



Terms and Conditions of Use of Digitised Theses from Trinity College Library Dublin

Copyright statement

All material supplied by Trinity College Library is protected by copyright (under the Copyright and Related Rights Act, 2000 as amended) and other relevant Intellectual Property Rights. By accessing and using a Digitised Thesis from Trinity College Library you acknowledge that all Intellectual Property Rights in any Works supplied are the sole and exclusive property of the copyright and/or other IPR holder. Specific copyright holders may not be explicitly identified. Use of materials from other sources within a thesis should not be construed as a claim over them.

A non-exclusive, non-transferable licence is hereby granted to those using or reproducing, in whole or in part, the material for valid purposes, providing the copyright owners are acknowledged using the normal conventions. Where specific permission to use material is required, this is identified and such permission must be sought from the copyright holder or agency cited.

Liability statement

By using a Digitised Thesis, I accept that Trinity College Dublin bears no legal responsibility for the accuracy, legality or comprehensiveness of materials contained within the thesis, and that Trinity College Dublin accepts no liability for indirect, consequential, or incidental, damages or losses arising from use of the thesis for whatever reason. Information located in a thesis may be subject to specific use constraints, details of which may not be explicitly described. It is the responsibility of potential and actual users to be aware of such constraints and to abide by them. By making use of material from a digitised thesis, you accept these copyright and disclaimer provisions. Where it is brought to the attention of Trinity College Library that there may be a breach of copyright or other restraint, it is the policy to withdraw or take down access to a thesis while the issue is being resolved.

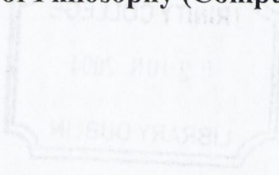
Access Agreement

By using a Digitised Thesis from Trinity College Library you are bound by the following Terms & Conditions. Please read them carefully.

I have read and I understand the following statement: All material supplied via a Digitised Thesis from Trinity College Library is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of a thesis is not permitted, except that material may be duplicated by you for your research use or for educational purposes in electronic or print form providing the copyright owners are acknowledged using the normal conventions. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone. This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

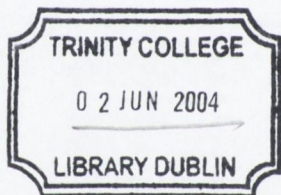
Towards Semantic Health Records

A thesis submitted to the
University of Dublin, Trinity College
for the degree of
Doctor of Philosophy (Computer Science).



Benjamin Jung,
Centre for Health Informatics,
Department of Computer Science,
Trinity College,
Dublin, Ireland.

October 2003.



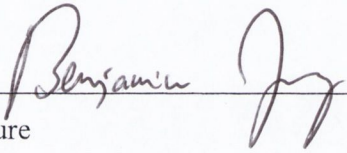
THOSIS
7986
7481

Declarations

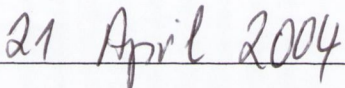
I, the undersigned, declare that this Thesis has not been previously submitted as an exercise for a degree to this or any other university.

I declare that all of the material contained in this Thesis, unless otherwise stated, is entirely my own work.

I agree that Trinity College Library may lend or copy this Thesis upon request.



Signature



Date

Dedication

To My First Child.

Acknowledgements

There are a number of people without whom this Thesis might never have been planned, researched, written and submitted and to whom I am greatly indebted:

This work would not have been possible without the efforts and enthusiasm of my Thesis supervisor, **Prof. Jane Grimson**. Throughout my time at Trinity College, she provided encouragement, constructive advice, academic supervision, good company, and lots of novel ideas. I would have been lost without her. Thank you!

I would like to thank all members of the **Centre for Health Informatics (CHI)** but in particular Gaye Stephens, who welcomed me on my first day and has continuously explained and taught the Irish way of life, including how to dance the Céilí. Paula has been another wonderful friend over the last years. I am sure I would have been less productive without her homemade buns for a late breakfast, even if it occasionally set off the fire alarm in the building. Special thanks also to the members of the **Dublin Health Informatics Group (DHIG)** comprising of members from the Dublin Institute of Technology, St. James's Hospital and Trinity College Dublin: Jesús (with whom I had some tough competitions not only on the bowling alleys), Damon (who hopefully enjoys the comfortable German coach), Bill (with whom I laughed about so many submarine stories) and Sebastien (who shared a passion for Italian cooking). There are many people I have met at numerous meetings of the **SynEx and Synapses consortia** and that I would like to thank for their fruitful discussions. I will not forget the excellent days that we had on and off the piste!

I am grateful to my many friends and colleagues in the **Knowledge and Data Engineering Group (KDEG)**: Cliff, Ken, Patrice, Sinead, Brian, John, Owen, Rita, Catherine, Dave, Steffen, Ruaidhri, Gordon, Catherine, Vinnie, Lucy, Mary and Declan. If I have forgotten anyone's name, I apologize. Your help was greatly appreciated.

I am most indebted to **Sonya**, my girlfriend, partner and hopefully wife-to-be (sic). I am amazed and thankful how patient and unselfish she handled this difficult and restless period of writing up the Thesis, while at the same time having so many worries and uncertainties herself.

I would also like to thank **my parents**, who supported me in all my decisions, in particular my desire to move to Ireland. I wish to thank my **Dublin friends** who made my stay so much more enjoyable, especially Deirdre (probably the best female soccer player in town) and Fergal (organiser of the annual Eurovision Song Contest nights).

Big thanks also to **LEO**¹, my constant companion over the past months who never got tired of recommending translations from German into English and vice versa.

Last but not least, I would like to remember the virtual patient **Barney Woods**, whose synonym and dummy medical data was used throughout the various project implementations. According to the records, he is the person with the poorest health in the country, but miraculously survived until the present day. Barney was a steady companion; may the CHI give him many more years to live!

Now that the end of this PhD is near, let me finish with a well known song². It expresses feelings very much alike to mine during those years at TCD. Although, I hope, I have not done things exclusively “my way”!

1. And *mf* $\frac{3}{4}$ C Em Gm6

now the end is near, and so I face the fi-nal
grets, I've had a few. But, then a gain, too few to
I've laughed and cried. I've had my fill, my share of

cur-tain.
men-tion.
los-ing.

My Way²

And now, the end is near;
And so I face the final curtain.
My friend, I'll say it clear,
I'll state my case, of which I'm certain.
I've lived a life that's full.
I've travelled each and ev'ry highway;
But more, much more than this,
I did it my way.
Regrets, I've had a few;
But then again, too few to mention.
I did what I had to do
and saw it through without exemption.
I planned each charted course;
Each careful step along the byway,
But more, much more than this,
I did it my way.

clear,
do,
side,

Yes, there were times, I'm sure you knew
When I bit off more than I could chew,
But through it all, when there was doubt,
I ate it up and spit it out.
I faced it all and I stood tall;
And did it my way.
I've loved, I've laughed and cried.
I've had my fill; my share of losing.
And now, as tears subside,
I find it all so amusing.
To think I did all that;
And may I say - not in a shy way,
"No, oh no not me,
I did it my way".
For what is a man, what has he got?
If not himself, then he has naught.
To say the things he truly feels;
And not the words of one who kneels.
The record shows I took the blows -
And did it my way!

case, of which
through with-out
all so a mus-ing. I o think I did all

Benjamin Jung, University of Dublin, Trinity College

October 2003

¹ <http://dict.leo.org/>

² "My Way" is the English version of the 1967 French song "Comme d'habitude" by Jacques Revaux, Gilles Thibault, Claude François. In 1968 Paul Anka wrote the English lyric and Frank Sinatra transformed it into a world hit.

Abstract

The benefit of information search and knowledge retrieval using the World Wide Web (WWW) has steadily declined in recent years due to the vast amount of unstructured data. The task of locating relevant information on the Internet has become increasingly difficult. Sophisticated search engines continuously index most of the online data but cannot readily handle situations where information is requested according to its semantic context. Information extracted without its semantic context can be difficult or even impossible to interpret, necessitating tedious and time consuming manual filtering of research results in order to locate the desired information.

However, web evangelists have proposed a solution to overcome these problems by defining an extension to the existing World Wide Web, namely the Semantic Web. The basic idea is that each information unit is given a well-defined meaning within its context by using metadata to further describe and classify the data. This allows autonomous agents to roam the Semantic Web, collect, understand, reason, evaluate and compare information in a way which is currently impossible.

So far, a small number of locally confined prototype implementations of the Semantic Web have been realised; a global pervasiveness is far from accomplished and some say that it will never happen. However, the potential benefits of the Semantic Web are real and it is well worth investigation how the concepts can be exploited in and adapted for information intensive domains such as healthcare.

This Thesis illustrates how Semantic Web concepts and technologies can be effectively applied in the process of developing Electronic Health Record (EHR) systems. The EHR is the entire collection of medical data relating to a single patient and covering the whole health history from cradle to grave. While there have been over four decades of research into the development of EHR systems, widespread adoption outside the individual healthcare organisations have always proved difficult. There are a number of technical challenges including the highly fragmented, heterogeneous and multimedia nature of health data and the lack of standards, although many of the barriers are of social and organisational type. In particular, it is proved difficult using existing approaches and technologies to meet, among others, four key EHR objectives, namely Maintainability, Extensibility, Reusability and Consistency (MERC).

The Semantic Web is a promising approach to overcoming existing obstacles and forms the basis towards a new concept – the Semantic Health Record (SHR).

This Thesis demonstrates that by using XML technologies, the SHR can go much further in meeting the MERC objectives than previous EHR approaches. It presents initial steps towards an integrated and multimedia EHR architecture, which will ultimately lead to the ubiquitously accessible, searchable and sharable SHR.

Publications related to this Ph.D.

- [1] BERRY, D., GRIMSON, W., GRIMSON, J., STEPHENS, G., FELTON, E., PERSANO, G., JUNG, B., KALRA, D., LLOYD, D., HURLEN, P., SKIJFELD, K. (1998) *Client Interfaces to Synapses Record Server using CORBA*: Conference Proceedings Object Management Group (OMG). Manchester, United Kingdom.
- [2] JUNG, B., GRIMSON, J. (1999) *eXtended Structured Query Language (xSQL)*: Conference Proceedings XML Europe 1999. Paris, France. pp. 315-324.
- [3] JUNG, B., GRIMSON, J. (1999) *Synapses/SynEx goes XML*: Conference Proceedings Medical Informatics in Europe (MIE) 1999. Ljubljana, Slovenia. pp. 906-911.
- [4] JUNG, B., BERRY, D., GRIMSON, J. (1999) *Using XML to Network Distributed Analytical Instruments – Back to the Future*: Proceedings of the 4th Annual Conference and Scientific Symposium of the Health Informatics Society of Ireland (HISI). Malahide, Ireland.
- [5] JUNG, B., GRIMSON, W., GRIMSON, J. (1999) *The EHCR – if not now, when?*: Conference Proceedings Towards an Electronic Healthcare Record in Europe (TEHRE) 1999. London, United Kingdom. pp. 51-55.
- [6] JUNG, B., ANDERSEN, E.P., GRIMSON, J. (2000) *Using XML for Seamless Integration of Distributed Electronic Patient Records*: Conference Proceedings XML Scandinavia 2000. Gothenburg, Sweden.
- [7] JUNG, B., ANDERSEN, E.P., GRIMSON, J. (2000) *SynExML as a vehicle for Electronic Patient Records – A status report from the SynEx project*: Conference Proceedings XML Europe 2000. Paris, France. pp. 339-346.
- [8] STEPHENS, G., JUNG, B., BRENNAN, B., PARDON, S., GRIMSON, J. (2000) *Using a Synapses Server in a Distributed Health Environment (DHE)*: Proceedings of the 5th Annual Conference and Scientific Symposium of the Health Informatics Society of Ireland (HISI). Saggart, Ireland.

- [9] JUNG, B., STEPHENS, G., GRIMSON, J. (2000) *XML based EPR Architectures: How do they relate to Synapses*: Conference Proceedings Towards an Electronic Healthcare Record in Europe (TEHRE) 2000. London, UK. pp. 79-92.
- [10] GRIMSON, J., STEPHENS, G., JUNG, B., GRIMSON, W., BERRY, D., PARDON, S. (2001) *Sharing Health-Care Records over the Internet*: IEEE Internet Computing, May/June 2001, pp. 49-58, IEEE Computer Society.
- [11] JUNG, B., MCKEOWN, J. (2001) *Adaptive Graphics*: Conference Proceedings XML 2001. Orlando, USA.
- [12] GRIMSON, W., JUNG, B., VAN MULLIGEN, E. M., VAN GINNEKEN, A., PARDON, S., SOTTILE, P.A. (2002) *Extensions to the HISA Standard – the SynEx computing environment*: Methods of Information in Medicine 05/2002; 41: 401-410, Schattauer GmbH - Verlag für Medizin und Naturwissenschaften.
- [13] JUNG, B., MCKEOWN, J. (2002) *Adaptive Graphics*: Conference Proceedings XML Europe 2002. Barcelona, Spain.
- [14] JUNG, B., NIXON, L.J.B. (2002) *Integrating multimedia components into a Semantic Web*: ERCIM Journal, No. 51, October 2002, pp. 38-39.
- [15] WANG, H., JUNG, B., AZUAJE, F.J., BLACK, N. (2003) *ecgML: Tools and Technologies for Multimedia ECG Presentation*: Proceedings XML Europe 2003. London, United Kingdom.
- [16] WANG, H., AZUAJE, F.J., JUNG, B., BLACK, N. (2003) *A markup language for electrocardiogram data acquisition and analysis (ecgML)*: BMC Medical Informatics and Decision Making, Vol. 3, Number 4.
- [17] LYNCH, P., JUNG, B. (2003) *Park And Go - An SVG Client Application*: Conference Proceedings SVG Open 2003. Vancouver, Canada.
- [18] JUNG, B. (2003) *XHTML and SVG: Publishing with concept*: Conference Proceedings SVG Open 2003. Vancouver, Canada.

- [19] JUNG, B., BROHAN, J., MAHER, R., O'SHEA, L.J., WADE, V. (2003) *Stirring XML: Visualisations in SVG*: Conference Proceedings SVG Open 2003. Vancouver, Canada.

Contents

Acknowledgements.....	iv
Abstract	vi
List of Tables	xvii
List of Figures.....	xviii
List of Code Sections.....	xx
Chapter 1. Introduction	1
1.1. Motivation and Justification	1
1.2. Goals and scope of the Thesis.....	2
1.3. Electronic Publishing with XML technologies.....	3
1.4. Research Approach and Contribution.....	3
1.5. Projects related to this Thesis	6
1.5.1. Synapses	6
1.5.2. SynEx.....	7
1.6. Thesis Organisation	7
Chapter 2. Background and Related Work	9
2.1. Introduction.....	9
2.2. Content Encoding	11
2.3. XML storage.....	14
2.4. XML Access and Selection.....	17
2.5. Query and Select.....	17
2.5.1. Document Object Model.....	17
2.5.2. Simple API for XML	18
2.5.3. Other API developments.....	18
2.5.4. xSQL.....	18

2.5.5. xSQL prototype implementation	19
2.5.6. XQuery	22
2.6. XML Security	23
2.6.1. Encryption Scenarios.....	25
2.7. Referencing XML Content	27
2.8. Data Transformation and Presentation	30
2.8.1. Plain text visualisation.....	31
2.8.2. Structural and markup visualisation	31
2.8.3. Visualisation through transformation.....	31
2.9. Semantic Web.....	33
2.9.1. Introduction	33
2.9.2. Expressing Meaning.....	34
2.9.3. Knowledge Representation.....	35
2.9.4. Ontology Definition	36
2.9.5. Relation between Semantic Web and Semantic Health Record.....	37
2.10. Summary	38
Chapter 3. Electronic Patient Record Architectures.....	39
3.1. Introduction	39
3.2. Electronic Health Records.....	40
3.2.1. Background	40
3.3. Terminology	41
3.3.1. Electronic Health Record	42
3.3.2. Alternative Terminology	43
3.4. Synapses EHR architecture	44
3.4.1. Introduction	44
3.4.2. Synapses Object Model (SynOM).....	46

3.4.3. Synapses Object Dictionary (SynOD).....	50
3.4.4. Assembling the record.....	51
3.5. Other Patient Record Architectures.....	52
3.5.1. HL7.....	52
3.6. CEN/TC251.....	53
3.7. Other Architectures.....	54
3.7.1. ASTM.....	54
3.8. Comparison.....	56
3.8.1. Synapses and CEN.....	56
3.8.2. Synapses and HL7.....	56
3.8.3. Synapses and ASTM.....	57
3.9. Summary.....	57
Chapter 4. Visual Modelling and Content Syntax.....	58
4.1. Introduction.....	58
4.2. Multi View Modelling.....	59
4.3. Record Structure Builder.....	60
4.3.1. Functionality.....	63
4.3.2. Internal model.....	65
4.3.3. Medical View.....	66
4.3.4. Technical View.....	67
4.4. Other software tools.....	68
4.5. The Syntax of Messaging Standards.....	69
4.5.1. EDI/EDIFACT.....	71
4.5.2. HL7.....	73
4.6. SynExML.....	75
4.6.1. The role of XML.....	75

4.6.2. Mapping SynOM classes.....	77
4.7. Comments.....	81
4.8. Summary	82
Chapter 5. Content Integration: Presentation/Exchange	83
5.1. Introduction	83
5.2. Background	84
5.2.1. Syntax transformations.....	85
5.2.2. Semantic Transformation	88
5.2.3. Combined Syntax/Semantic Transformation	90
5.3. Transformation with XSLT technology	90
5.4. SynEx	92
5.4.1. Server-side: XML Wrapper.....	93
5.4.2. WSDL, SOAP and UDDI.....	98
5.4.3. Client side: XML Presentation Engine.....	102
5.5. Equivalent use case.....	106
5.6. Comments.....	107
5.7. Summary	107
Chapter 6. Multimedia Component Integration into EHRs.....	109
6.1. Introduction	109
6.2. ASTM 1394-91.....	110
6.2.1. Background	110
6.2.2. Message Design.....	110
6.2.3. Scenario using EDIFACT syntax	111
6.2.4. Scenario using XML syntax.....	113
6.3. Multimedia Data Presentation: ecgML	116
6.3.1. Background	116

6.3.2. ecgML	117
6.3.3. ecgML visualisation	119
6.4. Imaging: DICOM-X	120
6.4.1. Background	120
6.4.2. DICOM syntax	121
6.4.3. DICOM-X	123
6.4.4. Comments.....	125
6.5. Summary	127
Chapter 7. Conclusions.....	128
7.1. Review of this Thesis	128
7.2. Summary of Thesis Contributions.....	130
7.3. Future Work	131
7.3.1. Content	131
7.3.2. Access: Web Services	132
7.3.3. Querying.....	133
Bibliography.....	135
Appendix	149
A. Typesetting Conventions.....	149
A.1. XML syntax components	149
A.2. Program Code.....	149
B. SynExML: Document Type Definition (ver. 2.2, GOLD release).....	149
C. SynExML: XML Schema (ver. 2.2, GOLD release).....	155
D. ASTM 1394-91: Document Type Definition	160
E. SynOM state charts of hierarchical behaviour	167
E.1. RecordFolder	167
E.2. FolderRIC.....	168

E.3.	ComRIC.....	168
E.4.	DataRIC.....	168
E.5.	RecordItem	169
F.	Synapses Server Interface Description.....	169
F.1.	Common Gateway Interface (CGI).....	169
F.2.	Web Services Definition Language (WSDL).....	170
G.	Synapses Consortium Members	174
H.	SynEx Consortium Members	176
I.	Acronym Glossary.....	177

List of Tables

Table 2-1: xSQL examples with equivalent SQL clauses	21
Table 2-2: XPath expression examples (select by hierarchy and predicate)	22
Table 4-1: EHR concept mapping	62
Table 4-2: HL7 segment table extract for Patient Identification (PID) Segment.....	73
Table 5-1: Mapping Levels: Human Communication vs. Computer Messaging	89
Table 6-1: ASTM 1394-91 message model.....	111
Table 6-2: IPv4 vs. IPv6 address space	115
Table F-1: Synapses Server: Request Record IDs (CGI).....	170
Table F-2: Synapses Server: Request Record Shape (CGI).....	170
Table F-3: Synapses Server: request ComRIC (CGI)	170

List of Figures

Figure 1-1: XML plausability among non-technical promotors.....	4
Figure 2-1: Electronic Publishing Architecture.....	10
Figure 2-2: XML vs. XML vocabulary	13
Figure 2-3: XML Encryption (Data Content).....	26
Figure 2-4: XML Encryption (Element Content).....	26
Figure 2-5: Classical vs. contemporary view of URIs, URLs, and URNs	28
Figure 2-6: LinkBase Architecture	30
Figure 2-7: XML Visualisations (plain text, structure, transformation).....	31
Figure 2-8: Server vs. Client side processing (XML-XSL).....	33
Figure 3-1: Component granularity (using the train analogy).....	45
Figure 3-2: SynOM Component Descriptions [GBG98].....	46
Figure 3-3: Class Hierarchy: Synapses Record Architecture	47
Figure 3-4: SynOM Aggregation.....	49
Figure 3-5: Assembling the EHR: The three-stage Synapses approach.....	51
Figure 4-1: Schema Development Interface (Technical and Medical View).....	59
Figure 4-2: Synapses Architecture.....	63
Figure 4-3: SynOM/SynOD datamodel (ER Diagram)	65
Figure 4-4: SynOD (Relational Model).....	66
Figure 4-5: Medical View in the RSB	67
Figure 4-6: Technical View in the RSB	68
Figure 4-7: EDIFACT syntax design (CED message segment)	72
Figure 4-8: LabInvestigation ComRIC (object view).....	78
Figure 5-1: Mapping scenarios with and without core agreed structures.....	84
Figure 5-2: Transformation Modelling with Transformation Hub.....	91

Figure 5-3: Server- vs. client side transformation and mapping of medical data.....	92
Figure 5-4: XML enabled Synapses Server.....	95
Figure 5-5: RetrieveRecordIDs (UML sequence diagram)	96
Figure 5-6: RetrieveRecordShape (UML sequence diagram)	97
Figure 5-7: RetrieveComRIC (UML sequence diagram)	98
Figure 5-8: Dublin Generic Synapses Client (GSC).....	104
Figure 5-9: Oslo web-client.....	105
Figure 5-10: Healthlink Architecture.....	106
Figure 6-1: ecgML single-source/multi-target transformations	119
Figure 6-2: DICOM Data Set	121
Figure 6-3: DICOM image in dedicated DICOM viewer (AccuLight).....	122
Figure 6-4: DICOM-X Visualisation.....	123
Figure 6-5: DICOM-X image layers.....	124
Figure E-1: RecordFolder state chart	167
Figure E-2: FolderRIC state chart	168
Figure E-3: ComRIC state chart	168
Figure E-4: DataRIC state chart	169
Figure E-5: RecordItem state chart.....	169

List of Code Sections

Code 2-1: SGML ELEMENT.....	11
Code 2-2: SGML ELEMENT Declaration.....	12
Code 2-3: Data-centric XML fragment	16
Code 2-4: Document-centric XML fragment	16
Code 2-5: URL with anchor reference (#phonenumbers).....	29
Code 2-6: URL with XPath expression reference (#id('phonenumbers')).....	29
Code 2-7: Simple RDF triplet ("Barney Woods is patient in St. James's Hospital")	35
Code 2-8: Simple RDF example using RDF/XML syntax.....	36
Code 4-1: CED message segment in XML syntax	73
Code 4-2: HL7 message.....	74
Code 4-3: HL7 message (see Code 4-2) translated into XML syntax.....	74
Code 4-4: LabInvestigation ComRIC (XML view).....	78
Code 4-5: DTD fragment to handle site-specific extension in SynExML.....	80
Code 5-1: Syntax Transformation on the basis of an XML document.....	86
Code 5-2: XSLT code for syntax transformation (based on syntax rules).....	87
Code 5-3: XSLT code for syntax transformation (based on semantic rules).....	88
Code 5-4: SynEx SOAP request.....	100
Code 5-5: SynEx SOAP response.....	101
Code 6-1: ASTM 1394-91 code	112
Code 6-2: ASTM 1394-91 code (XML).....	114
Code 6-3: ECG waveform data using EDIFACT syntax (HL7).....	117
Code 6-4: ecgML fragment.....	118
Code 6-5: DICOM Data Elements.....	121

Code A-1: Code Block Example 149

Code F-2: URL construct to access Synapses Record Server 169

Chapter	<h1>1</h1>
---------	------------

Introduction

Syntax and semantics are not separate enterprises – they go hand in hand. Every syntax should have a clear underlying semantics, and every semantics should have a syntactic form that clearly expresses its underlying model.

(Jonathan Robie) [Rob01]

1.1. Motivation and Justification

Extensible Markup Language (XML) is an open and flexible standard for storing, publishing and exchanging any kind of information. Using plain text files, XML content is marked up with special delimiters (“tags”) which structure and describe the content (“metadata”). Work on the early specification, unnamed at the time but provisionally titled by acronyms such as Minimal Generalized Markup Language (MGML), Simple Internet Markup Protocol Language (SIMPL) and XML [Con97], started in July 1996 by a few enthusiasts around Bray, Paoli and Speerberg-McQueen. A first working draft of the specification was issued in 1997. After only two years of forming the XML Working group, the first XML specification was officially published as a W3C recommendation [XML98]. A second edition (“XML 2e”) with minor changes was released in 2000 [XML00]. At the time of writing this Thesis, ideas are being discussed to merge fundamental parts of the core XML specifications into version 1.1 (and subsequently 2) of the XML specification, namely XML Namespaces, XML Include and XML Base.

In one of the earliest publicly available and promotional articles about XML and its future importance, the “father of XML” Jon Bosak describes XML as ideal for “*applications that require the Web client to mediate between two or more heterogeneous*

databases" [Bos98]. In particular, he illustrates a medical scenario, where a hospital receives computerised patient records to be included into the local healthcare information system. But instead of the traditional manual and tedious conversion of remote data, intelligent (XML based) mapping systems automatically execute the transformation between two or more heterogeneous systems. Nevertheless, he outlines the prime task and difficulty that has to be resolved before a successful exchange is possible: the definition of standard formats for seamless information exchange.

Theoretically speaking, the basic idea is straightforward, relatively easy to define and implement. In addition, data exchange between heterogeneous information systems is as old as the systems themselves and has been part of much research over the last two decades. After a more thorough and detailed analysis, the original task can be clearly separated into two distinct and independent key areas of further interest and investigation, namely transformation of syntactic and semantic structures.

1.2. Goals and scope of the Thesis

The goals of this Thesis are developments towards a new concept in relation to health records, the Semantic Health Record (SHR). In recent years, the management of medical data, in many respects very similar to managing information on the omnipresent World Wide Web (WWW), became increasingly difficult due to its heterogeneity and physical fragmentation. As the concept of the Semantic Web concept tries to solve this problem on a global scale, the Semantic Health Record architecture pursues an equivalent idea on a significantly smaller and restricted medical domain of Electronic Health Records (EHR).

Concepts of the Semantic Web have been proven successful and beneficial in overcoming problems related to Maintainability, Extensibility, Reusability and Consistency (MERC) of heterogeneous and highly fragmented data on the Web, including schemas and content. It is inevitable to successfully improve the quality of the MERC objectives in order to gain the promised advantages of EHRs by integrating information in a uniform way. However, apart from the MERC objectives, the medical domain and in particular EHR implementations have to deal with additional properties such as security (authentication, authorisation), reliability and availability.

The goal is to exploit effectively Semantic Web concepts for application in the EHR domain and allow for superior management of patient data.

The scope of this Thesis reaches from design to presentation issues related to the SHR. It starts with an investigation as well as the adaptation of one existing EHR model (Synapses) to an XML vocabulary to be used as a basis for the EHR, closely resembling a Semantic Web like architecture. New ideas to seamlessly incorporate multimedia components such as graphics and audio into the SHR are further important aspects and developments. Last but not least, potential routes to present and integrate remotely available SHR fragments into a local system are discussed.

All results emerging from the work undertaken are positioned within the intersection of two (at first sight unrelated) disciplines: Computer Science and Medicine. Nevertheless, Medical Informatics is a field of growing interest and importance and the focus of this Thesis is targeted at members of both groups. Therefore the author decided to describe medical aspects from a computer science perspective as well as computer science aspects from a medical perspective.

1.3. Electronic Publishing with XML technologies

The term **Electronic Publishing** is used throughout this Thesis in a media independent fashion, i.e. it does not solely reference the **textual** and traditional publishing process of documents and data. Instead, it includes publishing methodologies to generate alternative media formats such as **images, sound, animation** and **video**. An abstract publishing architecture together with the **XML family of languages**³ offers a solid base for a **single-source** and **multi-target** (especially multimedia target) publishing process. This model allows for transparent, independent and customer specific generation of data visualisations using, for instance, tabular text as well as graphical charts as representation formats.

1.4. Research Approach and Contribution

Together with the excitement of web technologies, XML and its related technologies have been described as the **silver bullet** that magically integrates all existing data sources into one easily accessible information pool. In particular marketing depart-

³ <http://www.cs.jyu.fi/~airi/xmlfamily.html>

ments promoted their new product releases most enthusiastically as **XML enabled**, pointing out that XML would be the simple solution for unsolved problems in the IT world, especially data exchange and integration. The marketing hype quickly went into overdrive and closely followed dot-com ex- as well as implosion (see **Figure 1-1**).

Today it is generally acknowledged that XML itself is not the silver bullet as it was seen by many. Nevertheless, XML is one component among many that does help and can ease information integration if exploited correctly [Mad01] [Fin00], often referred to as a “great pipe wrench” [UsGr98]. Advantages over existing data formats include the guarantee of a long life span for and accessibility of information due to its platform-, system- and vendor-independence. Furthermore, the possibility of domain specific and self-definable XML vocabulary definitions offers a rich and more intelligent document format and semantic. Last but not least, the availability of free XML applications to read, write and manipulate XML documents offers an improved and ubiquitous machine-machine communication [UsGr98] and making it the logical choice for documents.

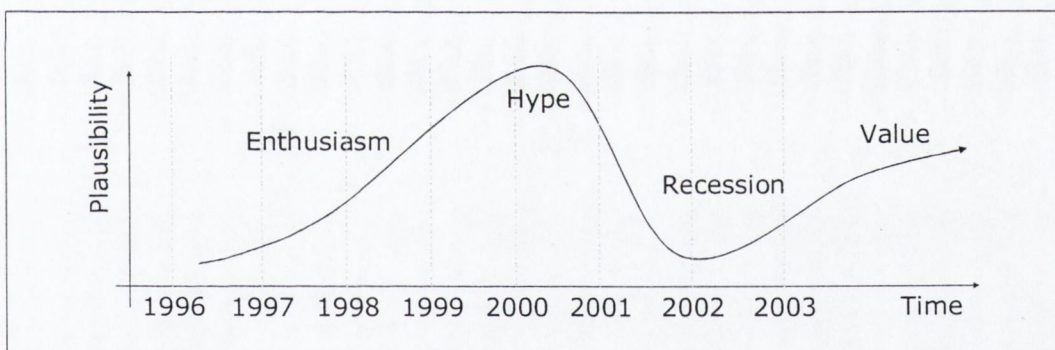


Figure 1-1: XML plausibility among non-technical promoters

The misconception and selling propaganda around XML lead to many difficulties during the research period of this Thesis. Being embedded into a multi-faceted research environment with close links to the medical as well as IT domain, rapidly changing specifications and early adaptations do not increase confidence in a new technology. The aggressive promotion of marketing departments of commercial institutions together with semi-professional technology pre-views lead to an unexpected reluctance to adopt the new technology by project members in the beginning. Later, marketing strategies paid off and healthy cynicism changed to an even more unexpected overeager and often thoughtless use. Nevertheless, the comprehension of

the real purpose and value of XML technologies (including its limiting aspects) led to a steep drop in marketing interest and a back to basics approach to academic research questions. Ongoing work is raising the level of acceptance and support to a final level with a clear understanding of its value and potential. It is finally widely accepted that XML *“must face many of the same challenges that plagued Electronic Data Interchange and database integration efforts of the past”* [Mad01] such as data heterogeneity, data mapping conflicts and lack of data description (metadata).

In addition to the technological difficulties, security policies in medicine lead to a lack of live (and possibly anonymous) patient data and delayed the evaluation process of the implementation. **Manually designed data** (*“dummy data”*) can only serve in an early implementation phase to demonstrate, analyse and highlight the various functions of an integrating system. In order to validate the findings, an appropriate amount of **real data** (*“live data”*) is needed.

This Thesis contributes in multiple ways to the ongoing research in the medical informatics as well as the pure IT domain:

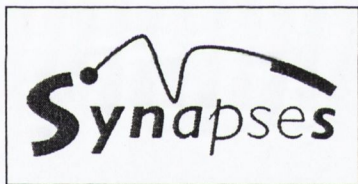
- It presents the state of the art in modularising XML based publishing processes with a focus on adapting methodologies to EHR modelling, exchange, presentation and integration.
- It illustrates the various approaches in (re-)defining EHR architectures with XML technologies as undertaken by standards organisations in recent months. How they relate and compare to the architectural design developed in projects such as Synapses (see Section 1.5.1) and SynEx (see Section 1.5.2) is also examined.
- It introduces and proposes a new concept – the Semantic Health Record (SHR) – to overcome typical obstacles of electronic health records such as high fragmentation, heterogeneity as well as its multimedia nature to meet four of the key EHR objectives namely Maintainability, Extensibility, Reusability and Consistency.
- It demonstrates the power of separating content and presentation information not only in the post-modelling phase (publishing), but also in the modelling phase, where for example structures of electronic patient records are designed.

- It defines the problem of heterogeneous model mapping and gives solution strategies as prototyped in the SynEx (see Section 1.5.2) project.
- It proves the feasibility of XML technologies to seamlessly integrate textual publishing with other multimedia publishing formats, such as graphics and audio, into a single electronic patient record.

Although XML technologies can help to improve and ease automatic data exchange between two heterogeneous information systems described as a deep integration strategy (see Chapter 5), the prerequisite of concise data description and model mapping is still inevitable. The XML specification [XML00] defines the **syntax** to describe ontologies and how to mark up data accordingly. Nevertheless, as mentioned in this chapter, XML is not the silver bullet that provides data exchange at your fingertips.

1.5. Projects related to this Thesis

1.5.1. Synapses

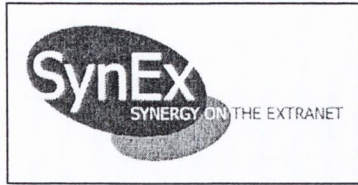


The Synapses Project was a three-year project funded under the European Union (EU) 4th Framework Health Telematics Programme. The consortium consisted of 26 partners from 14 different countries representing the health software industry sector, research institutes and universities as well as end-users through the participation of several hospitals (see Appendix G).

Synapses set out to solve problems of sharing data between autonomous information systems, by providing generic and open means to combine healthcare records or dossiers consistently, simply, comprehensibly and securely, whether the data passes within a single healthcare institution or between institutions [GrEtal96] [GBG98] [GGB98].

The author only joined the group of researchers, designers and implementers at Trinity College Dublin during the last months of the Synapses project and was therefore not involved in the important design and development phase of the project. However, he took part in various evaluation processes and helped to carry over the work into the successor project SynEx.

1.5.2. SynEx



The SynEx Project as a direct successor of Synapses Project was also funded under the EU 4th Framework Health Telematics Programme and assembled most of the previous partners and countries (see Appendix H).

The main goal of the two-year project was the integration of previously developed Synapses components with additional commercial modules into one Healthcare Information Systems Architecture (HISA) [HISA97].

The author of this Thesis has been actively involved in the research, design and implementation of various aspects of a patient record server as part of the SynEx project. He was in particular the driving force and evangelist behind the development of a graphical user interface (RSB, see Section 4.3) with Multi-View-Modelling (see Section 4.2) and XML functionality to support the patient record modelling process, the definition of the SynExML vocabulary to represent Electronic Patient Records based on the Synapses paradigm (SynExML, see Section 4.6) and the implementation of the generic Synapses web client (see Section 5.4.3).

1.6. Thesis Organisation

This Thesis is organised in the following way:

Chapter 2 deals with the technical and conceptual background as well as related research work undertaken by the author. Its outline follows the information processing model used in electronic publishing systems, i.e. starting from content definition and followed by topics such as storage, access and selection, transport, addressing and finally presentation. Aspects related to two of domains, namely **content definition** and **content presentation** will be explained in more detail in the following chapters.

The *next chapter* reviews the state of the art in declarative electronic patient record models. It starts with an explanation of the **Synapses paradigm** considering components such as object model (SynOM), object dictionary (SynOD) and the Record itself. This example is then compared to developments and standards released by international standards organisations including HL7, CEN/TC251 and ASTM.

Chapter 4 discusses history, features and cross-transformation of possible message and document **syntaxes**, including EDIFACT and XML. The second part deals with

one visual modelling technique and associated application (RSB) used to implement the Synapses model described in the previous chapter. Related to this, it finishes with a short excursion into data access using the Synapses Record Server.

Chapter 5 is dedicated to data exchange and information presentation. It starts with an explanation of **deep and shallow data integration** together with a prototype implementation in the SynEx framework for demonstration purposes. Advantages and disadvantages are pointed out by comparison to a similar installation (Healthlink). The most important concept related to XML technologies, i.e. the separation of content and presentation information as promoted in the previous chapters, demonstrates its power through the generation of application- and user-specific information presentation (customised, personalised).

Other important (multimedia) components and their integration into the Electronic Patient Record are discussed in *Chapter 6*. Use cases include the migration of an ASTM messaging standard for laboratory instruments from EDIFACT into XML syntax, an XML vocabulary to logically markup and present ECG data and finally a framework in which Scalable Vector Graphics (SVG) and its unique support is used for medical images in the Electronic Patient Record.

The *final chapter* summarises the main statements of this Thesis and indicates some considerations and directions for future work related to the topic, including a SynEx Web Service, an additional abstraction layer for the Synapses Record Server as well as video and 3D image annotation techniques using XML technologies, similar in design and implementation to the SVG image annotations described in the previous chapter.

Chapter

2

Background and Related Work

XML is an extremely versatile markup language, capable of labeling the information content of diverse data sources including structured and semi-structured documents, relational databases, and object repositories.

(Jonathan Robie) [Rob01]

2.1. Introduction

The possibility of separating content and presentation information is one of the key aspects in electronic publishing. Nevertheless, they are only start and end points in the overall process (“*From Content to Presentation*”) with a number of intermediate steps and areas that are important to address (see **Figure 2-1**). XML technologies are playing an ever increasing role in each of the various progression stages due to their simplicity, availability and wide application support. Additional XML philosophy and design principles include hierarchical data structures, embedded tagging and user-definable structures [UsGr98].

Standard Generalized Markup Language (SGML), a standard initially developed by the International Organisation for Standardisation (ISO) offering a common syntax for the publishing industry, specifies rules for the definition of a vocabulary (tag set or document markup language) rather than for a document language itself. Unfortunately, due to its flexibility in use and its syntactical varieties, SGML systems and applications became very difficult to maintain and affordable only to big publishing houses.

Being a simple subset of SGML, XML was primarily designed to add flexibility and extensibility to web publishing, but it was quickly realised that it can be used as a general syntax for documents of all kinds. XML is a simple but flexible way to create common information architectures, including the format as well as the data.

One prominent feature of XML based architectures is the differentiation between **document-centric** and **data-centric** markup. The former includes a high percentage of free-flow text or **mixed content** whereas the latter exclusively contains ELEMENTS with either text or element content, making it very similar to structures typically kept in databases.

Each level in the overall architecture, similar to the electronic publishing architecture depicted in **Figure 2-1**, plays a significant role in the idea of the **Semantic Web** [BHL01], which is the term ubiquitously used for data representation on the World Wide Web. But the basis of it all is the definition of presentation-free and content-only XML vocabularies. Combined with accurate meta-data (“*data about data*”) it makes automatic and autonomous interpretation and classification of information feasible. Data on the WWW might be meaningful to humans but is only effective and reliable to computer applications through the Semantic Web methodology.

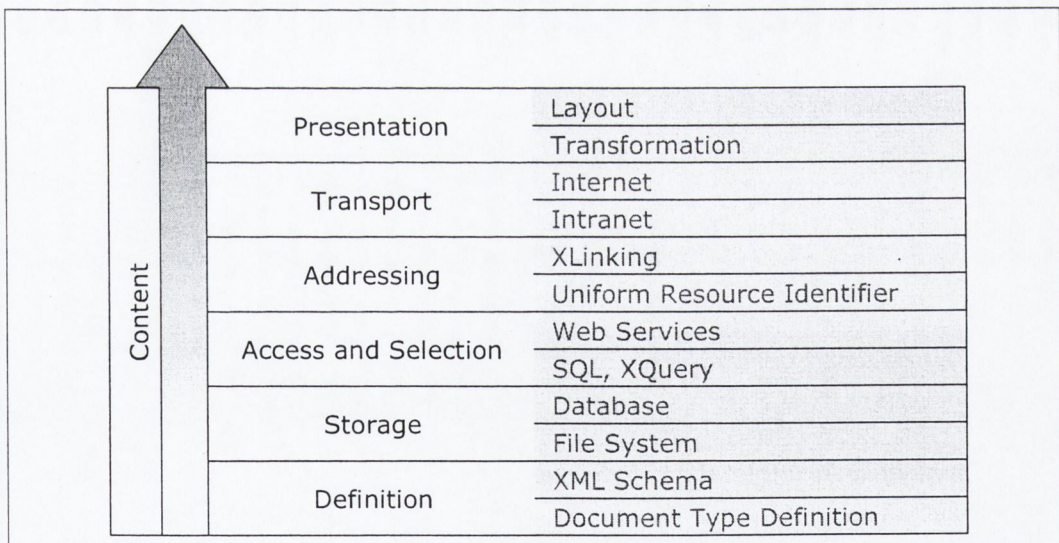


Figure 2-1: Electronic Publishing Architecture

Expressing meaning of data through meta-data is the foundation for knowledge representation on the web with ontologies providing a formal definition of relations between the various participating entities. This prerequisite is the basis for an often described and referenced scenario: Intelligent agents are independently surfing the

WWW, automatically gathering information, reasoning and evaluating the content and offering real knowledge to the user, which otherwise would have been inaccessible.

2.2. Content Encoding

The origins of content encoding in electronic documents can be traced back to the late 1960s when publishing companies started to use generic descriptive markers such as Heading and Paragraph. This variation made the document markup different from previous approaches in two noteworthy ways. Firstly, they did not use abstract symbols (such as a tab or linefeed characters) anymore to define fields and separate content in the document. Instead, a simple tag marked the beginning and the end of each ELEMENT. Secondly, they abandoned specific formatting markers (Format-1, Format-2) in favour of descriptive markers to identify the logical parts of the document. This was the start of conceptually separating content from presentation information, first described by William Tunnicliffe [Gol91]. At around the same time, IBM employee Charles Goldfarb together with Edward Mosher and Raymond Lorie invented the Generalized Markup Language (GML) in order to share documents between text editing, formatting, and information retrieval subsystems. GML was based on the same generic coding concept of separating content and presentation information. However, Goldfarb continued to extend this concept with ideas already known from hypertext documents [McA99]. These include short references, e.g. quotation-signs " or ' used as markup for quotations, link processes to instantiate document types, and concurrent document types. This work led in 1980 to a first working draft SGML specification, which over the next 6 years, resulted in the ISO 8879:1986 standard [ISO8879].

SGML is a very powerful platform-, system-, vendor- and version-independent meta-language, which is used to define document vocabularies. In principle, SGML standard describes the actual markup that is used in the document such as the use of the less-than sign \leq and the greater-than sign \geq together with a name to identify the start of a document Title as shown in **Code 2-1**:

```
1 <Title>History of the Electronic Health Record</Title>
```

Code 2-1: SGML ELEMENT

Furthermore it defines the syntax for rules to specify a class of documents (vocabulary, DTD). For example a `Chapter` ELEMENT must contain a nested single `Title` ELEMENT followed by at least one `Paragraph` ELEMENT as illustrated in **Code 2-2**:

```
1 <!ELEMENT Chapter - - (Title, Paragraph+)>
```

Code 2-2: SGML ELEMENT Declaration

Each single SGML document is one instance of the document class and can be validated using the rules in the Document Type Definition (DTD). The most important and widest used SGML vocabulary at the moment is the Hypertext Markup Language (HTML), the de facto standard for publishing content on the WWW. A first proposal⁴ was formulated by Tim Berners-Lee in 1989 when he worked as a researcher at the *Conseil Européen pour la Recherche Nucléaire* (European Laboratory for Particle Physics, CERN) and tried to persuade the management that a global hypertext system was in CERN's interests.

Often the distinction between markup syntax and vocabulary is not made precisely. Especially statements referring to “*the advantage of SGML over HTML*” clearly show misunderstanding and frequently cause confusion. HTML is an SGML vocabulary, i.e. uses SGML syntax and relates to SGML as “*apples relate to fruit*”.

As generic SGML had proved too complex for the web, HTML visibly showed its limitations as soon as the commercial world discovered the web. Browser manufacturers tried to overcome restrictions by adding proprietary functionality, i.e. indirectly extending HTML, which finally led to the time of “browser war”.

In the middle of the 1990s, a small group of people lead by Jon Bosak created a simple subset⁵ of the SGML standard to overcome its shortcomings and bring SGML functionality to the Internet. Similar to SGML, XML is a meta-language to define document markup vocabularies and presents methods to separate content from presentation. It retains the powerful features of SGML, such as extensibility, structure, validation, but ignores the complex features which simplifies and improves use and implementation.

⁴ <http://www.w3.org/History/1989/proposal>

⁵ For differences see <http://www.w3.org/TR/NOTE-sgml-xml-971215>.

Ten design goals were specified which laid the foundation for the development of XML [XML00]:

1. XML shall be straightforwardly usable over the Internet.
2. XML shall support a wide variety of applications.
3. XML shall be compatible with SGML.
4. It shall be easy to write programs that process XML documents.
5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
6. XML documents should be human-legible and reasonably clear.
7. The XML design should be prepared quickly.
8. The design of XML shall be formal and concise.
9. XML documents shall be easy to create.
10. Terseness in XML mark-up is of minimal importance.

