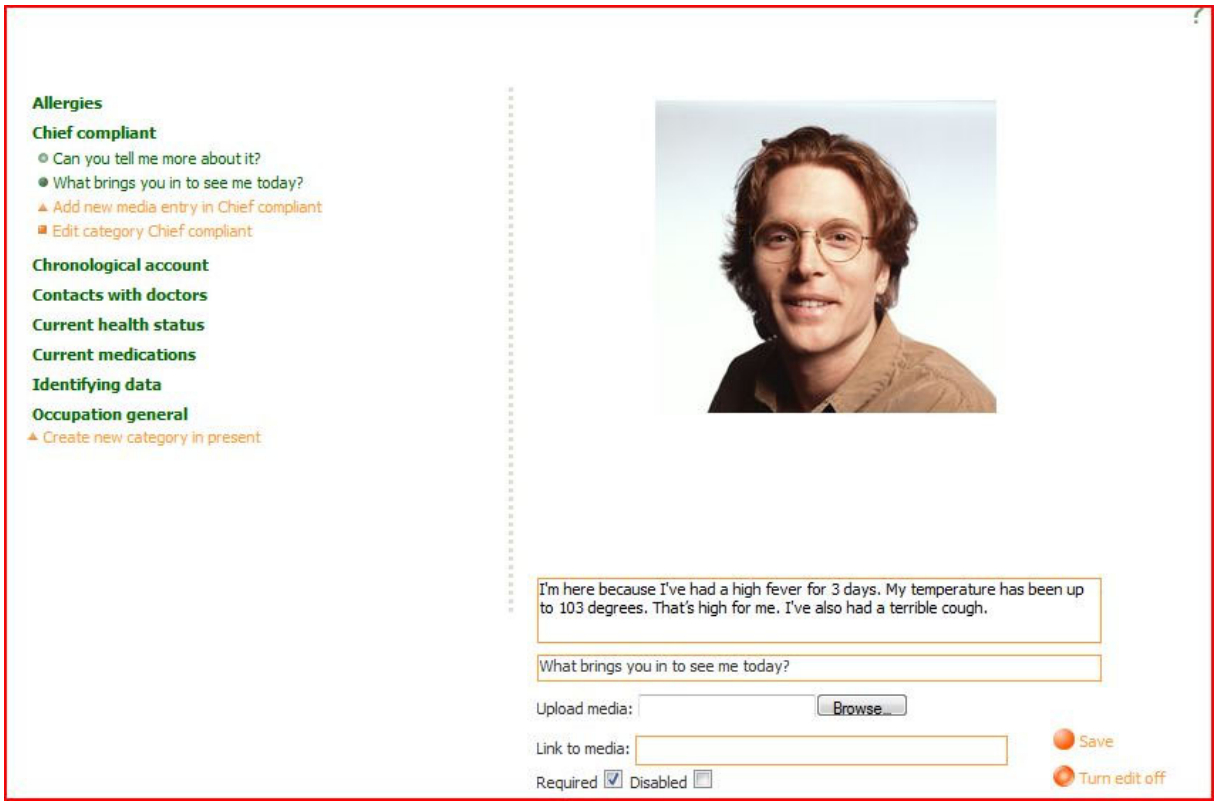


Fig. 40 Shows the Web-SP built-in editor

Mapping of the VP xml Medbiquitous standard to the different elements/modules of the player

Data from a Web-SP VP maps to the different MVP Virtual Patient elements as following:

- *PatientDemographics* for patient data from the introduction module
- *VPDText* for
- *InterviewItem* for medical history questions from the patient interview module
- *PhysicalExam* for physical exams from the examination module
- *DiagnosticTest* for lab tests and other tests from the investigation module
- *Intervention* for the therapy, care plans from the patient management module
- *Diagnosis* for the dx from the diagnosis module
- *XtensibleInfo* for Web-SP specific data

2.4.2 Export module

2.4.2.1.1 Functions related to export of VPs

A VP package can be exported using any of the available authoring environments. This feature is available to the owner(s) of the VP (aka as the VP author).

The following steps are undertaken to export a VP (Figure 41):

- Locate the VP to be exported
- Switch on the built-in authoring mode
- Click on the export link
- A pop-up windows displays a link to the VP package
- The VP package is downloaded by the user on it's local PC

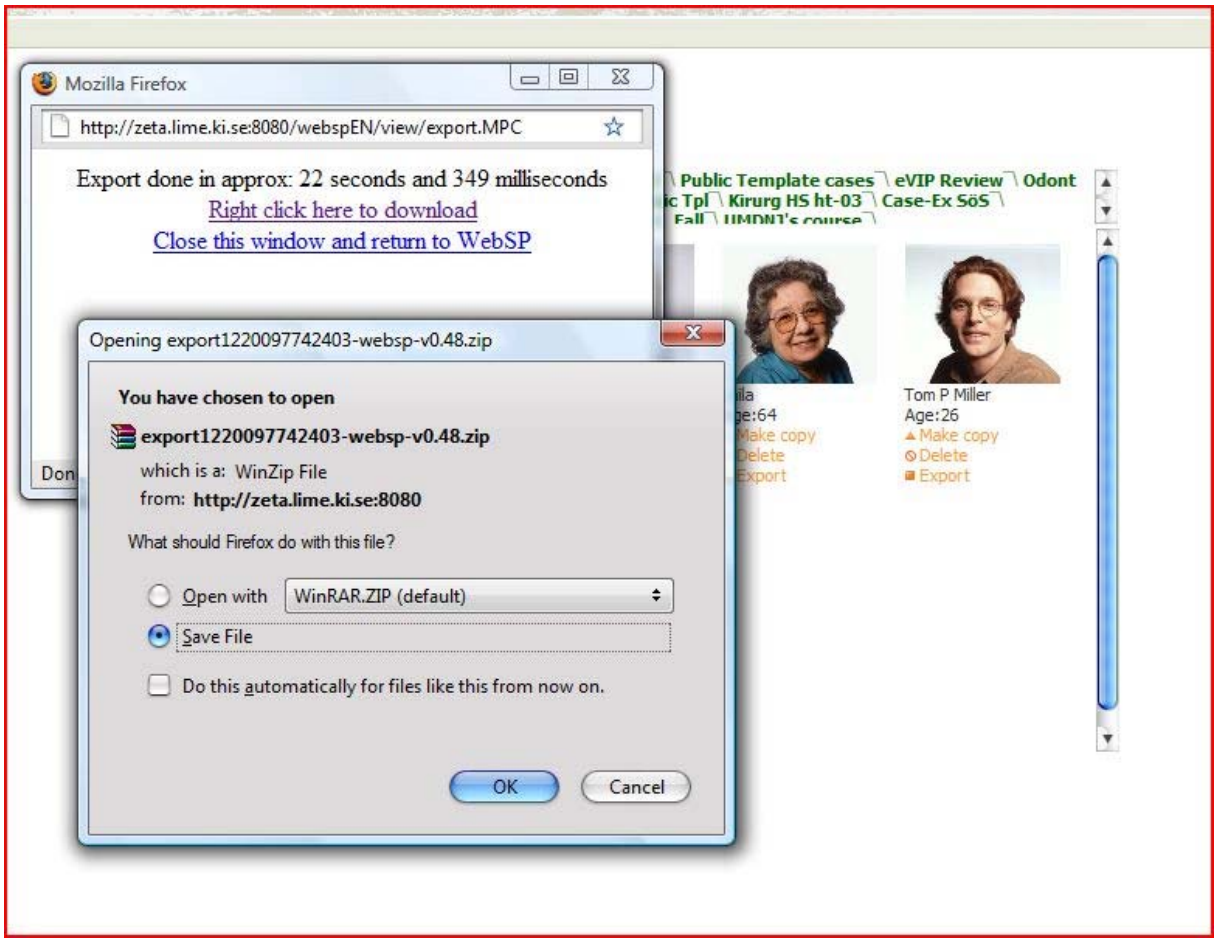


Fig. 41 Shows the different steps to export a VP package from Web-SP

2.4.2.1.2 Development of the export module

The export module was a developed over a longer period of time. The first version of the MVP supported by the Web-SP import/export tool included only the virtualpatient data xml. The technical/architecture approach adopted is very similar to the way Casus is exporting their VPs (reported earlier in this document)

2.4.3 Import module

2.4.3.1 Functions related to import of VPs

The import module is, in a large part, a reverse mirror of the export module. The main difference is at the presentation (gui) level – how the eViP package is imported.

A VP package is imported using one of the available authoring environments.

The following steps are undertaken to import the VP package:

- Switch on the authoring mode
- Locate the import form
- Upload the VP package from the local pc
- The uploaded VP is uploaded to the users pending area