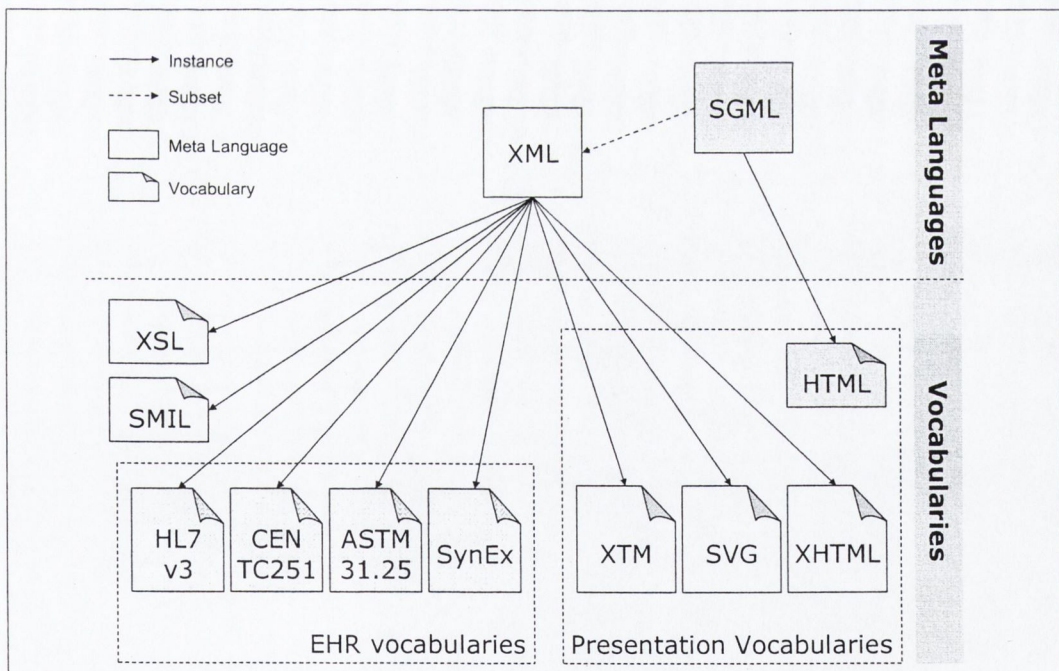


Figure 2-2: XML vs. XML vocabulary

Surprisingly, what had been a misunderstanding in the times of SGML arose again and questions such as whether “XML is the better solution over XHTML” (web publishing) or “HL7 should be favoured over XML” (medical informatics) were com-

monly asked at conferences and seminars. **Figure 2-2** shows the two distinctive layers of meta-languages and derived vocabularies which can be further categorised into EHR modelling vocabularies and presentation vocabularies. The difference is that presentation vocabularies have dedicated viewers, i.e. these vocabularies should solely be used for data visualisation purposes.

To summarise, three important objectives⁶ have been followed throughout the development of meta-languages from GML and SGML until XML:

- The notion of separating content and structure encoding from specifications for (print) processing,
- The notion of using names for markup ELEMENTS which identified text objects descriptively,
- The notion of using a (formal) grammar to model structural relationships between encoded text objects.

2.3. XML storage

XML documents can be classified as simple databases but “*only in the strictest sense of the term that is a collection of data*” [Bou03]. They are more accurately described as rich data collections which (by the nature of XML) include features such as self-describing structure and names as well as portability using Unicode, a system for “*the interchange, processing, and display of the written texts of the diverse languages of the modern world*”⁷. On the other hand, verbosity and data replication (caused by the hierarchical nature of XML when compared to the relational database model) indicates a serious weakness in XML documents when compared to databases. Techniques similar to normalisation, which would decrease verbosity and replication, are not always easy to define and maintain. Basically, XML documents are not enclosed by a data management system as databases are, e.g. Relational Database Management System (RDBMS), Object Oriented Database Management System (OODBMS), but specifications from the XML family of languages do often show an equivalent but simpler functionality:

⁶ <http://xml.coverpages.org/general.html#hist>

⁷ <http://www.unicode.org/>

- Storage is made available by the XML document itself.
- Data Definition Language (DDL) correspond to schemas such as DTD [XML00], XML Schema [XSD01a] [XSD01b] and RELAX-NG [RNG01].
- Data Manipulation Language (DML) is closely related to query languages (e.g. XQuery [XQL03], XPath [XPath99] and QUILT [CRF00]).
- Programming interfaces are resembled by Simple API for XML (SAX)⁸, Document Object Model (DOM)⁹ and Java Document Object Model (JDOM)¹⁰ interfaces.

But there are a number of advanced DBMS tasks that do not have corresponding functions in the XML world, including efficient storage, indexes, security, transactions and data integrity, multi-user access, triggers and queries across multiple documents [Bou03].

File systems are mainly used to store small sized XML documents with little consideration of performance and data integrity issues. But in order to utilise the above mentioned advanced DBMS functions, XML documents can be stored in databases with two options available: XML enabled databases as well as native XML databases. Native XML database systems are designed, optimised and implemented for the single purpose of maintaining XML documents. On the other hand, XML enabled database systems are traditional database systems (e.g. RDBMS, OODBMS) that are modified to handle XML documents. An additional layer in the architecture (XML interface) is responsible for importing and exporting data to and from the underlying database transparently to the user.

The ideal choice is very much dependent on the type of XML document, namely data- or document-centric. The term data-centric relates to XML documents where all ELEMENTS have exclusively zero-or-more ELEMENT-nodes or exactly one text-node as children. They show a reasonably recurring structure of fine grained data, the element order is of minor significance and their main purpose is portability and data exchange. The example in **Code 2-3** illustrates an XML fragment with the ELEMENT

⁸ <http://www.saxproject.org/>

⁹ <http://www.w3.org/DOM/DOMTR>

¹⁰ <http://www.jdom.org/>

`<biblioentry>` containing two child ELEMENTS (`<abbrev>`, `<bibliomisc>`), each containing exactly one text-node (content).

```

1 <biblioentry id="Spe01">
2   <abbrev>Spe01</abbrev>
3   <bibliomisc>
4     Spence, R. (2001) Information Visualization, Addison Wesley.
5   </bibliomisc>
6 </biblioentry>
```

Code 2-3: Data-centric XML fragment

Elements in document-centric documents on the other hand are made by humans and for human interpretation. Each ELEMENT contains zero-or-more text- or ELEMENT-nodes in any possible combination (mixed content). **Code 2-4** gives an example with the `<para>` ELEMENT containing two text-nodes separated by the empty-ELEMENT-node `<xref>`.

```

1 <para>
2   This aggregation, combined with the radial layout, creates the
3   smooth focus&plus;context
4   <xref linkend="Spe01"/>
5   effect. This enables users to view details of a localized region
6   without losing the context of the overall functional space.
7 </para>
```

Code 2-4: Document-centric XML fragment

As mentioned earlier, XML enabled databases are a combination of an existing traditional DBMS plus a wrapping XML layer that manages the schema mapping of hierarchical XML structures into the Relational Model. Many database manufacturers already equip their products with integrated XML wrappers; alternatively there are also a number of third-party middleware applications available which provide a similar functionality. In general, XML enabled databases are preferably used to store data-centric XML documents. The insignificance of sibling element order as well as the fine-grained and regular structure offers an easy to define and lossless mapping between the two architectures.

However, native XML databases are becoming increasingly popular for storing document-centric XML documents. Here, the order of sibling ELEMENTS is of significance and storing an XML document as a complete entity rather than splitting it over a number of tables improves accessibility and efficiency.

2.4. XML Access and Selection

Access to XML documents as well as selection and retrieval of XML fragments are the third level in the XML publishing architecture (see **Figure 2-1**). Two standard Application Programming Interfaces (API) based on fundamentally different methodologies (processing models) exist to process and validate XML documents: DOM and SAX. Both offer functionality to dynamically manipulate data in XML documents, e.g. add, edit and delete content and structure as well as navigate through the document to a varying degree. Both APIs are equally supported by most of the XML parsers, applications that read a sequential source, break it down into components and validate the result against a control model such as a DTD or XML schema.

2.5. Query and Select

Querying and selecting fragments has been one of the earliest requirements for XML documents; nevertheless the XML Query Language (XQuery) [XQL03] (still only W3C Working Draft) only made it into the XML family of languages recently. The road to agree on a widely accepted standard has been long. Over a period of many years (and this might sound like an eternity compared to some W3C developments among the XML family of languages), many approaches have been described and defined, including xSQL (see Section 2.5.4), XQL [Rob03], XML-QL [Detal98], Quilt [CRF00], SQL, and OQL. All of them included important aspects and contributed to the development of XQuery.

2.5.1. Document Object Model

The DOM is a *“platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents.”* [DOM98]. The document can be further processed and the results of that processing can be incorporated back into the presented page. The whole document has to be completely parsed, constructing a treelike structure in memory, before it can be accessed. After finishing the parsing process, each XML ELEMENT is exposed as a regular program object. Equivalent to object attributes, XML ELEMENT attributes and content are accessible via DOM methods which control the manipulation of and access to the document. Access to ELEMENTS based on hierarchical conditions such as a parent-child relationship as well as random access is proprietary to the DOM but not supported by SAX (see Section 2.5.2).

2.5.2. Simple API for XML

SAX is an event-based API that reports parsing events directly to an application through callbacks. For instance, during the parsing process, specific methods are called at the start (`startElement()`) and end (`endElement()`) of every XML ELEMENT and entities such as ELEMENT name and attributes are passed on as parameters. Compared to DOM, SAX is a member of the group of streaming APIs, which do not create an internal structure nor keep the document in memory. Its decreased use of system resources makes it an ideal API for large documents and the only choice for continuously streamed XML data.

SAX has become a de facto standard for processing and validating XML documents due to its streaming, speed and memory-friendly characteristics..

2.5.3. Other API developments

Two other APIs, better classified as generic meta-APIs, are of primary interest, Java Document Object Model (JDOM) and Java API for XML Processing (JAXP).

JDOM is a simplified API for efficient reading, manipulation, and writing of XML documents and XML data, especially trimmed for use in Java applications. It is a combination of the best features of DOM and SAX and integrates well with both. Compared to DOM, it is easier to use and performs faster due to optimised memory management. Its advances over SAX include random access to XML data and improved manipulation and output of the XML data. In addition access to methods proprietary to JDOM, support of the underlying DOM and SAX methods is still available.

JAXP is a pluggable abstraction layer over the two standard APIs, providing vendor independence of parsers, which also allows easy swapping of parser implementations.

2.5.4. xSQL

In the early days of XML, the availability of query applications specifically designed to work with XML documents was limited. This was the complete opposite to Extensible Stylesheet Language Transformations (XSLT), an XML vocabulary to transform XML documents for display in a browser, as the XSLT specification as well as applications have been released nearly simultaneously to XML itself. However, us-

ing the query mechanisms in XSLT (i.e. XPath), it soon became obvious that this technology would not be sufficient to define powerful queries such as those which can be expressed in SQL. It was important to investigate, define and implement a simple prototype that would help to promote the power of XML.

SQL is widely used, easy to learn and has a very clear syntax; these three features influenced the decision to use it as the basis of a query language and extend it with additional and more XML-specific functionality. The main focus of this work is on expanding the existing SQL syntax to apply it to the conditional selection of sub-trees and ELEMENTS from, and the merging (join and union) of pre-existing XML documents.

2.5.5. xSQL prototype implementation

Conditional selection of document fragments and conditional merging of two or more existing XML documents were the two main requirements for the xSQL engine. The prototype application extracts document fragments based on conditions which are defined by the user. It not only provided easy access to a collection of ELEMENTS with common ATTRIBUTES or tag-names but was also successful in creating a subset of the source XML document by means of ELEMENT filtering. Creating application specific data sets, filtering of relevant data or automatic statistical evaluation were among the expected scenarios. The second aspect, merging of two or more XML documents, provided the power to store data (similar to an RDBMS) in different files including their inter-ELEMENT relations. It reduced the risk of inconsistencies caused by data replication as well as maintenance time. Another advantage was the opportunity to merge locally as well as globally distributed document fragments. This is especially interesting in the healthcare area, where distributed Electronic Health Records are also becoming more and more geographically fragmented.

Information Viewpoint

The first step in designing an xSQL query mechanism was defining the connections between the main components in the model, namely storage, schema, access (query), transformation and presentation [JuGr99a]. The XML documents were retrieved from both the local hard disk and over the network and kept in memory as a Document Object Model (DOM) and stored in the underlying database. The xSQL engine resided within the application and automatically created the index tables for XML

document fragments. The xSQL query engine retrieved queries from the user and mapped them to the database internal query language. The result set was passed back to the xSQL engine, requesting specified ELEMENTS from the DOM and creating a hierarchical result document. This document was also stored and indexed in the database, allowing it as an input for further queries, similar to nested queries in SQL.

Computation Viewpoint

To allow for rigid synchronisation between the XML DOM in the application and the index table in the database a unique element/node identifier was defined. A string, concatenating the parent node unique identifier and its actual position within its siblings, was chosen to produce this “primary key”, e.g. 1. for the root node and 1.4. for the fourth child of the root node. These unique identifiers are stored together with the element information in the index table of the database.

The maintenance of this additional element identifier is easy and does not require many additional resources. The following methods were implemented:

Add a new node/element

The inclusion of a new element requires the following two operations. Firstly, all level-numbers of the elements level contained within the unique identifier of all siblings and their children following the new element must be increased by one. Then, the new element can be included at the chosen location.

Delete one node/element

Delete the intended element and all its children. Then decrease the level-numbers in the unique identifier of all the siblings, which followed the deleted element and all of their children by one.

Move one node/element

This is simply a combination of the previous two operations, firstly a deletion of the selected element and then its inclusion at the specified location.

This structure also gives the ability to check simply whether an ELEMENT A exists within the sub-tree of another ELEMENT B. If B's unique identifier is a prefix of A's identifier then A is a child of B, otherwise it is not.

The mapping from xSQL to SQL was implemented in the following way. xSQL-SELECT-clause parameters as well as xSQL-WHERE-clause parameters were mapped into SQL-WHERE-clause parameters. The SQL-SELECT-clause-parameter is always the uniqueID in order to retrieve the corresponding location within the DOM. xSQL-FROM-clause parameters are copied without change into SQL-FROM-clause parameters in the latest implementation.

Table 2-1 shows some examples of the xSQL to SQL mapping supported by the prototype xSQL engine.

Table 2-1: xSQL examples with equivalent SQL clauses

<u>xSQL</u>	<u>SELECT</u> <u>Diagnosis</u> <u>FROM</u> <u>ICURecord</u>
<u>SQL</u>	<u>SELECT</u> <u>uniqueID</u> <u>FROM</u> <u>ICURecord</u> <u>WHERE</u> <u>Elements.Name='Diagnosis'</u>
<u>xSQL</u>	<u>SELECT</u> <u>Diagnosis</u> <u>FROM</u> <u>ICURecord</u> <u>WHERE</u> <u>Diagnosis.content='Fever'</u>
<u>SQL</u>	<u>SELECT</u> <u>uniqueID</u> <u>FROM</u> <u>ICURecord</u> <u>WHERE</u> <u>Elements.Name='Diagnosis'</u> <u>AND</u> <u>Elements.Value='Fever'</u>
<u>xSQL</u>	<u>SELECT</u> <u>Diagnosis</u> <u>FROM</u> <u>ICURecord</u> <u>WHERE</u> <u>Diagnosis.attribute(Person)='Ron'</u>
<u>SQL</u>	<u>SELECT</u> <u>uniqueID</u> <u>FROM</u> <u>ICURecord</u> <u>WHERE</u> <u>Elements.Name='Diagnosis'</u> <u>AND</u> <u> ((Attributes.Name='Person')AND</u> <u> (Attributes.Value='Ron'))</u>
<u>xSQL</u>	<u>SELECT</u> <u>Diagnosis</u> <u>FROM</u> <u>ICURecord</u> <u>WHERE</u> <u>Diagnosis.content='Fever'</u> <u>OR</u> <u> Diagnosis.content='Cough'</u>
<u>SQL</u>	<u>SELECT</u> <u>uniqueID</u> <u>FROM</u> <u>ICURecord</u> <u>WHERE</u> <u>Elements.Name='Diagnosis'</u> <u>AND</u> <u> ((Elements.Name='Diagnosis') AND</u> <u> (Elements.Value='Fever'))</u> <u>OR</u> <u> ((Elements.Name='Diagnosis') AND</u> <u> (Elements.Value='Cough'))</u>

2.5.6. XQuery

XQuery is a superset of the XPath [XPath99] expression language. XPath was designed to address and reference parts of an XML document based on hierarchical structure or predicates (explicit conditions) (see **Table 2-2**). It operates on the abstract, logical structure of an XML document, rather than its surface syntax. Besides the selecting functionality, it also provides basic facilities for manipulation of strings, numbers and booleans.

Table 2-2: XPath expression examples (select by hierarchy and predicate)

	XPath expression	Description
Hierarchy (1)	<code>chapter//para</code>	Select all para elements that are descendant of the chapter element.
Hierarchy (2)	<code>chapter/para/title</code>	Select the title element that is child of para which itself is a child of the chapter element.
Predicate (1)	<code>para[position()=1]</code>	Select the para element that's at the first position among all para child elements.
Predicate (2)	<code>para[@type="intro"]</code>	Select the para element that contains an attribute with the name 'type' and value 'intro'.
Predicate (3)	<code>chapter[title]</code>	Select the chapter element that contains a child element with the name 'title'.

However, a query language for XML documents must provide more versatile features than just simply select and return XML document fragments as XPath does. In particular, a query language must be able to reformat and restructure result sets according to rules set in the query.

Compared to relational result sets in RDBMS, XQuery returns a collection of XML fragments, i.e. a set of hierarchical structures, and offers variability on the sort key (field) location. Element content and attribute values (or parts thereof) are taken as key information, generally from the immediate element, but any other element down in the hierarchy can be the source.

In addition, result sets might have to undergo a **structural transformation** before being returned to the querying application. This technique implies a programmatic

reorganisation of the XML fragment including modifications in the hierarchical element order and component changes from element to attribute or vice versa. On the other hand, **structural preservation** is required to preserve the relative hierarchy and sequence of input document structures in query results. For instance, in creating a Table of Contents, the support of structural preservation is necessary to select all chapter and section headings, without losing the hierarchical relationship between them. The XML specification presents two key attribute types to define **references** within the document, namely ID for identifier and IDREF for references to the identifier. This is similar to the structure of primary keys and foreign keys in RDBMS. They identify tuples and represent relationships; XQuery must similarly support the ID/IDREF construct without restrictions.

2.6. XML Security

Document encryption, the translation of data into secret code, is an integral part of many modern information systems, not only in medicine. The W3C issued two recommendations dealing with XML Security, namely XML Encryption [XEnc02] to encrypt XML documents and document fragments and XML signature [XSig02] to sign XML documents similar to the Public Key - Private Key methodology.

Numerous encryption methods and algorithms are available, varying in type (symmetric, asymmetric), key-size, speed and reliability. Besides, data security through encryption is separate from transport level security, such as HTTP using Secure Socket Layer (HTTPS), which encrypts and decrypts web content on the server as well as the client before sending it over the network. Both encryption algorithms and transport level security are beyond the scope of this Thesis. However, it is interesting to note the difference that XML makes in terms of document fragment encoding mechanisms.

The traditional way of encrypting documents is based on the simple fact that a document is a single, non-divisible entity, e.g. a file. To allow fragment encryption, the original content has to be dissected, separately encrypted and merged together into an archival file. But more importantly, this “all-or-nothing” technique does make the structural components of the document inaccessible as it does not differentiate between structure and content. As an example, documents generated by word-

processors do not let you encrypt each paragraph that exists under first-level headings; similarly, a result set returned from a DBMS cannot be partially encrypted.

In contrast, XML documents open a new approach to document encryption due to its plain text format, hierarchical data structure and distinct separation of content and structure. Suddenly, the limitation of “all-or-nothing” document encryption vanishes and fragment encryption becomes feasible. Even encrypting fragments with different keys within the same document is possible. This allows interesting constructs such as fragments within the same document being encrypted with different keys for different groups or people. Furthermore, because the encrypted content is embedded into another XML document, repeated encryption of the same fragment with varying keys is possible thus decreasing the chance of unauthorised decryption. This methodology is especially useful in the medical domain, where different user groups (e.g. doctor, nurse, and pharmacist) have different privileges and access rights to different parts of the record. More examples of the various possibilities are given in Section 2.6.1.

Encrypting XML documents or XML document fragments requires an additional step to convert the content into its canonical form. In other words, content has to be consistently serialised into an octet stream in order to avoid inconsistencies such as missing default attributes and their values. An XML document in canonical form meets the following criteria [CXML01]:

- The document is encoded in UTF-8 (Universal Transformation Format), a Unicode character set [RFC2279]. During the transformation process all Unicode characters are encoded into a variable length of bytes. This procedure assures that the Unicode characters corresponding to the familiar ASCII set will have the same byte values as ASCII and for example full text searches can be executed on the plain file.
- Both XML declaration and document type declaration (DTD) are removed from the XML document.
- Line breaks are normalised to `#xA` (UNIX notation) and character as well as parsed entity references are replaced.
- ATTRIBUTE values are normalised, including the use of double quotes for ATTRIBUTE value delimiters, insertion of default attributes to each ELEMENT and

replacement of special characters in ATTRIBUTE values and character content by their corresponding character references.

- CDATA sections are replaced with their character content.
- Empty ELEMENTS are converted to start-end tag pairs (`
</br>`).
- Whitespace¹¹ outside of the document ELEMENT and within start and end tags is normalised, i.e. consecutive whitespace is reduced to a single space. All whitespace in character content is retained (excluding characters removed during line feed normalization).
- Unnecessary namespace declarations are deleted from ELEMENTS.
- Lexicographic order is imposed on the namespace declarations and ATTRIBUTES of each ELEMENT

2.6.1. Encryption Scenarios

Figure 2-1 shows three different XML document fragments with graphical Tree-View representations above them. The original plain and unencrypted fragment is depicted on the far left side, containing Patient information such as Name, Address and date of birth (DOB) with the Address split into City and Country fields. This excerpt is part of a data-centric XML document with ELEMENTS surrounding either content without further markup or solely child elements (see Section 2.2).

The example in the middle reflects the state in which the textual content of the City ELEMENT is encrypted, leaving the structural information exposed, i.e. the existence of a City ELEMENT is visible as well as possible ATTRIBUTES contained within the start-tag. The XML code clearly shows that even the encrypted content of the City ELEMENT is valid XML and therefore does not break the validity of the embedding XML document. In the right example, content as well as structure of the City ELEMENT are encrypted; it only discloses the existence of data but hides its type i.e. ELEMENT with attached ATTRIBUTES and content, simple PCDATA or mixed content (see Section 2.2).

¹¹ Any contiguous sequence of spaces, tabs, carriage returns, and/or line feeds. (<http://dict.die.net/>).

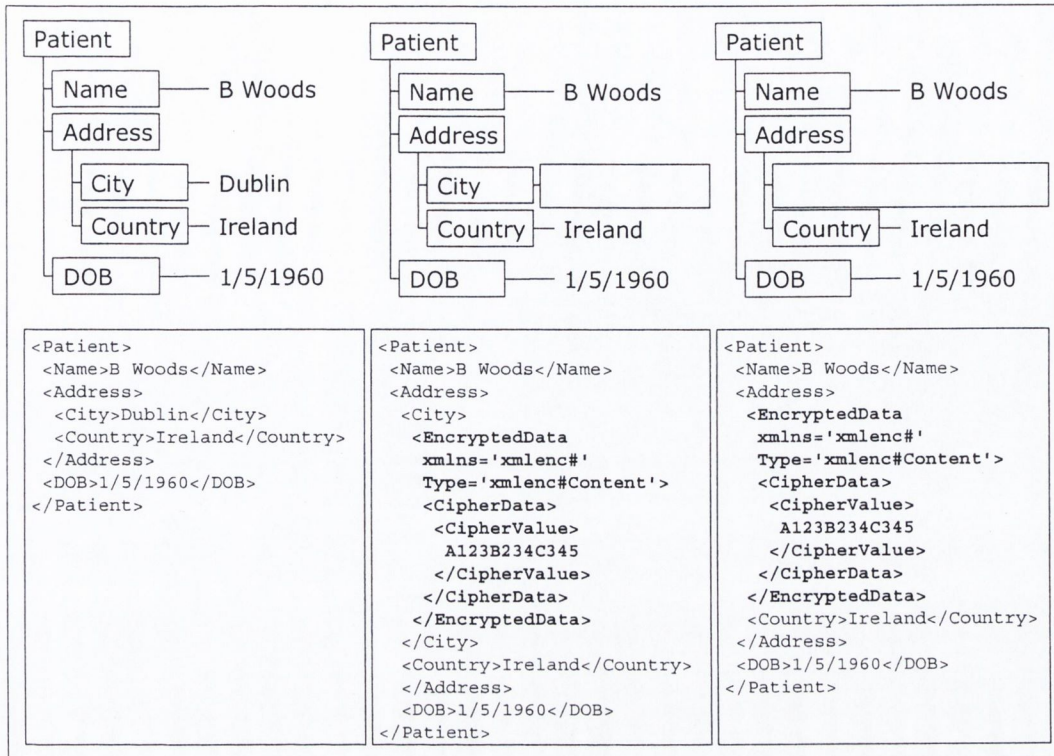


Figure 2-3: XML Encryption (Data Content)

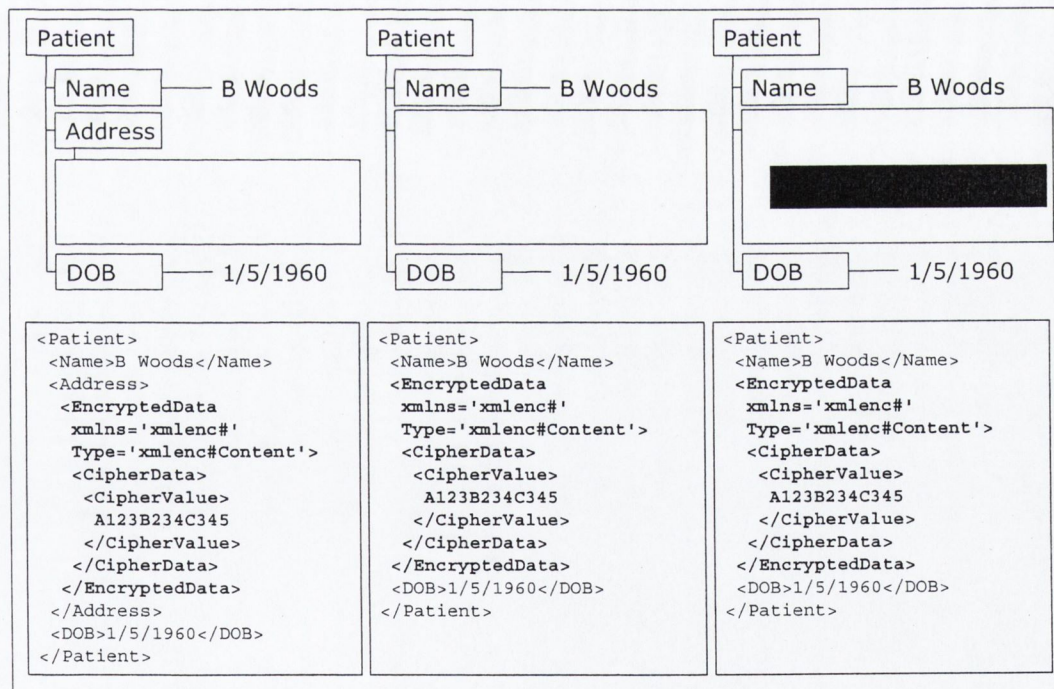


Figure 2-4: XML Encryption (Element Content)

Different types of encrypting elements with data content are shown in the previous scenarios. **Figure 2-4** illustrates equivalent situations with elements enclosing element content. On the left side, all content contained within the Address element is

encrypted. Nevertheless, the existence of the Address element as well as its potential attributes is still visible and accessible. A complete encryption of the Address element is shown in the middle, giving only evidence of the existence of content without revealing its type and value. Finally, the example on the right gives a picture of an XML fragment with double encryption. Firstly, the City ELEMENT was completely encrypted followed by another encryption of the Address ELEMENT, including the already encrypted City ELEMENT.

More details related to XML Signatures and asynchronous document encryption, for example Public Key – Private Key encryption, can be found in [XSig02].

2.7. Referencing XML Content

Peter Stein [Ste66] claims that the earliest accurate reference to a distinct piece of content was made in the late fifth-century by the legal scholar Ulpian, adding a comment to his lecture notes. The ancient memo states that a confirmation for a particular directive can be found in the *Regulae* ("Rules") of the third-century Roman jurist Modestinus, "*seventeen regulae from the end, in the regula beginning 'Dotis'...*". All earlier references have been vaguer, pointing to books - for example the bible - or speeches rather than precise locations within a document.

In the early 1960s, "references" became "links" and Ted Nelson formulated the term "Hypertext" for information units coupled by links which allows content navigation in non-sequential manner. The idea of connecting content with links became the fundamental paradigm in hypermedia systems, resulting in the ultimate application, which is the WWW and its widely projected successor, the Semantic Web (see Section 2.9). Traditional linking is primarily categorised in two ways. Firstly, the link behaviour can be static or dynamic, meaning that links are physically kept embedded within the information source or in independent LinkBases, external to the source and fed into the document on demand. Secondly, a distinction between implicit and explicit links can be drawn, making the link to the target information visible or transparent to the user [Letal99].

This section focuses on the aspect of defining and referencing a location rather than the linking functionality, defined in the more recent W3C recommendation related to XML Linking [XLin01]. Uniform Resource Identifier (URI), such as an Uniform Resource Locator (URL) or Uniform Resource Name (URN), are used to locate content

in an information space, for example the WWW or its projected successor, the Semantic Web.

Terms such as URI, URL and URN do still cause a lot of confusion among the web community. Misunderstandings mostly originate in the incompatibility of classical and contemporary view of URI partitioning, as described by the W3C [UPIG01] and illustrated in **Figure 2-5**.

The classical view distinguishes between two types of identifiers: Firstly location-based identifier which identify resources by their location using an URL, which in turn specifies a transport protocol such as HTTP (`http:`); secondly name-based identifiers, URN, that identify a resource by its name such as International Standard Book Number (ISBN) by its distinct namespace prefix `isbn:`. Both, URLs and URNs are non intersecting true subspaces of the URI space. The initial proposal allowed extending the URI space by a limited number of additional subspaces, such as an Uniform Resource Class (URC). But due to the lack of interest, this idea was never realised. The contemporary view is based on a different concept. It assumes that a number of URI schemes are specified within the web space; for example a URN scheme for name-based resource identification and a URL scheme for location-based resource identification. Each URN scheme can have an unlimited number of subspaces, often described as namespaces. One such example is the ISBN URN namespace with the `isbn` as the URN namespace identifier.

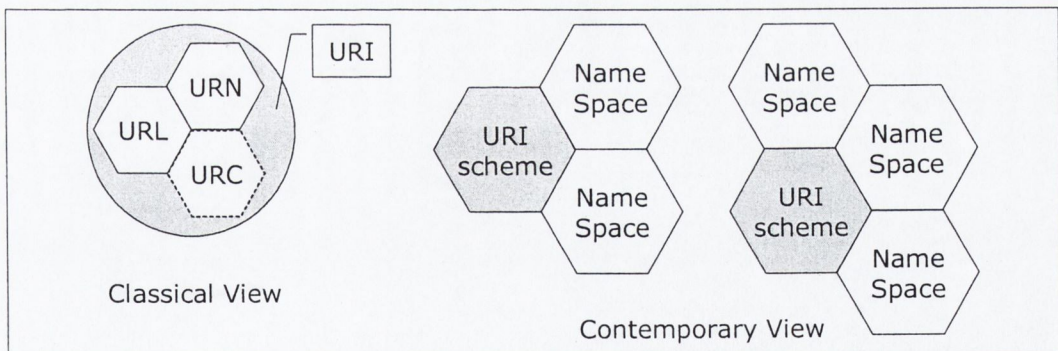


Figure 2-5: Classical vs. contemporary view of URIs, URLs, and URNs

At present, URLs are the predominant URI used on the WWW. A URL is a combination of external and internal identifiers, composed of a maximum of six different elements including:

1. Transport protocol that defines how to access the source, e.g. HTTP, FTP,

2. Host, e.g. a Web Server, often identified by a domain name rather than an IP address,
3. Port that listens to incoming requests on the host,
4. Virtual path to the location on the host,
5. Name of the resource that represents the content, including a physical file or a service (such as a script or application) which outputs a document,
6. Fragment identifier or position within the resource.

Elements 2-5 are used to reference a static file or application on the server that dynamically creates content similar to a file (external reference). In addition, element 6 can be appended to cite a position or fragment inside that document (internal reference). **Code 2-5** and **Code 2-6** show two different URL styles, both illustrating the generic syntax (line 1) followed by a specific example (line 2). The difference is in the fragment identification part: **Code 2-5** uses the simple anchor reference point syntax known from HTML, whereas **Code 2-6** uses an XPath expression as used in XML vocabularies, for example XHTML.

```

1 protocol://host:port/location/resource#position
2 http://www.cs.tcd.ie:80/Benjamin.Jung/index.php#phonenumber

```

Code 2-5: URL with anchor reference (#phonenumber)

```

1 protocol://host:port/location/resource#xpath_expression
2 http://www.cs.tcd.ie:80/Benjamin.Jung/index.php#id('phonenumber')

```

Code 2-6: URL with XPath expression reference (#id('phonenumber'))

Two recommendations of the XML family of languages are primarily related to linking and resource location: XML Linking Language (XLink) [XLin01] and XML Pointer Language (XPointer) [XPoi01]. XLink defines ELEMENTS which, when inserted into XML documents, describe links between two or more resources. It promotes advanced linking features besides the unidirectional linking as known from HTML, including metadata definitions associated with the link. Another feature is the native support for LinkBases, dedicated data stores that store and maintain all information about the link separate from the document. Documents only include keys as references to the link details in the LinkBase and the link itself is dynamically incorporated into the final document on demand. Figure 2-6 shows traditional linking

with links pointing to the whole target document (A) and a location within the target document (B). The second example gives a picture of a scenario using a LinkBase (C).

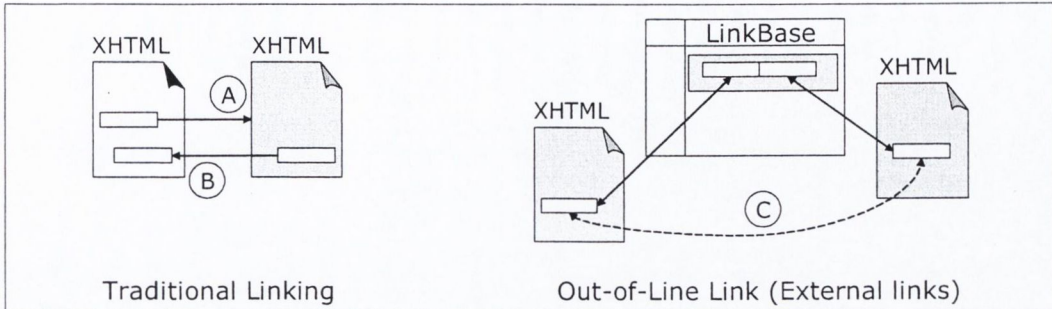


Figure 2-6: LinkBase Architecture

XPointer is a language used for fragment or location identification as part of a URI reference that locates a resource. In particular it provides references into the internal structures of XML documents such as ELEMENTS and character strings, whether or not they have an explicit ID attribute included. XPointer uses expressions based on XPath, a compact, non-XML syntax which is ideally suited to address parts of an XML document within the URI. The XPath syntax is similar to path notations in hierarchical file systems and operates on the abstract, logical structure of an XML document.

2.8. Data Transformation and Presentation

As described in Section 2.2, XML documents are basically plain text files representing a “*simple, common layer for tree structures in a character stream*”¹². Together with the separation of content from presentational information, automatic and autonomous processing of XML files is easy and straightforward for **software applications**. However, visualising XML documents for **human consumption** is essential and needs additional steps. At the time of writing of this Thesis, three different approaches are common practice, all of which are implemented as visualisation modes in most of the available XML editors and viewers.

¹² David Megginson, Megginson Technologies, Ltd., <http://www.megginson.com/>

2.8.1. Plain text visualisation

This is the simplest type of XML document visualisation using a plain character representation. No additional processing is done and the XML file is displayed “as is” in the presentation application such as a browser or editor (see **Figure 2-7**, left part). Navigating the document is not easy as both document markup and content are displayed intertwined and differentiating between them can be a tedious undertaking. Visualisation applications include plain text editors such as Notepad for Microsoft Windows or vi/vim for UNIX derivatives.

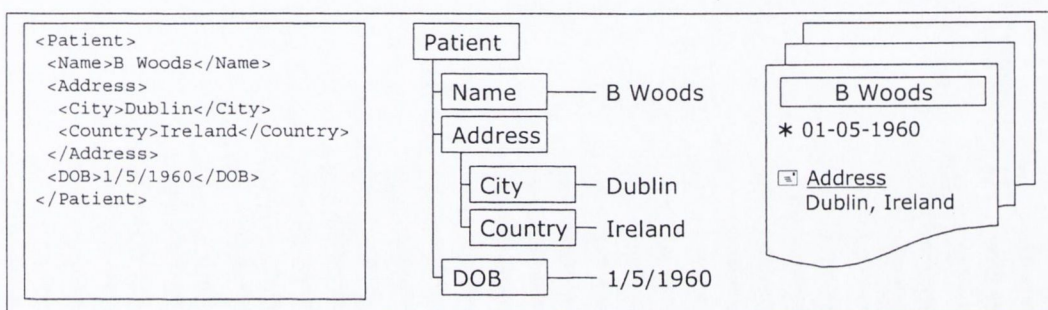


Figure 2-7: XML Visualisations (plain text, structure, transformation)

2.8.2. Structural and markup visualisation

By using structural visualisation, document markup is visually separated from the content to facilitate additional navigational help. Supplementary support comes in several variations including a tree-like view of the hierarchical document structure (see **Figure 2-7**, middle part). Markup visualisation, an improved version of the previously described plain text visualisation, includes simple structural or semantic enhancements, such as indented code and line breaks after closing tags to emphasise the hierarchical nature of the document or colour-coded text to distinguish between markup and content. XML editors such as XML Spy¹³ (Altova®) and XML Markup Editor¹⁴ (Topologi) support various modes of structural and markup visualisation.

2.8.3. Visualisation through transformation

If none of the above described visualisation methods produce a satisfying result, XML documents can be transformed into a visualisation format (PDF) or presentation vocabulary (XHTML, SVG). Each of these formats has an associated stand-

¹³ Altova, <http://www.altova.com/>

¹⁴ Topologi Pty, <http://www.topologi.com/>

alone display application or plug-in, which is suitable for rendering the data (see **Figure 2-7**, right part). The most obvious choice to define transformation rules for XML documents is XSL, another member of the XML family of languages, with its two parts XSLT (XSL stylesheet) for transforming and FO (XSL formatting objects) for formatting content.

XSLT offers a standard way to describe rules that transform structure and content of a source XML document into a target document with a possibly different structure. The process of creating the result document includes ELEMENTS from the source being filtered and reordered as well as adding arbitrary structures. The resulting document does not necessarily have to be an XML document again; transformations into non-XML formats such as plain text or even binary files are possible. During the conversion phase, an XSL processor reads the XML and XSLT documents, follows the rules in the XSL stylesheet and outputs the result.

FOs are another XML vocabulary and an intermediate step in a conversion from a proprietary XML source document into a presentation format like PDF. The initial step involves the insertion of formatting objects, i.e. typographic abstractions such as page, paragraph and table, into the result document. To enable a finer granularity, indents, word- and letter spacing, and widow, orphan, and hyphenation are controlled by associated formatting properties. Both, formatting objects and properties are solely intended for presentation purposes and contain presentation specific details like page-size and margin-width. In the second and final phase, the FO document is processed by an FO engine and the desired end-user format is created, e.g. PDF.

Figure 2-8 shows three different types of XSL transformation scenarios. A pure server side processing requires XML as well as XSL documents on the server (A). Depending on the properties of the requesting client, the server selects a stylesheet tailored for a specific hardware (software) and generates device specific (viewer specific) markup. The client on the other hand does not need any processing capabilities; instead a simple viewing application is sufficient. However, structural as well as logical information of the original source document (XML) is lost and automatic interpretation almost impossible. It should be noted that presentation vocabularies such as XHTML and SVG should only be used for data visualisation; further processing is not advised. Information could also be customised for a particular user group or individual user in a similar fashion

If hard- and software configuration allows the client to execute the transformation process, the server decreases its potential processing load by sending both XML and XSL files to the client (B). This has the advantage that the client is aware of the content's original markup. Moreover, if special visualisations are required that are not available as part of the server's stylesheet package, the client is able to load the appropriate stylesheet from other resources, possibly a local file store, and subsequently execute the transformation (C).

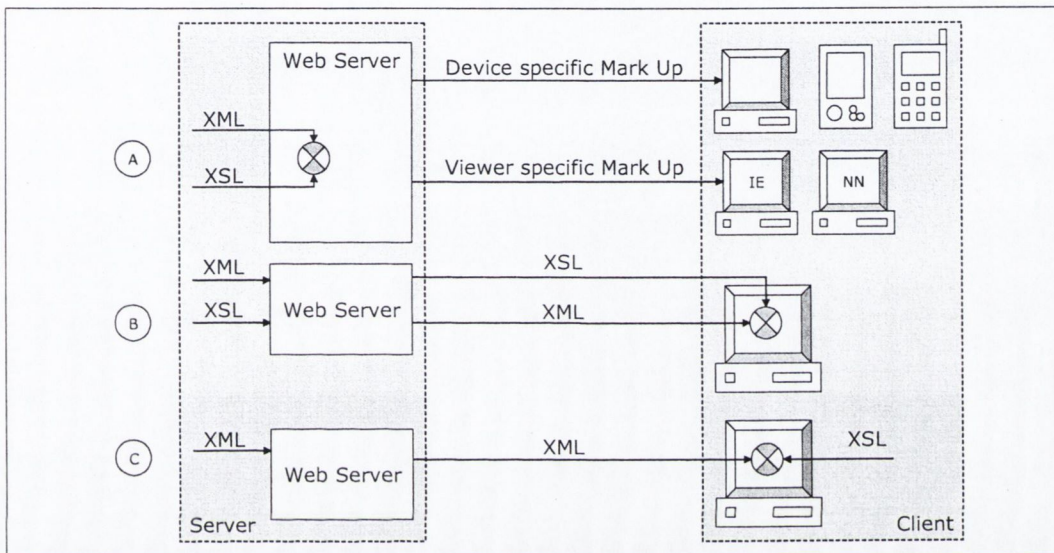


Figure 2-8: Server vs. Client side processing (XML-XSL)

2.9. Semantic Web

*Definition*¹⁵:

The Semantic Web is the representation of data on the World Wide Web. It is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners. It is based on the Resource Description Framework (RDF), which integrates a variety of applications using XML for syntax and URIs for naming.

2.9.1. Introduction

From the beginning of the World Wide Web, information was published in various formats (including text, image, audio and video), different languages, data structures and layouts. Heterogeneity of information and ubiquitous access were among the main reasons to provide and support individuality and independence; the relatively

¹⁵ World-Wide-Web Consortium (W3C), <http://www.w3.org/2001/sw/>

simple original web vocabulary (XHTML) lead to fast publishing and widespread distribution of information by authors ranging from individuals of all ages to commercial institutions. The web has always been an audio-visual medium and at the beginning, the emphasis of content creators has been put on presenting rather than representing information, deliberately concentrating more on design and layout than description of data. This practice is adequate for human consumption but makes it very difficult for machines to autonomously “understand”, interpret, relate and reason available information. Only well defined terminologies (vocabularies) bring meaning into the web and allow computerised agents roaming the information space to gather, process and assemble data. Metadata, i.e. data about data, can extend the current web and provide additional and necessary (semantic) details to support an automated process. Structural as well as semantic mapping between heterogeneous information models allows for better machine-machine cooperation and enables discovery, automation, integration and reuse of data across various applications [Dum00]. Once implemented on a broad scale, the Semantic Web will complement humans in searching (finding) information and browsing (navigating) the web through contextual assistance. It will expose data in a machine understandable format and provide access to independent agents (applications) through publicly available services.

The idea of the Semantic Web (SW) is based on three essential concepts, namely **Expressing Meaning, Knowledge Representation** and **Ontology Definition** [BHL01]. All elements have implicit relations to one or more members of the XML family of languages described earlier in this chapter.

2.9.2. Expressing Meaning

In the past, the assembly of information on the web was nearly exclusively limited to design and layout structures similar to those known from textual publishing, e.g. books and articles. The reason for this lies in the chosen SGML vocabulary (HTML), which was designed to simply present information rather than describing it, although simple descriptive tags were already included in the first draft, such as headings, paragraphs and lists. Nevertheless, the structural rules often opposed the logic, as heading ELEMENTS were kept outside the paragraph ELEMENT, for example, and therefore logically separated from the content. With the ubiquitous use of the WWW, data increasingly did not fit into the minimal HTML tag set anymore, which lead to peculiar adaptations of HTML tags including the mark up of the EHR in the medical

domain. Proprietary extensions to the HTML standard completed the confusion and caused a high degree of uncertainty and disbelief in a valuable future aspect of the WWW, especially for use in the medical informatics domain.

With the introduction of XML and its predominant feature of self-definable structure as well as tag names, WWW authors are not forced anymore to adapt pre-defined (presentation) markup vocabularies to their needs. Instead, expressing meaning by using a clear hierarchical structure together with descriptive tag names closely related to the original domain, leads to information that is not only comprehensible by humans but more importantly also by machines. Besides, this will create knowledge environments where humans and machines can cooperate in a more intelligent way than they ever did before.

2.9.3. Knowledge Representation

The availability of a substantial and structured information pool together with inference rules between the various information items of the pool are the two basic requirements to represent knowledge. XML is the ideal syntax to define structured information including its unique features of hierarchical data format and self-definable data containers, e.g. ELEMENT and ATTRIBUTE names. Additionally, the well established XML vocabulary Resource Description Format (RDF) [KlCa03] allows meaning to be expressed using simple statements. Each statement consists of an information triplet comprising a subject, a predicate and an object (value), where the value can act as a subject in its own right in another triplet combination. For instance, the information "*Barney Woods is a patient in St. James's Hospital*" can be expressed using RDF triplets as shown in **Code 2-7** and **Code 2-8** (XML syntax). In these examples, the subject *Barney Woods* is uniquely identified through a Social Security Number URI (using an `ssn` namespace), the predicate `patient-in` is part of (an imaginary) *HL7* namespace and another URI specifies the related institution (*St. James's Hospital*).

```
1  ssn:123-456-789      hl7:patient-in      http://www.stjames.ie
```

Code 2-7: Simple RDF triplet ("Barney Woods is patient in St. James's Hospital")

```
1  <?xml version='1.0'?>
2  <rdf:RDF xmlns:rdf='http://www.w3.org/TR/WD-rdf-syntax#'
3      xmlns:hl7='http://www.hl7.org/' >
4      <rdf:Description rdf:about="isbn:123-456-789">
```



```
5 <hl7:patient-in rdf:resource='http://www.stjames.ie'/>  
6 </rdf:Description>  
7 </rdf:RDF>
```

Code 2-8: Simple RDF example using RDF/XML syntax

In the most effective RDF environment, each participating entity (i.e. subject, predicate and object) is identified by a URI, which allows global linking and makes expressive representation of information possible.

2.9.4. Ontology Definition

The third fundamental component of the Semantic Web is the availability of Ontologies, described as “*an explicit formal specification of how to represent the objects, concepts and other entities that are assumed to exist in some area of interest and the relationships that hold among them*”¹⁶ or simply, a “*specification of a conceptualization*” [Gru93]. Generally, ontologies are a combination of taxonomy and certain related inference rules. Taxonomies provide a domain classification, i.e. a set of classes together with appending subclasses. A simple and often used ontology example is the address class which includes aggregation items such as street, house number, ZIP code, city and country. Inference rules uniquely associate for example the ZIP code and a combination of street/house number/city/country. Instances of these classes (e.g. specific streets, cities etc.) together with the ontology itself build knowledge bases.

As a result of XML’s feature to allow proprietary tag names, different terminologies for similar entities might cause problems for machine understanding and reasoning. Therefore it is important that ontologies offer a solution to support equivalence relations and overcome “*surface differences*” in XML vocabularies such as localised terms *Name* (English) and *Nombre* (Spanish) as well as synonyms *postcode* and *ZIP*. The use of ontologies allows for improved and automated exchange and integration between distributed, heterogeneous information systems as centralised vocabularies are not necessary anymore.

Since the beginning of the XML era, knowledge and ontology representation formats have been an integral part of the XML family of languages. RDF was the earliest

¹⁶ <http://dict.die.net/ontology/>

adaptor to express knowledge closely followed by DARPA Agent Markup Language (DAML)¹⁷, an extension to XML and RDF. Subsequent developments focused on languages designed for ontology descriptions and lead to the release of DAML+OIL¹⁸ (Ontology Interference Layer), which provides a rich set of constructs making data more machine readable and understandable. More recently the W3C released the first candidate recommendation of OWL¹⁹ which is conceptually similar to DAML+OIL.

2.9.5. Relation between Semantic Web and Semantic Health Record

The conceptual relation between the Semantic Web and an envisaged Semantic Health Record is clearly visible and multifaceted. EHR information is heterogeneous, highly fragmented and distributed. It is controlled by a number of service providers (e.g. hospitals, departments, GPs, laboratories, insurance companies) and available in multiple formats, languages and structures. Retrieval is often restricted or granted on an individual and exclusive basis through various network access methods. In an equivalent scenario to the Semantic Web, the SHR facilitates improved, but potentially complete, automatic content integration. Roaming agents are capable of collecting related information in order to reason, compare, assist and predict possible procedures such as treatments and guidelines. Medical standards organisations provide terminology and classifications to enrich data with valuable and necessary metadata which makes structural and semantic mapping a promising task. Additionally, sophisticated information retrieval within the contextual field can improve and support actions between healthcare professionals. Physical as well as virtual (through linking) content integration is achievable. There are many beneficiaries of the SHR, but predominately the patient. Doctors are able to easily retrieve and merge remote patient data, assuming that unique patient identifiers are in place and security measures are met. Similar or related medical cases in addition to medical guidelines are accessible for reconciliation and lead to an enhanced, i.e. faster and more accurate, treatment. This immediately directs to a decreased number of consecutive consultations and therefore less expensive therapy.

¹⁷ <http://www.daml.org/>

¹⁸ <http://www.daml.org/2001/03/daml+oil-index.html>

¹⁹ <http://www.w3.org/TR/owl-features/>

2.10. Summary

This chapter has reviewed members of the XML family of technologies that are fundamental to the development of a Semantic Web and an adaptation in the medical domain, the Semantic Health Record.

Firstly, it described in brief the history of content encoding, in particular in relation to markup languages such as SGML and XML. This is followed by a section about XML storage methodologies, including internal (in XML files) as well as external (in XML databases) solutions. Then an example of a simple query implementation together with the official version of the W3C recommendation was presented. A discussion on encryption of XML documents and XML fragments followed, before the URI addressing mechanism for web content and linking between content elements was explained. The chapter finished with a section about ways to transform XML documents for expressive presentation and a technical explanation of the three essential aspects of the Semantic Web, namely Expression Meaning, Knowledge Representation and Ontology Definition.

Chapter

3

Electronic Patient Record Architectures

Diversity of approaches will always be a feature of the internal implementation of EHCR systems. In terms of the record "one size does not fit all". Different specialities have different needs and different ways to store and present the information.

(Jane Grimson)[Gri01]

The ancient Romans built their greatest masterpieces of architecture, their amphitheatres, for wild beasts to fight in.

Voltaire

3.1. Introduction

Approaches to defining a semantic model and logical structure (architecture) for electronic patient records do not differ much from modelling other types of documents or databases. The overall process is generally broken up into three distinctive sub tasks, but not every method keeps them separate in the modelling process. The main three consecutive steps are as follows:

- Stage 1: Define the basic document building blocks as structural abstract entities. These are the smallest identifiable and independent (structural) objects within the patient record.
- Stage 2: Define a common model, i.e. a class hierarchy for patient record documents, using the abstract entities defined in stage 1. An instance of this class is often described as a structural skeleton, i.e. a document compiled from only structural components but without actual data.
- Stage 3: Instantiate a patient record object and populate this empty skeleton with live patient data retrieved from the system's data store, e.g. a database.

Data exchange and integration can be supported through level 1 or level 2, representing a pure structural (shallow) or combined structural and semantic (deep) exchange respectively [see Section 5.2]. Both strategies show significant advantages and disadvantages that do not allow a universally valid recommendation.

A number of standards organisations have begun to use XML technologies as the prime syntax for defining their semantic data structures as well as for the actual record. XML, what seemed to be an unbeatable technology to solve syntax differences between heterogeneous data models and making seamless integration easily achievable, turned out to be a fantastic instrument for information interchange and ubiquitous document access. Nevertheless, the original problems of finding a conformant terminology, ontology and finally document standard resides, although they are easier to deal with.

3.2. *Electronic Health Records*

3.2.1. Background

The management of electronic patient data was relatively easy as long as the data was collected, stored and viewed in a closed environment like a single hospital/department or doctor's practice. A single vendor provided compatible systems and a centralised system administrator was responsible for the smooth running of all installed components. Data exchange with external organisations was only possible by paper, mail, fax or telephone, which is clearly both time-consuming and error-prone. Later, the electronic exchange of patient information was carried out by means of physical delivery of data on storage devices such as tapes and disks. A network connection between the hospital and the 'outside world' was not available which avoided many of the problems associated with "open" systems. In particular safeguarding security and confidentiality of patient data was straightforward to implement by means of physical access restrictions (locks) and file access rights (tape, disk). However, feedback (such as comments) and results had to be manually entered and linked to the original entry, a repetitive task that is prone to error and inaccuracies. This approach is no longer sufficient. Increased computerisation throughout the health sector has given rise to a proliferation of independent (and heterogeneous) systems for storing patient data. However, the growing trend towards shared care requires that these systems are able to share their data. This has led to the development of projects such

as Synapses (see Section 1.5.1) [Syn01] [GBG98] [GGB98] and its successor, SynEx (see Section 1.5.2) [SynEx01] [FG99] that aim to provide healthcare professionals with integrated access to patient records and related data. One of the goals of SynEx is to bring together the work on Federated Healthcare Records from the Good European Health Record project [Ing95], emerging standards from CEN/TC251 (Comité Européen de Normalisation - Technical Committee 251) [TC251] and the Synapses project with the Information Systems perspective of middleware based Health Information System Architecture pre-standard HISA (Healthcare Information Systems Architecture) [FG99, HISA97]. The HISA standard defines six healthcare related common components and associated services in order to support the development of modular open systems in healthcare. On the other hand, the Synapses project exploited ideas from federated database technology [ShLa90] which provide client applications with an integrated view of data stored in heterogeneous, distributed database systems. At the heart of Synapses is the FHCR (Federated Health Care Record) server that accepts requests for data (in the form of clinical objects) from clients, decomposes them into queries against the connected "feeder" systems, where the data is actually stored, and integrates the responses dynamically. The Synapses project was concerned with the specification of an open standard for the server and its interfaces and for pragmatic reasons used an ad hoc mechanism for exchanging clinical data between feeders, server and client. An obvious candidate for such an exchange format would be based on a easy accessible syntax, such as XML (Extensible Markup Language) [XML00] together with an widely accepted semantic standard. This conceptual combination was pursued in the SynEx project.

The use of XML to exchange data between heterogeneous systems provides support for hierarchical structured patient data, user defined tags and machine understandable assertions for searching, reasoning and analysing EHR objects.

3.3. Terminology

Even before the "digital age", collecting information about a patient's health has been a labour-intensive and specialised task. Every medical domain has its own minimal dataset, i.e. the smallest amount of information to be collected. Visiting a large number of specialists during a person's lifespan accumulates an enormous amount of medical patient data. Every piece of information related to a patient's

health, is acquired as a result of a thorough investigation by a specialist. By nature, each medical domain as well as the specialists themselves propose and maintain their own set of minimal/optimal amount of data to be collected. This fact together with an emphasis on different aspects and a multifaceted interpretation focus of the EHR lead to numerous terminology variations.

However, more recently two distinct terms became widely recognised and accepted by standards organisations as well as healthcare institutions: Electronic Health Record (EHR) and Electronic Patient Record (EPR).

3.3.1. Electronic Health Record

The term EHR describes the complete collection of medical data, related to one single patient over his entire life and linked by a (globally unique) person identifier such as the Social Security Number (SSN) or Patient Record Identifier (PRI). This information pool, often also described as the longitudinal **Cradle-To-The-Grave Active Record**, does not necessarily have to be kept in a single environment. In fact, it is more likely that data, including text, images, audio and video, is distributed over a number of different physical locations and systems, maintained by diverse applications. Typically, the data is stored in heterogeneous data formats ranging from unstructured and semi-structured to fully structured, (virtually) ubiquitously accessible through various network types [Gri01].

EHR systems are developed to maintain EHRs in the most flexible (from a technology's point of view) as well as most comfortable (from a user's point of view) way in order to provide a holistic picture of the patient's health and medical history. Apart from the obvious patient care delivery and management tasks, support for care, financial and administrative processes as well as patient self-management are among the primary uses of such a system [DSD97]. Secondary uses include aspects in education, regulation, research, public health and policy support. In order to retrieve the EHR (or even EHR fragments), data from numerous information sources have to be requested and retrieved, transformed and customised, merged and integrated before finally being presented to the user. These methods, enriched and evaluated with further information from standards, guidelines and quality assurance processes, lead to the **Active Patient Record**.

3.3.2. Alternative Terminology

The EPR is typically a fragment of the EHR. It contains medical information of one particular patient with relation to e.g.:

- A specific **time period** in which the patient visited that institution. In a time where people relocate more often than ever before, changing the General Practitioner (GP) and/or hospital comes along with moving houses, cities, countries or even continents.
- A **specialist department** such as radiology which only keeps results of imaging examinations such as X-Ray and Magnet Resonance Tomography (MRT).
- A **domain specialist** such as a psychiatrist who primarily collects data related to the immediate domain.

Various (often equivalent) terms are used for the EHR and EPR in medical informatics literature. In essence, they are EPR, but giving indications to one specific focus or aspect within a domain:

Paediatric Electronic Medical Record (PEMR)

[AAP01] states that “*an essential function of a paediatric EMR system is to facilitate care that is accessible, family-centred, continuous, comprehensive, coordinated, compassionate, and culturally effective*”. Furthermore, “*the purpose of EMR systems is to compile and centralize all pertinent information related to a child's medical and non-medical care so as to ensure that optimal paediatric care is provided*”.

Computer Based Patient Record (CBPR)

The Computer-based Patient Record Institute (CPRI)²⁰ defines a CBPR as “*an electronically maintained [set of] information about an individual's lifetime health status and health care. The computer-based patient record replaces the paper medical record as the primary source of information for health care meeting all clinical, legal and administrative requirements. It is seen as a virtual compilation of non-redundant health data about a person across a life-*

²⁰ <http://www.cpri.org/>

time, including facts, observations, interpretations, plans, actions and outcomes. The CBPR is supported by a system that captures, stores, processes, communicates, secures and presents information from multiple disparate locations as required."

Problem Oriented Medical Record (POMR)

In 1968 Lawrence Weed [Wee68] defined the POMR where *"each patient was assigned one or more problems. Notes were recorded per problem according to the SOAP structure, which stands for subjective (S; the complaints as phrased by the patient), objective (O; the findings of physicians and nurses), assessment (A; the test results and conclusions, such as a diagnosis), and plan (P: the medical plan, e.g., treatment or policy). Besides further improvement in the standardization and ordering of the patient record, the main purpose of the problem-oriented SOAP structure is to give a better reflection of the care provider's line of reasoning."* [BeMu97]

Personal/Consumer Health Record (PCHR)

The American Society for Testing and Materials (ASTM) defines in the document E31.28 [E3128] the Personal (Consumer) Health Record as *"an electronic application where individuals can maintain and manage their health information and that of others for whom they are authorized in a private, secure, confidential environment that allows the individual or other authorized persons to access and share such information."*

3.4. Synapses EHR architecture

3.4.1. Introduction

The Synapses Record Architecture is based on the ENV12265 record architecture specification [ENV12265] (developed and issued by CEN/TC251²¹). It is described as a three stage paradigm, including the fundamental single class hierarchy SynOM (see **Figure 3-3**), a conceptual framework or skeleton of a record or record fragment (SynOD) and finally the actual Synapses patient record.

²¹ Comité Européen de Normalisation - Technical Committee 251, <http://www.tc251.org/>

One of the fundamental considerations in developing the Synapses Record Architecture was the definition of granularity for each of the components. The **Train Analogy** clearly explains and depicts (see **Figure 3-1**) the difference of two approaches. As a first option, a small number of generic but basic elements (LEGO® blocks as displayed in the left picture for example) are being engineered, which allows one to build a huge number of differently shaped model trains. As expected, the design might look a bit rough at the edges, but the variety of shapes through re-arranging parts offers enormous model flexibility. On the other hand, highly specialised elements can be assembled to form a stylish looking train. Unfortunately, a vast number of different components are needed. Each part is dovetailed and reusing them in other areas of the train proves very difficult if not impossible; the front window barely fits into the door frame for example.

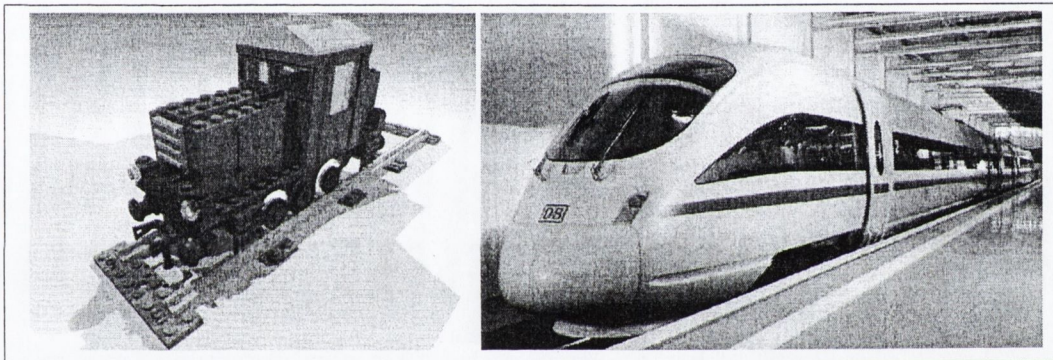


Figure 3-1: Component granularity (using the train analogy)

These two distinct approaches can be directly adapted to the design of an EHR architecture. Depending on a number of factors, but especially on the fundamental requirement of *flexibility in use* or *elegance in presentation*, one or the other of the above mentioned approaches is more suitable for the task.

The Synapses Record Architecture, following the ENV12265 research, applies the low granularity paradigm. A small number of EHR components are defined to allow for most flexible record construction. This implies that each user has to become familiar with the definition and characteristics of each component as well as the overall concept in order to create a well designed patient record. The Record Structure Builder (see Chapter 4) is a software tool that interactively supports the user in this task.

3.4.2. Synapses Object Model (SynOM)

The SynOM is a common object model and extends the model described in the pENV12265 from the *Comité Européen de Normalisation Technical Committee 251* (CEN/TC251) [TC251] [ENV12265]. It contains a set of seven base classes (see **Figure 3-2**), their attributes and hierarchical aggregation (see **Figure 3-3**), which will be used as general building blocks to define a SynOD (FHCRC shape or skeleton) and assemble the patient record.

Core Components

The seven base classes (core components) are divided into three main categories, namely into a set of structural-, data- and link components (see **Figure 3-2**). Structural components, i.e. RECORDFOLDER, FOLDERRIC and COMRIC, are used to build the high level organisation of the Synapses compliant EHR. This core structure is often also identified as the record skeleton, as it does not contain any actual patient data. Data components, i.e. DataRIC and RECORDITEM, on the other hand are used to provide context as well as containers for the actual patient and medical data, retrieved on demand from the underlying storage systems.

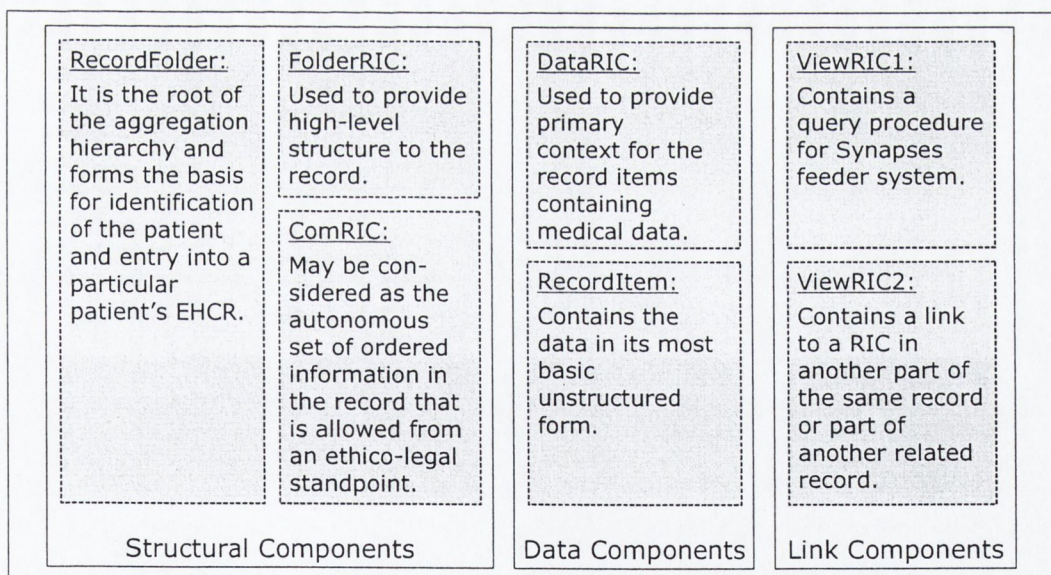


Figure 3-2: SynOM Component Descriptions [GBG98]

The last category consists of link and querying components. VIEWRIC1 ELEMENTS contain detailed information on how to query the underlying storage systems and include the results into the record. During the record assembly process VIEWRIC1 ELEMENTS are typically replaced with a hierarchical structure of data components.

The second link components, i.e. VIEWRIC2 ELEMENTS, provide internal as well as external linking between the different parts of a patient record.

All SynOM components are defined as part of the SynOM class hierarchy (see **Figure 3-3**). The basic object is a RecordComponent, specialising into RecordItem-Complexes (RIC) and RecordItem (RI) classes. OriginalRIC objects, i.e. objects that describe the structure of the patient record, include all structural components as well as the organising DataRIC ELEMENT. The tree structure of each record is rooted in a single particular RECORDFOLDER ELEMENT, instantiated from the class RECORD-FOLDER, which represents the overall record.

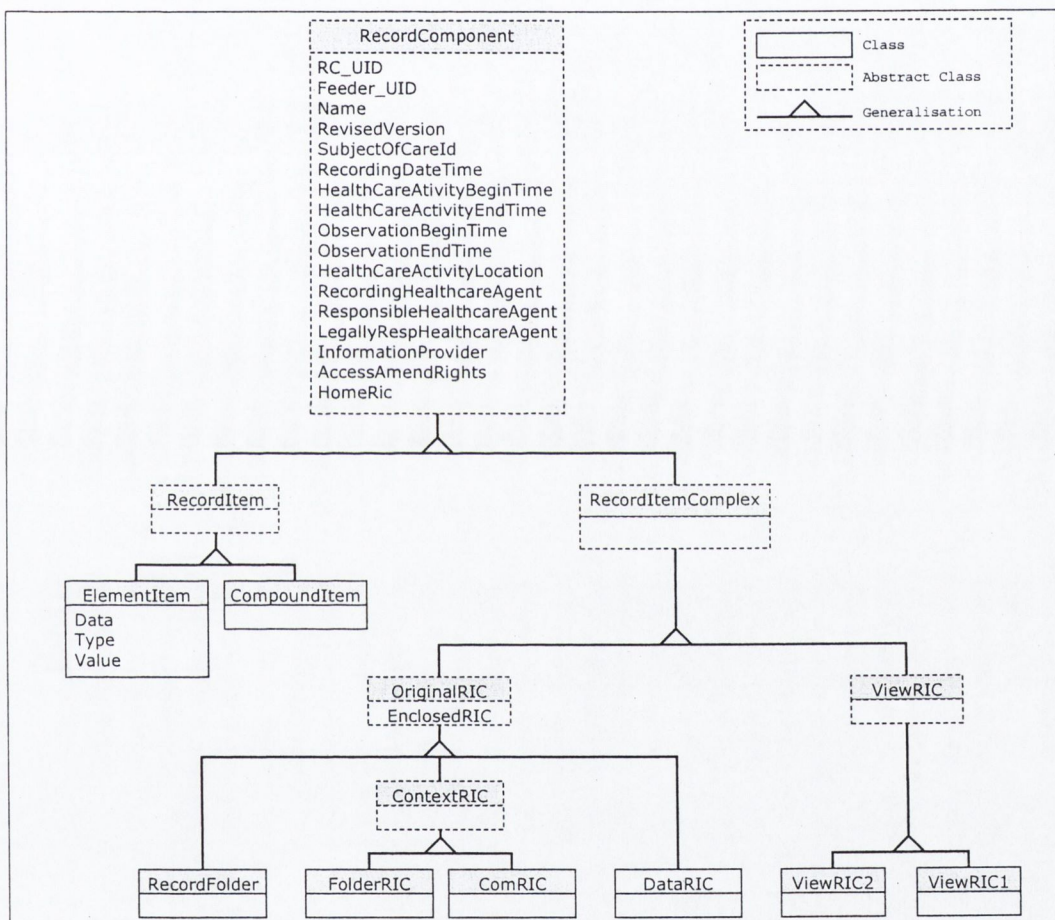


Figure 3-3: Class Hierarchy: Synapses Record Architecture

Below the root object will be a structure of folders (FOLDERRIC objects) and documents (COMRIC ELEMENTS). Each document will itself consist of a tree structure of objects, which can be DataRIC ELEMENTS and/or VIEWRIC1 ELEMENTS. The former contains information that is explicitly recorded in the record, while the latter is used to represent computed or derived information. In addition there are objects that rep-

resent links to other Synapses records, called VIEWRIC2 objects. These are the keys to the integration of Synapses records. They contain the unique identification of another RIC object, and they are used as follows. The root object of a record, or a folder within a record, may contain a single VIEWRIC2 object that references another record or folder object, respectively of the same class. In addition, a document may contain one or more VIEWRIC2 objects each referencing some subset of other documents. The RIC object referenced by a VIEWRIC2 object is kept either locally (intra- or inter-record) or remotely. If remote then the VIEWRIC2 object contains a Uniform Resource Identifier (URI) that identifies the server where the target RIC object resides.

In the actual Synapses Record Architecture implementation, a simplification of the original model has been put forward. COMPOUNDITEM objects are not considered and made obsolete, resulting in only one RECORDITEM (RI) specialisation, which is an ELEMENTITEM. This subsequently lead to the decision to use RECORDITEM instead of ELEMENTITEM. Link components, e.g. VIEWRIC1 and VIEWRIC2 ELEMENT, are specialisations of the VIEWRIC class.

Each RIC object instantiated from one of the OriginalRIC classes will have a small set of static, predefined attributes, e.g. as required for their unique identification, or the target address of a VIEWRIC2 object. However, most of the information content of a record, and all the medical information, exists in RI objects instantiated from the RECORDITEM class. That is, a set of RI objects can be attached to a structural RIC object and thus function as its dynamic attributes with actual data values such as a blood pressure measurement. The RI objects that belong to a particular RIC object can also be organized into a tree structure. This allows the information content of a record to be dynamically extended, including new information types that may not have been foreseen at the time the record itself was created.

Aggregation

At the time of defining and prototyping the first applications based on SynOM and SynOD architectures (such as the Record Structure Builder, see Chapter 4), XML technologies have not been widely implemented and accepted. Initial consensus was reached between the Dublin Synapses group of implementers to graphically define the SynOM component aggregation with the help of state charts. In a later stage, an

equivalent structure using e.g. a Document Type Definition (see Section 2.2) was developed to represent the aggregation in a more precise structure. Nevertheless, the state chart visualisation of the aggregation concepts is still used, especially in teaching and training of Synapses concepts to the less computer literate personnel.

Figure 3-4 describes the hierarchical specification of behaviour rules of the seven core SynOM classes through state charts. Each independent diagram in the figure contains a number of states, including exactly one start state and at least one end state. Looking at the RECORDFOLDER state chart, two paths through the system are possible (beginning from state 1) to build the hierarchy by attaching children to the tree. The first option adds a single VIEWRIC2 child to the RECORDFOLDER ELEMENT and finishes the building process (no more subsequent paths are available from here). Alternatively, option two adds one of the three ELEMENTS FOLDERRIC, COMRIC or RECORDITEM to reach state 2. Adding additional FOLDERRIC, COMRIC or RECORDITEM ELEMENTS is allowed but does not change the state of the system. In other words, a RECORDFOLDER can contain exactly one VIEWRIC2 or any number of FOLDERRIC, COMRIC or RECORDITEM ELEMENTS in any order.

As a special case of the norm, all states in the SynOM model are simultaneously end states, i.e. the building process can stop at any time.

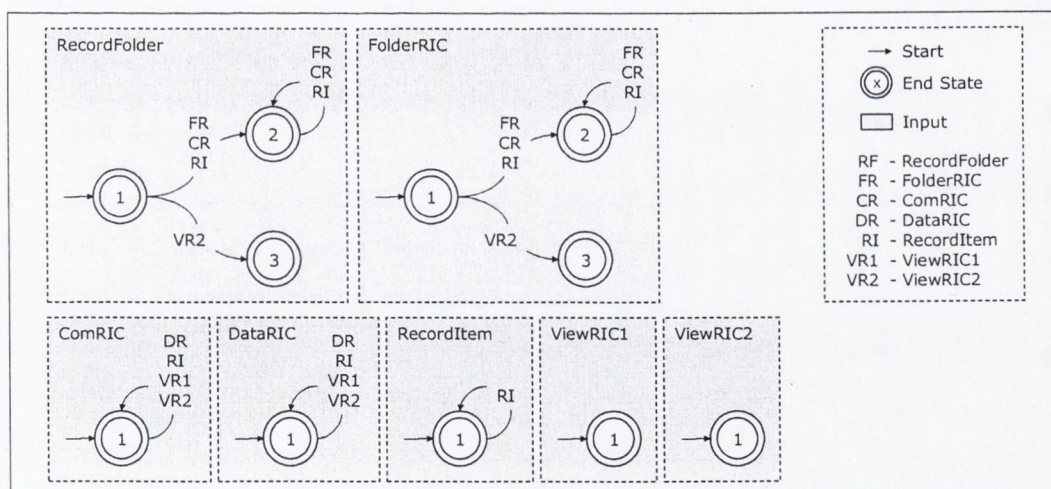


Figure 3-4: SynOM Aggregation

This aggregation model was implemented as part of the SynOM relational database and used in particular during the building process of the local SynOD. An advantage of this approach (often praised by the administrative users for its simplicity during the development phase of SRS and SynODs) is the fact that on-and-off changes to

the model do not require a recompilation of the SRS, as the model is dynamically loaded from the database at server startup. Additionally, the core SRS, initially developed for the medical domain, can easily be adapted to other models and reused in non-medical domains, such as banking.

The complete and more detailed model of the SynOM hierarchical behaviour as a set of state charts can be found in Appendix E. It illustrates and supports the more complex task of operating with multi-level SynODs, e.g. models with multiple levels of hierarchy. It is particularly targeted to flattened object models, e.g. in the form of XML documents, which contain indicators of the beginning and end of ELEMENTS (e.g. tags). This set does not simply help the visualising of the building process, but furthermore lays the foundation for validating existing SynODs.

3.4.3. Synapses Object Dictionary (SynOD)

The distinction between RIC and RI classes comprises of "vertical" grouping of the overall Synapses class hierarchy. In addition, the class hierarchy is split "horizontally" into a predefined set of base classes which are common to every Synapses server, called the SynOM, and an extendable set of classes that are derived from these SynOM classes, called the Synapses Object Dictionary (SynOD). The above RIC classes RECORDFOLDER, FOLDERRIC, COMRIC, etc, all belong to the SynOM, and they define the core part of Synapses' generic record model. The SynOD classes on the other hand, which are site specific and thus may differ for each Synapses server, are the classes from which the actual patient record objects are instantiated. Thus while every record object has the above SynOM characteristics and properties, they can also be customised to the needs of each individual site.

Healthcare professionals and healthcare institutions develop a locally viable and highly customised EHR skeleton (i.e. a template), which best represents their medical domain as well as the expected data. The basic components for this process are the earlier described seven SynOM base classes, which are equal to every Synapses compliant patient record. They are put together into a hierarchical SynOD structure according to the aggregation rules, labelled with a medically representative name and "virtually" connected to the underlying storage devices (feeder systems) by defining query details for retrieving actual patient data. The assembly of the base classes in a

number of SynOD templates and finally the patient record skeleton is (in most cases) unique to the developing institution, such as hospital, department or GP.

3.4.4. Assembling the record

Based on the Synapses methodology of three patient record development tiers (SynOM, SynOD and the record), two intermediate steps are needed to assemble a patient record (see **Figure 3-5**).

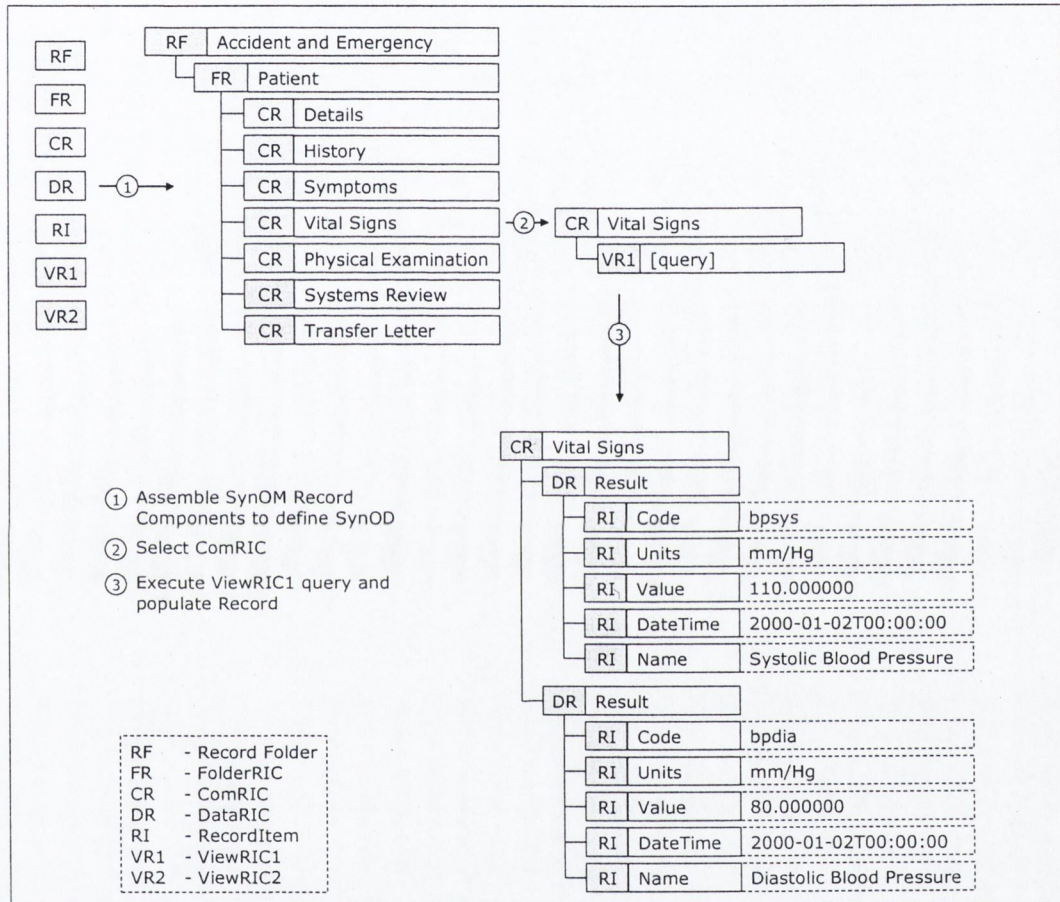


Figure 3-5: Assembling the EHR: The three-stage Synapses approach

The first phase (1) involves the creation of a skeletal architecture using the seven core SynOM components, the associated aggregation definitions and information (such as access-path and access-rights) about the underlying data stores. At that stage, each component is labelled with a medical concept (semantic mapping, see Chapter 4), which automatically allows for (at least) two visualisations including a technical (*technical view*) and a domain specific one (e.g. *medical view*). In a second phase (2 and 3), information from the feeder systems is requested according to the access details stored in a VIEWRIC1 ELEMENT in the record. For this purpose, VIE-

WRIC1 ELEMENTS typically contain information about feeder system location, access methods, a classification of access rights (e.g. based on user role and/or data type), query strings and a mapping definition to convert the returning result set into SynOD elements.

Figure 4-6 shows a screenshot of the RSB with a VIEWRIC1 element expanded in the TreeView. The Query node contains child nodes with information about the access method, data source and the actual query string, whereas the Result node defines the mapping from the returning result set into RECORDITEM XML ELEMENTS. At present, mapping definitions are defined as a simple match between the attribute name in the result set and the corresponding node name in the TreeView. As attribute names have to be unique within the result set, the placement (even in a deeper hierarchy) among the various Result nodes is a straightforward match defined by the name.

3.5. Other Patient Record Architectures

3.5.1. HL7

HL7 was founded in 1987 to “*provide standards for the exchange, management and integration of data that support clinical patient care and the management, delivery and evaluation of healthcare services. Specifically, to create flexible, cost effective approaches, standards, guidelines, methodologies, and related services for interoperability between healthcare information systems.*” [HLS01] This includes hospital information systems, clinical laboratory systems, enterprise systems and pharmacy systems.

The XML SIG group (later incorporated into the HL7 organisational hierarchy) was formed in early 1997. Since then, preliminary work was undertaken to develop the HL7 Reference Information Model (RIM), an object-oriented model of general EHR components. On the basis of the RIM, the Patient Record Architecture (PRA) model was designed, published and subsequently renamed to Clinical Document Architecture (CDA). The CDA consists of a number of predefined templates, organised by medical domain and activity within the hospital. The RIM serves as a core base and central schema that defines the semantics for all HL7 messages and documents [Detal99]. By using generic RIM components, the CDA is designed as a multi-level architecture with a strong focus on XML technologies. In fact, both PRA and CDA are designed as XML-based architectures.

The PRA model was developed as a hierarchical collection of document templates that forms the cradle-to-grave EHR when assembled. The concepts of levels or tiers relate to the document syntax and refer to varying degrees of required mark-up granularity and specificity. Indeed, it does not refer to the degree of granularity or depth of clinical information contained within the document. Every CDA document, independent of its level, contains a very rigid and detailed CDA header with information followed by a document body ELEMENT, which encloses the clinical information. Depending on the level, the clinical information is marked up to a varying degree. Level one compliance (Coded Header) offers complete interoperability for human-readable content, but does not specify encoding for interoperable machine processing beyond the header. Information, such as the traditional handwritten doctor's notes, is easily included as a free text ELEMENT. However, coding and classification of information is not supported.

The level two document body (Coded Structure) is structured into sections to support minimal machine processing. Level three documents (Coded Content) must be XML of sufficient structure and specificity to be consistent with the full version of the RIM and include coded header and coded structure of the lower levels 1 and 2.

The level concept provides a very flexible way to structure EHRs. It also allows the introduction of local structures with a decrease of level conformance to level one.

3.6. CEN/TC251

The scope of the work of CEN/TC251 is the "[...] *standardization in the field of Health Information and Communications Technology (ICT) to achieve compatibility and interoperability between independent systems and to enable modularity.*"

[CEN00]

CEN/TC251 started considering XML as the formal syntax for EPRs while revising European standard ENV12265 [ENV12265]. For that reason, the XML taskforce was formed in the middle of 1998. Its main purpose was the evaluation of XML as a globally accepted data structure to express data models for the various CEN specifications. ENV12265 does not include any XML related parts, but as the model is hierarchically organised, a mapping into a XML/SGML DTD will not cause any problem. In fact, the Synapses Object Model (SynOM), and as it follows the SynExML DTD, is based on the ENV12265 specification. The successor of ENV12265, Euro-

pean pre-standard prENV13606 [ENV13606] was finalised with the expert's outcome of the XML WG. The Health Informatics – Electronic Healthcare Record Communication specification comprises four different parts, namely Extended Architecture (part 1), Domain Termlist (part 2), Distribution Rules (part 3) and Messages for the Exchange of Record Information (part 4). At the moment, only part 4 contains a section about an explicit mapping from the native 13606-4 model into an XML DTD. Parts 1-3 remain without an XML DTD specification. Again, a straight and logical mapping of the existing model into its XML counterpart is imaginable and should be easily completed by following standard rules (see also ISIS99).

The ENV13606-4 XML DTD inherits many aspects from its predecessor. A very rigid header with information to identify the EPR's patient and originator as well as detailed sections about drug prescription and hospital internal administrative data (e.g. bed number etc). The main part of the EPR is kept flexible and can be defined by the medical institution. It uses modified ENV12265 concepts to code EPR hierarchy and medical information in the record.

More recently, the openEHR²² approach developed a standard, based on ideas from the ENV13606 family of standards and aspects of the Good Electronic Health Record (GEHR)²³ community. In this approach, two object models are of importance. The Record Model (RM) defines a set of patient related medical record elements, similar to the idea of the SynOM, but extended in size and specificity. A second model, the Archetype Model (AM), defines a number of archetypes or templates specific to domain or activity in the hospital.

3.7. Other Architectures

3.7.1. ASTM

Founded in 1898, ASTM (the American Society for Testing and Materials) is now one of the largest international voluntary standards organisations. ASTM is a non-profit organisation whose standards encompass metals, paints, plastics, textiles, petroleum, construction, energy, the environment, consumer products, medical services and devices, computerised systems, electronics, and many other areas.

²² <http://www.openehr.org/>

²³ <http://www.gehr.org/>

In its mission statement, ASTM declares

"[...] to be the foremost developer and provider of voluntary consensus standards, related technical information, and services having internationally recognised quality and applicability that

- *Promote public health and safety, and the overall quality of life;*
- *Contribute to the reliability of materials, products, systems and services; and*
- *Facilitate national, regional, and international commerce."*

[ASTM00]

The intent of subcommittee ASTM E31.25 is to develop standard electronic document representations of paper-based healthcare documents and forms. A goal of the subcommittee is to work together to enhance existing levels of interoperability among various XML/SGML standardisation efforts, products and systems in healthcare.

The scope of E31.25 is the development of standards and promotion of knowledge related to DTDs in healthcare. The subcommittee is developing implementation guides, sample document instances (versions of the document with XML mark-up) and a validation facility for verifying conformance to the voluntary ASTM standard DTDs. So far, ASTM has proposed lists for the following sections of an EPR: Discharge Summaries, Clinical Notes, Admission Notes, Operative Reports, Procedure Notes, Diagnostic Imaging and Prescriptions [ASTM00]. The DTDs are being developed under the current XML standards and will adapt as the XML standard evolves.

The first priority for E31.25 is to develop standard DTDs for transcribed documents. To reach this goal, the subcommittee co-ordinates with other ASTM subcommittees and outside organisations with related interests and standards, for instance, HL7.

E31.25 views the medical records as a collection of electronic documents and incorporates other efforts when possible. A significant work in progress is document analysis.

3.8. Comparison

The main differences between approaches from CEN/TC251/Synapses and HL7/openEHR are rooted in the granularity of already predefined elements. Whereas CEN/TC251/Synapses offer highest flexibility in creating hospital specific templates (SynOD), HL7/openEHR mandate a set of templates/archetypes to be applied by the institution. Both approaches have advantages as well as disadvantages. Giving more flexibility to the individual institutions allows for better and easier local adaptation. The institutions are able to develop the system according to their specific needs in mind instead of adapting hospital procedures and systems to a predefined EHR construct. Furthermore, the medical institution is able to implement an EHR immediately, instead of having to wait until archetypes for their domain are available. The disadvantage is that they need the technical and medical expertise to develop EHR templates/archetypes although supporting applications such as the RSB (see Section 4.3) make the task considerably easier. Mapping between individually developed SynODs will be more difficult than using the predefined template/archetype solutions.

3.8.1. Synapses and CEN

As the SynOM is based on a CEN pre-standard there are similarities between the concepts used in the Synapses approach and the current approach taken by CEN (prENV13606). The CEN approach is less flexible than the Synapses one as they have mandated a number of specific fields in the EHR header. In the actual EPR body, CEN uses concepts, which are similar to those, used in Synapses but the CEN concepts explicitly include more specific details e.g. the Related Agent attribute and the fact that each Record Component must include two attributes i.e. role and status.

3.8.2. Synapses and HL7

The structures used in HL7 are very different from those used in Synapses. The manner in which the Synapses concepts have been implemented by the Dublin Health Informatics Group (DHIG) is more comparable with level 3 of HL7 than with level 1. At the time of writing this Thesis, HL7 level 1 DTD is the only HL7 DTD available. To represent a Synapses EPR in HL7 would result in a very large XML document. This is due to the way in which the data is treated in both approaches. In HL7 the approach is to use blocks of free text to represent the EPR. This approach lends itself to

the HL7 idea of sections to represent the data. An EPR record represented in this fashion would not have a large amount of sections. The Synapses approach is to collect individual pieces of data from the data store and federate them as required by the user. It expects a section being included for each piece of data that Synapses collects in order to represent it in HL7 format. The granularity of the pieces of data depends on the requirements of the user. However, a piece of data in Synapses terms may simply be the value of a laboratory investigation. In HL7 this piece of data would need to be in its own section.

3.8.3. Synapses and ASTM

The approach taken by Synapses is a very flexible one that allows users to specify the data they would like to see in their EPR. ASTM on the other hand has been very specific in detailing the ELEMENTS to be included in various sections of the EPR. It is possible that these specific ASTM structures could be created using the Synapses approach of building SynODs. The proposed lists in ASTM could be represented as SynOD fragments in Synapses.

3.9. *Summary*

This chapter gave insight into three standards for Electronic Health Record Architectures.

It started with a background section about the electronic aspects of health records, followed by an explanation of the different record terminology used. Then, the Synapses Record Architecture is explained in detail, including the Synapses paradigm with the SynOM, SynOD and eventually the Record component. HL7 and CEN/TC251 are two standards organisations whose EHR approach has been described in the following two sections. The last part of this chapter compared and evaluated the HL7 and CEN/TC251 solutions against the Synapses methodology.

Chapter	<h1>4</h1>
---------	------------

Visual Modelling and Content Syntax

Syntax and semantics are the Yin and the Yang of the Web, and should be complementary to each other rather than independent — or worse, incompatible — from one another.

[PaSi02]

4.1. Introduction

The concept of separating content and presentation information into individual files undoubtedly increases the quality of the EHR key objectives: Maintainability, Extensibility, Reusability and Consistency (MERC). Content customisation in the course of information delivery is one of the most prominent aspects of modern publishing architectures. Nevertheless, a similar methodology during the (inevitably preceding) content definition and acquisition process is rarely applied, although it significantly advances the overall modelling process. Again, two access options are possible with a structural and a semantic view (dual mode).

Option one relates each entity in the model to a distinctive shape and/or pattern, which allows the user to graphically design and assemble a visualisation of the model without having to worry about the underlying data structure. The model itself (i.e. data structure) as well as the model visualisation (i.e. the graphic) is encapsulated (independent) which allows technology substitution by the use of alternative tools at any one time.

On the other hand, option two encapsulates not only one syntactic visualisation from the model, but additionally allows semantic visualisations for specialised tasks or

people. Various teams might work on different aspects of a model and prefer a view that is tailor-made to their specific assignment on the project such as a technical or medical task.

One important element of the Synapses project was the development and evaluation of a combined semantic and syntactic modelling application. The Record Structure Builder (RSB) was successfully used by both medical as well as technical personnel to design and maintain EPR models including the management of feeder system connections.

4.2. Multi View Modelling

Schema Development Interface (SDI) describes a combination of computer-computer and human-computer interfaces in order to customise the process of schema development. Similar to an API, which describes a programming interface of an application, SDI describes interfaces for schema development.

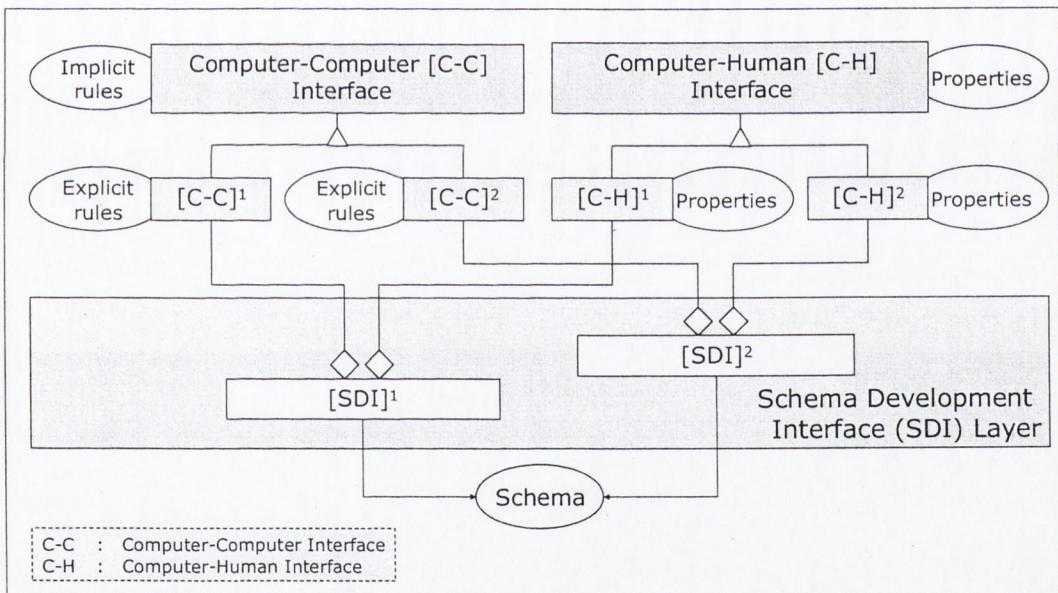


Figure 4-1: Schema Development Interface (Technical and Medical View)

An SDI incorporates two important aspects in customising the process of data modelling: Firstly, it defines the functions a user or user group can execute on a schema, such as adding and deleting ELEMENTS (computer-computer interface). This method is two-fold as constraints can be described in terms of implicit rules defined as part of the schema (e.g. "Element A is not allowed as child of element B.") as well as explicit rules in terms of user functionality (e.g. "User A is not allowed to add element

A at all.") (see **Figure 4-1**). In the previous example, IT personnel and medical personnel will be assigned different access rights to the schema.