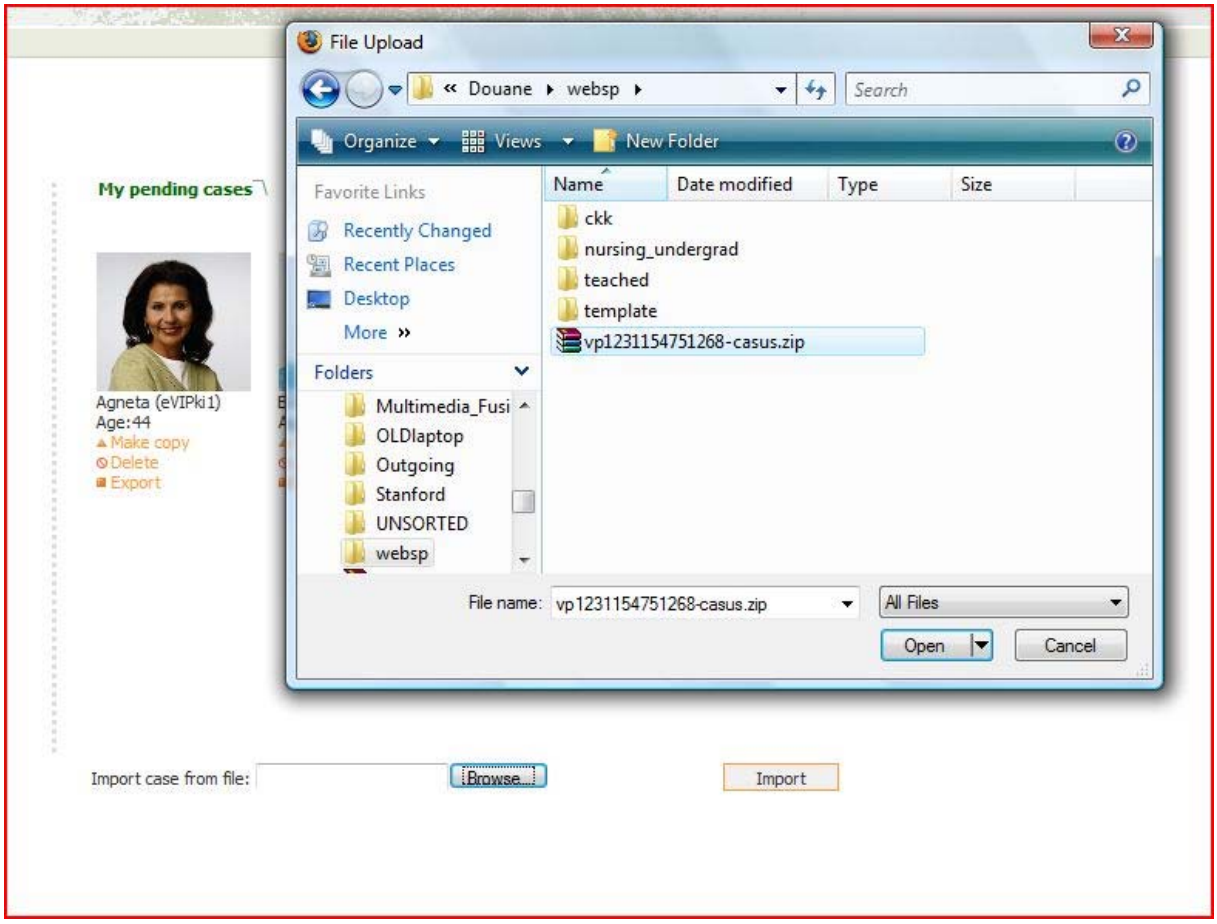


Fig. 42 Showing the different steps while importing a VP package to Web-SP

3 Testing for conformance

3.1.1 Levels of conformance

In order for a packaged virtual patient to be successfully imported into an eViP-compliant system, the package has to follow the eViP specifications. Checking for conformance can be described in four levels of increasing complexity. Ideally, a packaged VP would fulfil all four levels, but within the scope of the eViP project, compliance to level three was set as the goal for what is technically achievable at this stage of the project. Future projects would aim at conformance level 4 as more knowledge is acquired, during WP3, regarding the significance of the different educational approaches adopted by the different players.

Level 1 - Package validation

The first and lowest level of conformance implies that the archive structure and content conform with the eViP profile specifications. This means that the correct directory structure

and file names are used and that all required files are present. On this level, the contents of the files are disregarded. This validation is straightforward to automate and is part of the conformance checking tools (see below).

Level 2 - XML/XSD validation

The second level of conformance requires that the XML files are valid and well-formed relative to their schemas. This includes XML-id references inside and between XML files in the package, references to media resources, and references in the content of the meta file: all included references should be valid. Whether the package contains spurious, non-referenced items is outside the scope of this level of validation.

Level 2 conformance validation can also be automated and has indeed been implemented in two eViP conformance test suites (see below). It is to be regarded as good practice to demand from packages that are stored in the eViP repository that they fulfil this level of conformance. Moreover, to be able to export level-2 compliant VP packages is an essential part of eViP compliancy.

Level 3 - Import validation

The third level of conformance is less straightforward to define: the package must be imported by an eViP-compliant system in a **meaningful way**. It is best described in a set of desiderata. The bottomline is that an author has a clear profit from importing the package into the system.

- After import, enough data must be available to the target author in order to start repurposing of the case in the new system.
- Main sections of the VP package are recognised by the target VP system. After importing there is a clear separation of narrative/clinical data describing the case from data defining the activity model.
- As a threshold for level 3 conformance will hold: it takes less time to import the case automatically than create it anew.
- The package should not contain non-referenced items
- Optional: selected fragments from the package can be automatically imported into the target proprietary VP model. The import process may be customised and controlled by the user allowing a pragmatic conversion between different system models (*e.g while converting from a branched to linear model the system may display all available state nodes and the user selects manually the preferred path*). Manual extensions are usually required to make the case logically consistent.

It means that, for example, a simple import function written as an XSLT transformation that changes all text available in MVP XMLs elements into a long text string regardless of the semantics of elements would not fulfil those criteria.

This third level of conformance does not only concern the exported VP packages but also states what eViP compliant systems should be able to do. This could result in a circular definition: a VP package is level-3 conform (meaning: is imported meaningfully) if an eViP-complaint system can import it meaningfully. To break the circle, eViP compliance has been defined as being able to import a set of standardised VP packages (i.e. the test suite) in a meaningful way.

Level 4 - Runtime validation

The fourth level is the most demanding level of conformance. It states that the imported packaged virtual patient must run in an eViP-compliant system in a way that respects the original virtual patient. Again a set of desiderata is the best way to define this level of conformance:

- No case-related data is lost while importing the package
- The way the data is displayed reflects the main path as it was presented in the original system. The storyline of the case remains consistent.
- The educational value is retained.
- The learning objectives planned for the case in the original system are also achievable in the target system.

As with level 3, conformance to level 4 is as much a property of packaged virtual patients as it is of eViP-compliant systems. Moreover, a highly subjective aspect is added that has to be judged by authors or educationalists. Probably, level-4 conformance will only be defined relative to the source and target system: an exported Casus VP package could be level-four compliant for import in CAMPUS, but not for import in Web-SP. Exported OpenLabyrinth cases will hardly be level-four compliant to any of the other systems.

3.1.2 eViP compliancy of a VP system

From the above we agreed the following two demands for **eViP compliancy of a VP system**:

- An eViP compliant system must be able to export VP in packages that are conformant to level 2 - XML/XSD validation.
- An eViP compliant system must be able to import all cases in the test suite that are conformant to level three - Import validation.

Moreover, we can define an **eViP compatibility relation** between two VP systems to be on level 3 or 4 as follows:

- Two VP systems are compatible at level 3 if they can exchange VP packages at level 3 conformance.
- Two VP systems are compatible at level 4 if they can exchange VP packages at level 4 conformance.

We assume that all four current VP systems within eViP are mutually compatible at level 3.

3.1.3 Conformance application

An automatic conformance test suite can address several areas to check if the exported files are correctly structured according to the spec:

1. All the required files have to be inside the export file
2. All the XML files have to be syntactically correct
3. All the XML files have to be successful validated against the schema files
4. References inside and between the XML files have to be correct
5. References to media files have to succeed

6. All files have to be referenced in the meta file

To test those topics a free and open-source project has been started at http://code.google.com/p/mvp-evip-xslt-test-suite/. It can be run locally or with a webservice where anybody can upload an export and gets the test result using a web browser.

The result is generated as an XML or HTML file.

3.2 eViP Conformance Testing Suites

3.2.1 Introduction

- Short introduction into the premises and goals of conformance testing and its applications

As was already discussed in the previous paragraphs, the first two conformance levels of the eViP profile can be automatically checked by ancillary software, whereas the last two, due to their semantically advanced character, involve human intervention. This section focuses on the first two levels. Where possible, suitable validation software was selected from a list of already implemented tools. New modules were developed in those cases where no appropriate component was available on the market.

Validation against level 1 requires a simple check for the presence of a set of files prescribed by the eViP profile. From the technical point of view this task does not create any difficulties. The validation tool needs to unzip the package and compare its content with the correct list of files. However, because eViP profile is based on newly developed specifications, it was necessary to implement this feature anew.

The second level of conformance examines the common problem of well-formed and valid XML files. For that reason its solution required only the selection of a suitable tool from a long list of publicly available software. A very comprehensive and frequently updated list of tools for XML Schema validation is available on the W3C XML Schema page [30]. For the purpose of determining local schema-validity in the eViP profile JAXP and Xerces were used.

XML files comprising eViP profile conformant packages are usually interlinked by references of two kinds:

- Xpath expressions

Example: Content of DAMNode/ItemPath element in dataavailabilitymodel.xml

```
<DAMNodeItem display="immediately">
        <ItemPath>/VirtualPatientData/InterviewItem[@id="VPD1"]</ItemPath>
</DAMNodeItem>
```

- Identifier attributes

Example: *identifier* attribute of *resource* element in imsmanifest.xml

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

Conformance validation tools at this level need to verify the correctness of the existent references. This includes tests for verification of the existence of referenced elements and detection of unused elements. Such functionality was not present in the available software and had to be developed by the eViP consortium. A more detailed description follows in the sections below.

Rather than using each module from the set of validation tools separately, it was decided to develop a single application - so called eViP conformance testing suite - that would merge all tools for handling the whole validation process.

Two concurrent solutions emerged representing different implementation approaches. The conformance testing suite developed by KI uses in their analysis a Java implementation of the Document Object Model (DOM) interface, whereas HD developed a suit based on XSLT templates. Both applications are described in more detail in the following two sections.

3.2.2 Description of eViP conformance testing application developed by KI

The eViP Conformance testing application developed by KI is an open source web-based application with the primary aim to help developers/implementers to verify that a VP package is compliant with the eViP profile with regards to conformance levels 1 and 2.

Level 1 - Package validation
The first level of conformance implies that the archive structure and content is conformant with the eViP profile specifications. This means that the correct directory structure and file names are used and that all required files are present.

Level 2 - XML/XSD validation
The second level of conformance requires that the XML files are well-formed and validated against their schemas. This includes validation of XML-id references inside and between XML files in the package, references to media resources, and XPath references in the MVP content files.

The eViP Conformance application is downloaded from the official project site (http://code.google.com/p/mvptools/) and then installed locally by the tester. We envision that the tool could also be incorporated as part of a larger import/export process in the VP repositories.

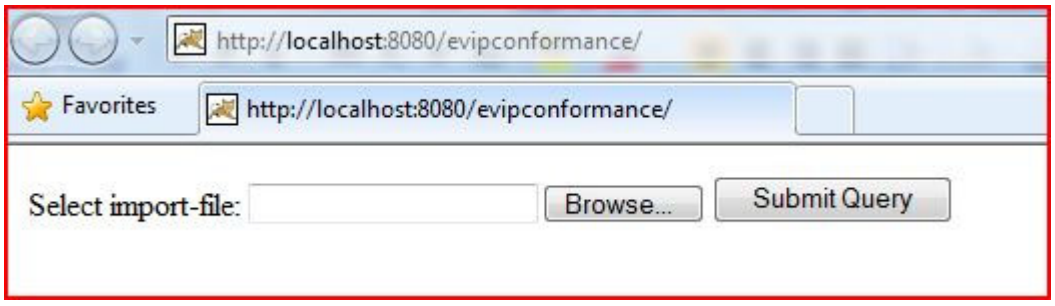


Fig. 43 The first step of the conformance testing process is the selection of the package to be tested

There are two types of outputs that can result from the testing process:

- Output A: The package is conformant
- Output B: The package is not conformant

In the case of output A, the eViP conformance application will notify the tester that the VP package successfully passed the conformance levels 1 and 2 and provides a summary of the media included in the package (and their location).