Secondly, SDI tailors the visualisation of the schema for the specific user group, including layout and localisation, i.e. language and country-specific diagrams (computer-human interface). This leads to different user interfaces for technical and medical users in our example and makes access to the underlying schema or data model transparent to the user.

Using the hospital environment as an example, each user group such as IT support staff as well as medical personnel, rely on a customised interface during each phase of the schema development cycle. Multi View Modelling (MVM), as defined by the author of this Thesis, is a design methodology where two or more Schema Development Interfaces (SDI) are available to define and maintain a single schema, e.g. a DTD or XML Schema. The Record Structure Builder (RSB) is an application with MVM functionality.

4.3. Record Structure Builder

In the early stages of the SynEx project it became apparent that medical personnel would only be able to design and develop SynODs with the help of an MVM supporting application to hide the technical aspects of the model. In order to make this possible, the author of this Thesis redesigned and redeveloped an existing simple prototype of a SynOD modelling tool to meet the new requirements.

The Record Structure Builder (RSB) is a graphical editing tool to support the design of SynODs as part of an EHR. In general, the RSB application is flexible enough to be easily customised to any kind of hierarchical document development. However, the initial implementation was tailored for managing EHR data based on Synapses (see Section 3.4.2) concepts. In this environment, the RSB application was equipped with a dual mode SDI (see Section 4.2) to allow technical and medical personnel to cooperatively create SynODs which subsequently serve as the underlying Synapses Record Server (SRS) data structures. At the core of this RSB customisation are two distinct views on the SynOD in order to separate the medical from the technical aspects of the various entities during the modelling process. This ensures that abstract technical terms such as **FolderRIC** and **ComRIC** are hidden from the medical personnel and replaced by expressions from a medical vocabulary such as **department**

and **test result**. In addition, each entity is associated with a self-explanatory icon or pictogram. This methodology hides the abstract SynOM classes from the medical personnel, allowing them to focus on high level knowledge modelling in medicine. However, the technical view provides access to low-level coding issues such as data storage and retrieval.

The Synapses/SynEx architecture closely resembles the reference architecture for system components as described by Sheth and Larson [ShLa90]. However, the **Data** as well as the **Database** layer are not considered in the visual process of defining and generating SynODs. They are understood to be autonomous entities and therefore encapsulated from the real clinical (i.e. patient-) data as well as the underlying storage system. The **Commands** layer includes two basic types of interaction components. Firstly, instructions how the SRS communicates with the data store, e.g. query clauses to pull data into the EHR and secondly commands that lets the RSB communicate with the SynOD. The RSB itself is represented in the **Processors** layer, together with the SRS and other SynEx applications. SynOM and SynOD relate to the **Schemas** tier and mappings between SynODs (e.g. local and remote implementations) as well as clinical and technical views of one particular SynOD are part of the **Mappings** layer.

With the help of the RSB, a three step process is necessary to generate a SynOD and connect the SRS to the appropriate Feeder Systems. It is important to mention that this procedure does neither include the choosing of a physical data storage system (e.g. a database) nor the insertion of (clinical) data as reasoned earlier.

Step 1: Mapping EHR components to SynOM objects

The initial step is related to the customisation of the SDI. A number of generic EHR ELEMENTS such as **patient record**, **department** and **examination** have to be conceptually linked to one of the seven core Synapses component objects (see Section 3.4.2). This is the most important part of the process as it constitutes the foundations for the mapping definition between the technical and medical components. **Table 4-1** shows an example of an EHR fragment with mappings between technical and medical concepts. It proved to be good practice for the medical personnel to start the definition process by creating an abstract model of the medical components they want to record, using modelling tools and diagrams such as the Unified Modelling Language

(UML) or Entity Relationship Diagrams (ERD). Different approaches are used to define the hierarchical organisation of medical data, often arranged in similar style to time-, source- or problem-oriented EHR systems.

The result of this mapping process, namely the information describing how technical SynOM concepts relate to parts of the local medical record, is stored as an extension to the SynOM in the local SynOM database.

Table 4-1: EHR concept mapping

	Technical View	Medical View
Structural Components	RecordFolder	Patient Record
	FolderRIC	Department Problem Visit
	ComRIC	Examination
Data Components	DataRIC	Entry
	RecordItem	Date Value Unit
Link Components	ViewRIC1	n/a
	ViewRIC2	n/a

Step 2: Develop the SynOD

Two parts are necessary to develop an initial SynOD. In the first phase, the instantiation of multiple FOLDERRICS (e.g. Haematology, Dermatology), COMRICS and a single RECORDFOLDER, i.e. the root node of the hierarchy is required. Typical questions for the designer of the SynOD would include “*Which departments are supplying record information?*” or “*What data does a department provide?*”. In the second phase, the user defines the hierarchy between the instantiations. The whole process is normally undertaken by medical personnel, using the medical view in the RSB (see Section 0).

Step 3: Setting up connections to the feeder systems

The last step in the overall process is undertaken by the technical staff. Using the previously defined SynOD and switching to the technical view, they are able to define the connections to the underlying feeder systems, the data sources where the ac-

tual patient data is stored. This includes information on access methods, e.g. Hypertext Transfer Protocol (HTTP) and Open Database Connectivity (ODBC), a concrete query string and details about mapping the returned result sets into SynOM objects.

4.3.1. Functionality

The RSB tool was written in Visual Basic 5.0. Meta-data, required to construct SynOM, SynOD and mapping information is contained in an RDBMS database system (Microsoft® Access). The SRS does not interact directly with the RSB. Instead, the SynOD data model together with associated information, that is stored in the SynOD database, provides a common interface for both applications (see **Figure 4-2**).

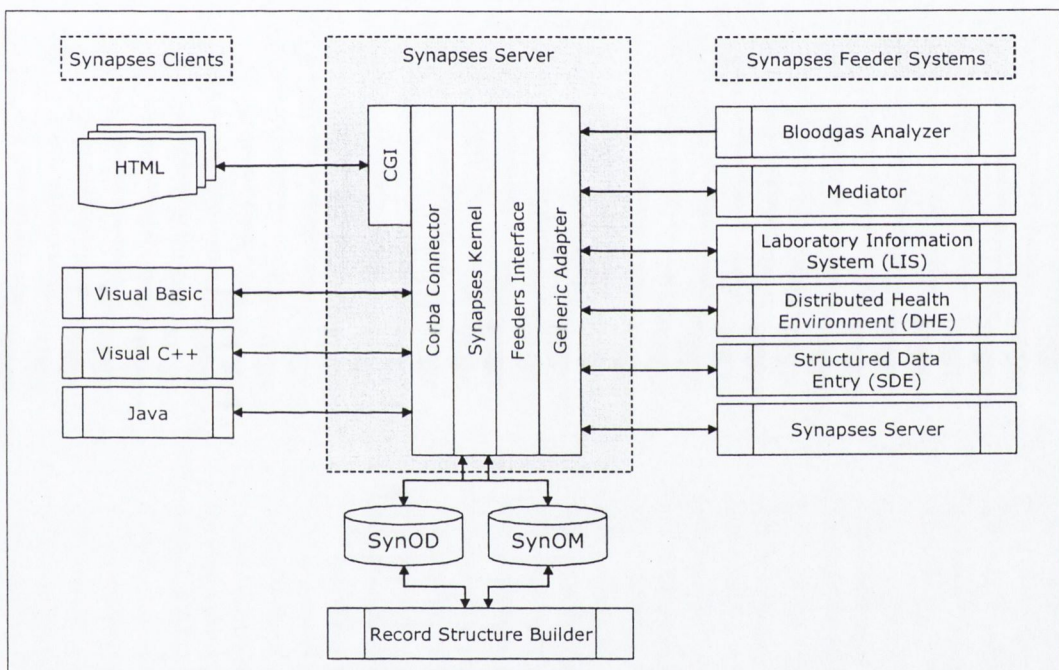


Figure 4-2: Synapses Architecture

The most prominent features of the RSB are all closely related to the SDI paradigm (see Section 4.2). For the development of SynOD models, two distinct SDIs have been developed, including one to be used by medical personnel (*medical view*) and a second one for the members of the IT support department (*technical view*). Both interfaces assist in constructing SynODs through context sensitive action menus in the TreeView. These menus are adapted on-the-fly as the user selects an action such as adding or deleting an entry from the TreeView. The system automatically retrieves

and enforces restrictions according to the user and schema permissions set in the SDI.

Another core function of the RSB is the **two-way mapping** of VIEWRIC1 ELEMENTS to query feeder system query aspects. In the first part, the VIEWRIC1 has to be mapped to a query string that is executed within the SRS, and retrieves the appropriate data from the feeder systems. At the time of writing, the query string was proprietary to the feeder system, i.e. it included a SQL command for an RDBMS or an XQuery string to query an XML database. An additional abstraction layer in combination with the existing Generic Adapter of the SRS would encapsulate the feeder system from the query string and provide more flexibility in integrating new storage systems (see Section 7.3.3).

A different part relates to result formatting and integration. On retrieving query results from the storage systems, the SRS has to convert the result sets proprietary to the underlying systems into XML fragments, which are subsequently included into the final SynExML document. Configuration information associated with both parts are included in the SynOD database and easily managed by the RSB.

Last but not least, the availability of an **XML generator** proved beneficial in the process of creating EHR SynODs and their subsequent use for exchanging EHR fragments over the Internet. At any stage in the development cycle, this add-on functionality to the RSB provided the SynOD authors with an instance (snapshot) of the actual model as an XML document, which is a SynExML document strictly speaking. Because the RSB does not have access to the **Generic Adapter** (see **Figure 4-2**) which manages the connections to the backend of feeder systems, this snapshot does not include any actual medical data. However, it describes the full structural layout of the final document, which can be used to test and evaluate data exchange scenarios in a hypothetical non-hospital environment, i.e. without live patient data and therefore not restricted by data protection.

4.3.2. Internal model

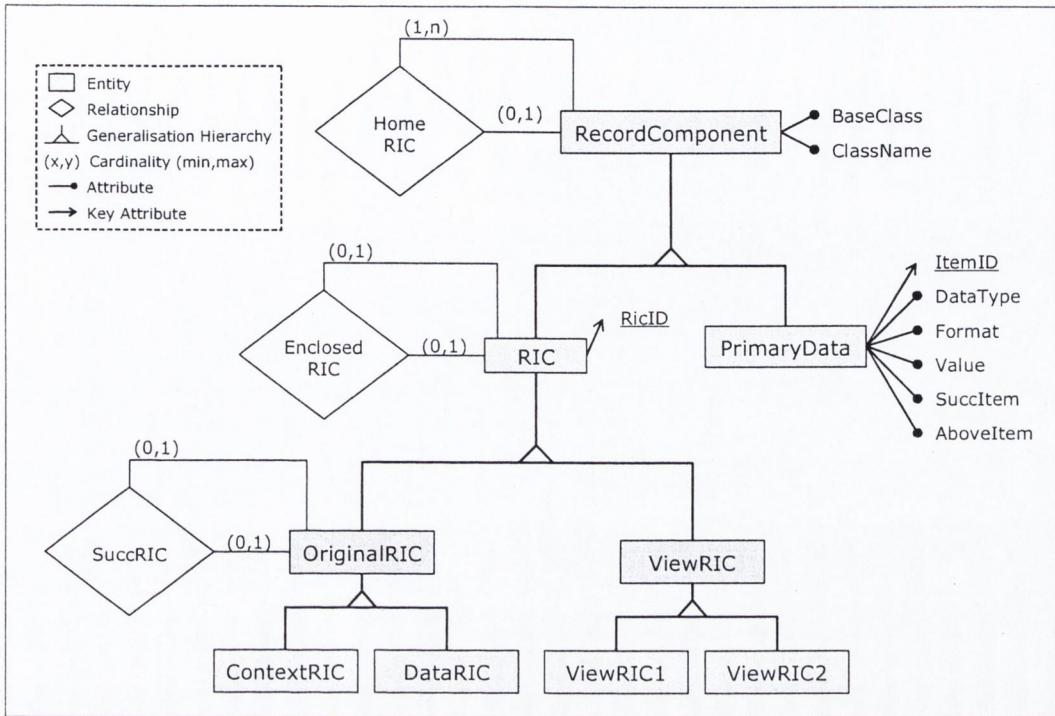


Figure 4-3: SynOM/SynOD datamodel (ER Diagram)

Figure 4-3 shows the Relational Model of the SynOD database. The prototype implementation used a RDBMS and as a result, key references (i.e. HomeRIC, EnclosedRIC, SucceedingRIC, AboveItem and SuccItem, see Figure 4-4) had to be added in order to replicate the hierarchical structure in a flat relational model. The additional structural information permits two-way conversion between the hierarchical data structure and its tabular representation and allows pre-order treewalking to build and navigate the SynOD.

The two diagrams on the right side of Figure 4-4 graphically explain the meanings of SynOM terms such as *HomeRIC* (pointer to the hierarchical parent), *EnclosedRIC* (pointer to the first hierarchical child) and *SucceedingRIC* (pointer to the next sibling in order), respectively *AboveItem* and *SuccItem*.

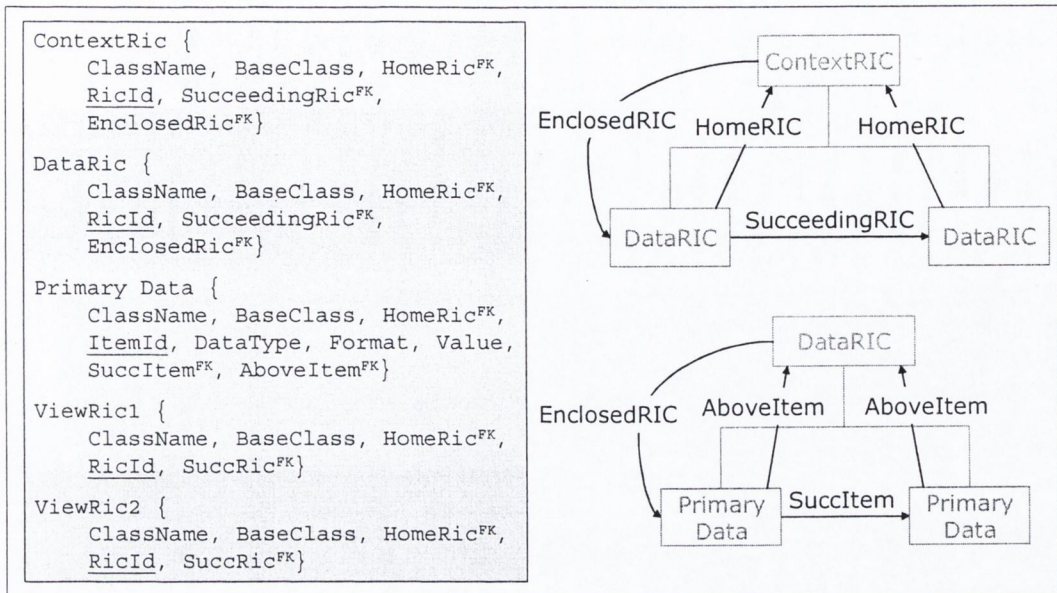


Figure 4-4: SynOD (Relational Model)

4.3.3. Medical View

As described in the previous sections, the RSB was used during the lifespan of the Synapses and SynEx projects to create hierarchical EHR models according to the Synapses EHR architecture (see Section 3.4). After an initial step of identifying a number of medical concepts and mapping them to their technical equivalent (SynOM class), a member of the medical personnel would develop a new EHR or EHR fragment for future use within the department using the RSB. If the department is already equipped with an operational SynOD, extensions are also easily achievable. **Figure 4-5** is a screenshot of the RSB in medical view (SDI). A TreeView Graphical User Interface (GUI component displays the hierarchical structure of the model using the previously defined medical concepts (left hand window). Each medical concept is associated with a distinct pictogram or icon to improve identification. In this picture, a global placeholder has been given to each medical concept. The right window of the application displays properties of the selected element, e.g. the *Vital Signs*. The SDI for the medical view hides all technical details as the medical developer does not need to know about the underlying SynOM structures. The expanded *QueryVR* ELEMENT in the TreeView, associated with a *VIEWRIC1*, contains information to connect to the feeder systems, which cannot be changed from the medical view. The only modifications allowed are adding, deleting and rearranging elements by their medical

concept and changing their *ClassName* value (in the properties window; right side of the application, showing the *ClassName* value “VitalSigns”).

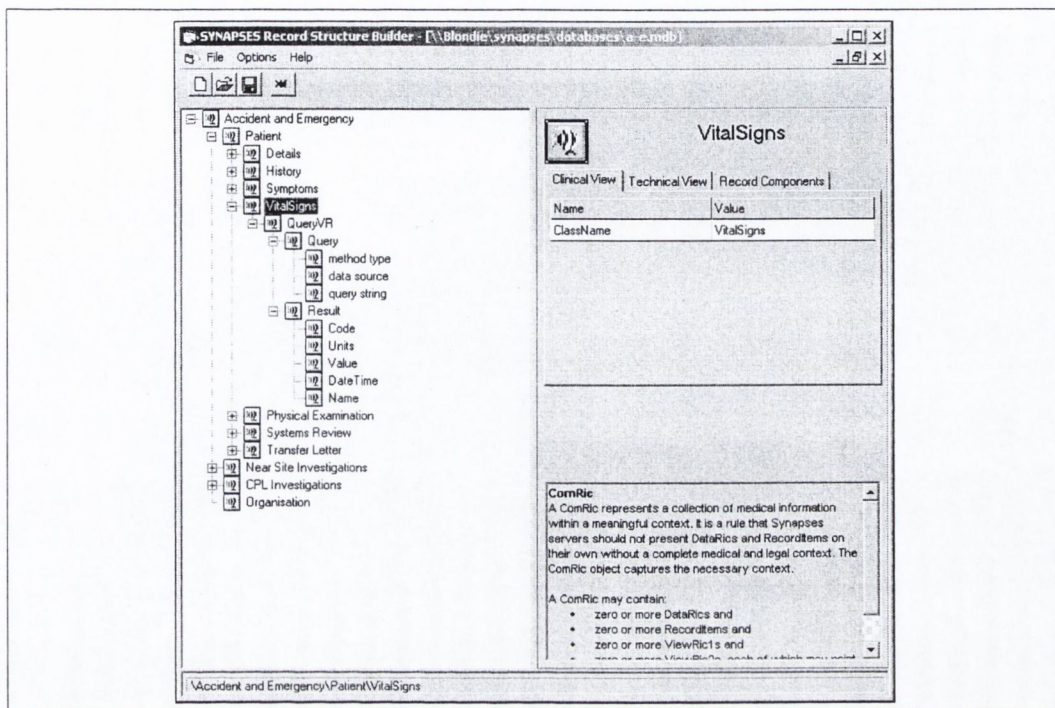


Figure 4-5: Medical View in the RSB

4.3.4. Technical View

The technical view (SDI) allows more modifications and offers extra functionality in the second step. **Figure 4-6** depicts a screenshot, similar to the one in **Figure 4-5**, but with a changed SDI (technical view). It also has (technical) pictograms associated with each of the SynOM classes which are applied in the TreeView. But more importantly, the number of accessible properties in the properties windows is increased and an additional field for the query string is available. Here, the technical developer specifies all settings related to the query mechanisms of the associated feeder system, including *method type* (e.g. ODBC, file system), *data source* (e.g. ODBC source name, file name) and finally the *query string* itself. Additionally, the mapping from the returning values to SynOM classes can be defined in *Result* element (child element of the *QueryVR* element).

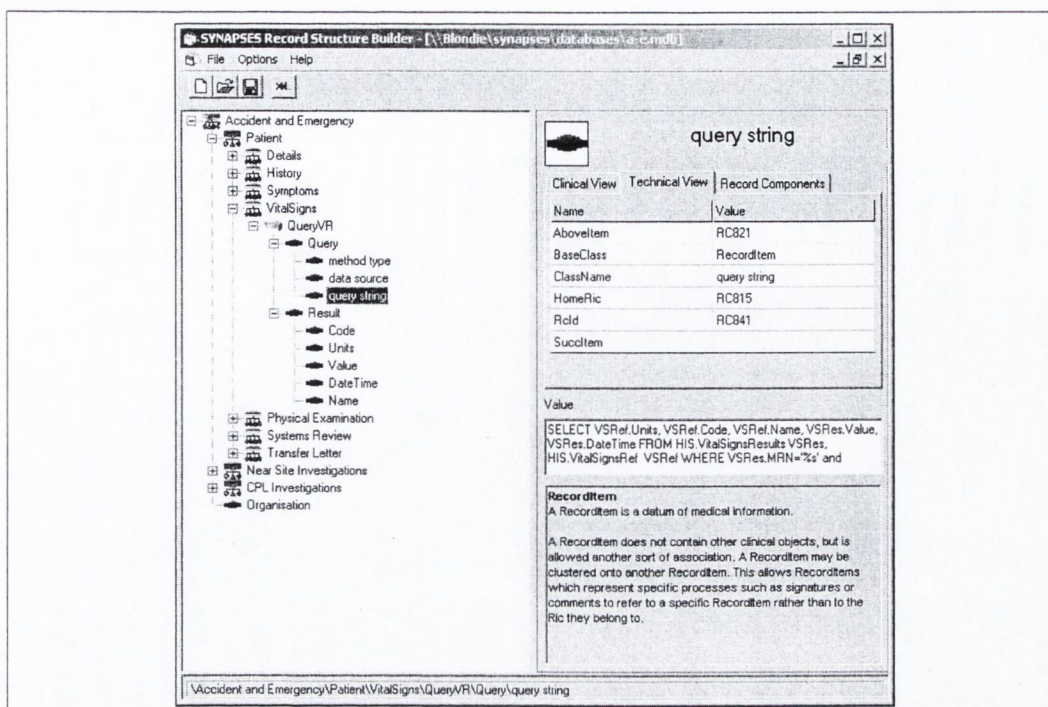


Figure 4-6: Technical View in the RSB

The differentiation between technical and medical view is also reflected in the use of two different link core SynOM components in the SynOD (see chapter 3.4.2), namely VIEWRIC1 and VIEWRIC2. The VIEWRIC1 component can only be edited from the technical view in the RSB and incorporates the query procedures for Synapses feeder systems. As described earlier, it contains detailed information on how to access the underlying feeder systems and map the result into XML syntax. On the other hand, the VIEWRIC2 component offers the functionality to link two EHR parts, independent of their location. This includes connections between EHR fragments that are stored locally or remotely and can even link from one person's EHR to another person's EHR (e.g. for mother-child relations).

As mentioned already, all information related to the query is proprietary to the associated feeder system. Ideally, an additional abstraction layer would provide further independence (see Section 7.3.3).

4.4. Other software tools

Schema development tools with similar or comparable features and functionality as the RSB are available from many software companies. They include dedicated DTD

and XML schema modelling applications²⁴ as well as modelling tools which have export and import methods which convert the internal (and proprietary) data model into DTD and XML Schema syntax. The combination of computer-computer and human-computer interface customisations is not unique to the RSB. However, the improved adaptation of the computer-computer interface determined by user preferences as well as schema logic is new to the field of XML schema modelling.

Modelling information is only the first step in the complex process of defining and setting up data exchange mechanisms between heterogeneous information systems. Another important aspect relates to the syntax and semantics of the messaging exchange format.

4.5. The Syntax of Messaging Standards

The development of messaging models and standards is multifaceted, similar to the concept of separating content and presentation information in the electronic publishing process. Three of the more apparent components of this methodology include an integrated modelling environment such as the RSB, a meta-language for the model such as a DTD or XML Schema, and a semantic vocabulary (SynExML, see Section 4.6). The meta-language together with the vocabulary automatically determines the document instantiation syntax, which could be SGML or XML in the case of a DTD and solely XML with XML schema.

There are many contradictory opinions on how to define the terms *message*, e.g. “*a communication in writing, in speech, or by signals*”²⁵, and *document*, a collection of messages, in the area of data exchange and in particular in relation to EHR systems.

It is very important to distinguish between standards for syntax and semantics definitions. A messaging standard has to be clear about both the syntax and semantics of the message. A common syntax ensures that a message can be read, whereas the semantic gives meaning to the message details. Unfortunately, the misconception of message and document often leads to confusion and misinterpretation as the following statement indicates:

²⁴ <http://www.xmlsoftware.com/dtd.html> lists a wide range of alternative tools.

²⁵ Merriam Webster Dictionary, <http://www.mw-com/>

“There are three messaging standards to consider, eXtensible Markup Language (XML), Health Level Seven (HL7) and United Nations Electronic Data Interchange for Administration, Commerce and Transport (UN/EDIFACT).”

[Mah01]

This citation is not correct as it compares two syntax/semantic standards, namely HL7²⁶ and EDIFACT²⁷ with the pure syntax standard (XML). HL7 as well as EDIFACT have both released semantic vocabularies to be used with their syntax, although in different domains (i.e. medical and trading). In the foreseeable future, these vocabularies will merge with XML syntax to create new messaging formats. In essence, highly specified, widely agreed and accepted standard semantics are combined with a highly accessible, easy to process and future-proof syntax.

As for this Thesis, a **message** describes a set of closely related information that, once created, never changes its composition and content. It is used as a fixed envelope with a distinct addressee (person or application) and a distinct lifespan to transmit information. After successful delivery, its only purpose is fulfilled and the message is subsequently deleted. For instance, a GP requests the finding of a blood investigation from a remote system. A message containing the results is created and sent on its way to the requester. This message might not reach the GP immediately for various reasons such as the GP not actually being connected to the system. As soon as the message is delivered, the GP system integrates the content of the message before the validity period of the message expires. On the other hand, **documents** are described as autonomous information collections, containing e.g. an entire EHR. Documents are constantly under construction, meaning that changes take place at irregular intervals, including modifications such as additions, annotations, corrections etc.

[Fox93] identifies and groups 23 prioritised formats for existing interchange formats to be used in healthcare. The evaluation covered properties such as efficiency, richness, complexity, ambiguity, flexibility, cost and practicality. Formats with priority one include ASN.1 (Abstract Syntax Notation 1), ASTM E1238, EDIFACT, EUCLIDES and ODA. At the time of publication, *Standard Generalised Markup Language* (SGML), the predecessor of XML, was categorised as “only” priority two.

²⁶ HL7 Homepage, <http://www.hl7.org/>

²⁷ EDIFACT Homepage (United Nations), <http://www.unece.org/trade/untdid/welcome.htm>

Nevertheless, with the advent of the first W3C recommendation for XML, all previously “first class” formats became undoubtedly second to SGML and XML. One of the first reports on SGML use for electronic patient records was published in 1996 [ALS96] and additional areas of application have been identified, including medical publishing, new drug submissions and clinical practice guidelines [DABM97].

4.5.1. EDI/EDIFACT

EDIFACT (Electronic Data Interchange for Administration, Commerce and Transport) defined and published a message structure and syntax [ISO9735] in order to electronically facilitate and support requirements and procedures related to the flow of information needed for the international transfer of goods. This specification is a basic requirement for paperless data and information exchange. A typical EDIFACT message (transaction) consists of data segments (“segment”), possibly framed by a header and trailer to create a transaction set, which is the typical EDI transmission unit. Each segment contains a string of data elements (“fields”), which in turn can be divided into data components (“component”). All elements are separated by delimiters. EDIFACT segments are closed by linefeed characters, i.e. each line contains exactly one segment of the message. The colon sign `:` is used to mark the beginning or end of EDIFACT fields and the plus-sign `+` divides EDIFACT components within one field.

Figure 4-7 shows an excerpt of an EDIFACT message which is used for trading information exchange scenario. Here in particular, a segment with computer environment details is presented taken from a software status report. The segment includes six fields, with the first field subdivided into three components. The first two components are mandatory, indicating the segment code qualifier (Computer Environment Details Code Qualifier, CED) and an identifier for the computer environment (Computer Environment Identification, 2). The last component of the first field as well as the following two fields are left empty. The fourth field contains the computer environment name (Linux) with succeeding fields for version (1.2) and release identifier (13). All other fields and components are optional and only added on demand. In this example, the last two fields are left empty and therefore not included in the message.

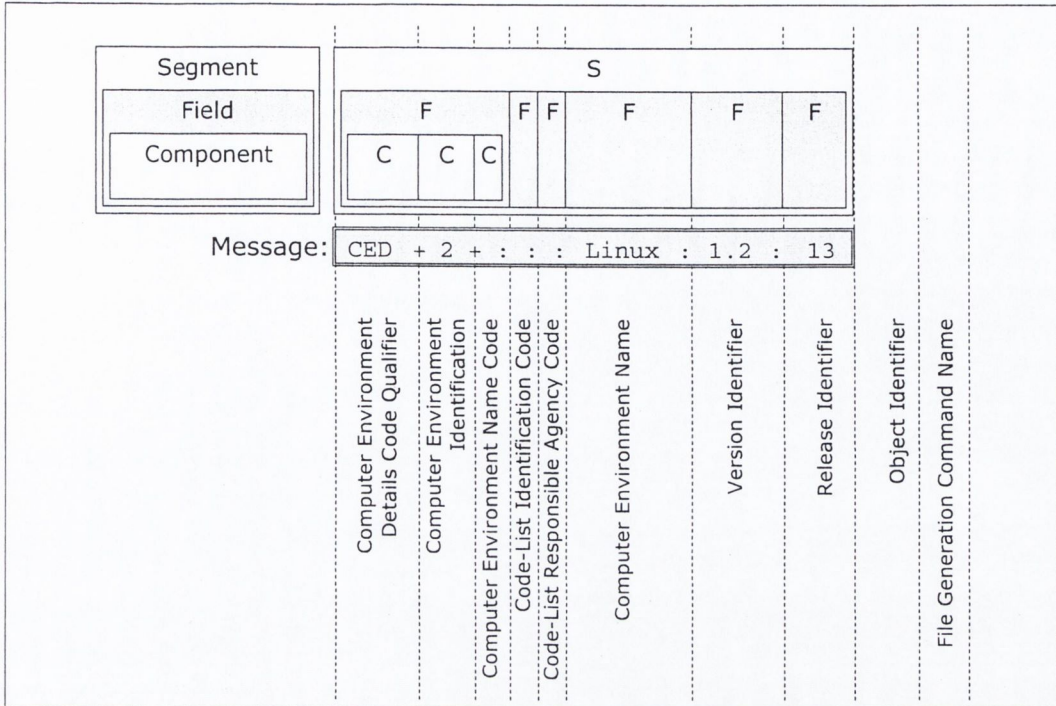


Figure 4-7: EDIFACT syntax design (CED message segment)

Software applications exist that automatically transform the less structured EDIFACT syntax into a more accessible XML document, e.g. a message for Computer Environment Details (CED) as shown in **Code 4-1**. It includes additional information from the EDIFACT specification such as resolved codes (line 10) and qualifier IDs (line 9). The “United Nations Directories for Electronic Data Interchange for Administration, Commerce and Transport” [UN00] includes tables on how to resolve codes into their corresponding text, e.g. the classification table 1501 (Computer environment details code qualifier) defines 2 as the “Code to identify the operating system, like DOS, VMS, etc. used in a computer environment”.

```

1 <message
2   xmlns="http://www.xml-edifact.org/LIB/xml-edifact-03/edifact.rdf"
3   xmlns:trsd="http://www.xml-edifact.org/LIB/xml-edifact-03/trsd.rdf"
4   xmlns:trcd="http://www.xml-edifact.org/LIB/xml-edifact-03/trcd.rdf"
5   xmlns:tred="http://www.xml-edifact.org/LIB/xml-edifact-03/tred.rdf"
6   xmlns:uncl="http://www.xml-edifact.org/LIB/xml-edifact-03/uncl.rdf"
7 >
8   <trsd:computer.environment.details>
9     <tred:computer.environment.details.qualifier uncl:code="1501:2">
10      Operating system
11    </tred:computer.environment.details.qualifier>
12    <trcd:computer.environment.identification>
13      <tred:computer.environment>Linux</tred:computer.environment>
14      <tred:version>1.2</tred:version>
15      <tred:release>13</tred:release>

```



```

16     </trcd:computer.environment.identification>
17     </trsd:computer.environment.details>
18 </message>

```

Code 4-1: CED message segment in XML syntax

4.5.2. HL7

The syntax used for HL7 messages (version 2.x) is similar to the EDIFACT syntax described in Section 4.5.1. Each message contains a set of segments, divided into fields; each field can be further divided into components. However, the required delimiter characters are a minor syntactical difference. HL7 defines default characters including the newline character (\r\n for Windows, \n for UNIX and \r for Macintosh operating systems) to separate segments, the pipe character | to separate fields and the caret ^ character to separate components. Additionally, the tilde character ~ is used to separate repeating elements in one component. The backslash character \ is used as an escape method, meaning that the following character should be interpreted a literal and not as a separating character. This is important if for example one of the delimiter characters is part of component content. If necessary, these default delimiting characters can be replaced with user-specific ones, which will be indicated in the second field of the message (**Code 4-2**, line 1).

Table 4-2: HL7 segment table extract for Patient Identification (PID) Segment²⁸

	SEQ	LEN	DT	R/O	RP	TBL#	ITEM	ELEMENT NAME
	1	4	SI	O			00104	Set ID - Patient ID
	2	20	CX	O			00105	Patient ID (External ID)
	3	20	CX	R	Y		00106	Patient ID (Internal ID)
	4	20	CX	O	Y		00107	Alternate Patient ID
	5	48	XPN	R	Y		00108	Patient Name
	6	48	XPN	O			00109	Mother's Maiden Name
	7	26	TS	O			00110	Date of Birth
	8	1	IS	O		0001	00111	Sex

²⁸ SEQ: sequence number; LEN: field length; DT: data type; R/O: required or optional element; RP: repeatable; TBL#: HL7 table for value definition; ITEM: unique identifier within HL7 V2.4 protocol.

```

1 MSH|^~\&|XRAY||CDB|||ORU^R01|K172|P
2 PID||PATID1234^5^M11||Jones^William||19610613|M
3 OBR||P8754^OE|XR1501^XR|71020^Chest X-ray
4 PA||198703281530|198703290800
5 OBX|1|TX|71020||It is a normal PA Chest X-ray|||||F

```

Code 4-2: HL7 message

Segment tables, available for each segment of an HL7 message, define segment type and structure. **Table 4-2** displays an extract of the segment table for the Patient Identifier (PID) segment as seen in **Code 4-2**, line2. Similar to EDIFACT syntax transformations, HL7 messages can automatically be converted into a different syntax, such as XML.

Code 4-3 shows an XML fragment of the HL7 v2.x message segment as displayed in **Code 4-2** (line 2). Here, each ELEMENT name is a combination of the segment name and the field sequence number within the segment (e.g. <PID.3>, line 2) or a data type combined with the component sequence number within the field (e.g. <CX.1>, line 3).

```

1 <PID>
2   <PID.3>
3     <CX.1>PATID1234</CX.1>
4     <CX.2>5</CX.2>
5     <CX.3>M11</CX.3>
6   </PID.3>
7   <PID.5>
8     <XPN.1><FN.1>Jones</FN.1></XPN.1>
9     <XPN.2>William</XPN.2>
10  </PID.5>
11  <PID.7>
12    <TS.1>19610613</TS.1>
13  </PID.7>
14  <PID.8>M</PID.8>
15 </PID>

```

Code 4-3: HL7 message (see **Code 4-2**) translated into XML syntax²⁹

The next version of HL7, namely version 3, will not only be based on a different EHR architecture, Clinical Document Architecture (CDA) [Detal01], but additionally uses XML as its native syntax.

²⁹ © LuMriX.net GmbH, Switzerland, <http://www.lumrix.com/hl7xml/>

4.6. SynExML

4.6.1. The role of XML

There are several advantages to the use of XML over existing syntax formats such as EDIFACT, to support the exchange of structured patient data between diverse sites. XML provides a hard- and software independent specification. Furthermore, both the Synapses based EHR model and XML are hierarchical/object-oriented data structures with the option to link their elements.

SynExML (see Appendix B), an XML vocabulary to mark up data according to the Synapses architectural model, is the first but fundamental step in creating easy to access EHR fragments, messages and documents, used as a possible basis for the Semantic Health Record (SHR). During the process of defining and evaluating the SynEx data exchange model, it was soon realised that the user requirements were a very close match with the XML design goals (see Section 2.2) that were published by the W3C [XML00].

The following section evaluates the adaptation of the ten standard design goals for XML (see Section 2.2) to the requirements of a SynExML DTD. It is important to mention that, although compared here, XML and SynExML are two different concepts: XML is a meta-language and defines the syntax of XML documents, whereas SynExML is an XML vocabulary and is used to define the SynOM semantics as described in Section 2.2.

1. XML shall be straightforwardly usable over the Internet.

SynExML shall be straightforwardly usable over the Internet, as projections show that virtually all hospitals and GPs will be connected to the Internet in the near future. Ideally, it should be used with Internet browsers that support XML as well as XSL transformation in order to reduce the processing on the server side, but more importantly to support the use of local stylesheets (see **Figure 5-3**).

2. XML shall support a wide variety of applications.

SynExML shall support a wide variety of applications. Indeed, using XML as the basic syntax, most existing applications are able to process SynExML easily, although to a varying degree. The creation of SynExML is implemented in an intermediate layer, strictly embedded between the data-storage- and the web-server application.

This implies no changes to the existing system; indeed only minor additions have to be made in order to enable XML functionality.

3. XML shall be compatible with SGML.

This does not apply to SynExML, because SynExML is an XML vocabulary and XML is a meta-language, used to create XML vocabularies (see Section 2.2). Furthermore, SGML will be superseded in the near future by XML.

4. It shall be easy to write programs that process XML documents.

As SynExML is an application of XML, every program that supports XML can be instantly adapted for use with SynExML. Due to the widespread use of proprietary software for medical applications, the use of XML and many freely available software tools might narrow the gap to establish data exchange at a broader level.

5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.

Optional Features in SynExML are allowed (see later in this section), but should be kept to the absolute minimum for compatibility reasons. This concept was used in early stages of the SynEx project in order to allow site specific extensions. However, it created difficulties during the implementation phase and was initially abandoned. At a second phase, the addition of site-specific data was crucial to some of the SynEx partners and it was agreed to allow the inclusion of one ELEMENT for site specific extension (see **Code 4-5**).

6. XML documents should be human-legible and reasonably clear.

As SynExML is specialised by the local institutions with semantic information, they have to provide a reasonably clear structure. Using the RSB, the user is supported in creating human-legible structures (using the medical SDI) and clear syntax (due to the continuing validation of the structure).

7. The XML design should be prepared quickly.

The SynExML core design was developed quickly, although the addition of site-specific extension increased the development time considerably, mainly due to achieving agreement between all parties.

8. The design of XML shall be formal and concise.

Following the Synapses concepts and semantics, SynExML is a formal and concise transformation of existing and well-developed structures into an XML DTD (see Appendix B) and XML Schema (see Appendix C).

9. XML documents shall be easy to create.

SynExML provides a structure that allows medical institutions to create their site-specific medical dictionaries easily and share them with others. The RSB with its MVM features offers the best solution to create SynExML documents.

10. Terseness in XML mark-up is of minimal importance.

Terseness in SynExML is of minimal importance; no abbreviations are used and everything is fully human-legible. However, the size of EHRs is often gigantic and the use of ELEMENT name size limitation, for example a maximum length of two or three characters, was discussed. Investigations showed that the size of ELEMENT as well as ATTRIBUTE names does not have much effect on the overall document length. Compression algorithms normally replace repeating character sequences with a single code throughout the document, which explains the lack of effect of name size lengths.

4.6.2. Mapping SynOM classes

The mapping from SynOD objects into XML ELEMENTS is a straightforward process, as indicated by the mapping table in **Figure 4-8**. All SYNOM base classes within the SynOD are mapped to XML ELEMENTS, except the VIEWRIC2, which is mapped to a link according to the elements from the XPointer and XLink specification. The XML document contains SynOD class attributes as XML ATTRIBUTES. All data retrieved from the feeder systems are mapped to XML ELEMENT content. This follows the CEN/TC251 XML Taskforce recommendation (discussed at the first CEN/TC251 XML Taskforce meeting in Gießen/Germany in 1999) that any patient data, explicitly validated and signed by the person in charge, has to be contained within XML ELEMENT content. Additional structural information is stored as XML ATTRIBUTES in the related XML ELEMENT.

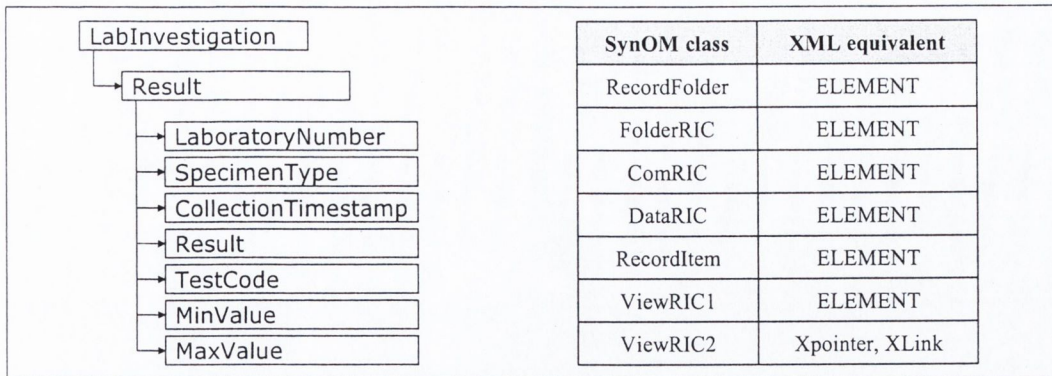


Figure 4-8: LabInvestigation ComRIC (object view)

Figure 4-8 shows a simplified diagram of a LabInvestigation object (COMRIC) within an EHR together with the mapping table of SynOM classes to XML equivalents. The root node (LabInvestigation) contains exactly one (of possibly many) Result objects (DATARIC), which itself acts as a container for a set of RECORDITEMS. On the other hand each RECORDITEM has a distinct value. An equivalent representation of the data structure using XML syntax, together with actual patient data, is displayed below in **Code 4-4**.

```

1 <LabInvestigationComRIC BC="ComRic" RCId="47" RecId="0123456">
2   <Result BC="RecordItem" RCId="50" DT="EMPTY">
3     <LaboratoryNumber BC="RecordItem" RCId="57">
4       S,96.0110835.R
5     </LaboratoryNumber>
6     <SpecimenType BC="RecordItem" RCId="58">B</SpecimenType>
7     <CollectionTimestamp BC="RecordItem" RCId="59">
8       3/3/1996 10.15
9     </CollectionTimestamp>
10    <Result BC="RecordItem" RCId="60">29</Result>
11    <TestCode BC="RecordItem" RCId="61">ALB</TestCode>
12    <MinValue BC="RecordItem" RCId="62">35.00000</MinValue>
13    <MaxValue BC="RecordItem" RCId="63">50.00000</MaxValue>
14  </Result>
15 </LabInvestigationComRIC>

```

Code 4-4: LabInvestigation ComRIC (XML view)

In the beginning it was assumed that a straightforward transformation from SynOM into an XML DTD (Document Type Definition) would not cause any problems. This later proved to be too optimistic as XML is based on a very flexible specification and allows 'colourful' variations.

The advent of XML presented a new solution to achieving an agreed common model for the exchange of Synapses EHRs. Not only did it provide a structure to define the model (DTD), but it also defined the syntax for the model implementation (instantiation, XML document). It was decided to investigate the use of the XML DTD as another modelling tool. Hopes in the new technology were high and local variations were accommodated by the introduction of optional (site-specific) extensions to the DTD. However, this rapidly led to a situation in which these local extensions exceeded the commonly agreed core part of the DTD by a factor of four. It quickly became clear that XML had not eased the problems associated with data exchange as had been hoped. Following a review, it was agreed to eliminate virtually all these locally proposed extensions and to concentrate on the common core, on which consensus was achieved. Unfortunately, there is no common methodology or standard that could be used as a guideline for this conversion. Due to the flexible nature of XML, many different (legal and acceptable) conversions are possible. The project decided to use basic DTD modelling as the only method, as this is the least limiting procedure but still clear to follow.

The second main goal for the introduction of XML as exchange format was to facilitate extensions to existing systems. The integration of the additional interfaces is relatively easy and quick to realise as the SynOM is already hierarchically organised and internally represented in object-oriented data structures. The transmission of structure and data within a single document, even a human readable one, is an advantage. Furthermore the Internet was the obvious choice for the basic transport mechanism as infrastructure and transport protocols are already in place. The connection to the Internet via web-server and scripting technologies, e.g. CGI (Common Gateway Interface), ASP (Active Server Pages), JSP (Java Server Pages), can be quickly achieved.

Last but not least, the separation of content (XML) and associated rules about how to present the content on the client side using e.g. *Extensible Stylesheet Language* (XSL) [XSL01] made the effort of developing *SynEx Markup Language* (SynExXML) in the SynEx project a success. Each of the SynEx data providers delivered a stylesheet, which describes how to present their data on the client side with the option of locally selecting another (preferred and/or personalised) view.

It was obvious that XML would be the ideal technology to use for wrapping up patient data and transmitting it between the various medical institutions.

It took nearly a year to develop a Document Type Definition (DTD) for the SynEx Markup Language (SynExML). It started with the initial design of a syntactical reference DTD (hub, core) that was agreed by all partners in the project, which were involved in the document architecture and data exchange group. In addition to the core format, each site declared specific extensions that were optionally integrated. As work and especially implementations progressed, it quickly became clear that a more rigid structure (i.e. with fewer options) would be necessary to provide and support the data exchange mechanisms between the various medical institutions. The latest version is back to a straight mapping of the existing Synapses Object Model [Syn98] into an XML DTD. Minimal extensions, which are mainly used to support site-specific functionality, are still present. A single `RCproperty` ELEMENT (see **Code 4-5**) was defined as part of the SynExML DTD (see Appendix B) to enable the inclusion of site specific data as a value-name pair. A redesign or upgrade of the SynExML DTD in the future should consider the use of XML namespaces [BHL99] for site-specific extensions.

```
1 <!ELEMENT RCproperty (#PCDATA) >
2 <!ATTLIST RCproperty Name CDATA #REQUIRED>
```

Code 4-5: DTD fragment to handle site-specific extension in SynExML

It is important to mention that the current version of SynExML gives greatest flexibility to the data providers (e.g. hospitals, GPs). This flexibility gives these data providers more control when adapting their current data schemas to the Synapses approach. However this flexibility decreases the ability to map record structures from one system to another easily.

Sharing of data (i.e. display of remote patient information within the local system) is the first step to automatically exchanging data and mapping it into the local system and structure.

Currently, only syntactical information is covered in SynExML. Semantic data is added by each institution while adapting their system to the SynExML specification. This makes mapping between systems and models extremely difficult and time con-

suming. The Semantic Web and ontologies (see Section 2.9) provide a first step to effectively approach this task.

4.7. *Comments*

It was felt that the RSB had a great influence on the acceptance of Synapses ideas and implementations, especially in the medical field. Its clear separation of medical and technical data followed the XML paradigm of separating content and presentation, i.e. the underlying content model and accompanying technical/medical views on the model. A clear and easy to operate user interface, similar to well-known navigation applications (e.g. file navigator) in style and use, helped to keep the learning curve low. Both parties involved, members of the medical and technical communities, were able to use the RSB after a short introduction and focus on setting up their individual environment. Results were almost immediate, as development teams were confident that they could make the necessary changes to their own view.

The customisation of data modelling is only one aspect of supporting data exchange and integration, paving the way to the Semantic Health Record. The second and more important part is content definition and content representation. EDIFACT and the related HL7 syntax do not provide enough flexibility in terms of access, processing and interconnecting fragments or complete EHRs. XML on the other hand, the core of the rapidly increasing XML family of languages, provides all the features needed to ease if not enable automated record exchange and integration. SHR, the medical adaptation of Semantic Web concepts for EHRs, might not be just “around the corner”, but continuing efforts on both sides, the medical to define semantics and the technical to define syntax and exploit technologies, should finally be successful.

As part of this enterprise, the implementations described in this chapter show the range of related domains from the beginning (content modelling) to the end (syntax definition). SynExXML has proven successful in a small to medium-sized and manageable environment such as the SynEx project. Since then, emerging technologies such as RDF and OWL (see Section 2.9) offer new theories and techniques to extend the Synapses concepts, and make the distribution more widespread and diverse. The RSB together with the SynExXML vocabulary mark the beginning of this endeavour.

4.8. Summary

This chapter has presented the Record Structure Builder for GUI based EHR modelling and an exchange format for EHRs based on the Synapses paradigm and XML syntax.

The first section explained the concept of Multi View Modelling with the aspect of personalised interfaces during each phase of the modelling process. The Record Structure Builder, an implementation with MVM functionality specifically designed to assist the development of SynODs, is part of the following section. To complete the pre-work of information exchange, the modelling process had to be finished with effective data exchange syntax. After an introduction to solutions from the EDI-FACT and HL7 groups, an approach based on the Synapses paradigm (SynExXML) has been explained and discussed in the last section of this chapter.

Chapter	<h1>5</h1>
---------	------------

Content Integration: Presentation/Exchange

One technically feasible way to implement seamless interchange of patient care records is simply to require all hospitals and health care agencies to use a single standard system dictated by the government [...]. In an environment where hospitals are going out of business on a daily basis and many health care agencies are in deep financial difficulty, however, a scheme that en masse is hardly practical.

The other way to enable interchange between heterogeneous systems is to adopt a single industry-wide interchange format that serves as the single output format for all exporting systems and the single input format for all importing systems. This is, in fact, the purpose for which SGML was initially designed, and XML simply carries on this tradition.

(Jon Bosak) [Bos98]

5.1. Introduction

Seamless content integration, transparent to the user, has always been the ultimate goal in decentralised heterogeneous information architectures. The mapping of syntactic as well as semantic structures is the key to information interoperability across multiple systems. Both forms rely on widely agreed and implemented standards, which it may be difficult to achieve within the community.

On the one hand, semantic standards (e.g. terminologies, vocabularies) are set through a difficult and often time-consuming procedure, compromising input derived from discussions and experience as well as advice by experts in the area. On the other hand, syntax mapping is a purely technical task. It advanced from managing proprietary (binary) and difficult to access and interpret (e.g. CSV, EDIFACT) formats to XML, a common computer-interpretable form that remains complete and

consistent when exchanged among different computer systems. This shift rendered the mapping between different syntactical forms obsolete.

In one of the earliest publications about XML technologies [Bos98], Jon Bosak clearly illustrates the fundamental requirements for two potential and generic data exchange scenarios. Both settings reference a necessary foundation for a semantic data exchange (core) that requires a fully agreed semantic standard. The difference is the depth or range of the standard. It ranges from “floating at the centre” allowing for an open system implementation to “stretching out” into every system but restricting flexibility (see **Figure 5-1**, scenario 2 and 3).

The SynEx project pursued an incremental approach to data integration. It started with a semi-semantic information presentation (shallow integration), which will gradually develop into a full semantic information exchange (deep integration).

5.2. Background

Figure 5-1 shows three different types of mapping scenarios. Scenario 1 on the left side illustrates *Multi-Mapping*, a situation where each institution participating in data exchange, has to provide a mapping to and from each other party. This approach provides total freedom to local sites with respect to data and information engineering, but considerable maintenance work has to be undertaken in order to keep the mapping files up-to-date. Although the number of mappings for each site increases linearly with additional parties in the network, the overall number $N_{total} = \frac{1}{2}(n \times (n - 1))$ increases with polynomial growth and quickly reaches a critical and barely manageable quantity.

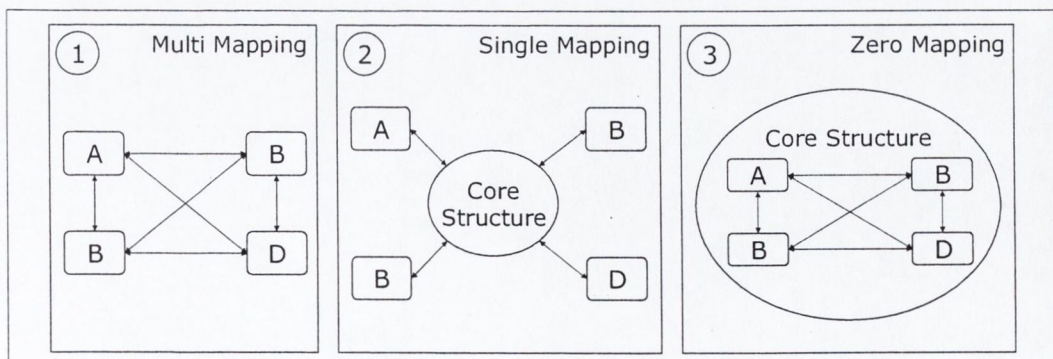


Figure 5-1: Mapping scenarios with and without core agreed structures

Scenario 2 provides a core structure which all parties support through *Single Mapping* to and from their local implementation. It limits each site to some extent as the local data model must be “mappable” to the core structure, e.g. each site should use an equivalent (but not necessarily equal) model. This implies that each site is free to design and organise its local data structures but must maintain a single (two-way) mapping. This approach ensures that the overall number of mappings required grows only linearly.

Finally, scenario 3 does not need a mapping algorithm as each site guarantees to implement the core model. *Zero Mapping* prevents local installations from high mapping maintenance work, but forces the implementation of the core structure, which most likely does not include properties required to handle local peculiarities.

Apart from defining the physical location of the mapping process, i.e. locally or centrally as described earlier, the logical level on which the mapping should occur has to be defined.

5.2.1. Syntax transformations

The transformation of syntactic structures is (theoretically speaking) relatively easy and can be achieved automatically and with only minimal human intervention. However, a precise definition of the source and the target model as well as a comprehensive mapping table is inevitable.

The most basic transformation is a plain syntax translation. It relates to converting information from one distinct source syntax into a different syntax (target) without structural reorganisation. An example is described in Section 4.5.1 (EDI/EDIFACT), illustrating two message fragments of exactly the same content and structure, but written out using EDIFACT as well as XML syntax. In other words, content representation in EDIFACT or XML syntax resembles alternative visualisations of the actual message content.

In a more complex scenario, items (e.g. elements) at a specific hierarchical location in an XML document are converted to attributes of another element. In this case, the transformation includes a restructuring of the content message (see **Code 5-1**) but without changing the message content. An attribute-to-element transformation was performed from the `position` ATTRIBUTE (left code snippet: line 4, line 7 and line 10) to ELEMENTS (right code snippet: line 4, line 8 and line 12). Similarly, an element

split occurred from the Name ELEMENT (left code snippet: line 5, line 8 and line 11) to a combination of Firstname and Lastname ELEMENTS (right code snippet: line 5/6, line 9/10 and line 13/14). Similar to the previous example, this transformation also accomplishes a structural re-organisation only without taking the wider context into consideration.

<pre> 1 <?xml version='1.0' ?> 2 3 <Team ID='Mari-Cha-IV'> 4 <Person position='Skipper'> 5 <Name>Robert Miller</Name> 6 </Person> 7 <Person position='Navigator'> 8 <Name>Mike Quilter</Name> 9 </Person> 10 <Person position='Trimmer'> 11 <Name>Brett Jones</Name> 12 </Person> 13 </Team> </pre>	<pre> 1 <?xml version='1.0' ?> 2 3 <Boat name='Mari-Cha-IV'> 4 <Skipper> 5 <Firstname>Robert</Firstname> 6 <Lastname>Miller</Lastname> 7 </Skipper> 8 <Navigator> 9 <Firstname>Mike</Firstname> 10 <Lastname>Quilter</Lastname> 11 </Navigator> 12 <Trimmer> 13 <Firstname>Brett</Firstname> 14 <Lastname>Jones</Lastname> 15 </Trimmer> 16 </Boat> </pre>
--	--

Code 5-1: Syntax Transformation on the basis of an XML document

An XSL transformation that is solely based on syntactic information is shown in **Code 5-2**. Note that all references in the stylesheet are based on syntax, e.g. hierarchy level, elements and attribute. It does not contain any references to markup and/or content information in the XML document.

```

1 <xsl:stylesheet
2   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3   version="1.0">
4
5   <xsl:output method="xml" indent="yes"/>
6
7   <xsl:template match="/">
8     <xsl:apply-templates select="*" mode="first_level" />
9   </xsl:template>
10
11  <xsl:template match="*" mode="first_level">
12    <Boat name="{@*}">
13      <xsl:apply-templates select="*" mode="second_level" />
14    </Boat>
15  </xsl:template>
16
17  <xsl:template match="*" mode="second_level">
18    <xsl:element name="{@*}">
19      <xsl:apply-templates select="*" mode="third_level" />

```



```

20     </xsl:element>
21 </xsl:template>
22
23 <xsl:template match="*" mode="third_level">
24     <Firstname>
25         <xsl:value-of select="substring-before(., ' ')" />
26     </Firstname>
27     <Lastname>
28         <xsl:value-of select="substring-after(., ' ')" />
29     </Lastname>
30 </xsl:template>
31
32 </xsl:stylesheet>

```

Code 5-2: XSLT code for syntax transformation (based on syntax rules)

In contrast, **Code 5-3** defines a transformation with the exact same output, but uses a number of semantic conditions. These include the existence of explicit hierarchies (line 37, Name as child of Person) and the presence of specific elements and element names as well as attributes and attribute names.

```

1 <xsl:stylesheet
2   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3   version="1.0">
4
5   <xsl:output method="xml" indent="yes"/>
6
7   <xsl:template match="/">
8       <xsl:apply-templates select="Team[./*[@position='Skipper'] and
9           ./*[@position='Navigator'] and
10          ./*[@position='Trimmer']] " />
11   </xsl:template>
12
13   <xsl:template match="Team">
14       <Boat name="{@ID}">
15         <xsl:apply-templates select="Person" />
16       </Boat>
17   </xsl:template>
18
19   <xsl:template match="Person[@position='Skipper']">
20       <xsl:call-template name="crew" />
21   </xsl:template>
22
23   <xsl:template match="Person[@position='Navigator']">
24       <xsl:call-template name="crew" />
25   </xsl:template>
26

```

```
27 <xsl:template match="Person[@position='Trimmer']">
28   <xsl:call-template name="crew" />
29 </xsl:template>
30
31 <xsl:template name="crew">
32   <xsl:element name="{@position}">
33     <xsl:apply-templates select="Name" />
34   </xsl:element>
35 </xsl:template>
36
37 <xsl:template match="Person/Name">
38   <Firstname>
39     <xsl:value-of select="substring-before(., ' ')" />
40   </Firstname>
41   <Lastname>
42     <xsl:value-of select="substring-after(., ' ')" />
43   </Lastname>
44 </xsl:template>
45
46 </xsl:stylesheet>
```

Code 5-3: XSLT code for syntax transformation (based on semantic rules)

5.2.2. Semantic Transformation

Transformation of semantic structures is far more complex than syntactic transformation and involves fundamental knowledge of the information domain, ideally provided by domain experts. Similar to natural language translation, numerous mapping levels are available. Each degree increase adds more depth to the transformation but (at the same time) presumes more robust (automatic) understanding of the surrounding context.

As described in the previous section, single entities of various lengths, such as a character or a phrase, are mapped from source to target model. Simple examples include the conversion of a single character to another character representation or the encoding of a single word independent of its meaning and context. In principle, the simple syntax transformation is a view change, e.g. expressing one concept through different visualisations, such as an English phrase or a standardised code.

[Kar00] describes three scenarios where integration tools are used to provide various levels of semantic interoperability. The tools are related to architectural locations and match closely with the three scenarios presented in **Figure 5-3**. They include a local file mapper (A), a middleware solution (B) including a common reference model and

finally a universal language, which has to be implemented by all entities participating in the data exchange and integration. Interoperability difficulties solely related to the content or data value are discussed as part of a detailed study [SSR94] which focuses on the potential resolvability of heterogeneity and problems arising such as incomplete or lossy transformations. Descriptions of situations where structural semantic heterogeneity and syntactic heterogeneity can cause mapping difficulties are described in [Col97].

Table 5-1 gives a brief tabular overview of the three central levels with examples from both scenarios, namely human and computer information exchange.

Table 5-1: Mapping Levels: Human Communication vs. Computer Messaging

	Human Communication	Computer Messaging
Language	English, German etc.	Document instance, e.g. SynExML document
Vocabulary	Words	Terms, codes, definitions, data dictionary
Character Set	Latin, Arabic, Chinese etc.	ASCII, UNICODE etc.

Starting with a mapping on the *Character Set* level, each single character from the source set is converted into a single character of the target space. “*Greeklisch*”³⁰ is a typical example used by many Greek authors to create Greek text using a computer keyboard with English characters. It is obvious that this conversion is a simple mechanical replacement: each character is replaced with a distinct substitution character regardless of its surrounding context. However, a syntactic conversion does not allow interpreting and understanding the target word without knowledge of the source model. For instance, “*Kalhmera*” in Greeklisch does only make sense if the reader does know the meaning of “*Καλημέρα*”, the letter-based equivalent in Greek.

Vocabulary Mapping describes a situation where a word or phrase from the source model is encoded into another word or code of the target model, e.g. the English word “*Flower*” is translated into the German word “*Blume*”; similarly, the medical condition “*Acute Appendicitis with peritoneal abscess*” relates to the ICD-10 code

³⁰ “*Greeklisch*” is the term for Greek language written with the Latin alphabet (‘English’). For example, the phrase “*Καλημέρα, πως είσαστε;*” (“Good morning, how are you?”) translates to “*Kalhmera, pws eisaste?*” in Greeklisch. © <http://www.wikipedia.org/wiki/Greeklisch/>

“K35.1” (International Classification of Diseases, revision 10) [ICD10]. Unlike the previous example, Vocabulary Mapping (in principle) is a combination of structural and semantic mapping and must take the context of the word into consideration. For example, the German word “Bank” translates into the English equivalents “bank” and “bench”. A solely structural conversion, e.g. word-by-word, most likely corrupts the logic of the sentence (“Please leave your cash with the person at the bench/bank.”). Automatic translation programs often use the word-by-word approach as a last resort, if the context is not automatically and sufficiently comprehended. Naturally, this method lacks quality and especially round-trip translations, e.g. English-German-English, clearly show the limitations of this technology.

5.2.3. Combined Syntax/Semantic Transformation

Last but not least, a combination of syntactic as well as semantic mapping proves to be the most challenging of all transformation tasks. Known from automated language translation, *Language Mapping* (see **Table 5-1**) is the most sophisticated but difficult mapping level. It includes combinations such as

- Syntactic mapping controlled by semantic properties:
Restructure entity-X from model-X into entity-Y from model-Y, but only if entity-X contains a logical reference to entity-Z.
- Semantic mapping controlled by syntactic properties:
Convert entity-X from model-X into entity-Y from model-Y, but only if entity-X also contains information about entity-Z.
- Semantic mapping controlled by semantic properties:
Convert entity-X from model-X into entity-Y from model-Y, but only if entity-X contains a logical reference to entity-Z.

5.3. Transformation with XSLT technology

XSL and in particular XSLT (see Section 2.8) is one technology for describing model mappings and implementing transformation functionality. An XSL transformation application (“XSL engine”) transforms XML documents, into the target document according to the rules of the associated XSL style sheet. Although the source document must be designed in XML syntax, the syntax of the target document varies with possibilities from plain text to binary, but is most often XML. XSL trans-

formations describing data mappings between two models A and B are created in two different files independently from each other. In an ideal environment (see **Figure 5-2**), a core and abstract transformation or mapping architecture (*Hub*) identifies and describes the relations between elements of the two document models. A compiler application creates two transformation files (*stubs*) with opposing characteristics, i.e. $\text{Transform}^{A \rightarrow B}$ and $\text{Transform}^{B \rightarrow A}$, which are used to subsequently transform instances of the A model into instances of the B model and vice versa.

Thorough investigation proved that an abstract description of the two-way mapping algorithm together with the implementation into (e.g. XSL) transformation rules is very complex. In particular the abstract definition and execution of semantic mappings proves extremely difficult with respect to content modifications. For instance, a round-trip scenario, where deletions take place in the transformation from A to B, B will consequently still carry these deletions in order to provide this data for the reverse transformation from B to A.

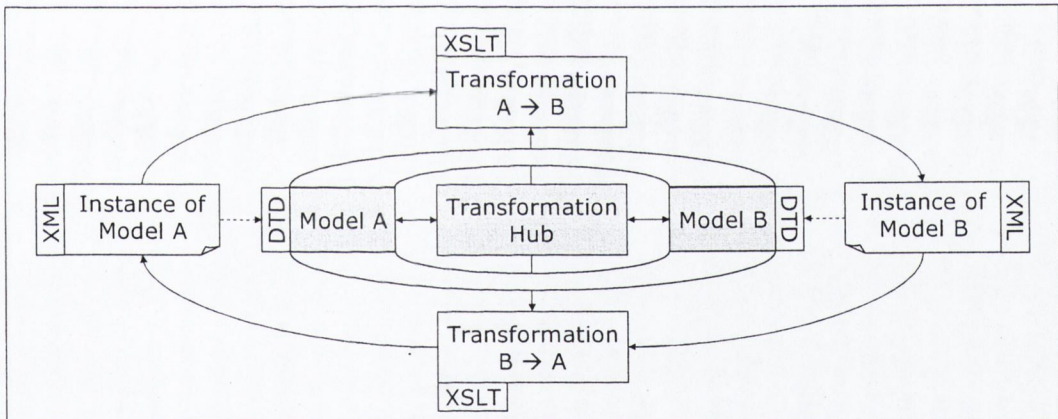


Figure 5-2: Transformation Modelling with Transformation Hub

The model mapping or transformation process can be implemented at two different physical locations, the server (originator) or client (receiver) site. Both approaches are known and widely used in electronic publishing scenarios using Internet technologies. Depending on the processing power, the availability of processing engines and the need for information processing after delivery, three different scenarios are common practice (see **Figure 5-3**):

- Graphic (A) describes a scenario where the receiving application does not have the processing power or facilities to execute the transformation. The server takes over the processing and delivers content in a format that the cli-

ent is capable of post-processing and presenting. Data is being customised with device specific mark-up (e.g. to be displayed on a Personal Digital Assistant, PDA, or mobile phone) or application specific mark-up, including browser applications such as Internet Explorer and Netscape Navigator.

- Option (B) is similar to (A) in that the server again delivers both files that are necessary to execute the transformation, i.e. the XML content file together with the XSL transformation rules. However, the receiving client application must be able to process both files.
- Option (C) is an extension of (B), exploiting the flexibility of location independent stylesheet application. It allows for an advanced and ideal scenario: Instead of using the transformation file pre-defined and delivered by the content originator, it applies transformation rules from a local file to the source content file in order to take care of local specifics.

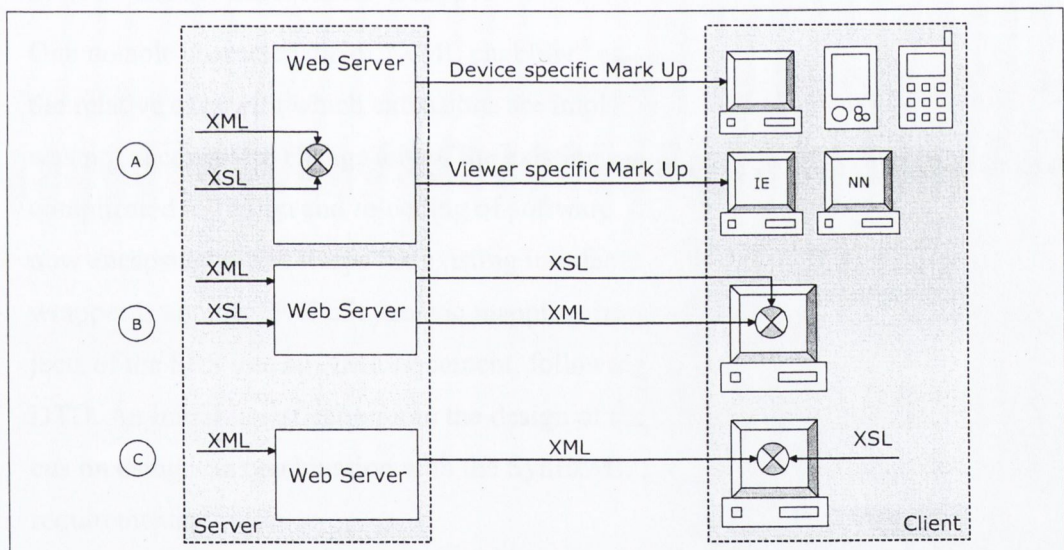


Figure 5-3: Server- vs. client side transformation and mapping of medical data

5.4. SynEx

Similar to the mapping and transforming concepts described in the previous sections, the task of data exchange can be divided into two distinct categories, which are characterised by their varying integration depths. The first category is comparable to the syntactic transformation method and described as sole information exchange through information visualisation (**shallow integration**). In other words, the local application seamlessly integrates the data, which was received from a remote source, by display-

ing it as part of the application. Further processing, e.g. transformation into the local data structure is not intended. XML documents for instance are typically represented by using a hierarchical software component such as a TreeView showing the hierarchical document structure with widely accepted contains/contained (parent/child) relationships. However, the data is only interpreted on a syntactical level and human knowledge is needed for further semantic interpretation. On the other hand, a **deep integration** strategy describes a scenario in which received data is first converted, adapted and integrated into the local data architecture before it is displayed as part of the application. The differences are clear: shallow integration's only purpose is the visualisation of remote data within the local application. Deep integration, however, extends the shallow integration method by additionally mapping the remote to the local data model and therefore allowing one to physically add (integrate) remote data into the local data structure.

5.4.1. Server-side: XML Wrapper

One notable characteristic of "XML enabling" existing and operational software is the relative ease with which extensions are implemented. In the case of the SRS, it was not necessary to change any of the existing modules and components through complicated re-design and re-coding of software. Instead, a minimal XML wrapper now encapsulates and wraps the existing interfaces. Strictly speaking, the XML wrapper is another level of syntactic mapping, transforming the proprietary data objects of the SRS into an XML document, following the definitions of the SynExML DTD. An initial investigation into the design of the XML wrapper, with a special focus on using it in combination with the SynExML format, identified the following requirements:

Requirement 1: Use of existing Synapses components

The re-use of existing Synapses components and code as well as the future use of Synapses in one HISA is the prime requirement of the XML wrapper. An already existing and working SRS shall be extended and completed with general XML functionality. Simplicity and speed of implementation are given emphasis over the integration of additional services.

Requirement 2: Platform and vendor independent solution

The W3C as the official institution behind the family of XML languages guarantees a platform and vendor neutral development and specification in the past, present and future.

Requirement 3: Easy access to data

The Document Object Model (DOM) and Simple API for XML (SAX) are platform- and language-neutral interfaces that will easily allow programs and scripts to dynamically process (i.e. access, update and format) content, structure and style of documents.

Requirement 4: Use of Metadata for further characterisation

XML allows machine-understandable data descriptions ('Metadata', 'data about data') to be included into the XML document. Metadata helps systems to enhance the automatic interpretation of information by network applications such as search engines, browsers and other autonomous network robots. It preserves the source data structure, includes additional semantic information and leads to better document discovery services (searching, reasoning, analysing and evaluating).

Requirement 5: Increase of speed

EHR often contains many hundreds or thousands of objects. One crucial weakness of the first SRS implementation was the implementation of Common Request Broker Architecture (CORBA) client-server environment between client and server together with the fine-grained SynOM objects. Thus, the CORBA client established new connections for every object transmission/migration. In the case of medical records based on SynOM paradigm, the number of record components easily reached into the thousands which slowed down the EHR transmission to an unacceptable level. This was caused by a maximum speed of approximately 50 connections per second at the time of implementation. The use of XML must address this issue and significantly reduce transmission time.

Requirement 6: No Loss of information

The solution must guarantee that neither patient data nor the data structure of the EHR is lost while mapping from system A to B. Round-trip mapping shall lead to an exact replication of the original EHR, including data and structure.

Figure 5-4 shows the conceptual integration of the XML wrapper into the existing SRS. The diagram, a simplified version of the SynEx architecture diagram (see **Figure 4-2**), shows SRS clients on the left side, the SRS itself in the middle and various feeder systems on the right side. In order to enable the SRS “talking XML dialect”, two additional methods were implemented to form the **XML wrapper**. RetrieveRecordShape() and RetrieveXMLComRic(). Both methods are fully integrated into the SRS and accessible via the already existing CORBA interface. At present, three CGI scripts (one for each XML wrapping method and one to retrieve a list of available patient records) allow web clients to virtually connect to the added XML wrapper methods by calling the associated CORBA interfaces. A future migration from CGI scripts to web services is advisable and proposed as part of this Thesis (see Section 5.4.2).

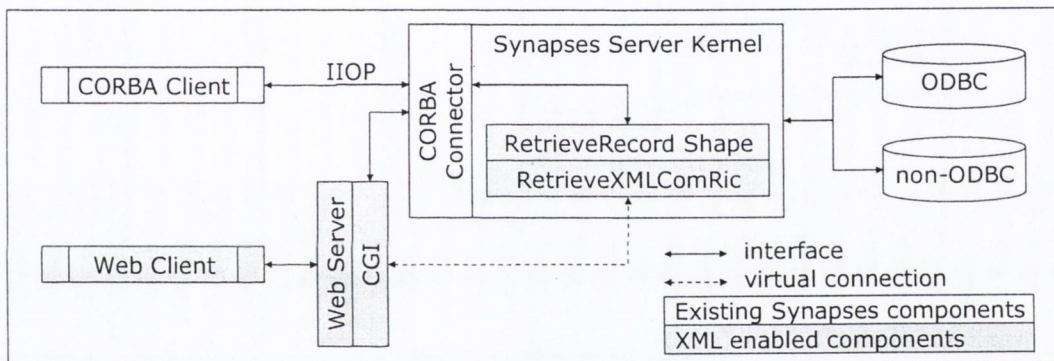


Figure 5-4: XML enabled Synapses Server

RetrieveRecordShape()

The RetrieveRecordShape() method wraps all SynOM classes down to the level of COMRIC (i.e. RECORDFOLDER » FOLDERRIC » COMRIC) and potential VIEWRIC2 instances into one single XML document. This file represents an (empty) fragment of the entire skeleton or shape of the FHCR; 'real' or 'live' data (i.e. patient data) from the connected feeder systems is not included. This allows the user to navigate through the structure of the FHCR on the client side and select a specific part of the FHCR (i.e. a COMRIC), which will be downloaded on request as one large bulk of data.

RetrieveXmlComRic()

The second method collects all SynOM classes within a COMRIC as root node down to its leaf nodes where the real data from the feeder systems is stored in RECORDITEM

elements. The COMRIC indicates a perfect logical location to split the EHR into smaller subsections, because of its intended representation of the smallest meaningful, autonomous set of medical information.

Figure 5-5, Figure 5-6 and Figure 5-7 display the interaction between the various SRS components, including the embedded XML wrapper, with the help of UML sequence diagrams.

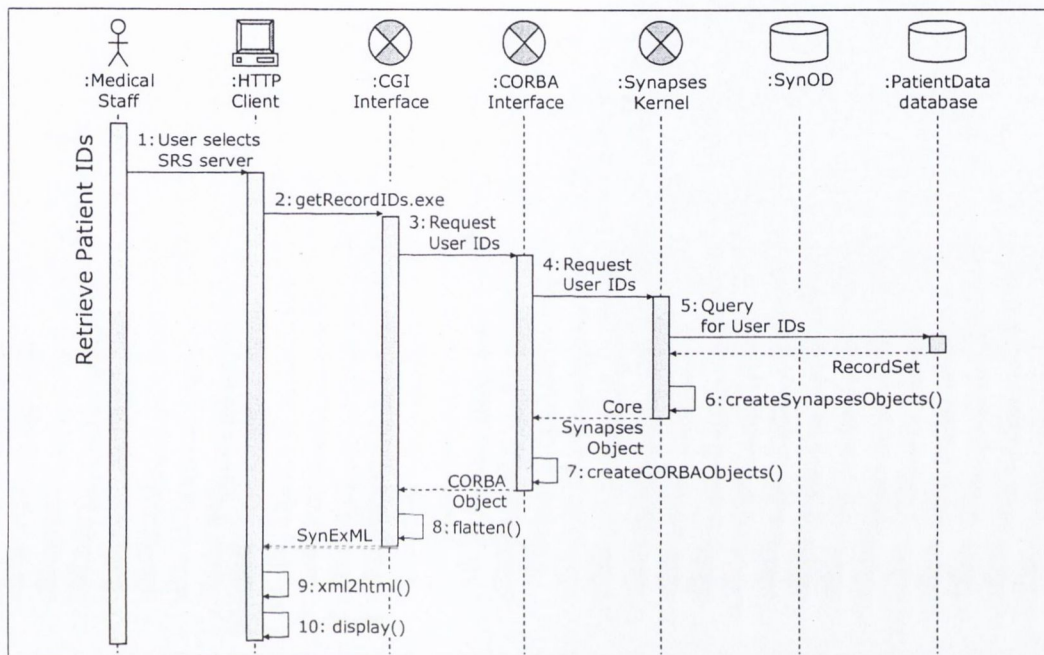


Figure 5-5: RetrieveRecordIDs (UML sequence diagram)

In the first scenario (*RetrievePatientIDs*), a member of the medical staff selects an entry from a list of (locally and remotely) available SRSs in a browser window. In this prototype implementation, user authorisation as well as authentication is not included. However, in a real world environment, access rights based on the user's role (e.g. doctor, nurse and administrator) as well as location (local or remote access) are an essential element in order to ensure confidentiality of patient data. The HTTP client (browser) submits the request to the CGI interface of the Web Server, which acts as a CORBA client. The request for patient IDs is forwarded through the CORBA interface to the SRS kernel. The SRS directly queries the database and retrieves a list of recordsets with the patient IDs of the available records, which are converted into Synapses Objects, CORBA objects and finally flattened into an XML document (*SynExML*). After receiving the XML document, local XSLT stylesheets are used to transform the SynExML message into XHTML code for display in the browser.

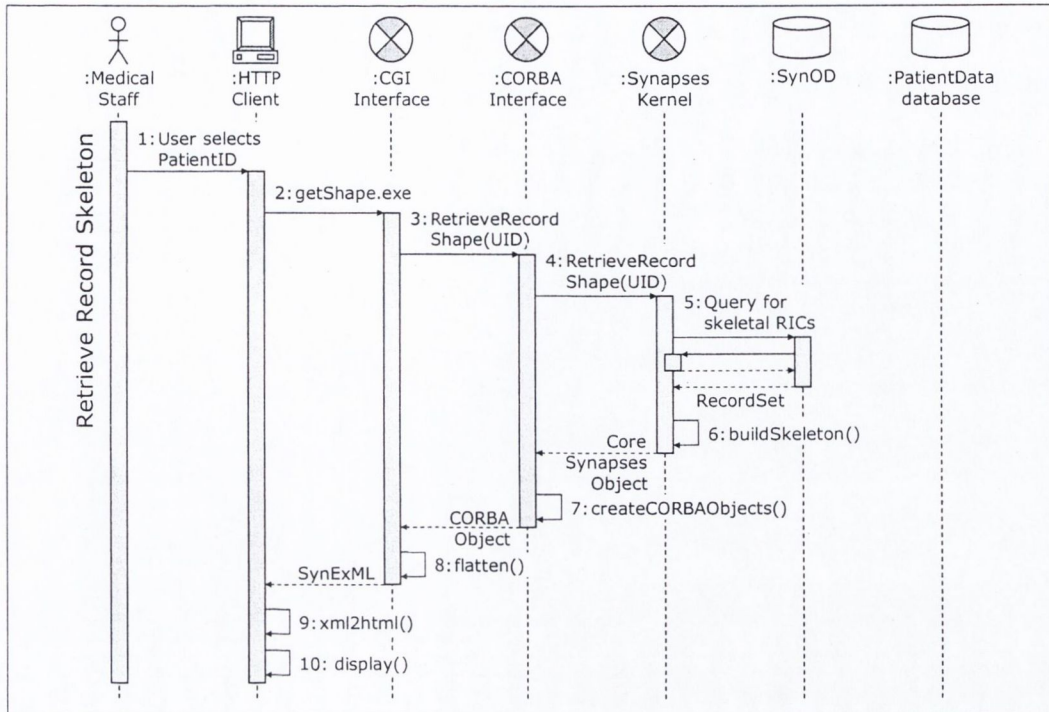


Figure 5-6: RetrieveRecordShape (UML sequence diagram)

In the following step, the user selects one patient ID which is passed to the CGI interface. Similar to the RetrievePatientIDs method, the request is passed through the CORBA layer to the SRS kernel. Here, a query is created to request all structural RICs (RECORDFOLDER, FOLDERRIC and COMRIC elements) and a hierarchical in memory representation of the empty record skeleton is created. On returning, the above described transformation into CORBA objects followed by the flattening procedure creates another SynExML message which is once more transformed into an XHTML document using local XSLT stylesheets. It is important to note that the process of querying and building a hierarchical structure involves repeating step 5 (see **Figure 5-6**) numerous times in order to recursively find all structural RICs.

Figure 5-7 graphically explains the request for a specific patient record's COMRIC. After a user navigates the skeletal patient record and selects a specific COMRIC, its object ID is sent via HTTP to the CGI interface, forwarded to the CORBA interface and finally received by the Synapses kernel. Here, a query is created to retrieve the COMRIC together with all containing SynOM elements from the SynOD database. Similar to the previous scenario, an in memory object-oriented representation of the patient record fragment is build. This fragment does not contain any patient relevant data so far. In a next step, the kernel connects (with information taken from the

SynOD database and now contained in the model) to the various data sources (e.g. the PatientData database) to request the actual patient data, such as name, date of birth and address. This information is added to the model and sent back to the client via CORBA and CGI interface, where the object-oriented model is flattened to a SynExML message. Local XSLT stylesheets are again applied to transform the XML document into XHTML code for display in the browser. Similar to the previous scenario, step 5 as well as 7 (see **Figure 5-7**) have to be repeated until all data is recursively requested from the SynOD and PatientData data sources and subsequently added to the record model.

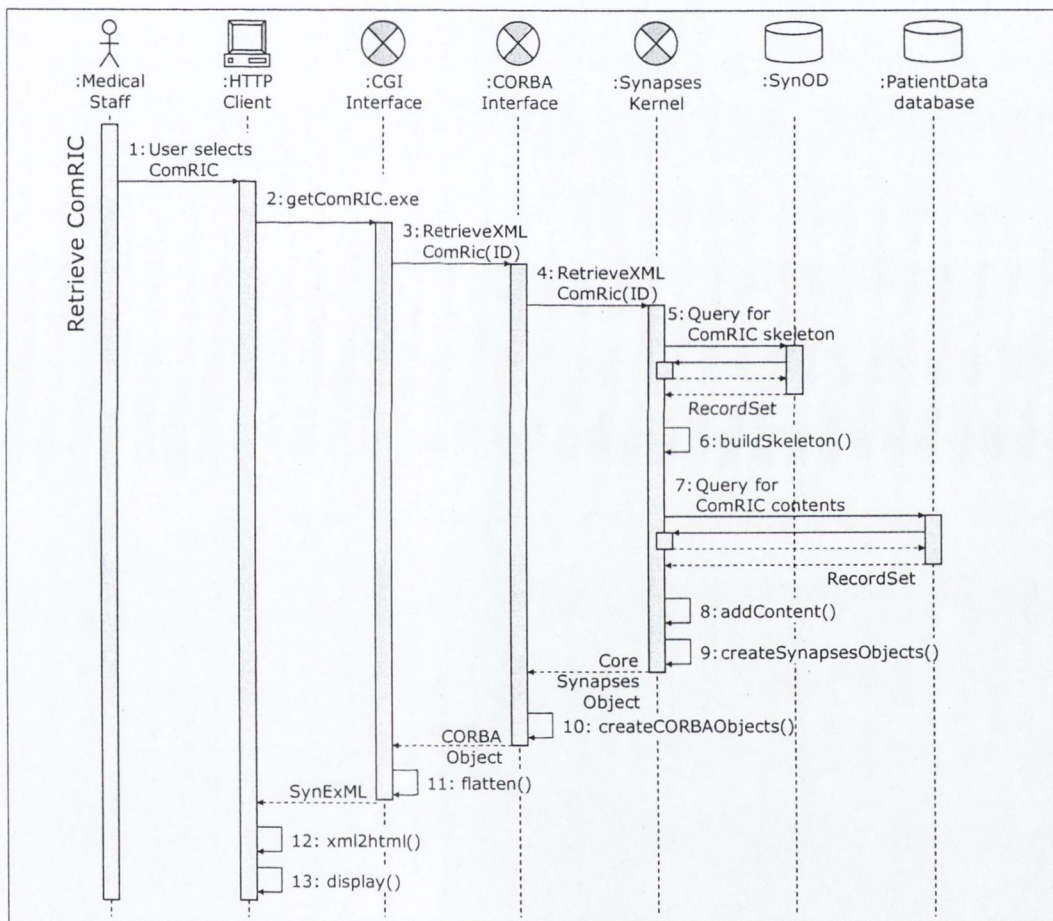


Figure 5-7: RetrieveComRIC (UML sequence diagram)

5.4.2. WSDL, SOAP and UDDI

At the present moment, the SRS exposes two interfaces that are accessible by a client, namely a CORBA and a web-based interface (CGI script) (see **Figure 5-4**). In order to connect to the CORBA interface, a client application has to be equipped with a CORBA component (*client-stub*), which is an exact counterpart to the server

component (*server-stub*). The client stub offers a set of methods that establishes and maintains the connection. By nature, these components are proprietary to each other.

On the other hand, CGI applications are small programs or scripts, which reside on a Web Server. Each application represents exactly one method, which can be accessed over an HTTP connection, potentially with parameter passing from the client.

Whereas CORBA interfaces are precisely defined and each stub works in perfect harmony with each other, CGI applications do not expose their interface structure. Available methods, possible input parameters as well as output formats are transparent to the user and have to be described in accompanying text documents.

Web services offer a combination of CORBA precision and CGI flexibility. The extension of the SRS with web service functionality is a logical enhancement of the existing version. Three elements are necessary to allow ubiquitous (but secure) access to patient data and to prepare the SRS for seamless SHR integration.

Firstly, a WSDL file has to be generated and made available on the Web Server in order to expose available methods, including constraints such as input parameter and output formats. A preliminary WSDL file, equivalent in its functionality to address the server methods exposed by the existing CGI interface, is included in Appendix E.2.

Secondly, client as well as server applications have to communicate by means of SOAP messages. The SOAP specification basically defines a message standard, consisting of SOAP requests and SOAP responses, using XML syntax.

Code 5-4 shows a SOAP request to the SRS, requesting the record shape (skeleton) of the patient with the ID Z99444386. Additional parameters define the Synapses-ServerID (:SynServ), the ServerID (localhost) and the UserID (1). All information is wrapped up into a SOAP Body ELEMENT, which itself is contained in a SOAP Envelope ELEMENT (SOAP-ENV:Envelope), and transmitted to the SRS.

```

1 <SOAP-ENV:Envelope
2   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
3   SOAP-ENV:encodingStyle=
4     "http://schemas.xmlsoap.org/soap/encoding/" >
5   <SOAP-ENV:Body>
6     <syn:getRecordShape
7       xmlns:syn="http://www.cs.tcd.ie/CHI/synapses/" >
8       <syn:SynapsesServerID> SynServ </syn:SynapsesServerID>

```



```

9      <syn:ServerID>localhost</syn:ServerID>
10     <syn:UserID>1</syn:UserID>
11     <syn:RecordID>Z99444386</syn:RecordID>
12     </syn:getRecordShape>
13   </SOAP-ENV:Body>
14 </SOAP-ENV:Envelope>

```

Code 5-4: SynEx SOAP request

Code 5-5 shows the corresponding response message to the SOAP request in **Code 5-4**. Similarly, this message contains a SOAP Envelope with a SOAP Body, which encloses the actual response information. In the case of **Code 5-5**, the content of the response is the patient record structure without any medical data (skeleton), marked up according to the SynExML specification. On receipt of the response message, the client application parses the file and extracts the SOAP Body in order to access and process the message information.

```

1 <SOAP-ENV:Envelope
2   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
3   SOAP-ENV:encodingStyle=
4     "http://schemas.xmlsoap.org/soap/encoding/">
5 <SOAP-ENV:Body>
6   <syn:getRecordShapeResponse
7     xmlns:syn="http://www.cs.tcd.ie/CHI/synapses/"
8     <RecordFolder ClassName="Accident and Emergency"
9       RCID="1" RecordID="Z99444386">
10      <FolderRIC ClassName="Patient" RCID="2">
11        <ComRIC ClassName="Details" RCID="15"/>
12        <ComRIC ClassName="History" RCID="515"/>
13        <ComRIC ClassName="Symptoms" RCID="623"/>
14        <ComRIC ClassName="VitalSigns" RCID="624"/>
15        <ComRIC ClassName="Physical Examination" RCID="644"/>
16        <ComRIC ClassName="Systems Review" RCID="658"/>
17        <ComRIC ClassName="Transfer Letter" RCID="672"/>
18      </FolderRIC>
19      <FolderRIC ClassName="Near Site Investigations" RCID="507">
20        <ComRIC ClassName="Urine Dipstick" RCID="680"/>
21        <ComRIC ClassName="Blood Gas" RCID="692"/>
22      </FolderRIC>
23      <FolderRIC ClassName="CPL Investigations" RCID="516">
24        <ComRIC ClassName="Renal Profile" RCID="695"/>
25        <ComRIC ClassName="Full Blood Count" RCID="696"/>
26        <ComRIC ClassName="Glucose Levels" RCID="697"/>
27        <ComRIC ClassName="Blood Cultures" RCID="698"/>
28        <ComRIC ClassName="Microscopy" RCID="699"/>
29        <ComRIC ClassName="MSU (Culture and Sensibility)"

```



```
30         RCID="700"/>
31     </FolderRIC>
32 </RecordFolder>
33 </syn:getRecordShapeResponse>
34 </SOAP-ENV:Body>
35 </SOAP-ENV:Envelope>
```

Code 5-5: SynEx SOAP response

The last step in preparing the SRS for migration to a Web Services environment is the setup of a local “Universal Description, Discovery and Integration” (UDDI) database, accessible via the network. UDDI is a universal registry and database for storing and finding web services that are available over the network, similar to a yellow pages directory in the telecommunications arena. However, the preferred alternative might be a submission of the local service descriptions (i.e. the WSDL file) to an existing global UDDI, as this ensures a wider “medical audience” and increased ways for data exchange.

A typical medical scenario portrays a doctor who needs more detailed information on a patient’s history. Although the patient now lives near his GP, he regularly travelled abroad and has attended a number of hospitals while being away. In order to obtain examination results from the various hospitals worldwide, the GP connects to the national as well as international UDDI server to query for appropriate web locations. After finding various sites, the client application immediately retrieves information about the access methods including required parameters as well as response formats. Using this information, the client is able to automatically generate a user interface for the GP to interact with and query the remote resources as well as interpret and format the response data in more or less detailed fashion, i.e. syntactical or syntactic/semantic interpretation (see Section 5.2).

What sounds simple in this (hypothetical) scenario is actually complex and technically as well as politically challenging. As mentioned before, the UDDI server does not provide the information itself, i.e. the clinical patient data. Instead, it responds with information about the services that allow access to the data. Web services have the advantage over existing systems that they expose their interfaces in a standardised and easy to understand (i.e. XML syntax) format, which is forwarded by the UDDI server. Software tools are needed that provide the business logic to interact with UDDI servers and evaluate, reason and classify the responses. A fundamental

requirement for national and international data access and furthermore exchange is the existence of a unique patient identifier in order to query, request and match EHR fragments. However, the development of such an ID is widely discussed among technical groups and political committees and out of the scope of this Thesis.

Needless to say, there are many other important issues that have to be dealt with before the SHR and ubiquitous access to and exchange of medical data becomes a regular practice, transparent to the user. Prominent conceptual examples are the data integration of hospital internal information systems, exchange of medical data such as laboratory reports on a regional and national scale (see Section 5.5) and the more technical tasks such as security, i.e. authorisation, authentication and encryption (see Section 2.6).