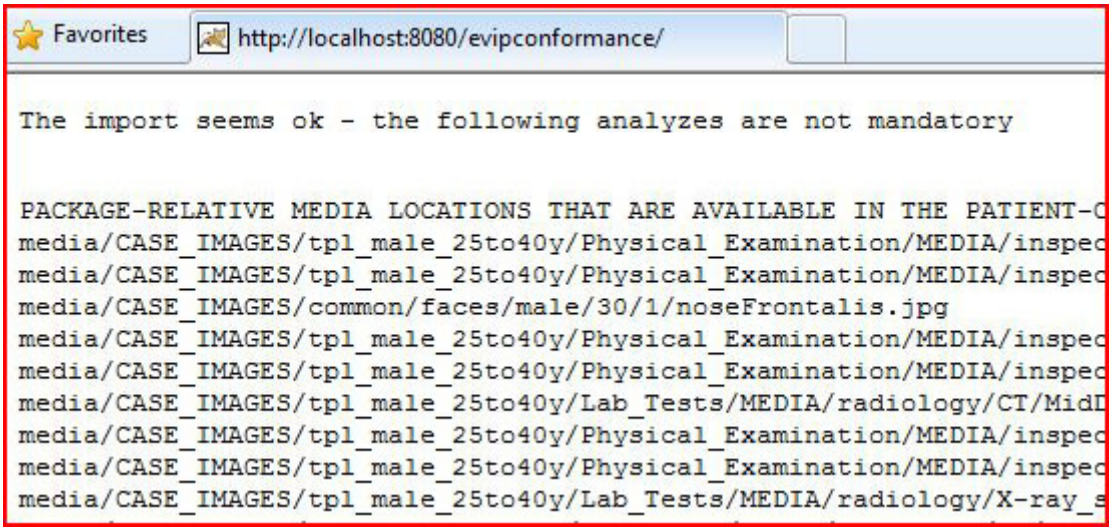


Fig. 44 Shows the result screen of a conformant VP package

Output B is returned if any of the conditions required by conformance levels 1 and 2 are not met. The role of the conformance testing application is then to indicate the error(s).

An example could be:

```
"Xml-problem at line 932:
cvc-complex-type.2.4.b: The content of element 'dam:DAMNode' is not complete.
One of '{"http://ns.medbiq.org/dataavailabilitymodel/v1/":DAMNodeItem}' is
expected."
```

The current version of the conformance testing application tool was helpful in assisting the eViP partners while making their import/export conformant with the eViP profile. As the MVP specification evolves, the application will need to be updated to test for the latest version. Since the source code is available, future efforts and improvements may be conducted by a wider community of developers. The reporting section of the application could be extended to further assist the testers and parts of conformance level 3 could be supported by the tool.

3.2.3 Description of XSLT-based eViP conformance testing application developed by HD

The XSLT conformance suite was developed for having a simple, yet powerful and easy to extend suite for testing of all kinds. The XSLT conformance application mostly just uses plain W3C standard XSLT to develop tests (currently there are 20) for all kind of areas:

- Schema validations
    - Schema Validation of virtualpatientdata.xml
    - Schema Validation of dataavailabilitymodel.xml
    - Schema Validation of activitymodel.xml
    - Schema Validation of imsmanifest.xml
- Files validations (missing files)
    - Files not referenced by imsmanifest.xml
    - Not found files referenced by imsmanifest.xml

- Reference validations (wrong or missing references to parts)
    - DAMNodes not referenced by any ActivitiyNodes
    - ActivityNodes with wrong references to DAMNodes
    - DAMNodeItem with wrong references in ItemPath
    - DAMNodeItem with wrong references in ItemComment
    - Unreferenced VPDText
    - Unreferenced InterviewItem
    - Unreferenced PhysicalExam
    - Unreferenced DiagnosticTest
    - Unreferenced Diagnosis
    - Unreferenced Intervention
- Semantic validations (e.g. checks for senseless attributes)
    - Duplicated IDs
    - Not recommended media references

An example test is written like this:

```
<test name="Files not referenced by imsmanifest.xml">
<output>
 <xsl:for-each select="$files//file">
  <xsl:if test="not($i//ims:resource[@href=current()/@full-path])">
   File: <xsl:value-of select="@full-path"/>
  </xsl:if>
 </xsl:for-each>
</output>
</test>
```

This test case looks for all files ($files//file) and checks if they are referenced by a resource definition in the imsmanifest.xml ($i).

The suite produces and XML file with the test results or a HTML file for easier reading. This makes it possible to integrate it into an application.

The suite can be run as a standalone (Java Runtime 1.6+ required) application or as a web service (for a Servlet container like Apache Tomcat).

The suite is open source (LGPL) available at http://code.google.com/p/mvp-evip-xslt-test-suite

For a quick test you can download the package, extract it and call the batch file:

```
cd <installation-path>
run-tests.bat <path-to-the-extracted-package>
```

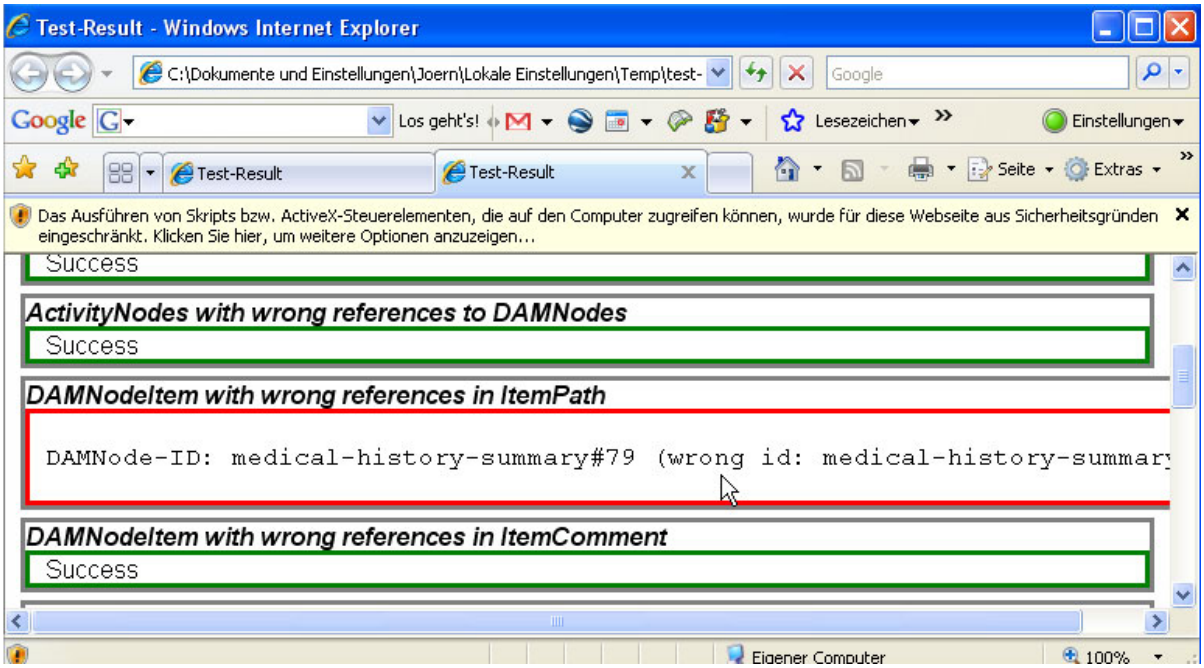


Fig. 45 XSLT-based eViP conformance testing application by HD –executed as command line tool

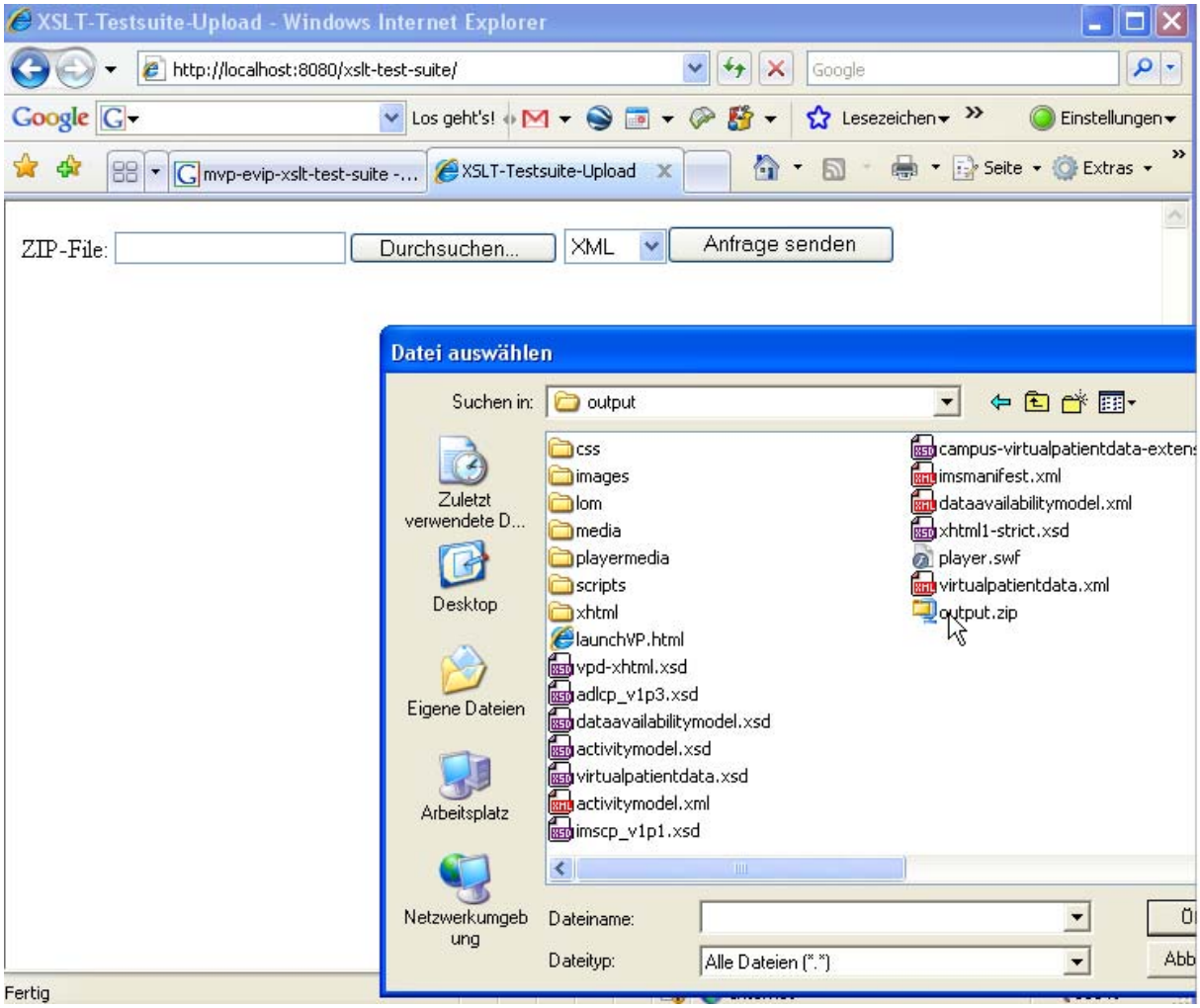


Fig. 46 XSLT-based eViP conformance testing application by HD –executed as web application

3.3 Other Tools and Resources useful in eViP profile conformance testing

The role of conformance testing suite is only to validate the packages - not correct them. In case an error is detected a brief message appears and it is the role of the developer to find a solution for the problem. This process can be supported by XML editors that support error analysis and prototyping of potential remedies. There is a great variety of free and commercial XML integrated development environments including <oXygen/>, Alchemist XML, Altova XML Spy, Butterfly XML, Liquid XML Studio, Steam XML, Stylus Studio, XmlShell and XMLwriter.