5.4.3. Client side: XML Presentation Engine

Synapses servers do not rely on a central catalogue service for retrieving information on where the various parts of a particular patient record reside. Instead, as explained above, this information is distributed such that every record on a particular server has the information required to access parts of it that reside in other servers. Remote record fragments are dynamically pulled into the local patient record on request. Having agreed on the SynExXML DTD as the common XML format for the information exchange, the hyperlink capabilities of the core SynOM components (VIEWSRIC1 and VIEWRIC2) in combination with state-of-the-art web technology (*XLink*) makes this an achievable task. SynEx project partners in Dublin as well as Oslo each developed a content aware data browser. These applications are more precisely described as Generic Synapses Clients (GSC) [JAG00a], as they are able to display and integrate record fragments from any remote source, as long as they are contained within a SynExXML document.

Early prototypes demonstrate the possibility of sharing records between two or more institutions and strengthen the usability and feasibility of presenting (and later integrating) locally and remotely stored data. GSCs were developed to simply visualise SynExXML patient records; additional functionality such as to update and return information to the underlying system was not envisaged. This is typically accomplished through locally available data input applications closely connected to the feeder system that stored the data.

Presentation of local data

This first example illustrates access to remotely stored data using the GSC developed in Dublin. As described in Section 5.3, the SRS architecture (and in particular the use of CGI scripts and later web services) allows customisation of data according to the requesting hardware device. **Figure 5-8** shows example screenshots from a browser application on two different devices, i.e. a desktop computer and a PDA, accessing the SRS over the local network. At the time of implementation, wireless network technology was not widely available, and navigating patient records on a PDA was achieved through local copies on the device. Regular synchronisation with the associated desktop computer keeps the patient information up-to-date. In the case of the Dublin GSC, all patient information is retrieved by the browser in SynExML format. Using XSL stylesheets which are likewise supplied by the server, the client transforms the data into XHTML, before it is displayed.

The Dublin GSC desktop application contains three components, closely related to the three core XML methods on the server (see Section 5.4.1). The left hand side (*“left panel”*) includes the components for managerial data as well as the structural representation of the patient record (skeleton). The *“right panel”* is reserved to display single COMRICs. The upper part of the *“left panel”* displays Record Details, such as Medical Record Number, data Origin and SynExML version, used to mark up the patient data. Below the record details, the skeletal part of the patient record is presented. This element resembles all data that is retrieved upon calling the RetrieveRecordShape (see **Figure 5-6**) CGI script on the server. Each entry corresponds to a RECORDFOLDER, FOLDERRIC or COMRIC. In the case of a COMRIC or VIEWRIC2, the entry is presented as a hypertext link. All other elements are presented as simple structural items, uncovering embedded child elements upon selection. By selecting a link in the hierarchical tree, the client automatically requests that specific COMRIC from the server via HTTP. Once the COMRIC data is received (SynExML), XSLT stylesheets transform the XML document into XHTML, which is displayed on the right side of the window. Depending on user preferences, this information is available in tabular format (see A, B in **Figure 5-8**), in name-value pairs (see C, D in **Figure 5-8**) or as plain XML (SynExML). Additional presentation formats to be used in the future include graphical (e.g. SVG) as well as audio (e.g.

ecgML) visualisations of SynExML documents, similar to scenarios described in Chapter 6.

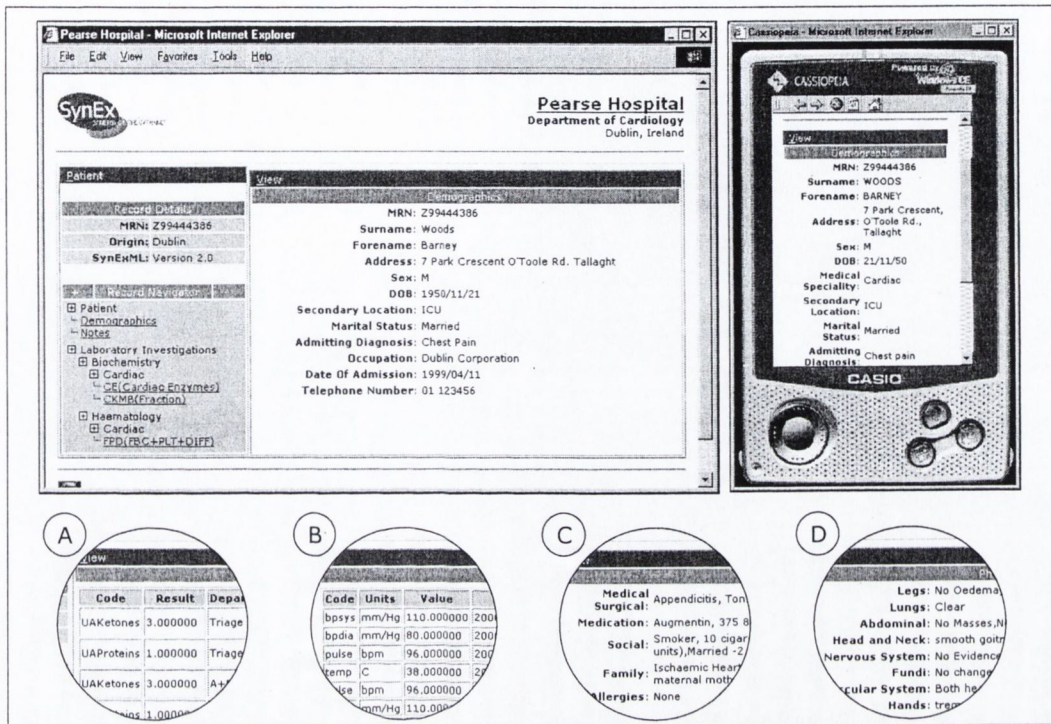


Figure 5-8: Dublin Generic Synapses Client (GSC)

The use of a PDA as navigation and presentation platform requires a different fragmentation of the data, as screen space and network bandwidth are limited. As for the desktop version, the patient record customisation for a PDA is made of two panels. However, in the case of the Dublin GSC, both panels are never visible at the same time. Instead the user switches between the two views, once the COMRIC is identified and downloaded to the device.

Presentation of remote data is transparent to the user, but different in terms of the technical realisation. In order to include data not contained in the local storage system, a VIEWRIC2 element is added to the local SYNOD that points to the remote resource. This part of the patient record, i.e. a remote fragment, is dynamically requested from the remote system upon selection of the VIEWRIC2.

For example, consider a scenario in which a user connects to a SynEx compliant site such as RH (Rikshospitalet) in Oslo, the national hospital of Norway. After the successful login at the local site, the user chooses which sets of available records and record fragments he would like to browse. Parts of a particular selected record may re-

side at other sites such as SJH (St. James's Hospital) in Dublin. The fact that the information content is distributed should be transparent to the user. If a document from RH is requested it will be retrieved from the current server, while if a document from SJH is requested then the client will forward the request to SJH transparently to the user and retrieve the requested document from there. The information to forward the request is entered at the first request of data from SJH and will be stored at RH whereas the data itself resides at SJH. Of course, using this technique might lead to multiple levels of redirection, e.g. if the patient record in Dublin links to another fragment in e.g. Germany. Nevertheless, it ensures a consistent storage of data at exactly one place and a set of properties could limit the number of followed redirections.

Figure 5-9 shows a screenshot of the Oslo GSC [JAG00a], representing exactly the above described scenario. In this case, the GSC is programmed as a stand-alone application but offers similar functionality to the web-based GSC in Dublin (see section 5.4.3). The left panel contains a hierarchical visualisation of the SynExML skeleton with a single COMRIC extracted in the right panel. In particular, the user selected the Demographics element which carries a link to the SRS in Dublin and data is loaded on-the-fly.

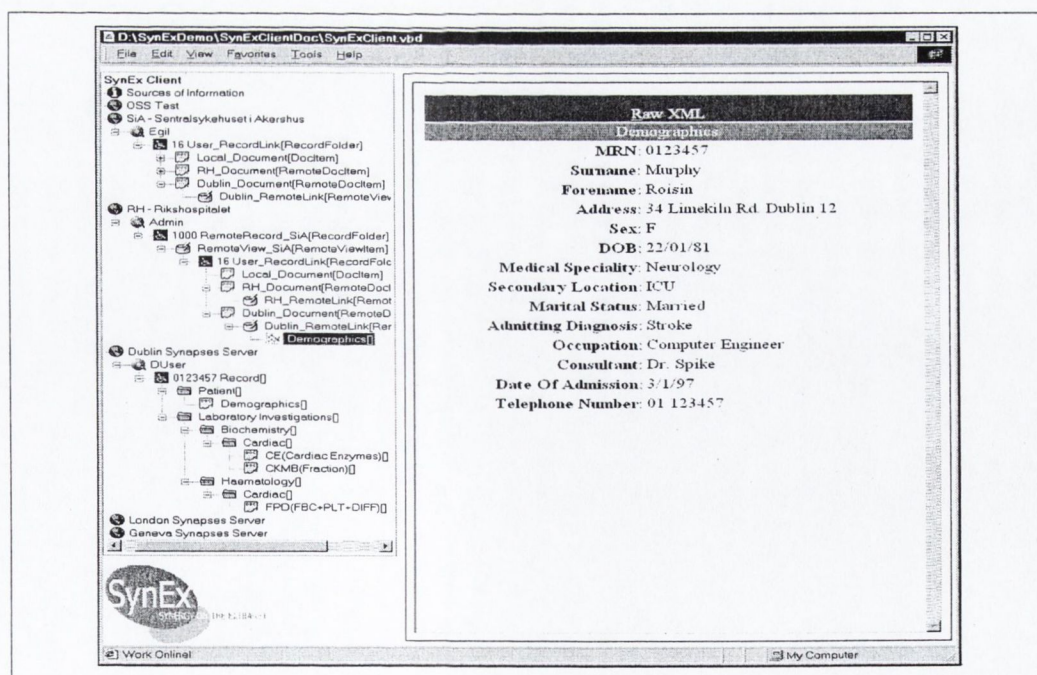


Figure 5-9: Oslo web-client

5.5. Equivalent use case

[DoMu02] describes another medical implementation of the shallow integration method (see Section 5.4), similar to the SynEx architecture. In this case, a number of distributed and heterogeneous hospital information systems act as the feeder systems, all accessible via a web interface (see **Figure 5-10**). The centralised *Healthlink Online* system (Healthlink) queries all available feeder systems at regular intervals, using the *Healthlink Bridge*, an interface proprietary to each hospital, and requests accumulated laboratory results. Upon retrieval, every result is transformed into the local data model and stored in the underlying database system. Instead of connecting to an indefinite number of hospital systems requesting laboratory results, the local GP, registered with Healthlink, is able to retrieve laboratory results for all his patients from one single Healthlink source. This not only simplifies the access to laboratory results, it also normalises the data format.

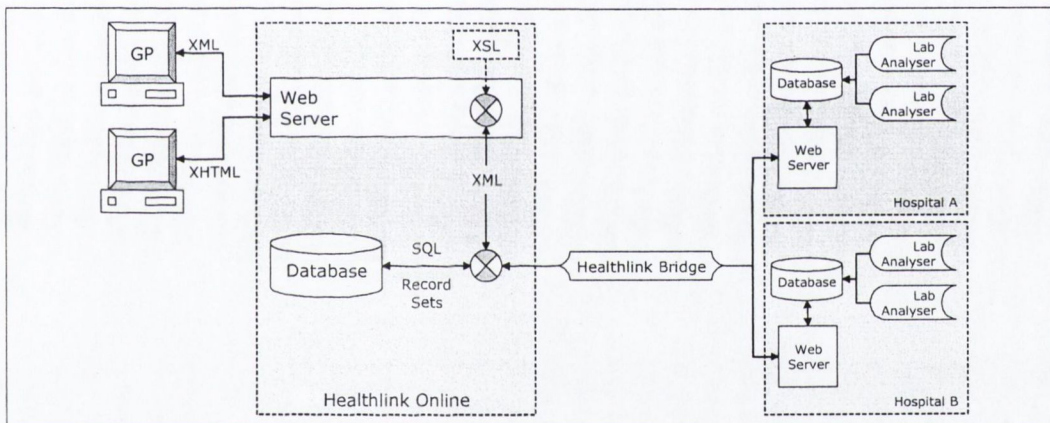


Figure 5-10: Healthlink Architecture

At present, results in Healthlink are stored in accordance with HL7 semantics converted on-the-fly from database result sets into XML syntax. In another step, each request is subsequently transformed into XHTML using XSL stylesheets before being transmitted to the GP. The use of the XHTML presentation vocabulary as a delivery format limits the automatic integration of laboratory results into the GP practice management system. This shallow integration approach provides a visualisation of the data (without HL7 specific markup included) and offers (manual) conversion inclusion into the local system. However, the use of XML syntax and HL7 semantics for the management of information in Healthlink creates an architecture that will eas-

ily scale into a deep integration extension in the future, pushing laboratory results in XML format and HL7 semantics right into the GPs local system.

5.6. Comments

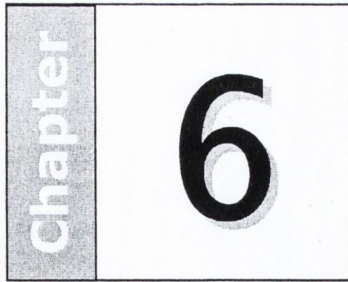
In summary, the development of the SynExXML DTD and its possible future use has brought the objective of seamless integration of patient information closer than originally envisaged. Agreement on a common syntax for the EHR and exchange mechanisms and the foreseeable progress in the development of web-based APIs made it work. However, the use of XML documents following the SynExXML specification, together with HTTP (Hypertext Transfer Protocol) support and a hardware- and software-independent data format do not by themselves provide a flexible exchange of patient records. More research has to be undertaken in the following two major areas before real, automated integration of patient information can take place:

1. XML by itself is only a transport format for data: easy and powerful, flexible and independent. Without applications, which interpret and present the XML document at the receiving site, the interpretation of the data is possible but highly complex. XSLT (XSL Transformations) and FO (Formatting Objects) provide a generic way to specify publishing rules for display in e.g. web-browsers.
2. The syntactical and semantic mapping between different models (i.e. SynODs in various medical institutions, but also for exchange with other EPR standards such as HL7 and CEN) still is and will be the main issue for automated data exchange between heterogeneous systems in the longer term. Because of its flexibility, XML can help to speed up the process of developing an integrated data exchange. Implementations within the SynEx project have demonstrated the display of patient data, conformant with the SynExXML DTD, in remote applications. This was envisaged as the first step in the integration process.

5.7. Summary

This chapter discussed content exchange solutions, in particular the differences between the presentation and integration approaches.

The first part explained the characteristics of syntactic, semantic and combined syntactic/semantic transformation. This was followed by a description of a hypothetical transformation hub that easily manages two-way conversions (based on XSL) between two heterogeneous data models. The second part used work undertaken in the SynEx project to illustrate the presentation method as an initial step in seamless data integration. Technical modifications to the SRS, a reformulation of existing CGI interfaces adhering now to Web Services standards and the implementation of a web based generic Synapses client complete this chapter. Last but not least, an equivalent installation, successfully operating in Ireland, is briefly outlined.

A graphic for Chapter 6. It consists of a vertical grey bar on the left with the word "chapter" written vertically in white. To the right of this bar is a white square containing a large, bold, black number "6".

Multimedia Component Integration into EHRs

6.1. Introduction

The previous chapters explained how XML technologies can be used to define and mark up Electronic Health Records and allow for example seamless data exchange between heterogeneous information systems. In order to entirely adapt the concept of the Semantic Web and create a foundation for the Semantic Health Record, further steps and considerations have to be taken. This includes enabling XML functionality in hard- as well as software components that are remote or peripheral to the core (textual) patient record.

This chapter presents three approaches demonstrating the syntactical adaptation of multimedia components (text, audio, graphics) for integration into the SHR. Beginning with the backend of the SRS, every connection to the feeder systems (see **Figure 4-2**) should transmit messages based on agreed standards and in XML syntax allowing easy access to and integration into the existing systems. It is a pure technical task to convert the existing messaging vocabularies from a proprietary syntax into XML format, keeping the well developed semantics untouched.

Secondly, using XML vocabularies as the core data format, customised transformations can provide visualisations into more than text, including graphics, audio and video. Interpretations of continuous time-series data such as ECG or Wave-Doppler records gain in particular from multimedia visualisation methods.

Last but not least, metadata is one of the key characteristics required to build the semantic web. Proprietary formats usually do not include metadata (e.g. images) or make it very difficult to access metadata with widely available tools. Ideally, metadata is kept together with the object, but in a format that is non-proprietary and easily accessible. An XML based graphics format (SVG) does provide these features and could make image integration into the EPR more complete and bring the idea of a Semantic Health Record one step closer to reality.

6.2. ASTM 1394-91

6.2.1. Background

In 1991 ASTM published two standards related to communication between clinical analysers and host computers, namely

- E1381 Standard Specification for the Low-Level Protocol in Transferring Messages Between Clinical Laboratory Instruments and Computer Systems [E1381] and
- E1394 Standard Specification for Transferring Information between Clinical Instruments [E1394].

Both standards were complementary and intended for point-to-point connections (for instance a serial connection) between instrument and computer system such as a *Laboratory Information System* (LIS) or a validating workstation.

E1381 describes a low-level protocol to permit reliable bi-directional exchange of information between instruments and computer systems. Areas included, for example, the low-level behaviour of two participating systems on error detection, error checking and an agreed low level message structure. The second standard, E1394, which is the subject of this section, describes the recommended logical message design that should be commonly 'understood' by both medical device and host computer. It outlines the hierarchical structure and describes the agreed terminology, providing sender and receiver with a good presentation of an E1394 message.

6.2.2. Message Design

E1394 defines five core elements as message fragments that can be arranged in hierarchical order. The **Header Record** contains information common to the entire message such as time of transmission; the **Patient Info Record** includes information about the subject of the request and its results. Information relating to an order for a test to be carried out by the instrument is part of the **Order Record** and **Result Record** contains data relating to a result that has been produced by the instrument. Finally, the **Request Record** contains information necessary to request the results of previously ordered tests. The less frequently used auxiliary elements are **Comment Record**, **Scientific Record** and **Manufacturer Information Record**.

Table 6-1: ASTM 1394-91 message model

	Values		
InstrumentModel	Analyzer ABG32		
SubjectID	012345		
SampleID	0000005		
MessageTime	13:01:56 09/09/1999		
	TestID	Value	Unit
Result1	pH	7.22	
Result2	pCO2	30.8	mmHg

Table 6-1 presents an ASTM 1394-91 message in tabular form. This example includes minimal identification information about the instrument type that has analysed the sample (instrument model identifier), the donor of the sample (subjectID), the sample itself (sampleID) and the time the message was sent (MessageTime). Following this, an unlimited number of results can be attached to the message, each including information about the test (TestID), the measured value and an associated unit.

6.2.3. Scenario using EDIFACT syntax

To show how an ASTM based instrument would operate, let us consider the case of an imaginary blood-gas analyser that supports ASTM over an RS 232 serial interface. Since the research was finished, RS232 was renamed to EIA 232F (Electronics Industry Association). Typically, such an instrument would not support all ASTM record types, as it requires a relatively unsophisticated communications interface. It is worth noting that this is a commonly cited reason why instrument interface implementers choose not to implement ASTM interfaces – it is difficult and thus expensive to implement the full specification. Consider the case where the instrument only sends information, in other words, the host computer is not permitted to use the request record to request particular results, nor to send any information to the instrument relating to a particular subject or result. The instrument merely sends all its result values to the host computer. In such a case it might be useful for the instrument to send information about the message itself using the header record. In fact, this information could be quite sparse. Perhaps it would send the date of transmission and the mode and name of the instrument. It would also be useful for the instrument to send some information to identify the subject of the tests. We could say that the in-

strument would require some numeric subject identifier, which would be entered by a user before inserting a sample into the instrument. This identifier would be the main content required in the patient information record. Next we would require the results, where each result would comprise at least a test identifier, a measured value, the units and the measurement timestamp. This information in our scenario would be contained in result records, one result record per measured value (see **Table 6-1** for a shortened message version related to this scenario).

Code 6-1 shows the converted model into an EDIFACT message, which would be transmitted between bloodgas analyser and host computer for further analysis, evaluation and finally storage. Equivalent to the earlier described model, the message contains information about the instrument and time the message was sent (line 1), the donator (line 2), the sample itself (line 3) and the measured results (line 4 to 6), including testID, value and unit. The distinct EDIFACT syntax made of segments, fields and components is clearly recognisable.

```

1 H|\^&|Analyzer ABG32^SJH005|||||||19990909130156
2 P|1||012345||
3 O|1|0000005|||||||
4   R|1|^^^pH|7.22|||||
5   R|2|^^^pCO2|30.8|mmHg|||
6   ...
7 L|1|N

```

Code 6-1: ASTM 1394-91 code

The following syntactical and procedural characteristics are interesting to note:

- The scenario only uses a small portion of the ASTM protocol. However the implementer of the host interface would have to redevelop basic functionality. This includes managing both the low-level parsing as well as the transformation logic in order to keep the message in a database. In fact, because of the implementation cost, one might say that the use of ASTM is unjustified in this case.
- The source and destination of the message is implicit as communication is point to point, single source-single destination.
- The use of delimiters imply a certain amount of redundancy in the message

- It is not easy to determine the exact meaning of the message, as there is no way to determine from the message itself, the purpose of the individual fields,

Additionally, a number of non-syntactical but technical characteristics are important to mention, including:

- The serial connection is simple, reliable and relatively easy to maintain.
- Only a single host can receive the message, as the underlying connection is of serial type.
- It is expected that the host is physically situated in close range to the instrument, because of practical limitations of RS232 connections.

6.2.4. Scenario using XML syntax

The equivalent scenario could also be implemented using XML syntax and commonly available HTTP over a TCP/IP network. Let us assume the following facts:

1. Each single room, which might contain any kind of medical-related data-retrieving instrument (e.g. blood-gas analyser, but also the front-desk, where the patient data is entered) has connections to the local hospital's Intranet. It can be assumed that a secure connection to the Internet can be made using the local Intranet.
2. Each instrument will have its own *Internet Protocol* (IP) address, is equipped with a hardware/software data storage unit (e.g. hard disk and database) and a network card with integrated web-server that can be connected to the local network. Alternatively, this functionality can be provided by a separate machine (computer), which then can be used as a bridge between the instrument and the network.

The easiest imaginable use-case is similar to the one described above (see Section 6.2.3). Each part of the original ASTM message is preserved. The main difference is a more readable and accessible syntactic structure for both, man and machine.

The following XML message/document based on the ASTM1394-91-DTD (see Appendix D) is equivalent to the ASTM message described in Scenario 1.

```
1 <?xml version="1.0"?>
2
3 <!DOCTYPE ASTM SYSTEM "ASTM1394-91.dtd">
4
5 <ASTM Designation='E1394'>
6   <MessageHeader>
7     <Instrument>
8       <Label>Analyzer ABG32</Label>
9       <Extension Type='IntenalNumber'>SJH005</Extension>
10    </Instrument>
11    <DateTime>19990909130156</DateTime>
12  </MessageHeader>
13  <Message>
14    <PatientInformation>
15      <ID Type='LaboratoryAssigned'>012345</ID>
16      <TestOrder>
17        <Specimen>
18          <ID>0000005</ID>
19        </Specimen>
20        <Result>
21          <UniversalTestID>pH</UniversalTestID>
22          <Value>7.22</Value>
23        </Result>
24        <Result>
25          <UniversalTestID>pCO2</UniversalTestID>
26          <Value Units='mmHg'>30.8</Value>
27        </Result>
28      </TestOrder>
29    </PatientInformation>
30  </Message>
31 </ASTM>
```

Code 6-2: ASTM 1394-91 code (XML)

The use of the above mentioned components and techniques will gain the following advantages over the use of the RS 232 serial interface:

1. The location of the instrument does not influence the access. Through the use of the TCP/IP protocol, the instrument will have a unique identifier (IP number) within the local Intranet; uniqueness could even be gained within the global Internet, although only version 6 of the Internet Protocol specification (IPv6) [RFC2373] would provide enough address space in order to assign a

dedicated IP number to every instrument³¹ (see **Table 6-2**). Moving the instrument from one room to another or from one building to another does not affect the system. This makes the XML TCP/IP approach suitable for *Near Patient Testing* (NPT) instruments including wireless network access.

Table 6-2: IPv4 vs. IPv6 address space

	IPv4	IPv6
Address format	X:X:X:X where X is a decimal number between 0 and 255	X:X:X:X:X:X:X:X where X is a hexadecimal number between 0 and FFFF
Maximum number of unique addresses	$(2^8)^4 = 256^4 \approx 4.3 * 10^9$	$(2^{16})^8 = 512^8 \approx 4.8 * 10^{21}$

2. Long and proprietary physical connections, for example cables, are not required to connect the instrument to the network. Since the widespread use of wireless networks, this aspect is no longer supportable.
3. The web-server functionality supports pull as well as push technology. Pull describes the situation when all data stored within the instruments storage system is sent to another system on request. In contrast, push technology sends the data without request at specific time-intervals to predefined targets.
4. Software to parse the XML document and provide a Document Object Model (DOM) for easy access to single elements or maintenance of the whole document is freely available. The source code as well as the binary files can be found for Java and C++. Dynamic Link Libraries (DLL) are also available for use in the Microsoft Windows environment.
5. Standard Tool support is very good and of a very high quality.
6. XML can support persistence, as XML files can be stored and retrieved from the file system. XML documents can also be merged easily using the DOM..

³¹ IPv6 addresses are made of 8 numbers, each in the hexadecimal range from 0000 to FFFF. This creates an address space of approximately $4.8 * 10^{21}$ unique addresses. According to Australian astronomers (<http://www.cnn.com/2003/TECH/space/07/22/stars.survey/>), this number is equivalent to the number of all grains of sand on all the world's beaches and deserts or the number of 1/10 of all stars in the known universe. In other words, each one of 10 billion people is assigned more than 420 billion addresses.

7. XML can be viewed in many different ways with comparative ease through the use of multiple style sheets. The processing of XML documents with stylesheets can take place on the server as well as on the client side. Stylesheets can be delivered from the default location (e.g. from the target source), but might be overwritten manually on the client side with local presentation rules. It only requires a modern Web browser to present reports coming from instruments.

6.3. *Multimedia Data Presentation: ecgML*

6.3.1. Background

Today Electrocardiography (ECG) is one of the most important non-invasive diagnostic methods, which can be performed at low cost and allows early recognition of coronary heart diseases. The growing demand for computerised ECG devices together with a high number of ECG device manufacturers and models resulted in a large quantity of heterogeneous data formats and structures. ECG records are digitised and transitionally stored in device internal memory before being transmitted into connected archival and documentation systems if required. Early computerised ECG devices used flat file formats as used in the MIT-BIH (Massachusetts Institute of Technology, Beth Israel Hospital) file library³² for storage. However, the lack of data analysis support, interoperability and integration of multiple resources required a change to more sophisticated database management systems. Apart from hard- and software enhancements, medical standards organisations such as CEN/TC251 and HL7 reviewed several data exchange formats in healthcare applications for possible adaptation of ECG data.

Code 6-3 shows an HL7 message with ECG waveform data. Compared to the much simpler MIT-BIH flat file format which only consists of consecutive numbers representing the ECG readings, this file already contains metadata (line 1-5) important for the interpretation of the ECG. The actual ECG data is included in encoded form starting from line 6.

³² PhysioNet, <http://www.physionet.org/>


```

1 OBX|1|ST|1002.2^Systolic Blood Pressure||120|mmHG||||F
2 OBX|2|ST|1002.3^Diastolic Blood Pressure||85|mmHG||||F
3 OBX|3|ST|93000.2^Ventricular Rate||58|BPM||||F
4 OBX|5|ST|93000.4^P-R Interval||148|ms||||F
5 OBX|6|ST|93000.5^QRS Duration||92|ms||||F
6 ZTF|||153095^133783^begin 644 WAV.DAT\X0D\X0A\M24DJ'.....

```

Code 6-3: ECG waveform data using EDIFACT syntax (HL7)

The DICOM standard, Supplement 30 [Sna99], concentrates on waveform interchange formats. It allows incorporation of diverse bio-signals associated with medical imaging and provides guidelines on how to represent ECG features.

6.3.2. ecgML

There is an increased need to promote the development of standards in order to support a seamless exchange and migration of ECG data as well as the native integration into Electronic Patient Records (EPR) and medical guidelines. Such models should be platform-independent, flexible and open to the scientific community. In the case of ECG data interpretation, an important pre-requisite is a comprehensive data description independent of the number of channels, instrumentation platform and type of experiments. Additionally, an ECG record should include professional comments and annotations relating to acquisition protocols, patient information and analysis findings.

The U.S Food and Drug Administration (FDA), together with a number of other institutions, has developed and published a specification for the exchange of time-series data [BKS02]. It defines a hierarchical structure for the body signals, including temperature, pulse, blood pressure, oxygen saturation and Electroencephalogram (EEG). In fact, it explicitly mentions its utilisation for Electrocardiogram (ECG) data and XML as the syntax of choice. The feature to store multiple records from one or more patients within a single file makes it different from other approaches. However, the authors did not carry the basic principle of separating content and presentation information completely. Elements which are clearly used to specify presentation information such as **MinorTickInterval**, **MajorTickInterval** and **LogScale** are still part of the described document structure and embedded into the DTD.

After more detailed analysis of the FDA proposal, it became obvious that the presentation elements that lead to the non-observance of the separation issue should be re-

moved. A hypothesis study [Wetal03b] assisted by the author of this Thesis among others, describes the adjustment of the FDA proposal to a single ECG content representation format. The most recent version of the ecgML XML vocabulary includes all components required for detailed ECG representation and already defined in the FDA proposal for general time-series data. Besides that, modifications have been done in order to create a data definition specific to ECG data and metadata. The preliminary recommendation supports multi-channel time-series ECG data, reuses concepts and nomenclatures from existing standards and applies the Unified Code for Units of Measure [Setal99] scheme where possible.

```

1 <Waveforms>
2   <XValues>
3     <XOffset dataType="time">00:00:00.000</XOffset>
4     <Duration dataType="time">00:30:06.000</Duration>
5     <SampleRate unit="Hz">360</SampleRate>
6   </XValues>
7   <YValues>
8     <RealValue>
9       <From dataType="time">00:00:00.000</From>
10      <To dataType="time">00:00:01.000</To>
11      <Data>-0.145,-0.145,-0.145,-0.145,-0.145 ...</Data>
12    </RealValue>
13  </YValues>
14 </Waveforms>

```

Code 6-4: ecgML fragment

The XML fragment in **Code 6-4** shows the core part of an ecgML document, containing the waveform data. It starts with information about the start (line 3) and duration (line 4) of the examination as well as the chosen sampling rate (line 5). Data from the actual reading is contained within line 9-11.

Apart from providing a fundamental terminology and syntax for ECG time-series data and associated metadata, ecgML offers simplified data access enabling improved data mining and pattern recognition on heterogeneous software platforms and applications. Additional research is undertaken to discover the value of ecgML in implementing automated decision support models such as case-based reasoning [Wetal03b].

6.3.3. ecgML visualisation

ecgML files do exclusively contain pure (raw) ECG data in plain text format. Information on how to present this data is not part of the XML vocabulary. Instead, the ecgML document has to pass one or more transformations in order to be converted from the single source ecgML file into multiple presentation formats. Single-source multiple-target transformation for presentation purposes has its origins in the textual publishing domain. Often, a single information source such as an article would have to be transformed into a number of output formats. This could include PDF for paper print, XHTML for web publishing and Wireless Markup Language (WML) for display on wireless devices. To keep maintenance low and avoid inconsistencies, the content is kept solely as single source and transformation files are developed to generate the various presentation formats. Changes in the content will reflect automatically in all output files. Furthermore, the complete collection of existing content can be transformed into emerging output formats by developing just another set of transformation files.

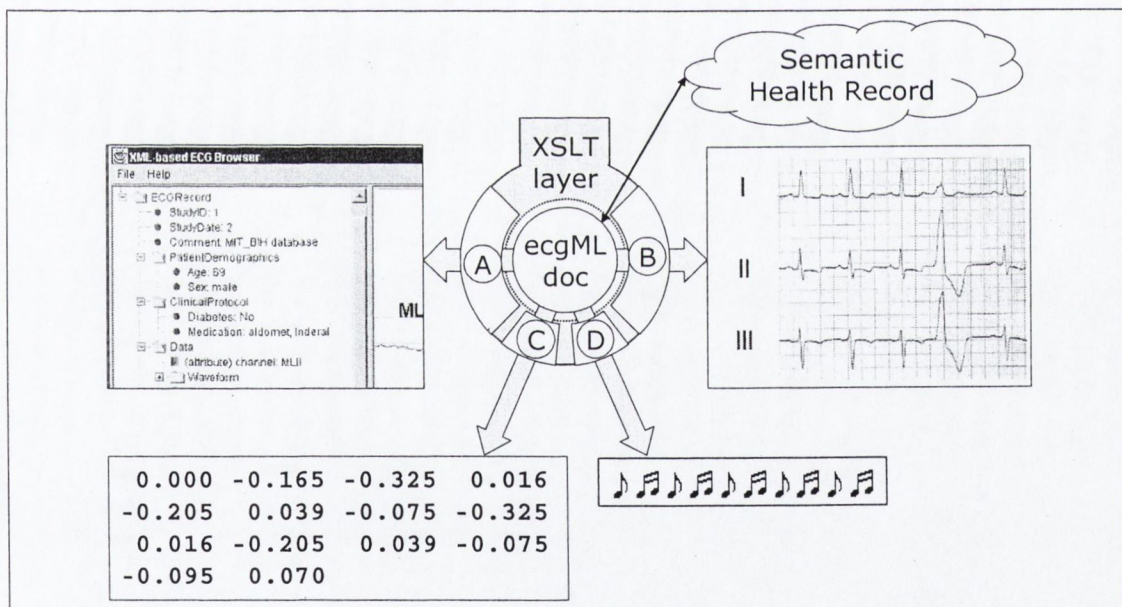


Figure 6-1: ecgML single-source/multi-target transformations

Figure 6-1 illustrates such an example using ecgML as the single source document format at the core. To begin with, each ecgML file can be used as an SHR fragment, providing seamless integration and access to ECG data. In addition, a number of visualisation methods are possible, using individually crafted or pre-defined transformations, e.g. in form of XSLT files. The option (A) indicates a transformation scenario which prepares the original data for display in a dedicated ecgML viewer

application [Wetal03a], whereas option (B) converts the ecgML document into a generic SVG document, offering improved graphical view and analysis known from traditional ECG paper prints. Statistical analysis applications such as Matlab^{®33} generally import data from comma or tab delimited data files, which transformation option (C) offers. Last but not least, option (D) proposes another inter-media transformation, i.e. from text to audio. This method is still under investigation and further tests will show the validity and advantage of a combined audio-visual ECG presentation format over proprietary, binary and semi-binary (DICOM) formats.

6.4. *Imaging: DICOM-X*

6.4.1. Background

The acquisition of specialised image processing computers by hospitals in the early 1970s can be seen as the first indications in the medical world of a change in image storage formats from traditional film and paper towards electronic files [Nem03]. At the time, standardisation issues in medical imaging have not been given much attention resulting in rapid prototyping and implementations using proprietary formats together with dedicated applications. Image exchange was realised by means of swapping removable media or the use of low bandwidth networks, making the entire process cost- and time consuming. From 1985 onwards, a joint initiative by the American College of Radiology (ACR) and the National Electrical Manufacturers Association (NEMA) issued a number of publications to standardise terminology, information structure and file encoding and allow for meaningful information exchange. The latest release (version 3.2) of this specification [DIC03] includes among others, sections about information object definitions (i.e. metadata) such as images, patients, studies and reports. Summarising, it can be said that the development of the DICOM format laid the foundation for the development of general Picture Archiving and Communications Systems (PACS) which seamlessly integrate with existing medical information systems. However, DICOM is a binary image format and prevents direct access (via an application) and indirect access (via a link) to and from its content.

³³ <http://www.mathworks.com/>

6.4.2. DICOM syntax

DICOM images are stored in serial access data files. Each file is virtually divided into three core parts, namely **File Meta Information**, **DICOM Data Sets** and the actual binary **Image Data** as shown in the top level of **Figure 6-2**. The DICOM Data Set contains a sequence of **Data Elements**, each being identified by a Tag, a four digit Group number (e.g. 0008) followed by a four digit Data Element number (e.g. 0023). Tags with an even Group number relate to information fields pre-defined by the DICOM standard. In contrast, odd Group numbers are reserved for Data Elements adding specialisations, extensions and privatisations to the image, such as vendor specific information. The Tag is followed by an optional Value Representation field, i.e. the data type, a field to determine the length of the Value Field and the Value Field itself.

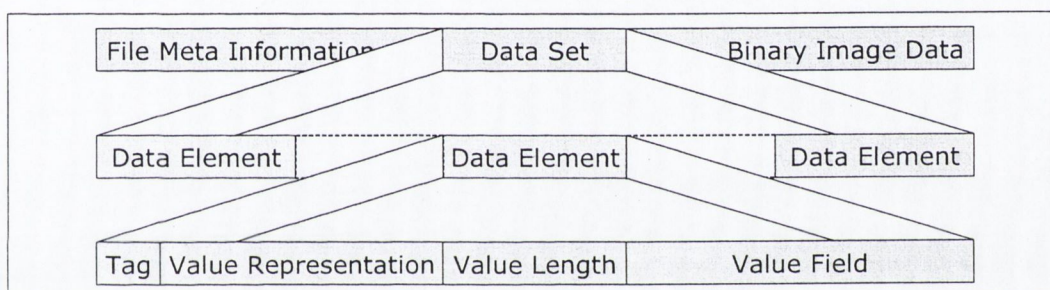


Figure 6-2: DICOM Data Set

1	(0008, 0023)	Image Date	[1999.05.05]
2	(0008, 0033)	Image Time	[10:52:32.510000]
3			
4	(0010, 0010)	Patient's Name	[Anonymized]
5			
6	(0018, 1000)	Device Serial Number	[519]
7	(0018, 1030)	Protocol Name	[ADULT BRAIN/U]
8			
9	(0028, 0002)	Samples per Pixel	[1]
10	(0028, 0004)	Photometric Interpretation	[MONOCHROME2]

Code 6-5: DICOM Data Elements

Code 6-5 shows representations of DICOM Data Elements from the Groups **Identifying** (0008), **Patient Information** (0010), **Acquisition Information** (0018) and **Image Representation** (0028). Each line starts with the Tag, the recognition code consisting of Group and Data Element identifiers, followed by the field descriptor, defined in the DICOM standard but not included in the DICOM image file, and a po-

tential value, embraced by brackets to point out possible spaces at the beginning or end of the value.

The File Meta Information part is a mandatory header element of the DICOM file and located before the DICOM Data Set. It contains an unstructured file preamble (128 bytes) followed by the DICOM prefix (4 bytes) and a number of Data Elements of the 0002 Group. The 0002 Group defines properties which relate to the transfer syntax including encoding rules such as byte ordering and compression type.

Physically last in the DICOM file, but by no means least, comes the actual image data, which follows the DICOM Data Set. This information is stored in binary format, representing pixel, a point in two-dimensional (2D) space, or voxel, a point in three-dimensional (3D) space.

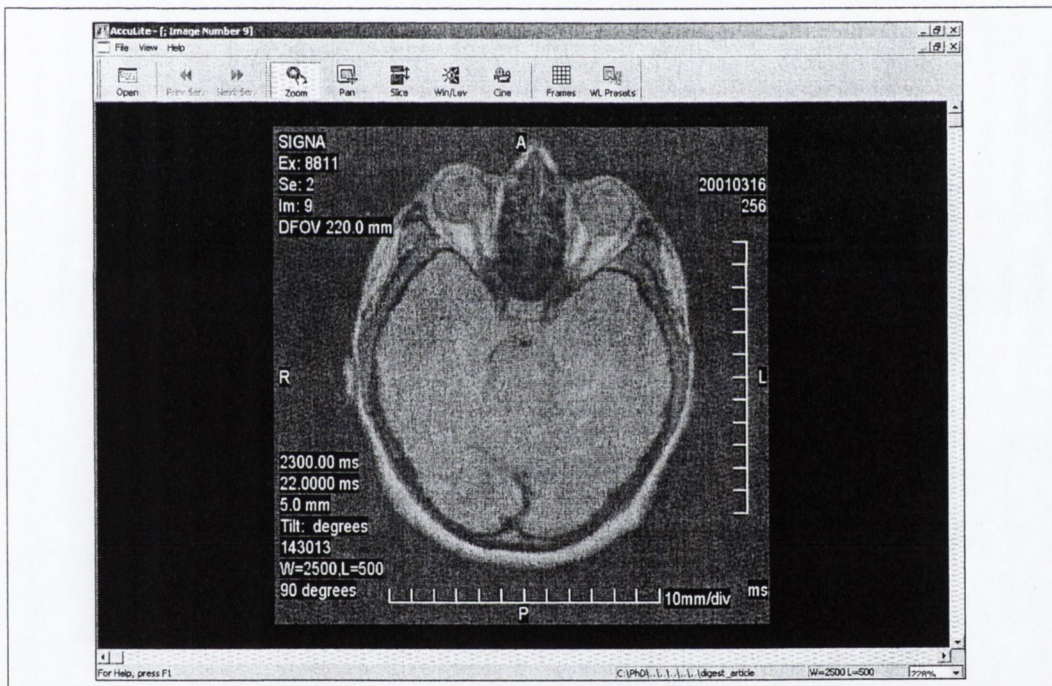


Figure 6-3: DICOM image in dedicated DICOM viewer (AccuLight³⁴)

A screenshot from a dedicated DICOM viewer is presented in **Figure 6-3**. The AccuLight³⁴ application solely operates as a viewer, i.e. it does not provide any functionality to maintain (add, update) the image, but offers possibilities to change presentation parameters such as saturation and brightness. The meta information from the DICOM Data Elements is extracted from the file and displayed in various layers on

³⁴ <http://www.accuimage.com/>

top of the image. Although hospital or department specific information can easily be added to DICOM images (using the odd numbered Group Data Elements), most other systems would not be able to add (non-standardised) metadata. In fact, they simply display metadata (in the best case), but more likely just ignore it. Moreover, Data Elements store information in textual format, which make graphical additions, such as shapes and lines, reasonably complex to encode and decode.

6.4.3. DICOM-X

DICOM-X is the result of a feasibility study undertaken by the author of this Thesis to reformulate the most recent version of the DICOM standard in XML format. The research on this topic is ongoing and a more comprehensive result, including schema design, prototype implementation, evaluation and publication are expected.

The goal of this effort is to prepare medical images for seamless integration into the Semantic Health Record architecture. Reformulation implies that the semantic information (as described and defined by domain experts in the original standard) is left untouched and metadata is presented with identical descriptors using equivalent data formats. Nevertheless, a major syntactical change from binary to XML syntax is proposed. This process will ensure that experience and knowledge from past standard development cycles is combined with rich functionalities provided by members of the XML family of languages (see Chapter 2).

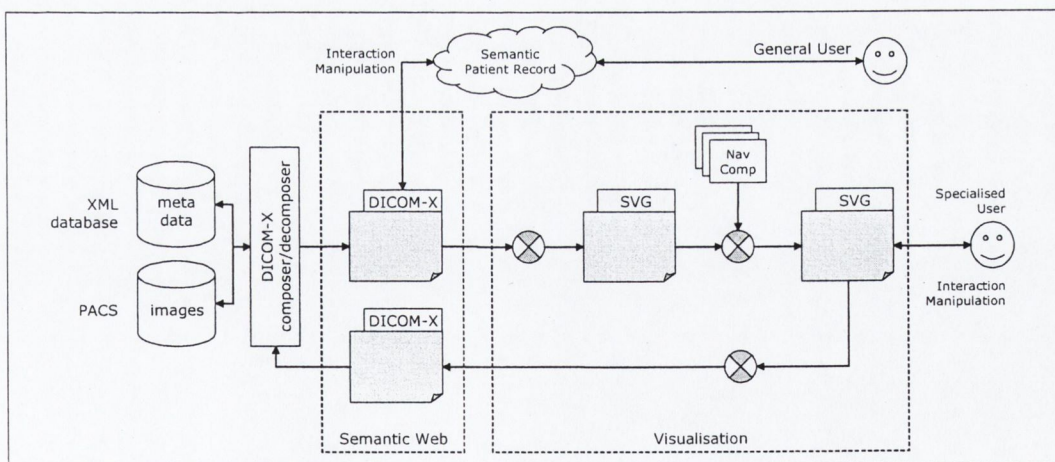


Figure 6-4: DICOM-X Visualisation

Figure 6-4 illustrates the entire data flow process. Images and meta-information are kept in separate dedicated local data stores, e.g. an image database for the images and an XML database for the metadata. On request, the DICOM-X composer retrieves

both the image as well as the associated metadata and generates a DICOM-X document on the fly. The resulting document can be viewed as an individual image item or could act as an SHR fragment when exchanged with remote systems. Visualisation engines transform the information from DICOM-X data format into a multi-layered SVG image (**Figure 6-5**). In a further step, supplementary navigational components (add-ons) are added to support the user in interacting with the layers. Programmed using JavaScript to seamlessly integrate with the SVG viewer, they provide easy access to the enclosed image or images and metadata, including functions to shuffle images, add annotations, highlight areas of interest and measure angles and distances. These navigational elements are ideally standardised and freely available for inclusion. Depending on the physicians needs, specialised methods can easily be developed, added and distributed.

The image together with changes in the metadata can be stored in two ways. Firstly, the user has the opportunity to save and possibly integrate the SVG image into the local system. Because the SVG file contains images, metadata as well as managerial components, it provides an autonomous data snapshot. Secondly, modifications in the metadata can be sent back to the originating database. This procedure makes a reverse transformation from SVG into DICOM-X format necessary, followed by decomposition into the underlying database structure.