3.3.1 Description of conformance tests in Altova XML Spy

In the development and testing process of the eViP profile we often used Altova XML Spy. The XML Spy is currently one of the most powerful and recognised XML editors available on the market.

XML Spy enables editing of eViP conformant XML files using either the traditional text-based view or enhanced grid view. The text view facilitates the editing by expandable and collapsible elements, auto-completion of code (insertion of mandatory elements or attributes) and auto-formatting feature. The grid view presents XML in a hierarchical table format especially useful in analysing of large files as it is often the case for virtual patients (Fig. 47).

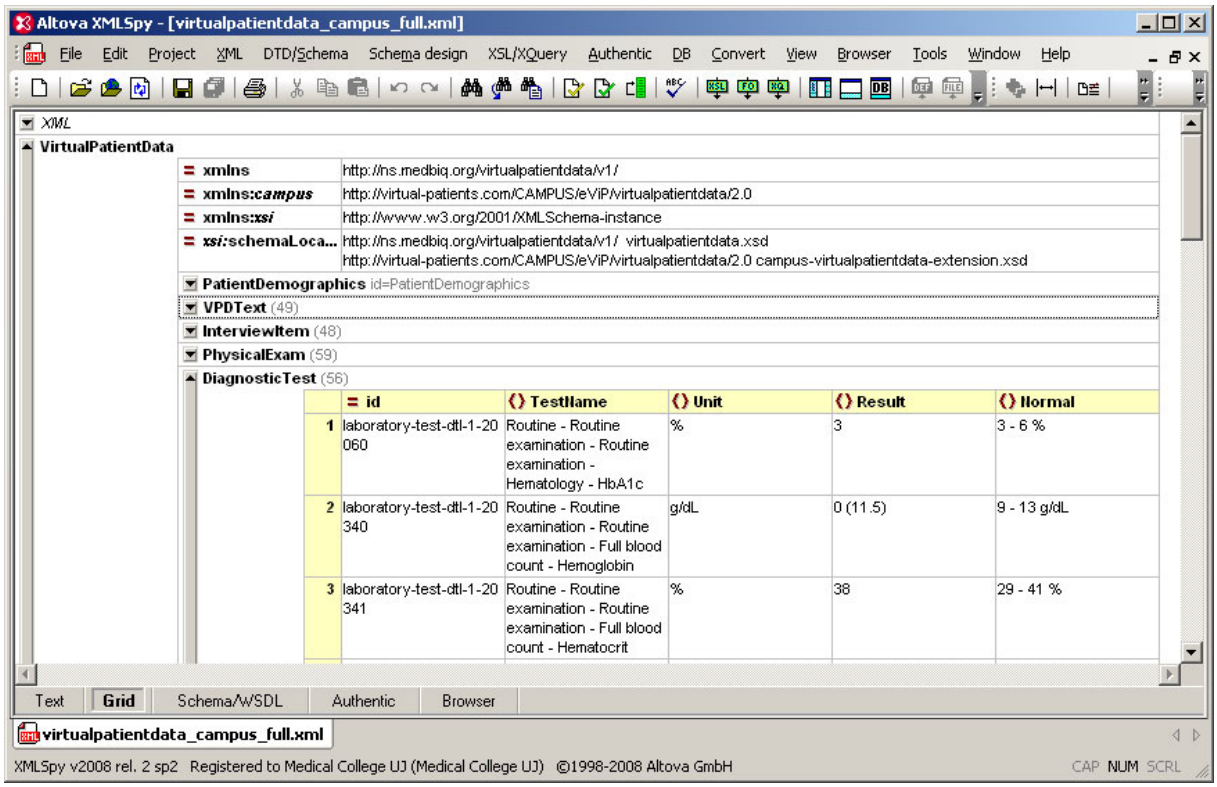


Fig. 47 Altova XML Spy - Grid view of MVP virtual patient data file.

Requested element can be easily located by an advanced find/replace function or through XPath queries. XML Spy includes also an integrated XSLT processor.

XML Spy has specialised features to facilitate browsing of XML Schema files through a special Schema/WSDL view. All mandatory and optional element definitions can be visualised in a text view, table format or as a expandable tree.

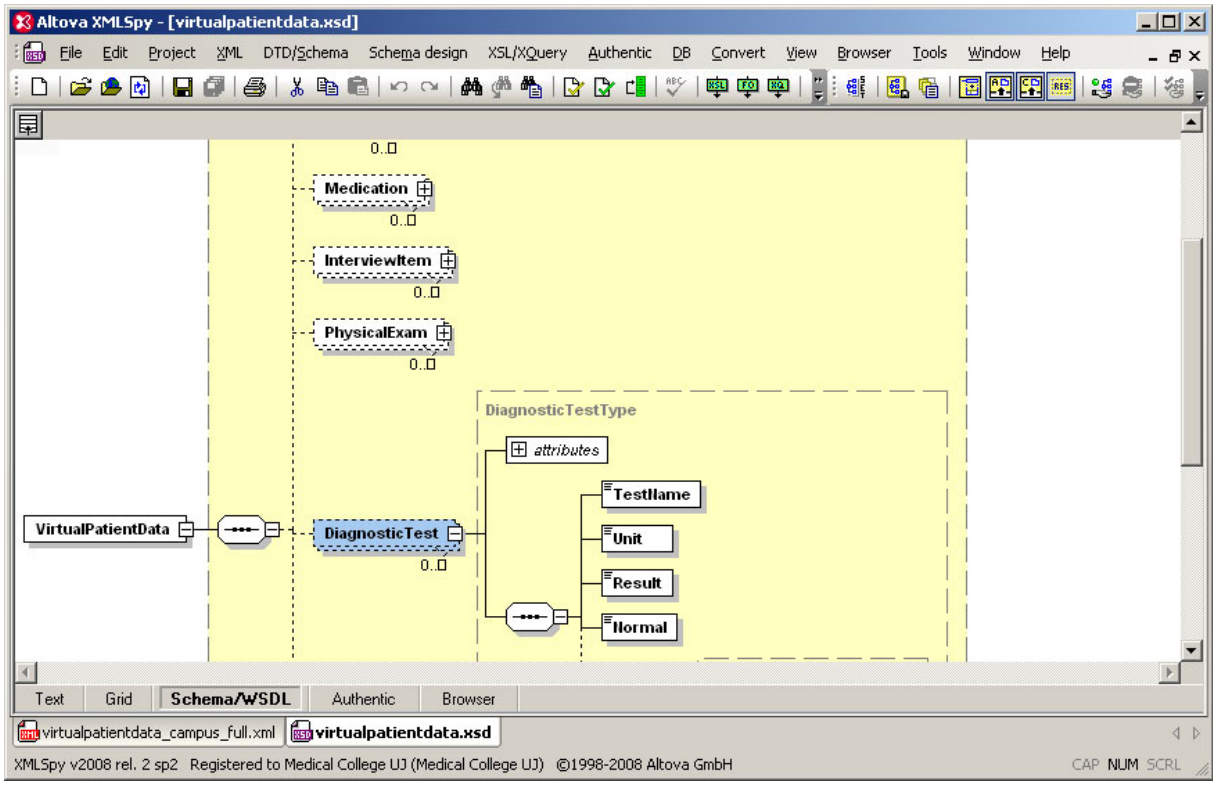


Fig. 48 Altova XML Spy –graphical view of XML Schemas

XML Spy examines well-formedness and validity of edited XML files. However, in contrast to the test suites XML Spy is only able to check for syntax and schema errors but e.g. not verify the existence of references resources and elements.

Unfortunately, XML Spy is commercial and available for Microsoft Windows only.

3.4 Test cases

Following virtual patients has been selected as test cases:

Test Case 1

VP System:	CAMPUS VP
Model:	Semi-linear, Terminology based
Patient id:	evip:vp:1000263
Patient name:	Katrin M.
Description:	Bacterial meningitis; procedures with suspected meningitis. Differential diagnoses: pneumonia, urinary tract infection
Authors:	Benjamin Hanebeck

Test Case 2

VP System:	CAMPUS Key Feature
Model:	Linear
Patient id:	evip:vp:1000263
Patient name:	Katrin M.
Description:	Bacterial meningitis; procedures with suspected meningitis. Differential

diagnoses: pneumonia, urinary tract infection

Authors: Benjamin Hanebeck

Test Case 3a

VP System: CASUS

Model: Linear (assessmentitems implemented as QTI

Patient id: evip:vp:1000131

Patient name: 43 year old saleswoman presented with dyspnea

Description: This case shows criterias for radiography of the chest.

Authors: M. Fischer, M. Maleck, F. Schenk

Test Case 3b

VP System: CASUS

Model: Linear (assessmentitems implemented as VPDText

Patient id: evip:vp:1000131

Patient name: 43 year old saleswoman presented with dyspnea

Description: This case shows criterias for radiography of the chest.

Authors: M. Fischer, M. Maleck, F. Schenk

Test Case 4

VP System: OpenLabyrinth

Model: Branched

Patient id: evip:vp:1000007

Patient name: Joseph Ansah's

Description: This is a demonstration virtual patient of Malaria

Authors: Arnold Somasunderam

Test Case 5

VP System: Web-SP

Model: Semi-linear

Patient id: evip:vp:1000114

Patient name: Tom P Miller

Description: Patient with symptoms of pneumonia

Authors: Dept Lime, Karolinska

In addition, one case has been created artificially to demonstrate QTI features in virtual patient packages:

Test Case 6

VP System: CASUS

Model: Linear with QTI Test Items

Description: Example case with all answertypes (except long menu and cloze) for evip

Authors: Inga Hege

3.5 Validation of imported cases in target systems - pilot study

Selected examples from the test cases in section 3.4 have been imported into target VP systems to validate the conformance at the third level.

Import between linear and semi-linear systems worked very well. In Fig. 49 an imported case from CASUS (Test Case 3) is visible in the authoring tool of CAMPUS.

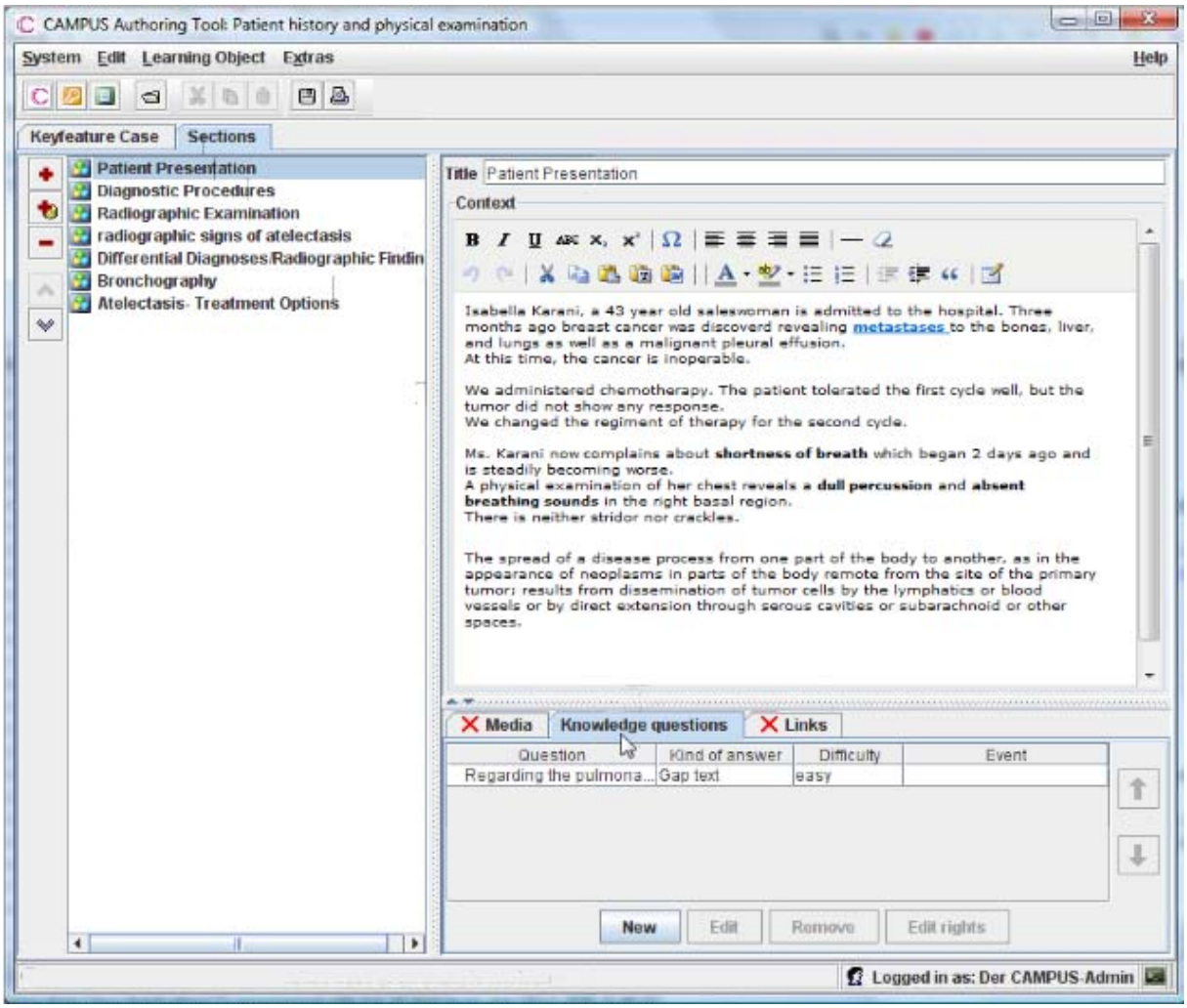


Fig. 49 Test Case 3 (CASUS) in CAMPUS Authoring Tool

The text in the CAMPUS player is well formatted and images are in place (Fig. 50). QTI assessment items have been automatically imported and are ready for use (Knowledge question tab in Fig. 49). Even less popular question types like sorting or network questions are transferable to the authoring tool and operable in the CAMPUS player (Fig. 51). The transfer of cases the other way around (from semi-linear into linear) has also been tested and is demonstrated for the systems CASUS and CAMPUS in Fig. 52 (Test Case 2) and CASUS and Web-SP in Fig. 53 (Test Case 5 – it resulted in an unusual long case with lot short cards). The third level of conformance for these models has been reached.