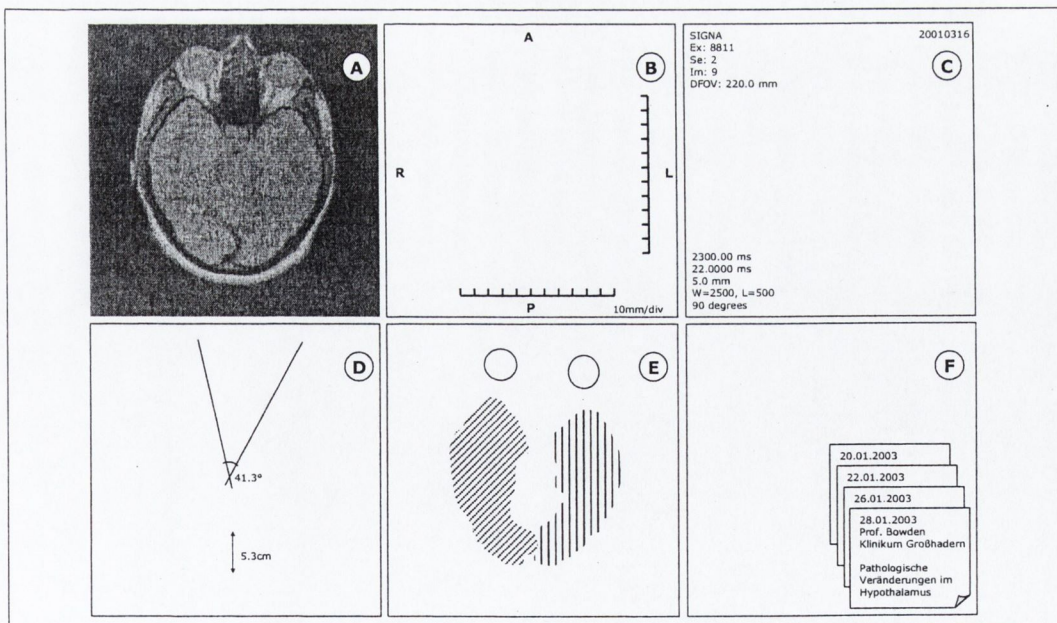


Figure 6-5: DICOM-X image layers

Figure 6-5 shows a potential DICOM-X visualisation using SVG image functionality. It is composed of a number of different layers, spread out for better presentation here. The background layer (A) holds a single original raster image or a set of consecutive raster images, e.g. an X-ray, a collection of CT- or MRT images. The remaining layers (B) to (F) contain metadata, such as scales and dimension information (B), patient data and image properties (C), angles and distances (D), prominent areas and shapes (E) and finally staff annotations (F). The original DICOM standard provides guidelines and methods to add and customise metadata in order to serve specialised medical domains and local distinctions. To complete the system, the development of a dedicated navigational component to handle the new specialisation is required.

6.4.4. Comments

Today, the DICOM standards committee participates in the development and maintenance of international standards for communication of biomedical diagnostic and therapeutic information [DIC03]. DICOM images are used in almost every medical discipline that uses digital images and associated data. A reformulation as described in this chapter would increase compatibility and workflow efficiency between imaging systems. More importantly, DICOM-X would easily facilitate the integration of medical images into a future SHR.

As of writing this Thesis, a reformulation of the existing DICOM standard has not attracted much attention from the scientific domain (medical as well as medical informatics). Nevertheless, comparable exercises in the past have shown the impact that XML syntax can have on binary data formats, including SVG [SVG01] in imaging, VoiceXML [VXML00] in audio and Rich Vector Markup Language (RVML)³⁵ in website development and management. In the medical domain, initial work has been published [LeHu03] which explains how to express metadata of DICOM images in XML syntax, in particular the Structured Reporting (SR) part of the standard. However, this work was undertaken without attention and inclusion of the actual image data into the final format.

³⁵ <http://www.kinesisssoftware.com/>

Advantages of the proposed DICOM-X format are multifaceted and naturally do conform to generally accepted XML advantages:

Plain Text Format and true XML

DICOM-X files are platform- vendor- and application independent data collections, accessible and readable in the foreseeable future. Although the development cycle of computer systems will lead to improved but currently unknown data formats, the fundamental option to read and write plain text files is undisputable and will be part of any future software application. This basic principle establishes an opportunity for true independence and long-term interoperability between heterogeneous systems.

Additionally, by using XML, DICOM-X images facilitate managing multiple UNICODE³⁶ charactersets even within the file, take advantage of a significant number of XML supporting applications, utilise data manipulation and transformation through standardised APIs as well as XSLT.

Searchable and selectable text

The three main aspects of the Semantic Web (and similarly of the Semantic Health Record) are the ability to express meaning, represent knowledge and define ontologies (see Section 2.9). XML syntax allows intelligent agents to automatically search and detect information associated with DICOM-X images, even if embedded within the images. Additionally, the use of the intelligent linking mechanisms as described and specified in the XLink and XPointer recommendations (see Section 2.4) makes it possible to define textual and graphical elements on the various layers as link anchors or link targets.

Open standard

The use of open standards for syntax (XML), semantic (DICOM) and visualisation (SVG) guarantees a flexible, accessible, transparent and future-proof implementation covering all levels of the data and application architecture (server, middleware, client).

³⁶ <http://www.unicode.org/>

Scripting and animation

Scripting and animation is part of the core SVG specification and supported by every SVG viewer. Using scripting languages such as ECMAScript or JavaScript, images can programmatically be accessed and manipulated at runtime on the client. Together with SMIL [SMIL01], integrated audio and video sequences enrich SVG to become a very powerful multimedia delivery format that is considered superior even to long established and widely adapted technologies like Macromedia Flash.

SVG supports descriptive as well as programmatic animation. Using descriptive animation, the start and end state of the animation are defined by the author and the application (viewer) calculates a smooth transition of colours, shapes or locations. On the other hand, programmatic animations are more flexible

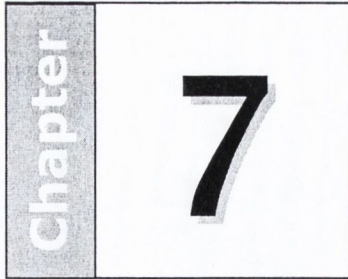
Scalable and zoomable

Although the effective value of this feature is limited by the resolution of the raster image used as a background layer, visualisations of the metadata do gain advantage. Nevertheless, separation of metadata and image offers the chance to keep the background image in its original size and zoom into e.g. metadata that is not graphically related to the image (such as textual annotations).

6.5. Summary

This chapter outlined three scenarios and illustrated solutions on how to make multimedia components ready for integration into a coming Semantic Health Record architecture.

The first part described the transformation of an EDIFACT based message format for clinical instruments to XML syntax, retaining its original semantic. The following part discussed a similar scenario. However this time waveform data from an ECG device is transformed into XML syntax to be seamlessly integrated into the SHR. The last section outlined a transformation of graphical metadata from the binary DICOM format into XML syntax and subsequently combined and compiled with the actual image data.



Conclusions

7.1. Review of this Thesis

This Thesis has investigated how concepts from the Semantic Web can be successfully adapted building an equivalent architecture in the domain of EHR - the Semantic Health Record. Contributions cover many aspects of the deployment line, including design and development of the underlying XML vocabulary for EHR architecture, reformulation of existing multimedia syntax formats, transmission and accessibility advances and finally presentation and integration aspects.

This Thesis has started in **Chapter 2** with a summary of technological aspects of the Semantic Web. In particular a review of the members of the XML family of technologies that play important roles in creating the SHR. The history of electronic content encoding has been discussed with a special focus on the meta-languages SGML and XML. An overview of XML data management (storage), including internal (files) as well as external (databases) solutions and query mechanisms followed. Security is of great importance to medical applications; thus encryption of XML documents and XML document fragments has been described. In order to embed highly fragmented EHR data components, the concept of URI addressing, known from the World Wide Web to a certain extent, as well as the improved linking mechanism defined as part of the XLink and XPointer specifications has been explained. Three methods of content mapping, namely syntactic, semantic and combined syntactic/semantic, have been part of the next section. XSL transformations, a technology to define transformation rules, have been the focus in the next section, in particular in relation to model mapping and information presentation. The second chapter finished with an explanation of the three essential aspects of the Semantic Web, namely expression meaning, knowledge representation and ontology definition.

Chapter 3 gave insight into three standards for Electronic Health Record Architectures.

It started with a background section about the electronic aspect of health records, followed by an explanation of the different record terminologies used in more recent literature. The Synapses paradigm of SynOM, SynOD and patient record – the three basic architectural components for the Synapses Record – has been explained in detail. This part included a section about the SynOM core elements and their aggregations, followed by a technical description of assembling a record skeleton (template) and generating the actual record (with patient data).

HL7 and CEN/TC251 are two standards organisations whose EHR approach has been briefly described in the following two sections. The last part of this chapter compared and evaluated the HL7 and CEN/TC251 solutions with the Synapses methodology.

Chapter 4 has presented the Record Structure Builder, an application for graphically modelling EHR skeletons (templates) and an exchange format for EHRs, based on the Synapses paradigm and XML syntax.

The first section explained the concept of Multi View Modelling with the aspect of personalised interfaces, supporting the user in each phase of the modelling process. The MVM methodology was implemented in the Record Structure Builder, a graphical application to assist the various user groups, e.g. technical programmer, medical use, in the development of SynODs, a skeleton of a patient record fragment. The following section of Chapter 4 described two messaging syntax formats, the nearly identical EDIFACT and HL7, followed by SynExXML, an XML vocabulary to represent SynOM and SynODs was presented.

Chapter 5 discussed two different content exchange methodologies, in particular the fundamental differences between presentation (shallow integration) and deep integration approaches.

The first part explained the characteristics of syntactic, semantic and combined syntactic/semantic transformation. The technical challenge of mapping between multiple heterogeneous information sources suggested the development of a core mapping model, which has to be supported by all sites in order to allow seamless information exchange. To improve the maintenance of the mapping process, the idea of a virtual transformation hub that easily manages two-way conversions (based on XSL) between two heterogeneous data models, was proposed. In the second part, two imple-

mentations are explained that enabled XML document transfer between server and client in the Synapses project. Firstly, an XML wrapper was added to the existing SRS, allowing the client to retrieve all data in XML syntax (SynExML). An upgrade of the existing system to a Web Services architecture was proposed with details on definition of the server interface (WSDL) and message format (SOAP). Secondly, the implementation of a generic web client has been shown, that supports shallow integration. The chapter finished with a description of an equivalent (shallow integration) installation (Healthlink), recently launched in Ireland, to view and transmit laboratory results from the hospital to the general practitioner.

Finally, **Chapter 6** has given detailed examples on how to modify and prepare existing multimedia formats for integration into the SHR. It started with the reformulation of the existing E1394-91 specification, a message standard to transfer information between clinical instruments, into an XML vocabulary. Conserving the well defined and established semantic, which was approved by the ASTM standards organisation, the syntax of the EDIFACT based message format was converted and evaluated. In the second part, the XML vocabulary for continuously streamed waveform data was evaluated and improved by removing information that is only necessary for presentation. It is recommended to separate content and presentation information in XML document. The last part of Chapter 6 outlines the definition of an XML vocabulary based on the DICOM semantic. It explains the novel idea of combining a binary image with its textual metadata in one single XML document, in this case an SVG image. This method offers easier access and more flexible processing capabilities than other purely binary solutions.

7.2. *Summary of Thesis Contributions*

The research contributions of this Thesis can be summarised as follows:

- A state of the art summary of all members of the XML family of technologies closely related to the Semantic Web and need for adaptation to build a SHR.
- A brief comparison of the three main EHR architecture implementations with the focus on syntactic and semantic diversity.
- The redesign and redevelopment of an application to support the definition of EHR and EHR fragments according to the Synapses methodology.

- The reformulation of the Synapses Object Model as DTD and XML Schema, including a Web Services interface definition to access the model.
- An analysis of syntactic, semantic and combined syntactic/semantic transformation options to enable mapping between heterogeneous information models.
- The design, evaluation and development of a web based client application to present and navigate local as well as remote EHR fragments.
- A reformulation of a message standard to transfer information between clinical instruments (EDIFACT to XML syntax).
- An adjustment to the existing proposal for an ECG XML vocabulary in order to adhere to the concept of separating content and presentation information in XML architectures.
- A hypothesis study to markup image metadata in XML and combine the binary image data together with the textual metadata in a single XML file for easy migration and access.

7.3. Future Work

The Semantic Health Record is clearly in an early development stage at the time of writing this Thesis. It is questionable whether the Semantic Web as proposed by Tim Berners-Lee [BHL01] will become as popular, ubiquitous and record-breaking as the World-Wide-Web; but without a doubt it is going to have an enormous impact in clearly defined, standardised and specialised “niche” areas such as the Electronic Patient Record. Nevertheless, more research and prototyping has to be undertaken before the Synapses Record Server can be expanded to serve Semantic Health Records. The following subsections outline three areas that need further research and investigation to complete and prepare the Synapses Record Server for future competitiveness.

7.3.1. Content

The seamless integration of multimedia content into the EPR is of fundamental importance to conclude major structural aspects of the architectural development. At present, the Synapses Record Server is equipped with a limited functionality to inte-

grate and serve two-dimensional still-images as part of the EPR, e.g. using an image database as feeder system. However, this approach lacks in various areas primarily related to adding and maintaining image metadata. The DICOM standard has solved these issues, although the use of proprietary and binary data formats does not supply for EPR integration into the Semantic Web and DICOM-X (see Section 6.4) might be a promising initiative.

As described throughout this Thesis, a key requirement for content integration into a Semantic Web like structure is the use of XML as its basic syntax. On the other hand, the semantic structure (terminology) is ideally be adopted from existing non-XML vocabularies or defined in cooperation with the appropriate standards organisations.

Other multimedia file formats are of similar interest. These include **3D imaging**, e.g. Virtual Reality Markup Language (VRML) or Extensible 3D Graphics (X3D) [X3D02], and **audio** as well as **video**. Only if we make all available medical information easily and ubiquitously accessible through the Semantic Health Record, will we gain the highest and most complete care for our health. Using XML technologies as the syntactic foundation, tasks previously not imaginably become reality such as metadata annotations, automatic classification and connection of information, context sensitive search, native integration of information from secondary data sources (e.g. medical guidelines) and sophisticated linking between every fragment of the EPR, independent of the media source.

7.3.2. Access: Web Services

In the time of the semantic web, each system participating in an EHR environment should provide and comply with ubiquitous remote access methods such as web services. The SRS is currently not enabled to participate in web service based transactions and communication. Nevertheless, taking into account the existence and support of SynExXML by the core SRS, only the following components have to be added:

- A Simple Object Access Protocol (SOAP) layer on the server, which encodes and decodes HTTP header together with the XML file before being sent over the network (HTTP).

An example of a SOAP message is given in Section 5.4.2.

- The interface description of exposed SRS services using Web Services Definition Language. This document specifies exactly which services are part of the interface, including access methods, input parameters and return format. A preliminary and unofficial version of the WSDL file, coherent with the existing CGI interface, is given in Appendix F.2.
- The entry into one of the available Universal Description, Discovery and Integration (UDDI) directories. Similar to the Yellow Pages in traditional phone companies, UDDI offers a web-based distributed directory with standardised access methods to allow businesses and institutions to advertise and locate existing services on the Internet.

7.3.3. Querying

Last but not least, an additional layer to enhance the abstraction level of the feeder system interface would ease the maintenance of the existing Synapses Record Server. In the most recent SRS implementation, all queries to retrieve the actual medical data from the feeder systems are stored in the SynOD database. Each query is stored as a query string, using syntax and semantics which is proprietary to the feeder system application. For instance, to retrieve patient data from a Patient Information System (PIS) implemented on top of a Relational Database Management System (RDBMS), a SQL query string would be stored in the SynOD. Likewise, an XQuery string is used to retrieve laboratory data from an XML database. In the event of a technology change at the feeder system level, e.g. a migration from an RDBMS to an XML database, every single query in the SynOD related to that particular feeder system would have to be modified to reflect the new condition.

One solution to improve the actual situation is the integration of another abstraction layer as part of the Generic Adapter (see **Figure 4-2**) which encapsulates the feeder system from the query definition. This allows one to define queries in an abstract and application independent format, making the feeder systems transparent to the query author. Information about the various types of feeder systems (e.g. RDBMS, OODBMS, XML database) and the mapping from the generic query language to the application-specific language is maintained in the abstraction layer. As a result, necessary changes in the feeder system configuration such as technology migration, in-

volve adapting the property settings in the new abstraction layer but changes in the SynOD database are avoided.

Nevertheless, it is understood that the definition of a generic query language is a difficult and complex task. Each storage system (in the widest sense) has its unique characteristics and associated query languages are specialised to utilise these. Creating a superclass of query languages might become counter productive and degenerate into an academic exercise with little or no practical use.

Bibliography

- [AAP01] AMERICAN ACADEMY OF PEDIATRICS (2001) *Special Requirements for Electronic Medical Record Systems in Pediatrics*: Pediatrics, Volume 108, Number 2, August 2001, pp. 513-515.
- [ALS96] ALSCHULER, L., LINCOLN, T.L. and SPINOSA, J. (1996) *Medicine for SGML*: Conference Proceedings GCA SGML 1996. Boston, USA.
- [ASTM00] AMERICAN SOCIETY FOR TESTING AND MATERIALS (2003) *Providing the value, strength, and respect of marketplace consensus* [online]. Mission Statement. Available from:
<http://www.astm.org/NEWS/Mission2.html> [Accessed 5th August 2003].
- [BeEtal98] BERRY, D., GRIMSON, W., GRIMSON, J., STEPHENS, G., FELTON, E., PERSANO, G., JUNG, B., KALRA, D., LLOYD, D. HURLEN, P. and SKIFFELD, K. (1998) *Client Interfaces to Synapses Record Server using Corba*: Proceedings Object Management Group Conference 1998. Manchester, United Kingdom.
- [BeGr92] BELL, D. and GRIMSON, J. (1992) *Distributed Database Systems*. Addison-Wesley Longman, Reading, Massachusetts, USA.
- [Bel00] BELLON, E. (2000) *Semantic Web opportunities for the Electronic Patient Record and medical image handling*: Building a Global Business, Symposium Euromedia 2000. Antwerp, Netherlands.
- [BeMu97] VAN BEMMEL, J.H., MUSEN, M.A. (eds) (1997) *Handbook of Medial Informatics*. Springer Verlag. Houten/Diegem.
- [BFJ99] BERRY, D., FELTON, E. and JUNG, B. (1999) *Record Structure Builder – User Manual v0.1*. Unpublished Draft, Dublin Health Informatics Group (DHIG).
- [BHL99] Bray, T., Hollander D., Layman, A. (1999) Namespaces in XML. [online]. W3C Recommendation (14 January 1999), W3C. Available

- from: <http://www.w3.org/TR/REC-xml-names/> [Accessed 14 November 2002].
- [BHL01] BERNERS-LEE, T., HENDLER, J., and LASSILA, O. (2001) *The Semantic Web*: Scientific American. Volume 284, Number 5, May 2001, pp. 34-43.
- [BKS02] BROWN, B., KOHLS, M. and STOCKBRIDGE, N. (2002) *FDA XML Data Format Design Specification* [online]. Design Specification (Draft B), US Food and Drug Administration. Available from: http://www.cdisc.org/discussions/EGC/FDA_XML_Data_Format_Design_Specification_DRAFT_C.pdf [Accessed 5th August 2003].
- [Bos98] BOSAK, J. (1997) *XML, Java and the future of the web*: World Wide Web Journal. Volume 2, Issue 4, Winter 1997, pp.219-228.
- [Bou03] BOURRET, R. (2003) *XML and Databases* [online]. Available from: <http://www.rpbouret.com/xml/XMLAndDatabases.htm> [Accessed 5th August 2003].
- [Bra98] BRAY, T. (1998) *Element sets: A minimal basis for an XML Query Engine* [online]. Available from: <http://www.w3.org/TandS/QL/QL98/pp/sets.html> [Accessed 5th August 2003].
- [BSG99] BISBAL, J., STEPHENS, G. and GRIMSON, J. (1999) *Generic Access to Synapses EHCR Data*: Proceedings of the 4th Annual Conference and Scientific Symposium of the Health Informatics Society of Ireland. Malahide, Ireland.
- [CBS96] CONNOLLY, T., BEGG, C. and STRACHAN, A. (1996) *Database Systems – A Practical Approach to Design, Implementation and Management*. Addison Wesley. Harlow, UK.
- [CEN00] COMITÉ EUROPÉEN DE NORMALISATION, TECHNICAL COMMITTEE 251 (2000) *Scope* [online]. Available from: http://www.centc251.org/GInfo/scope_Tc251.htm [Accessed 5th August 2003].

- [Cetal01] CHRISTENSEN, E., CURBERA, F., MEREDITH, G. and WEERAWARANA, S. (2001) *Web Services Description Language (WSDL) 1.1* [online]. W3C Note, 15.03.2001. Available from: <http://www.w3.org/TR/wsdl> [Accessed 5th August 2003].
- [Col97] COLOMB, R.M. (1997) *Impact of Semantic Heterogeneity on Federating Databases*: The Computer Journal, Vol. 40, No. 5.
- [Con97] CONNOLLY, D. (Edt.) (1997) *XML Principles, Tools, and Techniques*, The World Wide Web Journal, Volume 2, Issue 4, Fall 1997.
- [CPT99] CEN/TC 251 WG IV PT36 (1999) *Clinical Analyser Interfaces to Laboratory Information Systems, First Working Document (FWD)*, maintained by Comité Européen de Normalisation, Technical Committee 251.
- [CRF00] CHAMBERLIN, D., ROBIE, J., FLORESCU, D. (2000) *Quilt: An XML Query Language for Heterogeneous Data Sources*: Lecture Notes in Computer Science, Springer Verlag.
- [CXML01] BOYER, J. (2001) *Canonical XML Version 1.0* [online]. W3C Recommendation (15 March 2001). Available from: <http://www.w3.org/TR/xml-c14n> [Accessed 30th August 2003].
- [DABM97] DOLIN, R.H., ALSCHULER, L., BAY, T., MATTISON, J.E. (1997) *SGML as a Message Interchange Format in Healthcare*: Proceedings AMIA 1997, Annual Fall Symposium 1997. pp. 635-639.
- [Detal98] Deutsch, A., Fernandez, M., Florescu, D., Levy, A., Suciu, D. (1998) XML-QL: A Query Language for XML. [online]. Note World Wide Web Consortium (19 August 1998), W3C. Available from: <http://www.w3.org/TR/1998/NOTE-xml-ql-19980819/> [Accessed 12th May 2000].
- [Detal99] DOLIN, R., ALSCHULER, L., BEHLEN, F. ET.AL. (1999) *HL7 Document Patient Record Architecture: An XML Document Architecture Based on a Shared Information Model*: Proceedings AMIA Annual Symposium 1999: 52-6. Washington/DC, USA.

- [Detal01] DOLIN, R.D., ALSCHULER, L., BEEBE, C., BIRON, P.V., LEE BOYER, S., ESSIN, D. KIMBER, E., LINCOLN, T., MATTISON, J.E. (2001) *The HL7 Clinical Document Architecture*. J Am Med Inform Assoc.2001;8:552 –569.
- [DIC03] NATIONAL ELECTRICAL MANUFACTURERS ASSOCIATION (NEMA) (2003) *Digital Imaging and Communications in Medicine (DICOM) Part 1-16* [online]. Available from: <http://medical.nema.org/dicom/2003.html> [Accessed 12th January 2003].
- [DOM98] WOOD, L. (edt) (1998) *Document Object Model Specification Version 1.0* [online]. W3C Working Draft (16 April 1998), W3C. Available from: <http://www.w3.org/TR/1998/WD-DOM-19980416/> [Accessed 12th September 2003].
- [DoMu02] DOOGUE, O., MURPHY, K. (2002) *From the East and into the Midwest – electronic patient test results merge seamlessly into the GP's practice management system!*: Conference Proceedings Health Informatics Society Ireland (HISI) 2002. Dublin, Ireland.
- [DSD97] DICK, R.S., STEEN, E.B., DETMER, D.E. (eds) (1997) *The Computer-Based Patient Record: An Essential Technology for Health Care (Revised Edition)*. Institute of Medicine, National Academy Press Washington, USA.
- [Dum00] DUMBILL, E. (2000) *The Semantic Web: A Primer* [online]. XML.com, 1st November 2000. Available from: <http://www.xml.com/pub/a/2000/11/01/semanticweb/> [Accessed: 14th July 2001].
- [E1381] NATIONAL COMMITTEE FOR CLINICAL LABORATORY STANDARDS (NCCLS) (2001) *Standard Specification for Low-Level Protocol to Transfer Messages Between Clinical Laboratory Instruments and Computer Systems (LIS1-A)*. Vol. 23 No. 7, (formerly ASTM E1381-02).
- [E1394] ASTM Designation: E 1394 – 91, *Standard Specification for Transferring Information Between Clinical Instruments and Computer Sys-*

- tems, American Society for Testing and Materials. West Conshohocken, U.S.A.*
- [E3128] ASTM E31.28 (2001) *Standard Specification for Relationship Between a Person (Consumer) and a Supplier of an Electronic Personal (Consumer) Health Record*, E2211-02.
- [ElNa00] ELMASRI, R., NAVATHE, S.B. (2000) *Fundamentals of Database Systems*. Third edition. Addison Wesley. Reading, Massachusetts, USA.
- [ENV12265] CEN/TC251 WG I (1995) *Electronic Healthcare Record Architecture*, Final Draft 2 (prENV12265). not publicly available.
- [ENV13606] CEN/TC251 (1999) *Health Informatics – Electronic Healthcare Record Communication, Part 1: Extended Architecture, Part 2: Domain Term List, Part 3: Distribution rules, Part 4: Messages for the exchange of information*. [online]. Available from:
<http://www.centc251.org/WItems/listwis.htm#WGI> [Accessed 24th July 2003].
- [FG99] FERRARA, F.M., GRIMSON, W. (1999) *The holistic architectural approach to integrating the healthcare record in the overall system*: Proceedings MIE 1999. IOS Press, pp. 847-852.
- [Fin00] FINKELSTEIN, C. (2000) *The Enterprise: XML Is Not a Silver Bullet*. DMReview, October 2000.
- [Fox93] CEN/TC251 (1993) *Investigation of Syntaxes for Existing Interchange Formats to be used in Healthcare* [online]. Maintained by Comité Européen de Normalisation, Technical Committee 251. Available from:
<http://miginfo.rug.ac.be:8001/centc251/fox/word0222.htm> [Accessed 12th July 2003].
- [GBG98] GRIMSON, W., BERRY, D., GRIMSON, J., STEPHENS, G., FELTON, E., GIVEN, P. AND O'MOORE, R. (1998) *Federated healthcare record server – the Synapses paradigm*: Int J Med Inf. 1998 Oct-Dec; 52 (1-3):3-27.

- [GGB98] GRIMSON, J., GRIMSON W., BERRY D., STEPHENS G., FELTON E., KALRA, D., TOUSSAINT, P. AND WEIER, O. (1998) *A CORBA-based integration of distributed electronic healthcare records using the Synapses approach*: IEEE Transactions on Information technology in Biomedicine, Vol.2, No.3, Sept 1998, pg. 124-138.
- [GGH00] GRIMSON, J., GRIMSON, W., HASSELBRING, W. (2000) *The SI challenge in health care*: Communications of the ACM, Volume 43, Issue 6 (June 2000), pp. 48-55.
- [Giv02] MCGIVNEY, J. (2002) *Implementation and Use of a DICOM based Centralised Image Store*, MSc Thesis, Centre for Health Informatics, University of Dublin, Ireland.
- [Gol91] GOLDFARB, C. (1991) *The SGML Handbook*, Oxford University Press.
- [GrEtal96] GRIMSON, J. ET.AL. (1996) *Synapses – Federated Healthcare Record Server*: Proceedings of the Medical Informatics in Europe conference (MIE) 1996. IOS Press, pp. 695-699.
- [GrEtal01] GRIMSON, J., STEPHENS, G., JUNG, B., GRIMSON, W., BERRY, D., PARDON, S. (2001) *Sharing Health-Care Records over the Internet*: IEEE Internet Computing, May/June 2001, pp. 49-58, IEEE Computer Society.
- [Gri01] GRIMSON J. (2001) *Delivering the electronic healthcare record for the 21st century*. Int J Med Inf. 2001 Dec; 64(2-3):111-27.
- [Gru93] GRUBER, T.R. (1993) *A translation approach to portable ontologies*: Knowledge Acquisition, 5(2):199-220.
- [HISA97] CEN/TC251 WG I (1997) *Healthcare Information System Architecture Part 1 (HISA) Healthcare Middleware Layer*. Final Draft prENV 12967-1, not publicly available.
- [HLS01] HEALTH LEVEL 7, *Homepage*. Available from: <http://www.hl7.org/> [Accessed 2nd December 2001].

- [ICD10] WHO (1996) *ICD-10, Volume 1: Tabular List, Volume 2: Instruction Manual, Volume 3: Alphabetical Index*. World Health Organisation, Geneva, Switzerland.
- [Ing95] INGRAM D. (1995) *The Good European Health Record: Health in the new communication age*, Laires, M.F., Ladeira, M.F., Christensen, J.P. (eds.), IOS, pg. 66-74.
- [ISIS99] ISIS XML/EDI PROJECT (2000) *Mapping from UML Generalised Message Descriptions to XML DTDs*. [online]. Available from: <http://www.tieke.fi/isis-xmledi/D2/UmlToDtdMapping05.doc> [Accessed 10th January 2000].
- [ISO8879] INTERNATIONAL STANDARDS ORGANISATION (ISO) (1986) *Information Processing - Text and Office Systems - Standard Generalized Markup Language (SGML)*. [online]. ISO 8879:1986. Available from: <http://www.iso.ch/cate/d16387.html> [Accessed 2nd July 2002].
- [ISO9735] ISO-TC154 (1988) *Electronic data interchange for administration, commerce and transport (EDIFACT) - Application level syntax rules*, ISO 9735:1988.
- [JAG00a] JUNG, B., ANDERSEN, E.P., GRIMSON, J. (2000) *Using XML for Seamless Integration of Distributed Electronic Patient Records: Conference Proceedings XML Scandinavia 2000*. Gothenburg, Sweden.
- [JAG00b] JUNG, B., ANDERSEN, E.P., GRIMSON, J. (2000) *SynExML as a vehicle for Electronic Patient Records – A status report from the SynEx project: Proceedings XML Europe 2000*. Paris, France. pp. 339-346.
- [JBG99] JUNG, B., BERRY, D., GRIMSON, J. (1999) *Using XML to Network Distributed Analytical Instruments – Back to the Future?: Proceedings of the 4th Annual Conference and Scientific Symposium of the Health Informatics Society of Ireland (HISI)*. Malahide, Ireland.
- [JGG99] JUNG, B., GRIMSON, W., GRIMSON, J. (1999) *The EHCR – if not now, when?: Proceedings TEHRE 2000*. London, United Kingdom. pp. 51-55.

- [JSG00] JUNG, B., STEPHENS, G., GRIMSON, J. (2000) *XML based EPR Architectures: How do they relate to Synapses*: Conference Proceedings Towards an Electronic Healthcare Record in Europe (TEHRE) 2000. London, UK. pp. 79-92
- [JuGr99a] JUNG, B., GRIMSON, J. (1999) *eXtended Structured Query Language (xSQL)*: Conference Proceedings XML Europe 1999. Paris, France. pp. 315-324.
- [JuGr99b] JUNG, B., GRIMSON, J. (1999) *Synapses/SynEx goes XML*: Conference Proceedings Medical Informatics in Europe (MIE) 1999. Ljubljana, Slovenia. pp. 906-911.
- [Kar00] KARSAI, G. (2000) *Design Tool Integration: An Exercise in Semantic Interoperability*: Conference Proceedings 7th IEEE Conference and Workshop on the Engineering of Computer Based Systems. Edinburgh, Scotland.
- [KlCa03] KLYNE, G., CARROLL, J.J. (eds) (2003) *Resource Description Framework (RDF): Concepts and Abstract Syntax*. [online]. W3C Working Draft (05 September 2003), W3C. Available from: <http://www.w3.org/TR/2003/WD-rdf-concepts-20030905/> [Accessed 10th September 2003].
- [LeHu03] LEE, K.P., HU, J. (2003) *XML Schema Representation of DICOM Structured Reporting*. J Am Med Inform Assoc. 2003, 10:213-223.
- [Leta199] LEWIS, P.H., HALL, W., CARR, L.A. AND DE ROURE, D. (1999) *The Significance of Linking*. [online]. Computing Surveys, volume 31, number 4, Available from: http://www.cs.brown.edu/memex/ACM_HypertextTestbed/papers/20.html [Accessed 14th July 2003].
- [Mad01] MADNICK, S.E. (2001) *The Misguided Silver Bullet: What XML Will and Will NOT Do to Help Information Integration*. Center for eBusiness@MIT. Paper 111. Massachusetts Institute of Technology, Boston, USA.

- [Mah01] O'MAHONI, B. (2001) *Getting It Together: A Messaging Standard for Primary Care in Ireland (version 2.0)*. Report commissioned by the National General Practice Information Technology (GPIT) Working Group.
- [McA99] MCALEESE, R. (edt) (1999) *Hypertext – theory into practice*. Intellect Ltd. Exeter, United Kingdom.
- [Nem03] NATIONAL ELECTRICAL MANUFACTURERS ASSOCIATION (NEMA) (2003) *DICOM: Strategic Document* [online]. Version 2.1. Available from:
http://medical.nema.org/dicom/geninfo/dicom_strategy/Strategy_2003-02-07.htm [Accessed 14th July 2002].
- [NHS225] NHS INFORMATION AUTHORITY (2001) *Health Level Seven (HL7)*. [online]. Chapter 225. Available from:
<http://www.nhsia.nhs.uk/step/pages/handbook.asp> [Accessed 30th August 2003].
- [PaSi02] PATEL-SCHNEIDER, P., SIMÉON, J. (2002) *The Yin/Yang Web: XML Syntax and RDF Semantics*: Proceedings of the 11th International conference on World Wide Web. Honolulu, Hawaii, USA. pp. 443-453.
- [Rec99] RECTOR, A.L. (1999) *Clinical terminology: Why is it so hard?:* Methods of Information in Medicine 1999; 38: 239-252.
- [RFC2279] YERGEAU, F. (1998) *UTF-8, a transformation format of ISO 10646*, RFC 2279.
- [RFC2373] HINDEN, R., DEERING, S. (1998) *IP Version 6 Addressing Architecture*. Internet Engineering Task Force, Request for Change 2373, RFC2373.
- [RFC2376] WHITEHEAD, E., MURATA, M. (1998) *XML Media Types*. [online]. Internet Engineering Task Force, Network Working Group. Available from: <http://www.ietf.org/rfc/rfc2376.txt> [Accessed 2nd May 2002].

- [RNG01] CLARK, J. (2001) *RELAX NG Specification*. [online]. Committee Specification (3 December 2001). Available from: <http://www.oasis-open.org/committees/relax-ng/spec-20011203.html> [Accessed 14th August 2002].
- [Rob01] ROBIE, J. (2001) *The Syntactic Web – Syntax and Semantics on the Web*: Proceedings of the XML 2001 conference. Orlando, USA.
- [Rob03] Robie, J. (2003) The Design of XQL. [online]. Available from: <http://www.ibiblio.org/xql/xql-design.html> [Accessed 24th August 2003].
- [Setal99] SCHADOW, G., McDONALD, C.J. ET AL (1999) *Units of Measure in Clinical Information Systems*: JAMIA. 6(2); Mar/Apr 1999; pp. 151-162.
- [SFJ95] STARREN, J., FRIEDMAN, C., JOHNSON, S.B. (1995) *The Columbia Integrated Speech Interpretation System (CISIS)*: Proceedings of the 19th Annual Symposium on Computer Applications in Medical Care (SCAMSI). New Orleans, USA. pp. 985.
- [ShLa90] SHETH, A.P., LARSON, J.A. (1990) *Federated database systems for managing distributed, heterogeneous and autonomous databases*: ACM Computing Surveys, Vol. 22, September 1990, pp. 183-235.
- [SJS99] SEOL, Y., JOHNSON, S.B., STARREN, J. (1999) *Use of the Extensible Stylesheet Language (XSL) for Medical data Transformation*: Proceedings AMIA Annual Symposium 1999. Washington, USA. pp. 142-146.
- [SMIL01] AYARS, J., BULTERMAN, D., COHEN, A., DAY, K., HODGE, E., HOSCHKA, P., HYCHE, E., JOURDAN, M., KIM, M., KUBOTA, K., LANPHER, R., LAYAÍDA, N., MICHEL, T., NEWMAN, D., VAN OSSENBRUGGEN, J., RUTLEDGE, L., SACCOCIO, B., SCHMITZ, P., TEN KATE, W. (2002) *Synchronized Multimedia Integration Language (SMIL 2.0)* [online]. W3C Recommendation (7 August 2001), W3C. Available from: <http://www.w3.org/TR/smil20/> [Accessed 2nd July 2002].

- [Sna99] SNAVELY, D. (edt) (1999) *DICOM - Supplement 30: Waveform Interchange*. DICOM Standards Committee, Working Group 1 - Cardiac and Vascular Information. Rosslyn, Virginia, USA.
- [SSR94] SCIORE, E., SIEGEL, M., ROSENTHAL, A. (1994) *Using semantic values to facilitate interoperability among heterogeneous information systems*. ACM Transactions on Database systems, Vol. 19, No. 2, pg. 254-290.
- [Ste66] STEIN, P. (1966) *Regulae iuris: From juristic rules to legal maxims*, University Press, Edinburgh.
- [StEtal00] STEPHENS, G., JUNG, B., BRENNAN, B., PARDON, S., GRIMSON, J. (2000) *Using a Synapses Server in a Distributed Health Environment (DHE)*: Proceedings of the 5th Annual Conference and Scientific Symposium of the Health Informatics Society of Ireland (HISI). Saggart, Ireland.
- [SVG01] FARRAILOLO, J. (Edt.) (2001) *Scalable Vector Graphics (SVG) 1.0 Specification* [online]. W3C Recommendation (4 September 2001), W3C. Available from: <http://www.w3.org/TR/SVG/> [Accessed 8th September 2003].
- [Syn01] SYNAPSES, *Homepage*. [online]. Maintained by Dublin Health Informatics Group. Dublin, Ireland. Available from: <http://www.cs.tcd.ie/synapses/> [Accessed 12th August 2003].
- [SynEx01] SYNEX, *Homepage*. [online]. Maintained by GESI srl, Rome, Italy. Available from: <http://www.gesi.it/synex/> [Accessed 1st June 2002].
- [SysCon02] SYSTEM CONCEPTS LTD (2002) *The main stages in developing a European Standard* [online]. Available from: <http://www.system-concepts.com/stds/hsebox5-2.html> [Accessed 25th February 2002].
- [TC251] CEN/TC251, *Homepage*. [online]. Maintained by Comité Européen de Normalisation, Technical Committee 251. Available from: <http://www.centc251.org/> [Accessed 14th December 2001].
- [UN00] UN ECONOMIC COMMISSION FOR EUROPE (2000) *United Nations Directories for Electronic Data Interchange for Administration, Com-*

- merce and Transport* [online]. United Nations, Geneva, Switzerland. Available from:
<http://www.unece.org/trade/untdid/d01a/tred/tred1501.htm> [Accessed 2nd September 2003].
- [UPIG01] URI PLANNING INTEREST GROUP, W3C/IETF (2001) *URIs, URls, and URNs: Clarifications and Recommendations 1.0*. [online]. W3C Note (21 September 2001), W3C. Available from:
<http://www.w3.org/TR/2001/NOTE-uri-clarification-20010921/> [Accessed 14th August 2003].
- [UsGr98] USDIN, T., GRAHAM, T. (1998) *XML: Not a Silver Bullet, But a Great Pipe Wrench: Standard View*. Vol. 6, No. 3, 09/1998.
- [VXML00] BOYER, L., DANIELSEN, P., FERRANS, J., KARAM, G., LADD, D., LUCAS, B., REHOR, K. (2000) *Voice eXtensible Markup Language (VoiceXML™) version 1.0*. [online]. W3C Note (5 May 2000), W3C. Available from: <http://www.w3.org/TR/voicexml/> [Accessed 14th September 2003].
- [Wee68] WEED, L.L. (1968) *Medical records that guide and teach*: N Engl J Med 1968; 278: 593-600 and 652-7.
- [Wetal03a] WANG, H., JUNG, B., AZUAJE, F.J., BLACK, N. (2003) *ecgML: Tools and Technologies for Multimedia ECG Presentation*: Proceedings XML Europe 2003. London, United Kingdom.
- [Wetal03b] WANG, H., AZUAJE, F.J., JUNG, B., BLACK, N. (2003) *A markup language for electrocardiogram data acquisition and analysis (ecgML)*: BMC Medical Informatics and Decision Making, Vol. 3, Number 4.
- [X3D02] WEB 3D CONSORTIUM (2002) *Extensible 3D (X3D™) Graphics*, International Standard, ISO/IEC FCD 19775:200x, Final Committee Draft.
- [XEnc02] EASTLAKE, D., REAGLE, J. (EDTS.) (2002) *XML Encryption Syntax and Processing*. [online]. W3C Recommendation (10 December

- 2002), W3C. Available from: <http://www.w3.org/TR/xmlenc-core/> [Accessed 15th January 2003].
- [XLin01] DEROSE, S., MALER, E., ORCHARD, D. (2001) *XML Linking Language (XLink) Version 1.0*. [online]. W3C Recommendation (27 June 2001), W3C. Available from: <http://www.w3.org/TR/xlink/> [Accessed 14th July 2003].
- [XML98] BRAY, T., PAOLI, J., SPEERBERG-MCQUEEN, C. M. (1998) *Extensible Markup Language (XML) 1.0*. [online]. W3C Recommendation (10 February 1998), W3C. Available from: <http://www.w3.org/TR/1998/REC-xml-19980210> [Accessed 1st December 2001].
- [XML00] BRAY, T., PAOLI, J., SPEERBERG-MCQUEEN, C. M., MALER, E. (2000) *Extensible Markup Language (XML) 1.0 (Second Edition)*. [online]. W3C Recommendation (6 October 2000), W3C. Available from: <http://www.w3.org/TR/REC-xml/> [Accessed 1st December 2001].
- [XPath99] CLARK, J., DEROSE, S. (1999) *XML Path Language (XPath) Version 1.0* [online]. W3C Recommendation (16 November 1999), W3C. Available from: <http://www.w3.org/TR/xpath> [Accessed 2nd August 2003].
- [XPoi01] DEROSE, S., MALER, E., DANIEL JR. R. (2001) *XML Pointer Language (XPointer) Version 1.0* [online]. W3C Last Call Working Draft (8 January 2001), W3C. Available from: <http://www.w3.org/TR/WD-xptr> [Accessed 8th July 2003].
- [XQL03] BOAG, S., CHAMBERLIN, D., FERNANDEZ, M.F., FLORESCU, D., ROBIE, J., SIMÉON, J. (2003) *XQuery 1.0: An XML Query Language* [online]. W3C Working Draft (02 May 2003), W3C. Available from: <http://www.w3.org/TR/2003/WD-xquery-20030502/> [Accessed 2nd August 2003].
- [XSD01a] THOMPSON, H.S., BEECH, D., MALONEY, M., MENDELSON, N. (2001) *XML Schema Part 1: Structures* [online]. W3C Recommendation (2 May 2001), W3C. Available from: <http://www.w3.org/TR/xmlschema-1/> [Accessed 15th August 2003].

- [XSD01b] BIRON, P.V., MALHOTRA, A. (2001) *XML Schema Part 2: Datatypes* [online]. W3C Recommendation (02 May 2001), W3C. Available from: <http://www.w3.org/TR/xmlschema-2/> [Accessed 15th August 2003].
- [XSig02] EASTLAKE, D., REAGLE, J., SOLO, D. (eds) (2002) *XML-Signature Syntax and Processing* [online]. W3C Recommendation (12 February 2002), W3C. Available from <http://www.w3.org/TR/xmlsig-core/> [Accessed 2nd August 2003].
- [XSL01] ADLER, S., BERGLUND, A., CARUSO, J., DEACH, S., GRAHAM, T., GROSSO, P., GUTENTAG, E., MILOWSKI, A., PARNELL, S., RICHMAN, J., ZILLES, S. (2001) *Extensible Stylesheet Language (XSL) - Version 1.0* [online]. W3C Recommendation (15 October 2001), W3C. Available from: <http://www.w3.org/TR/xsl/> [Accessed 4th January 2002].

Appendix

A. Typesetting Conventions

This Thesis uses typesetting conventions that are worth pointing out.

A.1. XML syntax components

References to XML keywords (e.g. ELEMENT, ATTRIBUTE and ENTITY) do always use SMALL CAPITAL LETTERS. In contrast, “element” and “attribute” denote components of a non-XML system, for instance an EPR model or database schema.

A.2. Program Code

Code blocks are marked by use of non-proportional font. All lines are numbered for distinct referencing (see **Code A-1**). Line numbers do not belong to the code.

```
1 <!ELEMENT RCproperty (#PCDATA) >
2 <!ATTLIST RCproperty Name CDATA #REQUIRED>
```

Code A-1: Code Block Example

Inline code sections use non-proportional font throughout this document.

Example: The ELEMENT RCproperty requires exactly one ATTRIBUTE Name of type CDATA.