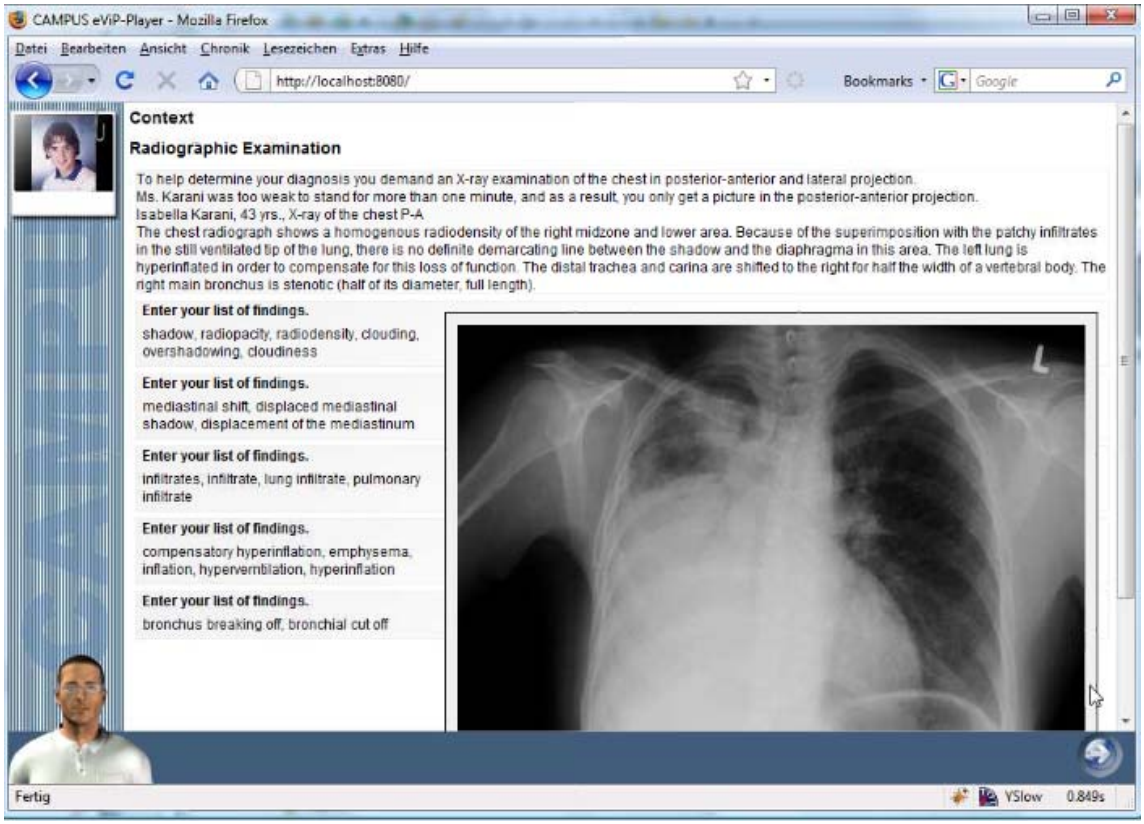


Fig. 50 Test Case 3 (CASUS) in CAMPUS eViP-Player

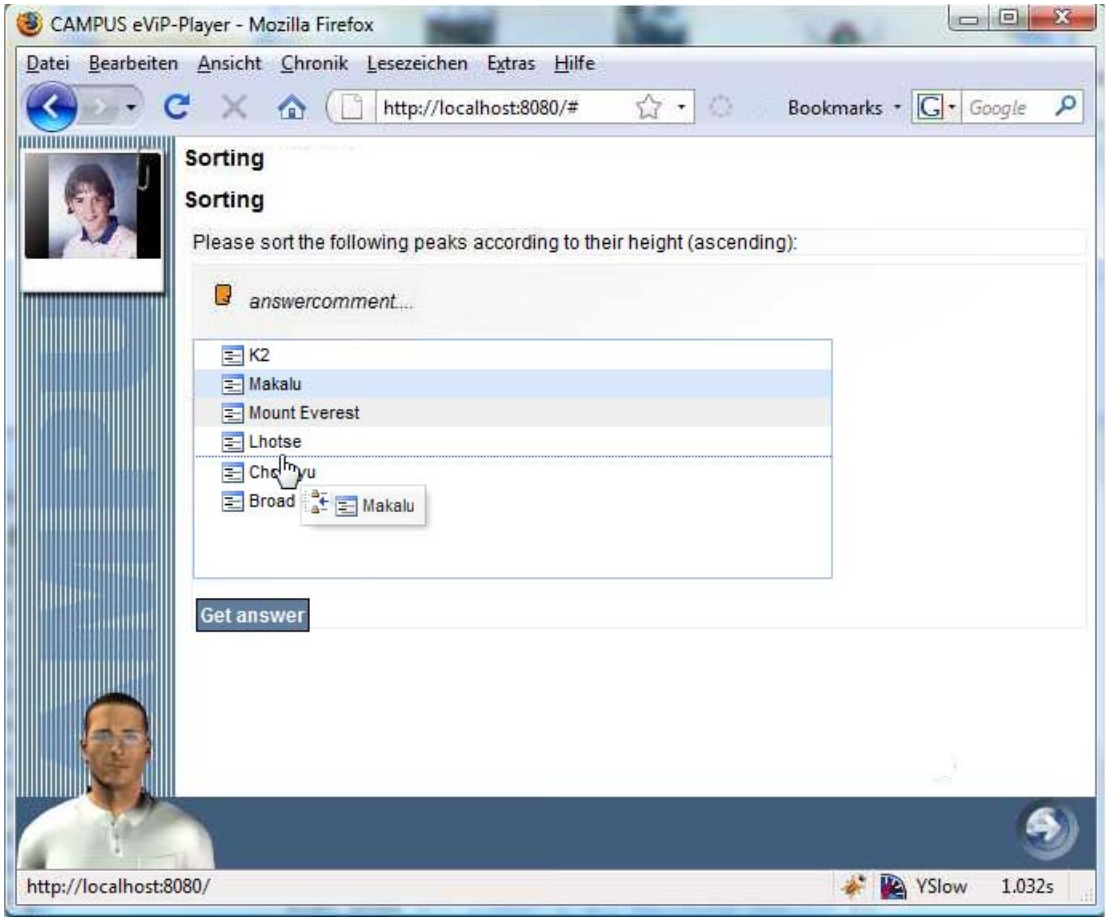


Fig. 51 Test Case 6 (CASUS QTI) in CAMPUS eViP-Player - Sorting question

Fig. 52 Test Case 2 (CAMPUS) in CASUS player

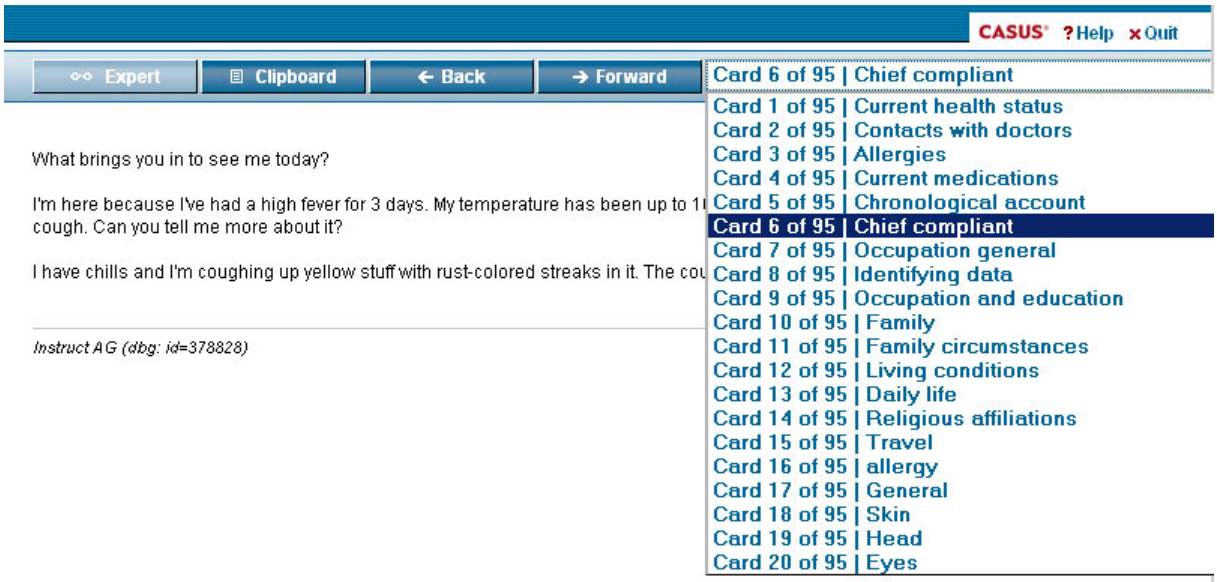


Fig. 53 Test Case 5 (Web-SP) in CASUS player

The most difficult transfer - from branched into (semi-)linear models - has also been achieved at the third conformance level. The result of importing of the test case 4 into CAMPUS is presented in Fig. 54. Importing of the same case into CASUS is demonstrated in Fig. 55. Pragmatic methods (described in more detail in paragraph 2.2) have been used to “linearise” the branches. In CASUS for instance, links available in the original case have been converted to text strings and the nodes ordered roughly. In all cases of branched VPs imported into (semi-) linear, manual post-processing operations were needed.

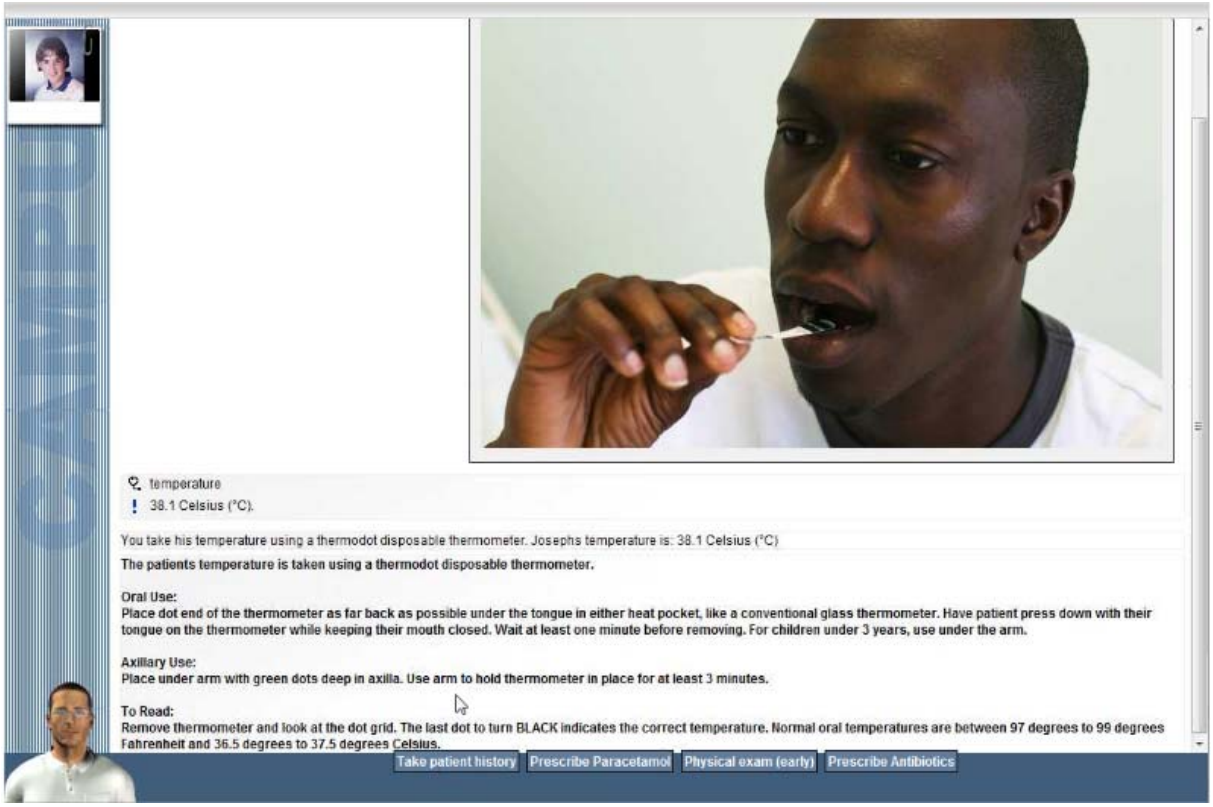


Fig. 54 Test Case 4 (OpenLabyrinth) in CAMPUS eViP player

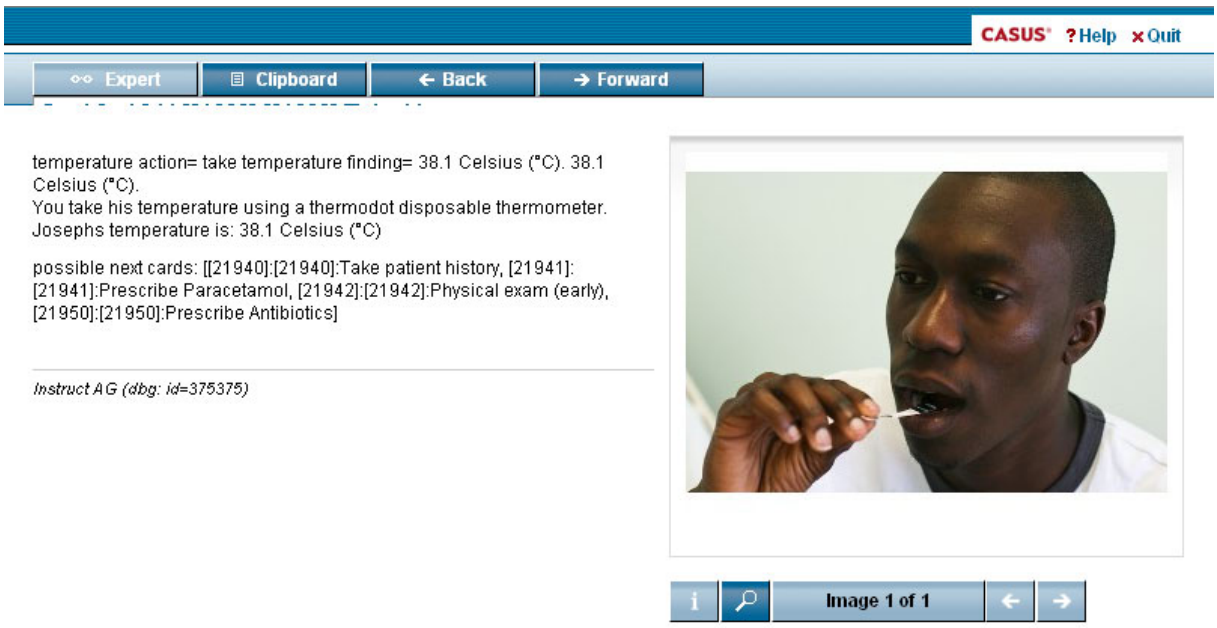


Fig. 55 Test Case 4 (OpenLabyrinth) in CASUS player

3.6 Statistics of MVP usage in eViP application profile

3.6.1 Introduction

A simple tool has been implemented to calculate statistics of the implemented VP cases. It gives an interesting insight into what elements of MVP model are useful for each partner systems.

The first part of the obtained data (→ annex 1) contains tables on how frequently individual elements of the MVP schema implemented in the eViP profiles have been used in all tested VP cases. Some schema elements may occur in more than one place in the XML DOM-tree. For these cases the frequency of values has added together. The number of used elements cannot be taken as an indicator of the quality of implementation but rather a measure of the compatibility between the MVP model and proprietary models (or simply, as a feature of given VP model class).

The second part shows the number of characters (without spaces) in text elements that are children of the given element (a top-down walk through the tree). This demonstrates how "text intensive" are those elements. The relative values are the absolute number of characters divided by the frequency of occurrence of the element in the document.

3.6.2 Results

Results for 6 test VP cases are available in annex 1.

3.6.3 Conclusions

Based on the results obtained for test cases which are presented in annex 1 early conclusions may be drawn:

1. eViP's semi-linear models like CAMUS VP and Web-SP use many specialised fields like BodyPart or InterviewItem with relatively few general purpose text elements like VPDText in the Virtual Patient Data description

2. eViP's Linear cases (like Casus or CAMPUS Key Feature) are characterised by the frequent use of text-heavy VPDText elements in the Virtual Patient Data description

3. Distinctive feature of branched cases is the relatively high number of Link and Rule elements in the Activity Model.

4. All elements in the DAM model proved to be useful

5. Three (CAMPUS, Casus, Web-SP) out of four systems used the XtensibleInfo element to extend the specification

6. Not used (so far) by the eViP application profile were MVP elements expressing:

   a) Conditional rules
   b) Counter
   c) Timers
   d) Specialised body location indicators (eg. ProximalOrDistal, InferiorOrSuperior)
   e) High level classification fields like species, breed

For general conclusions more eViP conformant cases need to be tested.

4 Summary

The purpose of deliverable 2.2 was to implement the eViP profile (D2.1) in all partner systems and test that these allow compliant import and export of VP content.

4.1 Successes & challenges

All four VP systems successfully implemented the eViP profile. The variation in strategies to achieve this deliverable has provided us with an increased understanding of the different VP systems which is also of interest to the wider technical community.

The learnings by all the partners involved in the implementation of the eViP profile have the potential to be generalised and adopted by future implementers of the profile for different systems. The successful mix of several specifications (MVP xml, Healthcare LOM, SCORM) is innovative and has inspired the international healthcare standardisation community.

The eViP project was the first of its kind to define levels of conformance and managed to release the only conformance testing applications available to date. These applications may play an important role in enabling the increased sharing of conformant VP packages.

The variation between the functionalities of the different eViP VP systems is a strength of the project but has also created challenges that will need to be addressed by the project or in other related projects. It is envisaged that further work will be carried out in accordance with work package 3 of the project.

4.2 Future work

A natural next step is to increase the number of case studies (work in progress) to systematically test all the possible scenarios between the four VP systems.
Furthermore, we will actively disseminate the results, both in Europe and elsewhere, to ensure a wider adoption of the eViP profile. The validity and generalisability of our findings will be challenged as more VP systems get involved in similar work.

5 References

[1] Balasubramaniam, C. & Poulton, T. (2008), 'eViP: Electronic virtual patients', *The Newsletter of the Higher Education Academy Subject Centre for Medicine, Dentistry and Veterinary Medicine* **01.16**, 6-7.

[2] Ellaway, R. (2004), 'Modeling virtual patients and virtual cases', *MedBiquitous E-Learning Discourse*.

[3] Ellaway, R. & McGee, J. (2006), 'Unlocking virtual patients', *The Newsletter of the Higher Education Academy Subject Centre for Medicine, Dentistry and Veterinary Medicine* **01.12**, 16-17.

[4] Ellaway, R.; Poulton, T.; Fors, U.; McGee, J. B. & Albright, S. (2008), 'Building a virtual patient commons.', Med Teach 30(2), 170–174.

[5] Ellaway, R.; Cameron, H.; Ross, M.; Laurie, G.; Maxwell, M. & Pratt, R. (2006), Clinical Recordings for Academic Non-clinical Settings, JISC Project.

[6] Ellaway, R.; Cameron, H.; Ross, M.; Laurie, G.; Maxwell, M. & Pratt, R. (2007), Digital Images in Education: Realising the Vision, JISC Collections, chapter Towards a Clinical Commons: Using Clinical Recordings in Academic Non-Clinical Settings, pp. 101-120.

[7] Fischer, M. R. (2000), *Use of Computers in Medical Education (Part II)*, Zeitschrift fur Hochschuldidaktik, chapter CASUS – An authoring and learning tool supporting diagnostic reasoning, pp. 87–98.

[8] Herry R.;Patel M. (2000), 'Application profiles: mixing and matching metadata schemas', Ariadne Issue 25, http://www.ariadne.ac.uk/issue25/app-profiles/intro.html [accessed 22/12/2008]

[9] Huang, G.; Reynolds, R. & Candler, C. (2007), 'Virtual patient simulation at US and Canadian medical schools.', *Acad Med* **82**(5), 446--451.

[10] Grunwald, T. & Corsbie-Massay, C. (2006), 'Guidelines for cognitively efficient multimedia learning tools: educational strategies, cognitive load, and interface design.', Acad Med 81(3), 213–223.

[11] Page, G., Bordage, G., Allen, T. (1995): Developing key-feature problems and examinations to assess clinical decision-making skills. Academic Medicine, 70; 194-201

[12] Pfaehler M & Holzer M.(2008) 'Erweiterung des QTI-Standards zur Unterstützung von Long-Menu-Fragen sowie Abbildung spezifischer Metadaten in QTI 2.1', Workshop der Arbeitsgruppe "Computerunterstützte Lehr- und Lernsysteme in der Medizin" der GMDS, Saarbrücken, 34-36

[13] Rubin, J. & Chisnell, D. (2008), Handbook of Usability Testing, Second Edition: How to Plan, Design, and Conduct Effective Tests, Wiley Publishing, Inc..
[14] Ruderich, F.; Bauch, M.; Haag, M.; Heid, J.; Leven, F. J.; Singer, R.; Geiss, H. K.; Jünger, J. & Tönshoff, B. (2004), 'CAMPUS--a flexible, interactive system for web-based, problem-based learning in health care.', *Stud Health Technol Inform* **107**(Pt 2), 921--925.
[15] Sedgewick, R. (2003), Algorithms in Java, Third Edition, Part 5: Graph Algorithms, Addison Wesley.
[16] Shortliffe EH; & Cimino J. (Eds.) (2006), 'Biomedical Informatics – Computer Applications in Health Care and Biomedicine', Springer Science
[17] Smothers, V. & Azan, B. (2008), 'MedBiquitous Virtual Patient Specifications and Description Document v.0.48', Technical report, MedBiquitous Consortium.
[18] Smothers, V. & Azan, B. (2008), 'MedBiquitous Virtual Patient Player Specifications and Description Document v.0.48', Technical report, MedBiquitous Consortium.
[19] Smothers, V.; Greene, P.; Ellaway, R. & Detmer, D. E. (2008), 'Sharing innovation: the case for technology standards in health professions education.', *Med Teach* **30**(2), 150--154.
[20] Zary, N.; Johnson, G.; Boberg, J. & Fors, U. G. H. (2006), 'Development, implementation and pilot evaluation of a Web-based Virtual Patient Case Simulation environment--Web-SP.', *BMC Med Educ* **6**, 10.
[21] ADL Sharable Content Object Reference Model http://www.adlnet.gov/scorm
[22] IMS Application Profile Guidelines Overview http://www.imsglobal.org/ap/apv1p0/imsap_oviewv1p0.html [accessed 22/12/2008]
[23] IMS Meta-data Best Practice Guide IEEE 1484.12.1-2002 Standard for Learning Object Metadata http://www.imsglobal.org/metadata/mdv1p3pd/imsmd_bestv1p3pd.html [accessed 22/12/2008]
[24] IMS QTI Homepage http://www.imsglobal.org/question [accessed 22/12/2008]
[25] IMS Question and Test Interoperability Overview Version 2.1 Public Draft (revision 2) Specification http://www.imsglobal.org/question/qtiv2p1pd2/imsqti_oviewv2p1pd2.html [accessed 22/12/2008]
[26] IMS Question and Test Interoperability Implementation Guide Version 2.1 Public Draft (revision 2) Specification http://www.imsglobal.org/question/qtiv2p1pd2/imsqti_implv2p1pd2.html [accessed 22/12/2008]
[27] IMS Question and Test Interoperability Assessment Test, Section, and Item Information Model Version 2.1 Public Draft (revision 2) Specification http://www.imsglobal.org/question/qtiv2p1pd2/imsqti_infov2p1pd2.html [accessed 22/12/2008]
[28] OpenLabyrinth http://sourceforge.net/projects/openlabyrinth/ [accessed 22/12/2008]
[29] OpenLabyrinth User Guide http://labyrinth.mvm.ed.ac.uk/documents/labyrinth_userguide.pdf [accessed 22/12/2008]
[30] W3C XML Schema Tools http://www.w3.org/XML/Schema#Tools [accessed 22/12/2008]
[31] W3C Timed Text (TT) Authoring Format 1.0 – Distribution Format Exchange Profile (DFXP) http://www.w3.org/TR/2006/CR-ttaf1-dfxp-20061116 [accessed 22/12/2008]

6 Annex 1 Statistics of MVP usage in eViP application profile

6.1 Frequency of occurrence of MVP elements in all 6 test packages

Description in 3.6 (Statistics of MVP usage in eViP application profile)

virtualpatientsdata.xml

MVP Element	CAMPUS VP Test Case 1	CAMPUS KF Test Case 2	CASUS Test Case 3a	CASUS Test Case 3b	OpenLabyrinth Test Case 4	Web-SP Test Case 5	Used in eViP?
Action	0	0	0	0	12	756	+
Age	1	0	0	0	1	1	+
Appropriateness	0	0	0	0	0	0	not used
BodyPart	31	0	0	0	1	756	+
Breed	0	0	0	0	0	0	not used
CoreDemographics	1	0	0	0	1	1	+
DemographicCharacteristic	0	0	0	0	1	0	+
Description	59	0	0	0	13	756	+
Diagnosis	14	0	0	0	0	1	+
DiagnosisName	14	0	0	0	0	1	+
DiagnosticTest	56	0	0	0	24	333	+
Dose	0	0	0	0	6	0	+

ExamName	59	0	0	0	12	756	+
Finding	59	0	0	0	12	756	+
Frequency	0	0	0	0	6	0	+
FrontOrBack	0	0	0	0	0	734	+
InferiorOrSuperior	0	0	0	0	0	0	not used
Intervention	10	0	0	0	0	0	+
InterventionName	10	0	0	0	0	0	+
InterviewItem	48	0	0	0	35	190	+
Item	0	0	0	0	0	0	not used
Likelihood	14	0	0	0	0	0	+
Location	0	0	0	0	1	832	+
LocationOnBody	31	0	0	0	1	756	+
Media	0	0	0	0	1	832	+
MediaPath	0	0	0	0	1	832	+
Medication	0	0	0	0	6	0	+
MedicationName	0	0	0	0	6	0	+
Name	1	0	0	0	1	1	+
Normal	56	0	0	0	24	333	+
Organization	0	0	0	0	0	0	not used
PatientDemographics	1	0	0	0	1	1	+
PatientID	1	0	0	0	1	1	+
PhysicalExam	59	0	0	0	12	756	+
ProximalOrDistal	0	0	0	0	0	0	not used
Question	48	0	0	0	35	190	+
Race	0	0	0	0	1	0	+
Response	48	0	0	0	35	190	+
Result	56	0	0	0	24	333	+
Results	0	0	0	0	0	0	not used
RightOrLeft	0	0	0	0	0	646	+
Route	0	0	0	0	6	0	+
Section	0	0	0	0	0	0	not used
Sex	1	0	0	0	1	1	+
Species	0	0	0	0	0	0	not used
TestName	56	0	0	0	24	333	+
Title	0	0	0	0	1	0	+
Unit	56	0	0	0	24	333	+
VPDText	49	28	22	25	23	1	+
VirtualPatientData	1	1	1	1	1	1	+
XtensibleInfo	1	0	1	0	0	1	+

dataavailabilitymodel.xml

MVP Element	CAMPUS VP Test Case 1	CAMPUS KF Test Case 2	CASUS Test Case 3a	CASUS Test Case 3b	OpenLabyrinth Test Case 4	Web-SP Test Case 5	Used in eViP?
DAMNode	54	28	10	10	38	100	+
DAMNodeItem	246	38	29	29	202	1376	+
DAMNodeLabel	28	28	10	10	0	100	+
DAMNodePath	0	0	0	0	0	97	+
DataAvailabilityModel	1	1	1	1	1	1	+
ItemComment	18	0	3	3	0	0	+
ItemOrder	0	0	29	29	202	26	+
ItemPath	246	38	29	29	202	1376	+
XtensibleInfo	1	0	0	0	0	0	+

activitymodel.xml

MVP Element	CAMPUS VP Test Case 1	CAMPUS KF Test Case 2	CASUS Test Case 3a	CASUS Test Case 3b	OpenLabyrinth Test Case 4	Web-SP Test Case 5	Used in eViP?
ActivityModel	1	1	1	1	1	1	+
ActivityNode	28	28	7	7	37	3	+
ActivityNodeA	27	0	6	6	77	6	+
ActivityNodeB	27	0	6	6	77	6	+

ActivityNodes	1	1	1	1	1	1	+
And	0	0	0	0	0	0	not used
ConditionalRule	0	0	0	0	0	0	not used
Content	28	28	7	7	37	3	+
Counter	0	0	0	0	0	0	not used
CounterActionRule	0	0	0	0	0	0	not used
CounterInitValue	0	0	0	0	0	0	not used
CounterLabel	0	0	0	0	0	0	not used
CounterOperator	0	0	0	0	0	0	not used
CounterPath	0	0	0	0	0	0	not used
CounterRuleValue	0	0	0	0	0	0	not used
CounterRules	0	0	0	0	0	0	not used
CounterUnitsPrefix	0	0	0	0	0	0	not used
CounterUnitsSuffix	0	0	0	0	0	0	not used
Counters	0	0	0	0	0	0	not used
Link	27	0	6	6	77	6	+
Links	1	0	1	1	1	1	+
Nand	0	0	0	0	0	0	not used
NavigateGlobal	0	0	0	0	37	0	+
NodeSection	1	1	3	3	1	3	+
Nor	0	0	0	0	0	0	not used
Operand	0	0	0	0	0	0	not used
Operator	0	0	0	0	0	0	not used
Or	0	0	0	0	0	0	not used
Probability	0	0	0	0	37	0	+
Properties	1	0	0	0	1	0	+
Relation	0	0	0	0	0	0	not used
Rule	0	0	0	0	0	0	not used
RuleMessage	0	0	0	0	0	0	not used
RuleRedirect	0	0	0	0	0	0	not used
Rules	0	0	0	0	37	0	+
Services	0	0	0	0	0	0	not used
Timer	0	0	0	0	0	0	not used
TimerDeltaSeconds	0	0	0	0	0	0	not used
TimerDirection	0	0	0	0	0	0	not used
TimerRules	0	0	0	0	0	0	not used
Value	0	0	0	0	0	0	not used
Weighting	0	0	0	0	0	0	not used
XtensibleInfo	0	0	0	0	0	0	not used

6.2 Relative text weight of MVP elements in all 6 test packages

Description in 3.6 (Statistics of MVP usage in eViP application profile)

virtualpatientsdata.xml

MVP Element	CAMPUS VP Test Case 1	CAMPUS KF Test Case 2	CASUS Test Case 3a	CASUS Test Case 3b	OpenLabyrinth Test Case 4	Web-SP Test Case 5
Action	0	0	0	0	17,2	7,9
Age	5	0	0	0	4	4
Appropriateness	0	0	0	0	0	0
BodyPart	10,2	0	0	0	3	5,1
Breed	0	0	0	0	0	0
CoreDemographics	25	0	0	0	35	39
DemographicCharacteristic	0	0	0	0	20	0
Description	18,8	0	0	0	28	3
Diagnosis	21,7	0	0	0	0	462
DiagnosisName	17,7	0	0	0	0	462

DiagnosticTest	116,9	0	0	0	42,1	74
Dose	0	0	0	0	22,2	0
ExamName	18,8	0	0	0	16,7	31,6
Finding	37,4	0	0	0	26,3	13,4
Frequency	0	0	0	0	30,8	0
FrontOrBack	0	0	0	0	0	4,6
InferiorOrSuperior	0	0	0	0	0	0
Intervention	26,4	0	0	0	0	0
InterventionName	26,4	0	0	0	0	0
InterviewItem	127,4	0	0	0	87,4	84,8
Item	0	0	0	0	0	0
Likelihood	4	0	0	0	0	0
Location	0	0	0	0	29	85
LocationOnBody	10,2	0	0	0	3	13,5
Media	0	0	0	0	78	140
MediaPath	0	0	0	0	49	55
Medication	0	0	0	0	70,3	0
MedicationName	0	0	0	0	9,7	0
Name	8	0	0	0	11	18
Normal	5,4	0	0	0	5,5	7,1
Organization	0	0	0	0	0	0
PatientDemographics	25	0	0	0	55	39
PatientID	6	0	0	0	9	13
PhysicalExam	80,4	0	0	0	90,2	210,7
ProximalOrDistal	0	0	0	0	0	0
Question	84	0	0	0	44,7	32,5
Race	0	0	0	0	7	0
Response	43,4	0	0	0	42,7	52,3
Result	10	0	0	0	15,5	17,4
Results	0	0	0	0	0	0
RightOrLeft	0	0	0	0	0	4,5
Route	0	0	0	0	7,7	0
Section	0	0	0	0	0	0
Sex	6	0	0	0	4	4
Species	0	0	0	0	0	0
TestName	96,4	0	0	0	13,3	16,5
Title	0	0	0	0	13	0
Unit	5	0	0	0	4,7	4,4
VPDText	338	977,9	347	315,9	188,7	70
VirtualPatientData	91479	27380	8108	7897	9967	220656
XtensibleInfo	56918	0	474	0	0	19996

dataavailabilitymodel.xml

MVP Element	CAMPUS VP Test Case 1	CAMPUS KF Test Case 2	CASUS Test Case 3a	CASUS Test Case 3b	OpenLabyrinth Test Case 4	Web-SP Test Case 5
DAMNode	305,6	126,4	188,8	178,9	253,6	756,1
DAMNodeItem	64,6	70,6	57,7	54,3	47,7	54,1
DAMNodeLabel	21,5	30,6	21,4	21,4	0	11,2
DAMNodePath	0	0	0	0	0	51
DataAvailabilityModel	16502	3540	1888	1789	9636	75612
ItemComment	52	0	57	57	0	0
ItemOrder	0	0	1	1	1,7	1,1
ItemPath	60,8	70,6	50,8	47,4	46	50,5
XtensibleInfo	0	0	0	0	0	0

activitymodel.xml

MVP Element	CAMPUS VP Test Case 1	CAMPUS KF Test Case 2	CASUS Test Case 3a	CASUS Test Case 3b	OpenLabyrinth Test Case 4	Web-SP Test Case 5
ActivityModel	6916	1983	1188	1188	12273	958
ActivityNode	68,9	70,8	48	48	57	71,3
ActivityNodeA	92,4	0	71	71	66	62
ActivityNodeB	92,3	0	71	71	66	62
ActivityNodes	1929	1983	336	336	2109	214
And	0	0	0	0	0	0
ConditionalRule	0	0	0	0	0	0
Content	68,9	70,8	48	48	52	71,3
Counter	0	0	0	0	0	0
CounterActionRule	0	0	0	0	0	0
CounterInitValue	0	0	0	0	0	0
CounterLabel	0	0	0	0	0	0
CounterOperator	0	0	0	0	0	0
CounterPath	0	0	0	0	0	0
CounterRuleValue	0	0	0	0	0	0
CounterRules	0	0	0	0	0	0
CounterUnitsPrefix	0	0	0	0	0	0
CounterUnitsSuffix	0	0	0	0	0	0
Counters	0	0	0	0	0	0
Link	184,7	0	142	142	132	124
Links	4987	0	852	852	10164	744
Nand	0	0	0	0	0	0
NavigateGlobal	0	0	0	0	2	0
NodeSection	1929	1983	112	112	2109	71,3
Nor	0	0	0	0	0	0
Operand	0	0	0	0	0	0
Operator	0	0	0	0	0	0
Or	0	0	0	0	0	0
Probability	0	0	0	0	3	0
Properties	0	0	0	0	0	0
Relation	0	0	0	0	0	0
Rule	0	0	0	0	0	0
RuleMessage	0	0	0	0	0	0
RuleRedirect	0	0	0	0	0	0
Rules	0	0	0	0	5	0
Services	0	0	0	0	0	0
Timer	0	0	0	0	0	0
TimerDeltaSeconds	0	0	0	0	0	0
TimerDirection	0	0	0	0	0	0
TimerRules	0	0	0	0	0	0
Value	0	0	0	0	0	0
Weighting	0	0	0	0	0	0
XtensibleInfo	0	0	0	0	0	0

7 Annex 2 MVP XML and eViP LOM Specifications

7.1 The MVP specification

The data specification for the exchange and reuse of virtual patients used by the eViP profile originated from the 'MedBiquitous Virtual Patient' (or MVP) specification.
A document describing the structure of the XML schemas (version 0.48) within the MVP specification in detail is available here:
http://mvptools.googlecode.com/files/VirtualPatientSpecification_0.48_08May2008.pdf

7.2 eViP LOM

The eViP profile uses the Healthcare LOM (version 1.0) to describe the VP package.
LOM is one of the standards used by the SCORM reference model for interoperability of online learning content. LOM provides descriptive information about a learning object. Just as a label on a container provides information on what's inside, learning object metadata provides information on a learning module, including the title, author, description, keywords, educational objective, and other relevant information.
Healthcare LOM is based on and is a profile of the Institute of Electrical and Electronics Engineers (IEEE) 1484.12.1–2002 Standard for Learning Object Metadata (LOM) and the Extensible Markup Language (XML) Schema Definition Language Binding for Learning Object Metadata (IEEE 1484.12.3-2005) developed by the IEEE Learning Technology Standards Committee.

A document describing the Healthcare Learning Object Metadata (Healthcare LOM) in detail is available here:
http://mvptools.googlecode.com/files/HealthcareLOMSpecification1.0.pdf