B. SynExML: Document Type Definition (ver. 2.2, GOLD release)

The most recent release (version 2.2, GOLD release) of following DTD was published as part of the SynEx project deliverables.

```
1 <!-- ===== -->
2 <!-- -->
3 <!-- Name: SynExML -->
4 <!-- Version: 2.2 (GOLD release) -->
5 <!-- Date: 26/07/2000 -->
6 <!-- Copyright: SynEx Consortium -->
7 <!-- -->
8 <!-- Editor: -->
9 <!-- Benjamin JUNG (TCD, <benjamin.jung@cs.tcd.ie>) -->
10 <!-- -->
11 <!-- Contributing editor: -->
12 <!-- Tony AUSTIN (UCL, <t.austin@chime.ucl.ac.uk>) -->
```

```

13 <!-- -->
14 <!-- Contributing authors: -->
15 <!-- Jose ANDANY (HUG, <jose.andany@dim.hcuge.ch>) -->
16 <!-- Egil P. ANDERSEN (SHS, <egil.paulin.andersen@nr.no>) -->
17 <!-- Stephane SPAHNI (HUG, <stephane.spahni@dim.hcuge.ch>) -->
18 <!-- Yigang XU (Broussais, <xu@hbroussais.fr>) -->
19 <!-- Vladimir YURPALOV (IBEX, <vdy@ibex.ch>) -->
20 <!-- Andrei EMELIANENKO (IBEX, <ave@ibex.ch>) -->
21 <!-- Dipak KALRA (UCL, <d.kalra@chime.ucl.ac.uk>) -->
22 <!-- ===== -->
23
24 <!-- ===== -->
25 <!-- GENERAL COMMENTS -->
26 <!-- It is recommended to use the ISO Date/Time format to -->
27 <!-- express Times and Dates in ELEMENT content and -->
28 <!-- ATTRIBUTE values. -->
29 <!-- ===== -->
30
31 <!-- ===== -->
32 <!-- % RICattributes -->
33 <!-- Attributes common to every RIC in the Synapses Server -->
34 <!-- specification; i.e., attributes defined in class -->
35 <!-- Record Component and class RIC in the Synapses Object -->
36 <!-- View. -->
37 <!-- -->
38 <!-- Class RIC inherits class Record Component in the -->
39 <!-- Synapses Server specification. -->
40 <!-- -->
41 <!-- Record Component is the root class in the Synapses -->
42 <!-- Object View. The Object View contains the classes -->
43 <!-- from which objects constituting actual healthcare -->
44 <!-- records are instantiated. The Synapses Class View -->
45 <!-- contains classes from which objects constituting -->
46 <!-- healthcare record classes are instantiated -->
47 <!-- ===== -->
48
49 <!ENTITY % RICattributes "ClassName CDATA #REQUIRED
50 RCID ID #REQUIRED
51 RecordID CDATA #IMPLIED
52 LogUserID CDATA #IMPLIED
53 LogTime CDATA #IMPLIED
54 InvalidationUserID CDATA #IMPLIED
55 InvalidationTime CDATA #IMPLIED">
56
57 <!-- ===== -->
58 <!-- % RIAttributes -->
59 <!-- Attributes common to every RecordItem in the Synapses -->

```



```

60 <!-- Server specification; i.e., attributes defined in -->
61
62 <!-- class Record Component and class RecordItem in the -->
63 <!-- Synapses Object View. -->
64 <!-- -->
65 <!-- Class RecordItem inherits class Record Component in -->
66 <!-- the Synapses Server specification. -->
67 <!-- ===== -->
68
69 <!ENTITY % RIAttributes "ClassName CDATA #REQUIRED
70 RCID ID #REQUIRED
71 RecordID CDATA #IMPLIED
72 LogUserID CDATA #IMPLIED
73 LogTime CDATA #IMPLIED
74 InvalidationUserID CDATA #IMPLIED
75 InvalidationTime CDATA #IMPLIED
76 EventBeginTime CDATA #IMPLIED
77 EventEndTime CDATA #IMPLIED">
78
79 <!-- ===== -->
80 <!-- % CommonRICAttributes -->
81 <!-- CommonRICAttributes are attributes of RIC's that are -->
82 <!-- not defined in the Synapses specification, but which -->
83 <!-- all sites agree to add to this DTD. -->
84 <!-- -->
85 <!-- The Language attribute is used to specify the language-->
86 <!-- used for terms within the element to which it belongs.-->
87 <!-- ===== -->
88
89 <!ENTITY % CommonRICAttributes "Type CDATA #IMPLIED
90 Language CDATA #IMPLIED">
91
92 <!-- ===== -->
93 <!-- % CommonRIAttributes -->
94 <!-- CommonRIAttributes are attributes of RecordItem's -->
95 <!-- that are not defined in the Synapses specification, -->
96 <!-- but which all sites agree to add to this DTD. -->
97 <!-- -->
98 <!-- The Language attribute is used to specify the -->
99 <!-- language used for terms within the element to which -->
100 <!-- it belongs. -->
101 <!-- -->
102 <!-- The DataType attribute is used to specify type of -->
103 <!-- data value carried by the RecordItem to which it -->
104 <!-- belongs. -->
105 <!-- ===== -->
106
107 <!ENTITY % CommonRIAttributes "Type CDATA #IMPLIED

```

```

108                                     Language CDATA #IMPLIED
109                                     DataType CDATA #IMPLIED">
110
111 <!-- ===== -->
112 <!-- SynExML (SynEx Markup Language) -->
113 <!-- A SynExML file can contain a set of RecordFolder's, -->
114 <!-- FolderRIC's and ComRIC's in any sequence. -->
115 <!-- -->
116 <!-- Source specifies from where the XML is produced -->
117 <!-- ===== -->
118
119 <!ELEMENT SynExML (RecordFolder | FolderRIC | ComRIC)*>
120 <!ATTLIST SynExML Version CDATA #REQUIRED
121                Source CDATA #REQUIRED>
122
123 <!-- ===== -->
124 <!-- RCproperty -->
125 <!-- RCproperties are (name,value) pairs. -->
126 <!-- They are not part of the Synapses Server -->
127 <!-- specification, but they are included to support -->
128 <!-- site-specific attributes. That is, conceptually they -->
129 <!-- should be considered a site-specific addition to the -->
130 <!-- ATTLIST for a particular element (e.g. the -->
131 <!-- RecordFolder), and they are only included as nested -->
132 <!-- elements within e.g. RecordFolder for DTD-technical -->
133 <!-- reasons. For this reason they must always be the -->
134 <!-- first elements within the element to which they -->
135 <!-- belong (when parsing e.g. RecordFolder its attributes -->
136 <!-- should be known). -->
137 <!-- ===== -->
138
139 <!ELEMENT RCproperty (#PCDATA)>
140 <!ATTLIST RCproperty Name CDATA #REQUIRED>
141
142 <!-- ===== -->
143 <!-- RecordFolder (a healthcare record) -->
144 <!-- In Synapses every healthcare record is rooted in a -->
145 <!-- single RecordFolder object, and the structure of a -->
146 <!-- HCR is seen as a tree-structure of RIC's with -->
147 <!-- hyperlinks (ViewRIC2's) between them. -->
148 <!-- The elements that can be nested within a RecordFolder -->
149 <!-- element is as specified in the Synapses Server -->
150 <!-- specification; i.e., either a single ViewRIC2, or a -->
151 <!-- set of ComRIC's and/or FolderRIC's in any sequence. -->
152 <!-- In Synapses RecordItem's are used to represent data -->
153 <!-- values (as "dynamic attributes") attached to a -->
154 <!-- particular RIC (a RIC as a structural element in a -->
155 <!-- HCR). Thus beside its RIC children, a RecordFolder -->

```



```

156 <!--      can also contain a set of RecordItem's.      -->
157 <!-- ===== -->
158
159 <!ELEMENT RecordFolder
160      (RCproperty*,
161      ( (ComRIC | FolderRIC | RecordItem)* |
162      (ViewRIC2, RecordItem*) ) )>
163 <!ATTLIST RecordFolder %CommonRICAttributes;
164      %RICAttributes;>
165
166 <!-- ===== -->
167 <!--      FolderRIC (a healthcare folder)      -->
168 <!--      The elements that can be nested within a FolderRIC -->
169 <!--      element is as specified in the Synapses Server -->
170 <!--      specification (the same as for a RecordFolder - -->
171 <!--      RecordFolder is a specialisation of FolderRIC in -->
172 <!--      Synapses). -->
173 <!-- ===== -->
174
175 <!ELEMENT FolderRIC
176      (RCproperty*,
177      ( (ComRIC | FolderRIC | RecordItem)* |
178      (ViewRIC2, RecordItem*) ) )>
179 <!ATTLIST FolderRIC %CommonRICAttributes;
180      %RICAttributes;>
181
182 <!-- ===== -->
183 <!--      ComRIC (a healthcare document)      -->
184 <!--      The elements that can be nested within a ComRIC -->
185 <!--      element is as specified in the Synapses Server -->
186 <!--      specification; i.e., a set of DataRIC's, ViewRIC1's -->
187 <!--      and/or ViewRIC2's in any sequence. -->
188 <!--      In addition it can contain a set of RecordItem's -->
189 <!--      representing data values (as "dynamic attributes") -->
190 <!--      attached to this ComRIC. -->
191 <!-- ===== -->
192
193 <!ELEMENT ComRIC
194      ( RCproperty*,
195      (DataRIC | ViewRIC1 | ViewRIC2 | RecordItem)* )>
196 <!ATTLIST ComRIC %CommonRICAttributes;
197      %RICAttributes;>
198
199 <!-- ===== -->
200 <!--      DataRIC (a "field" within a healthcare document) -->
201 <!--      The elements that can be nested within a DataRIC -->
202 <!--      element is as specified in the Synapses Server -->
203 <!--      specification; i.e., a set of more DataRIC's, -->

```

```

204 <!-- ViewRIC1's and/or ViewRIC2's in any sequence. -->
205 <!-- In addition it can contain a set of RecordItem's -->
206 <!-- representing data values (as "dynamic attributes") -->
207 <!-- attached to this DataRIC. -->
208 <!-- ===== -->
209
210 <!ELEMENT DataRIC
211     ( RCproperty*,
212       (DataRIC | ViewRIC1 | ViewRIC2 | RecordItem)* )>
213 <!ATTLIST DataRIC %CommonRICAttributes;
214
215           %RICAttributes;>
216
217 <!-- ===== -->
218 <!-- ViewRIC1 (a "computed field" within a healthcare -->
219 <!-- document) -->
220 <!-- In Synapses a ViewRIC1 is similar to a DataRIC except -->
221 <!-- that its RecordItem's (its data values as "dynamic -->
222 <!-- attributes") are computed on demand. -->
223 <!-- ===== -->
224
225 <!ELEMENT ViewRIC1 (RCproperty*, RecordItem*)>
226 <!ATTLIST ViewRIC1 %CommonRICAttributes;
227           %RICAttributes;>
228
229 <!-- ===== -->
230 <!-- ViewRIC2 (a hyperlink between RIC's in two healthcare -->
231 <!-- records) -->
232 <!-- A ViewRIC2 specifies a link either to another RIC -->
233 <!-- within the same record, to a RIC within another record-->
234 <!-- at the same server, or to a RIC within another record -->
235 <!-- at another server. The Destination element specifies -->
236 <!-- the link target. -->
237 <!-- ===== -->
238
239 <!ELEMENT ViewRIC2 (RCproperty*, Destination?, RecordItem*)>
240 <!ATTLIST ViewRIC2 %CommonRICAttributes;
241           %RICAttributes;>
242
243 <!ELEMENT Destination EMPTY>
244 <!ATTLIST Destination ServerID CDATA #REQUIRED
245           RecordID CDATA #REQUIRED
246           RCID CDATA #REQUIRED>
247
248 <!-- ===== -->
249 <!-- RecordItem -->
250 <!-- RecordItem is defined within the Synapses Server -->
251 <!-- specification, but it is not defined with any content -->

```



```

252 <!--      (DataItem, as a specialisation of RecordItem, is just -->
253 <!--      included as an example (page 6 in the computational -->
254 <!--      viewpoint)). An implementation of a Synapses Server -->
255 <!--      is therefore free to define the content of -->
256 <!--      RecordItem's as suits it best. However, their purpose -->
257 <!--      are as "dynamic attributes" to RIC's; i.e. RIC's -->
258 <!--      define the structure of HCR while RI's contain the -->
259 <!--      data values attached to them. Therefore, to make -->
260 <!--      their "value" explicit, a Value element is added to -->
261 <!--      their DTD definition. -->
262 <!--      To allow for RecordItem's to define tree-structures -->
263 <!--      of values, "RecordItem*" is added to the DTD -->
264 <!--      specification. -->
265 <!--      As for the RIC's defined above, RCproperty* is only -->
266 <!--      meant to be used for extending the ATTLIST with -->
267 <!--      site-specific attributes. -->
268 <!--      It is recommended to attach childelements of -->
269 <!--      RecordItem in the following order: RCproperty, -->
270 <!--      ElementItem, LinkItem, RecordItem, #PCDATA. It is -->
271 <!--      also also recommended to keep the #PCDATA in a single -->
272 <!--      'data-island'. -->
273 <!--      ===== -->
274
275 <!ELEMENT RecordItem
276      ( #PCDATA | RCproperty | RecordItem )*>
277
278 <!ATTLIST RecordItem %CommonRIAttributes;
279      %RIAttributes;>
280
281 <!-- ===== END OF SynExML v.2.2 ===== -->
282

```

C. SynExML: XML Schema (ver. 2.2, GOLD release)

Trang³⁷, a multi-format schema converter based on RELAX NG, was used to convert the SynExML DTD from Appendix B into XML Schema. As DTDs are not as expressive as XML schemas, various modifications had to be made manually. These include deleting comments, adding data types (e.g. line 26), deleting repeating attribute definitions (e.g. in `<xs:attributeGroup name="RIAttributes">` content) and subsequently adding attributeGroup references (e.g. line 31).

³⁷ <http://www.thaiopensource.com/relaxng/trang.html>

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- ===== -->
3 <!-- -->
4 <!-- Name: SynExML -->
5 <!-- Version: 2.2 (GOLD release) -->
6 <!-- Date: 30/07/2003 -->
7 <!-- Copyright: SynEx Consortium -->
8 <!-- -->
9 <!-- This XML schema resembles the SynExXML DTD (GOLD release) -->
10 <!-- in its entirety. It was extended with XML schema specific -->
11 <!-- functionality such as data typing. -->
12 <!-- For comments on the various elements, please refer to the -->
13 <!-- original DTD documenttation. -->
14 <!-- -->
15 <!-- ===== -->
16
17 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
18           elementFormDefault="qualified">
19
20   <xs:attributeGroup name="RICattributes">
21     <xs:attribute name="ClassName" use="required"/>
22     <xs:attribute name="RCID" use="required" type="xs:ID"/>
23     <xs:attribute name="RecordID" type="string"/>
24     <xs:attribute name="LogUserID" type="string"/>
25     <xs:attribute name="LogTime" type="xs:dateTime"/>
26     <xs:attribute name="InvalidationUserID" type="xs:dateTime"/>
27     <xs:attribute name="InvalidationTime" type="xs:dateTime"/>
28   </xs:attributeGroup>
29
30   <xs:attributeGroup name="RIattributes">
31     <xs:attributeGroup ref="RICattributes"/>
32     <xs:attribute name="EventBeginTime" type="xs:dateTime"/>
33     <xs:attribute name="EventEndTime" type="xs:dateTime"/>
34
35   </xs:attributeGroup>
36
37   <xs:attributeGroup name="CommonRICattributes">
38     <xs:attribute name="Type" type="xs:string"/>
39     <xs:attribute name="Language" type="xs:string"/>
40   </xs:attributeGroup>
41
42   <xs:attributeGroup name="CommonRIattributes">
43     <xs:attributeGroup ref="CommonRICattributes"/>
44     <xs:attribute name="DataType" type="string"/>
45   </xs:attributeGroup>
46
47   <xs:element name="SynExXML">
48     <xs:complexType>

```



```
49     <xs:choice minOccurs="0" maxOccurs="unbounded">
50         <xs:element ref="RecordFolder"/>
51         <xs:element ref="FolderRIC"/>
52         <xs:element ref="ComRIC"/>
53     </xs:choice>
54     <xs:attribute name="Version" use="required" type="xs:string"/>
55     <xs:attribute name="Source" use="required" type="xs:string"/>
56
57 </xs:complexType>
58 </xs:element>
59
60 <xs:element name="RCproperty">
61     <xs:complexType mixed="true">
62         <xs:attribute name="Name" use="required" type="xs:string"/>
63     </xs:complexType>
64 </xs:element>
65
66 <xs:element name="RecordFolder">
67
68     <xs:complexType>
69         <xs:sequence>
70             <xs:element minOccurs="0" maxOccurs="unbounded"
71                 ref="RCproperty"/>
72             <xs:choice>
73                 <xs:choice minOccurs="0" maxOccurs="unbounded">
74                     <xs:element ref="ComRIC"/>
75                     <xs:element ref="FolderRIC"/>
76                     <xs:element ref="RecordItem"/>
77                 </xs:choice>
78                 <xs:sequence>
79                     <xs:element ref="ViewRIC2"/>
80                     <xs:element minOccurs="0" maxOccurs="unbounded"
81                         ref="RecordItem"/>
82                 </xs:sequence>
83             </xs:choice>
84         </xs:sequence>
85         <xs:attributeGroup ref="CommonRICAttributes"/>
86         <xs:attributeGroup ref="RICattributes"/>
87     </xs:complexType>
88 </xs:element>
89
90 <xs:element name="FolderRIC">
91     <xs:complexType>
92         <xs:sequence>
93             <xs:element minOccurs="0" maxOccurs="unbounded"
94                 ref="RCproperty"/>
95             <xs:choice>
96                 <xs:choice minOccurs="0" maxOccurs="unbounded">
97                     <xs:element ref="ComRIC"/>
```

```
98         <xs:element ref="FolderRIC"/>
99         <xs:element ref="RecordItem"/>
100     </xs:choice>
101     <xs:sequence>
102         <xs:element ref="ViewRIC2"/>
103         <xs:element minOccurs="0" maxOccurs="unbounded"
104             ref="RecordItem"/>
105     </xs:sequence>
106 </xs:choice>
107 </xs:sequence>
108 <xs:attributeGroup ref="CommonRICAttributes"/>
109 <xs:attributeGroup ref="RICAttributes"/>
110 </xs:complexType>
111 </xs:element>
112
113 <xs:element name="ComRIC">
114     <xs:complexType>
115         <xs:sequence>
116             <xs:element minOccurs="0" maxOccurs="unbounded"
117                 ref="RCproperty"/>
118             <xs:choice minOccurs="0" maxOccurs="unbounded">
119                 <xs:element ref="DataRIC"/>
120                 <xs:element ref="ViewRIC1"/>
121                 <xs:element ref="ViewRIC2"/>
122                 <xs:element ref="RecordItem"/>
123             </xs:choice>
124         </xs:sequence>
125         <xs:attributeGroup ref="CommonRICAttributes"/>
126         <xs:attributeGroup ref="RICAttributes"/>
127     </xs:complexType>
128 </xs:element>
129
130 <xs:element name="DataRIC">
131     <xs:complexType>
132         <xs:sequence>
133             <xs:element minOccurs="0" maxOccurs="unbounded"
134                 ref="RCproperty"/>
135             <xs:choice minOccurs="0" maxOccurs="unbounded">
136                 <xs:element ref="DataRIC"/>
137                 <xs:element ref="ViewRIC1"/>
138                 <xs:element ref="ViewRIC2"/>
139                 <xs:element ref="RecordItem"/>
140             </xs:choice>
141         </xs:sequence>
142         <xs:attributeGroup ref="CommonRICAttributes"/>
143         <xs:attributeGroup ref="RICAttributes"/>
144     </xs:complexType>
145 </xs:element>
```



```
146
147 <xs:element name="ViewRIC1">
148   <xs:complexType>
149     <xs:sequence>
150       <xs:element minOccurs="0" maxOccurs="unbounded"
151         ref="RCproperty"/>
152       <xs:element minOccurs="0" maxOccurs="unbounded"
153         ref="RecordItem"/>
154     </xs:sequence>
155     <xs:attributeGroup ref="CommonRICAttributes"/>
156     <xs:attributeGroup ref="RICAttributes"/>
157   </xs:complexType>
158 </xs:element>
159
160 <xs:element name="ViewRIC2">
161   <xs:complexType>
162     <xs:sequence>
163       <xs:element minOccurs="0" maxOccurs="unbounded"
164         ref="RCproperty"/>
165       <xs:element minOccurs="0" ref="Destination"/>
166       <xs:element minOccurs="0" maxOccurs="unbounded"
167         ref="RecordItem"/>
168     </xs:sequence>
169     <xs:attributeGroup ref="CommonRICAttributes"/>
170     <xs:attributeGroup ref="RICAttributes"/>
171   </xs:complexType>
172 </xs:element>
173 <xs:element name="Destination">
174   <xs:complexType>
175     <xs:attribute name="ServerID" use="required"/>
176     <xs:attribute name="RecordID" use="required"/>
177     <xs:attribute name="RCID" use="required"/>
178   </xs:complexType>
179 </xs:element>
180
181 <xs:element name="RecordItem">
182   <xs:complexType mixed="true">
183     <xs:choice minOccurs="0" maxOccurs="unbounded">
184       <xs:element ref="RCproperty"/>
185       <xs:element ref="RecordItem"/>
186     </xs:choice>
187     <xs:attributeGroup ref="CommonRIAttributes"/>
188     <xs:attributeGroup ref="RIAttributes"/>
189   </xs:complexType>
190 </xs:element>
191 </xs:schema>
192
193 <!-- ===== END OF SynExML v.2.2 ===== -->
```

D. ASTM 1394-91: Document Type Definition

The following DTD was generated from the reference model described in [E1394].

```
1 <!-- ===== -->
2 <!-- -->
3 <!-- Name: ASTM 1394-91 DTD -->
4 <!-- Version: 0.1 -->
5 <!-- Date: 11/09/1999 -->
6 <!-- Copyright: Benjamin Jung -->
7 <!-- -->
8 <!-- Editor: -->
9 <!-- Benjamin JUNG (TCD, <benjamin.jung@cs.tcd.ie>) -->
10 <!-- -->
11 <!-- Contributing editor: -->
12 <!-- Damon BERRY (DIT, <dberry@docsee.kst.dit.ie>) -->
13 <!-- -->
14 <!-- ===== -->
15
16 <!ENTITY % Person 'ID*, Name?, Address?, Telephone*'>
17
18
19 <!ELEMENT ASTM
20 ( MessageHeader?, Message+, MessageTerminator? )>
21 <!ATTLIST ASTM
22 Designation ( E1394 | CENTC251PT36 ) #REQUIRED
23 AccessPassword CDATA #IMPLIED
24 ProcessingID ( P | T | D | Q ) #IMPLIED
25 ControlID CDATA #IMPLIED>
26
27 <!ELEMENT Message
28 ( PatientInformation*, RequestInformation*, Scientific*,
29 ManufacturerInformation? )>
30
31
32 <!-- ===== -->
33 <!-- Message Header Record (ASTM 1394-91, Chapter 7) -->
34 <!-- ===== -->
35 <!ELEMENT MessageHeader
36 ( Comment*, AccessPassword?, Sender+, Receiver,
37 SpecialInstruction?, Version, DateTime )>
38
39 <!ELEMENT AccessPassword ( #PCDATA )>
40
41 <!ELEMENT Sender ( (Instrument | %Person;), Characteristics* )>
42
43 <!ELEMENT Instrument ( Label, Extension* )>
```



```
44 <ATTLIST Instrument Type CDATA #IMPLIED>
45
46
47 <!ELEMENT Address ( Street?, City?, State?, Zip?, Country? )>
48
49 <!ELEMENT Receiver ( %Person; )>
50
51 <!ELEMENT Version ( #PCDATA )>
52
53 <!ELEMENT SpecialInstruction ( #PCDATA )>
54
55 <!ELEMENT DateTime ( #PCDATA )>
56 <!ATTLIST DateTime Type CDATA #REQUIRED>
57
58 <!ELEMENT ID ( #PCDATA )>
59 <!ATTLIST ID Type CDATA #IMPLIED>
60
61 <!ELEMENT Name ( Last, First?, Middle?, Suffix?, Title? )>
62
63 <!ELEMENT Street ( #PCDATA )>
64
65 <!ELEMENT City ( #PCDATA )>
66
67 <!ELEMENT State ( #PCDATA )>
68
69 <!ELEMENT Zip ( #PCDATA )>
70
71 <!ELEMENT Country ( #PCDATA )>
72
73 <!ELEMENT Telephone ( #PCDATA )>
74
75 <!ELEMENT Characteristics ( #PCDATA )>
76
77 <!ELEMENT Last ( #PCDATA )>
78
79 <!ELEMENT First ( #PCDATA )>
80
81 <!ELEMENT Middle ( #PCDATA )>
82
83 <!ELEMENT Suffix ( #PCDATA )>
84
85 <!ELEMENT Title ( #PCDATA )>
86
87
88 <!-- ===== -->
89 <!-- Patient Information Record (ASTM 1394-91, Chapter 8) -->
90 <!-- ===== -->
91 <!ELEMENT PatientInformation
```

```
92         ( Comment*, %Person;, MothersMaidenName?,
93           Birthdate?, Sex?, RaceEthnicOrigin?, Address?,
94           AttendingPhysician*, Special*, Height?, Weight?,
95           Diagnosis*, ActiveMedication*, Diet?, Practice*,
96           AdmissionDate?, DischargeDate?, Location?,
97           ADCC*, Religion?, MaritalStatus?,
98           IsolationStatus*, Language?, Hospital?, Dosage?,
99           TestOrder* )>
100 <!ATTLIST PatientInformation
101           AdmissionStatus ( OP | PA | IP | ER ) #IMPLIED>
102
103 <!ELEMENT MothersMaidenName ( #PCDATA )>
104
105 <!ELEMENT Birthdate ( #PCDATA )>
106
107 <!ELEMENT Sex ( #PCDATA )>
108
109 <!ELEMENT RaceEthnicOrigin ( #PCDATA )>
110
111 <!ELEMENT AttendingPhysician ( %Person; )>
112
113 <!ELEMENT Special ( #PCDATA )>
114
115 <!ELEMENT Height ( #PCDATA )>
116
117 <!ELEMENT Weight ( #PCDATA )>
118
119 <!ELEMENT Diagnosis ( #PCDATA )>
120 <!ATTLIST Diagnosis Type ( known | suspected) #IMPLIED>
121
122 <!ELEMENT ActiveMedication ( #PCDATA )>
123
124 <!ELEMENT Diet ( #PCDATA )>
125
126 <!ELEMENT Practice ( #PCDATA )>
127
128 <!ELEMENT AdmissionDate ( #PCDATA )>
129 <!ATTLIST AdmissionDate Status CDATA #IMPLIED>
130
131 <!ELEMENT DischargeDate ( #PCDATA )>
132
133 <!ELEMENT Location ( #PCDATA )>
134
135 <!ELEMENT ADCC ( #PCDATA )>
136 <!ATTLIST ADCC Nature CDATA #IMPLIED>
137
138 <!ELEMENT Religion ( #PCDATA )>
139
```



```

140 <!ELEMENT MaritialStatus ( #PCDATA )>
141
142 <!ELEMENT IsolationStatus ( #PCDATA )>
143
144 <!ELEMENT Language ( #PCDATA )>
145
146 <!ELEMENT Hospital ( Service, Institution )>
147
148 <!ELEMENT Service ( #PCDATA )>
149 <!ATTLIST Service ID CDATA #REQUIRED>
150
151 <!ELEMENT Institution ( #PCDATA )>
152 <!ATTLIST Institution ID CDATA #REQUIRED>
153
154 <!ELEMENT Dosage ( Category, SubGroup* )>
155
156 <!ELEMENT Category ( #PCDATA )>
157
158 <!ELEMENT SubGroup ( #PCDATA )>
159
160
161 <!-- ===== -->
162 <!-- Test Order Record (ASTM 1394-91, Chapter 9) -->
163 <!-- ===== -->
164 <!ELEMENT TestOrder
165     ( Comment*, UniversalTestID, RequestDateTime?,
166       Specimen?, OrderingPhysician?, Users?, Laboratory?,
167       ResultsReportedDateTime?, InstrumentCharge?,
168       InstrumentSectionID?, NosocomialInjectionFlag?,
169       Result* )>
170 <!ATTLIST TestOrder
171     Priority      ( S | A | R | C )          #IMPLIED
172     ReportType   ( O | C | P | F | X | I | Y | Z | Q )
173                                     #IMPLIED>
174 <!ELEMENT UniversalTestID ( #PCDATA )>
175
176 <!ELEMENT Specimen
177     ( ID, InstrumentID, Collection+,
178       DangerCode?, RelevantClinicalInformation?,
179       ReceivedDateTime?, Descriptor?, Type?, Source?,
180       WardOfCollection?, Service?, Institution? )>
181 <!ATTLIST Specimen
182     ActionCode ( C | A | N | P | L | X | Q ) #IMPLIED >
183
184 <!ELEMENT OrderingPhysician ( %Person; )>
185
186 <!ELEMENT Users ( AuxData* )>
187

```

```

188 <!ELEMENT Laboratory ( AuxData* )>
189
190 <!ELEMENT ResultsReportedDateTime ( #PCDATA )>
191
192 <!ELEMENT InstrumentCharge ( #PCDATA )>
193
194 <!ELEMENT InstrumentSectionID ( #PCDATA )>
195
196 <!ELEMENT NosocimialInjectionFlag ( #PCDATA )>
197
198 <!ELEMENT InstrumentID ( #PCDATA )>
199
200 <!ELEMENT Collection ( Collector?, DateTime+, Volume? )>
201
202 <!ELEMENT Volume ( #PCDATA )>
203
204 <!ELEMENT Collector ( %Person )>
205
206 <!ELEMENT DangerCode ( #PCDATA )>
207
208 <!ELEMENT RelevantClinicalInformation ( #PCDATA )>
209
210 <!ELEMENT ReceivedDateTime ( #PCDATA )>
211
212 <!ELEMENT Descriptor ( #PCDATA )>
213
214 <!ELEMENT Type ( #PCDATA )>
215
216 <!ELEMENT Source ( #PCDATA )>
217
218 <!ELEMENT WardOfCollection ( #PCDATA )>
219
220
221 <!-- ===== -->
222 <!-- Result Record (ASTM 1394-91, Chapter 10) -->
223 <!-- ===== -->
224 <!ELEMENT Result
225     ( Comment*, UniversalTestID, Value, ReferenceRange*,
226       DOCInINV?, Identification?, DateTime* )>
227 <!ATTLIST Result
228     Status ( C | P | F | X | I | S | M | R | N | Q | V )
229                                     #IMPLIED
230     NatureOfAbnormality ( A | S | R | N ) #IMPLIED>
231
232 <!ELEMENT Value ( #PCDATA )>
233 <!ATTLIST Value
234     Units CDATA #IMPLIED
235     ResultAbnormal ( L | H | LL | HH | LLL | HHH | N | A |

```



```

236             U | D | B | W ) #IMPLIED>
237
238 <!ELEMENT ReferenceRange ( Value )>
239 <!ATTLIST ReferenceRange Type ( upper | lower ) #REQUIRED>
240
241 <!ELEMENT DOCInINV ( #PCDATA )>
242
243 <!ELEMENT Identification ( InstrumentOperator?, Verifier? )>
244
245 <!ELEMENT InstrumentOperator ( #PCDATA )>
246
247 <!ELEMENT Verifier ( #PCDATA )>
248
249
250 <!-- ===== -->
251 <!-- Comment Record (ASTM 1394-91, Chapter 11) -->
252 <!-- ===== -->
253 <!ELEMENT Comment ( #PCDATA )>
254 <!ATTLIST Comment ID CDATA #IMPLIED
255             Source ( P | L | I ) #REQUIRED
256             Type ( G | T | P | N | I ) #REQUIRED >
257
258
259 <!-- ===== -->
260 <!-- Request Information Record (ASTM 1394-91, Chapter 12) -->
261 <!-- ===== -->
262 <!ELEMENT RequestInformation
263             ( Comment*, UniversalTestID?, RangeID, RequestDateTime*,
264             Physician*, AuxData* )>
265 <!ATTLIST RequestInformation
266             Status ( C | P | F | X | I | S | M | R | A | N | O |
267             D ) #REQUIRED>
268
269 <!ELEMENT RangeID ( PatientID?, SpecimenID?, AuxData* )>
270 <!ATTLIST RangeID Type ( start | end ) #REQUIRED>
271
272 <!ELEMENT PatientID ( #PCDATA )>
273 <!ATTLIST PatientID Type ( start | end | all ) #IMPLIED
274             Source CDATA #IMPLIED>
275
276 <!ELEMENT SpecimenID ( #PCDATA )>
277 <!ATTLIST SpecimenID Type ( start | end | all ) #IMPLIED>
278
279 <!ELEMENT AuxData ( #PCDATA )>
280 <!ATTLIST AuxData Type CDATA #IMPLIED>
281
282 <!ELEMENT RequestDateTime ( #PCDATA )>
283 <!ATTLIST RequestDateTime

```

```
284         Type ( begin | end ) #REQUIRED
285         Nature ( S | R ) #REQUIRED>
286
287 <!ELEMENT Physician ( %Person; )>
288
289 <!-- ===== -->
290 <!-- Message Terminator Record (ASTM 1394-91, Chapter 13) -->
291 <!-- ===== -->
292 <!ELEMENT MessageTerminator ( Comment* )>
293 <!ATTLIST MessageTerminator
294         Code ( Nil | N | R | E | Q | I | F ) #REQUIRED>
295
296
297 <!-- ===== -->
298 <!-- Scientific Record (ASTM 1394-91, Chapter 14) -->
299 <!-- ===== -->
300 <!ELEMENT Scientific
301         ( Comment*, AnalyticalMethod?, Instrumentation,
302           Reagents*, UnitsOfMeasure*, QualityControl?,
303           SpecimenDescriptor?, SpecimenID?, Container?, Analyte?,
304           Result?, CollectionDateTime?, ResultDateTime?,
305           AnalyticalProcessingSteps?, Patient? )>
306
307 <!ELEMENT AnalyticalMethod ( #PCDATA )>
308
309 <!ELEMENT Instrumentation ( EMPTY )>
310 <!ATTLIST Instrumentation
311         ManufacturerCode CDATA #REQUIRED
312         InstrumentCode CDATA #REQUIRED>
313
314 <!ELEMENT Reagents ( #PCDATA )>
315
316 <!ELEMENT UnitsOfMeasure ( #PCDATA )>
317
318 <!ELEMENT QualityControl ( #PCDATA )>
319
320 <!ELEMENT SpecimenDescriptor ( #PCDATA )>
321
322 <!ELEMENT Container ( #PCDATA )>
323
324 <!ELEMENT Analyte ( #PCDATA )>
325
326 <!ELEMENT ResultValue ( #PCDATA )>
327 <!ATTLIST ResultValue Unit CDATA #REQUIRED>
328
329 <!ELEMENT ResultDateTime ( #PCDATA )>
330
331 <!ELEMENT AnalyticalProcessingSteps ( #PCDATA )>
```



```

332 <!ELEMENT Patient ( Diagnosis?, Birthdate?, Sex?, RaceEthnicOrigin?
333 )>
334
335
336 <!-- ===== -->
337 <!-- Manufacturer Information Record (ASTM 1394-91, Chapter 15) -->
338 <!-- ===== -->
339 <!ELEMENT ManufacturerInformation ( Comment*, AuxData* )>
340
341 <!-- ===== END OF ASTM 1394-91 v.0.1 ===== -->
342

```

E. SynOM state charts of hierarchical behaviour

The following five state charts were developed as one basic element of the first RSB prototype. It was used in the process of defining, building and evaluating XML formatted documents from internal, object-oriented SynODs structures.

Each of the five modules contains separate entry and exit points, labelled with a unique identifier. The transition from one state to the next is initiated by the opening tag (referenced as e.g. RF⁺) or closing tag (referenced as e.g. RF⁻) of an ELEMENT. In addition, most transitions include a condition (top part of the transition box) and computing (bottom part of transition box) aspect to keep track of the hierarchy level of the ELEMENT.

E.1. RecordFolder

A RECORDFOLDER is the root of the aggregation hierarchy and forms the basis for identification of the patient and entry into a particular patient's EHCR [GBG98]. See **Figure E-5** for a legend of the diagram.

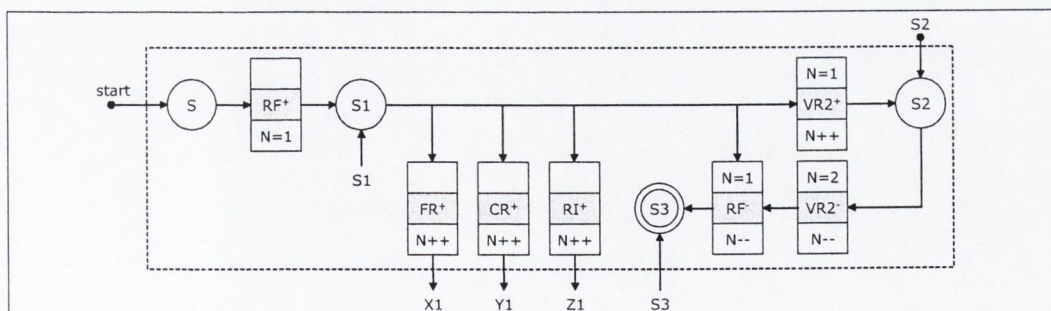


Figure E-1: RecordFolder state chart

E.2. FolderRIC

A FOLDERRIC is used to provide high-level structure to the record [GBG98]. See

Figure E-5 for a legend of the diagram.

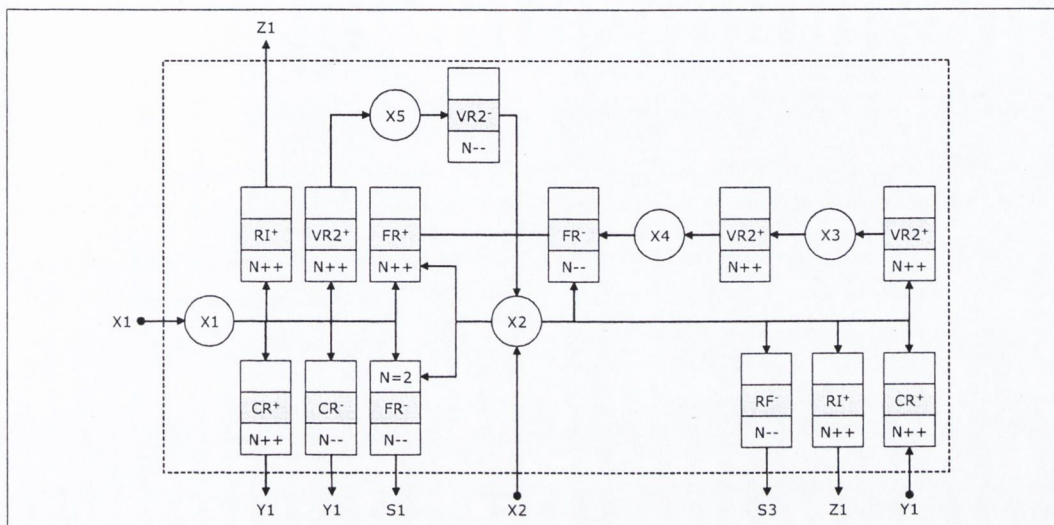


Figure E-2: FolderRIC state chart

E.3. ComRIC

A COMRIC may be considered as the autonomous set of ordered information in the

record that is allowed from an ethico-legal standpoint [GBG98]. See **Figure E-5** for

a legend of the diagram.

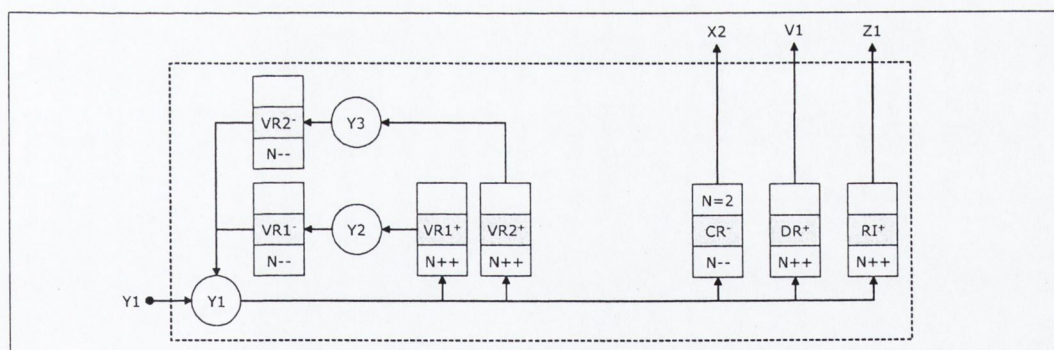


Figure E-3: ComRIC state chart

E.4. DataRIC

A DATARIC is used to provide primary context for the record items containing medi-

cal data [GBG98]. See **Figure E-5** for a legend of the diagram.

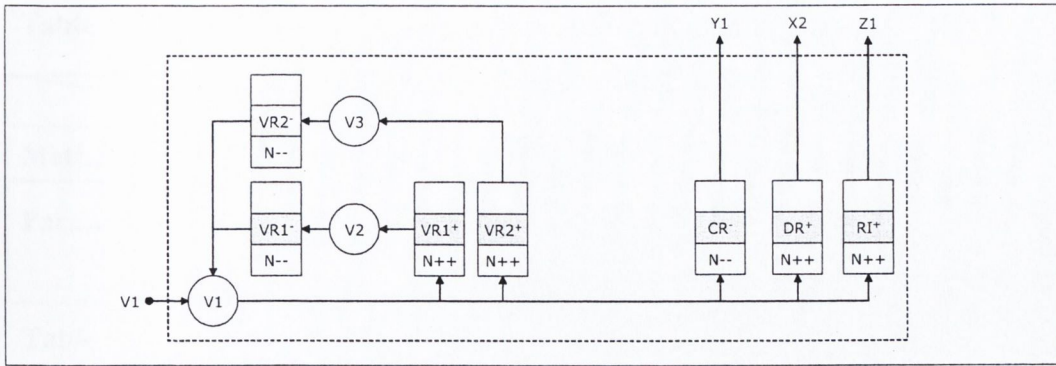


Figure E-4: DataRIC state chart

E.5. RecordItem

A RECORDITEM contains the data in its most basic unstructured form [GBG98].

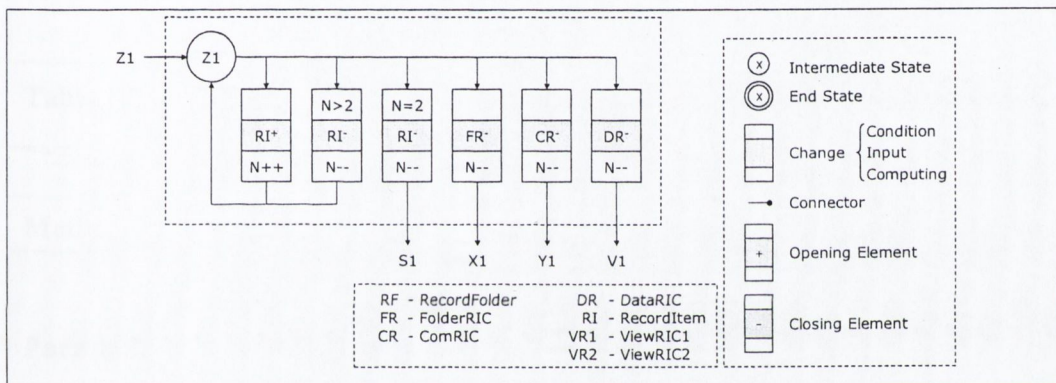


Figure E-5: RecordItem state chart

F. Synapses Server Interface Description

F.1. Common Gateway Interface (CGI)

The most recent implementation of a Synapses web interface uses CGI technology. A URL has to be assembled according to the following scheme calls in order to access the Synapses Record Server via the web.

1 `http://domain:port/path/method?parameter1+parameter2+...`

Code F-2: URL construct to access Synapses Record Server

See Section 5.4.1 for a more detailed description of the following API methods.

Table F-1: Synapses Server: Request Record IDs (CGI)

	Name	Example
Method	<u>getRecordIDs.exe</u>	
Parameter	SynapsesServerID ServerID UserID	<u>:SynServ</u> <u>localhost</u> <u>1</u>

Table F-2: Synapses Server: Request Record Shape (CGI)

	Name	Example
Method	<u>getShape.exe</u>	
Parameter	SynapsesServerID ServerID UserID RecordID	<u>:SynServ</u> <u>localhost</u> <u>1</u> <u>Z99444386</u>

Table F-3: Synapses Server: request ComRIC (CGI)

	Name	Example
Method	<u>getComRIC.exe</u>	
Parameter	SynapsesServerID ServerID UserID RecordID ComRicID	<u>:SynServ</u> <u>localhost</u> <u>1</u> <u>Z99444386</u> <u>51</u>

F.2. Web Services Definition Language (WSDL)

A simple WSDL definition to expose the Synapses Record Server as a web service.

```

1  <!-- ===== -->
2  <!-- -->
3  <!-- Name: WSDL for Synapses Record Server -->
4  <!-- Version: 1.0 -->
5  <!-- Date: 25/09/2002 -->
6  <!-- -->
7  <!-- Author: -->
8  <!-- Benjamin JUNG (TCD, <benjamin.jung@cs.tcd.ie>) -->
9  <!-- -->
10 <!-- ===== -->
11
12 <?xml version="1.0"?>
13
14 <definitions name="SynapsesRecordServer"
15     targetNamespace="http://namespaces.snowboard-info.com"
16     xmlns:tns=

```



```
17     "http://www.cs.tcd.ie/CHI/synapses/SynapsesRecordServer.wsdl"
18 xmlns:syn=
19     "http://www.cs.tcd.ie/CHI/synapses/SynapsesRecordServer.xsd"
20 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
21 xmlns="http://schemas.xmlsoap.org/wsdl/">
22
23 <import namespace="http://www.cs.tcd.ie/CHI/synapses/"
24     location="http://www.cs.tcd.ie/CHI/synapses/SynExML-2.2.xsd"/>
25
26 <types>
27     <xsd:schema
28         targetNamespace="http://www.cs.tcd.ie/CHI/synapses/"
29         xmlns="http://www.w3.org/1999/XMLSchema">
30
31         <!-- ===== -->
32         <!-- first of all, let's define all schema types that -->
33         <!-- we are going to use later in the wsdl file -->
34         <!-- (message elements) -->
35         <!-- ===== -->
36
37         <element name="SynapsesServerID" type="string"/>
38         <element name="ServerID" type="string"/>
39         <element name="UserID" type="nonNegativeInteger"/>
40         <element name="RecordID" type="string"/>
41         <element name="ComRicID" type="nonNegativeInteger"/>
42
43         <element name="IDs" type="string"
44             minOccurs="0" maxOccurs="unbounded"/>
45
46         <element name="fault">
47             <complexType>
48                 <all>
49                     <element name="errorMessage" type="string"/>
50                     <element name="errorNumber" type="integer"/>
51
52                 </all>
53             </complexType>
54         </element>
55
56     </schema>
57 </types>
58
59
60 <!-- ===== -->
61 <!-- the following message elements define the potential -->
62 <!-- transactions described in this wsdl file -->
63 <!-- ===== -->
64
```

```

65 <message name="RecordIdRequest">
66   <part name="SynapsesServerID" element="tns:SynapsesServerID" />
67   <part name="ServerID" element="tns:ServerID"/>
68   <part name="UserID" element="tns:UserID"/>
69 </message>
70
71 <message name="RecordIdResponse">
72   <part name="IDs" element="tns:IDs"/>
73 </message>
74
75 <message name="RecordShapeRequest">
76   <part name="SynapsesServerID" element="tns:SynapsesServerID" />
77   <part name="ServerID" element="tns:ServerID"/>
78   <part name="UserID" element="tns:UserID"/>
79   <part name="RecordID" element="tns:RecordID"/>
80 </message>
81
82 <message name="RecordShapeResponse">
83   <part name="RecordFolder" element="syn:RecordFolder"/>
84 </message>
85
86 <message name="ComRicRequest">
87   <part name="SynapsesServerID" element="tns:SynapsesServerID"/>
88   <part name="ServerID" element="tns:ServerID"/>
89   <part name="UserID" element="tns:UserID"/>
90   <part name="RecordID" element="tns:RecordID"/>
91   <part name="ComRicID" element="tns:ComRicID"/>
92 </message>
93
94 <message name="getComRicResponse">
95   <part name="ComRic" element="syn:ComRic"/>
96 </message>
97
98 <message name="srs_exception">
99   <part name="body" element="tns:fault"/>
100 </message>
101
102
103 <!-- ===== -->
104 <!-- next: we have to relate the various messages that are -->
105 <!-- part of an operation to the PortType -->
106 <!-- ===== -->
107
108 <portType name="SRS_PortType">
109   <operation name="getRecordIDs">
110     <input message="RecordIdRequest"/>
111     <output message="RecordIdResponse"/>
112     <fault message="srs_exception"/>

```



```

113     </operation>
114
115     <operation name="getRecordShape">
116         <input message="RecordShapeRequest"/>
117         <output message="RecordShapeResponse"/>
118         <fault message="srs_exception"/>
119     </operation>
120
121     <operation name="getComRIC">
122         <input message="ComRicRequest"/>
123         <output message="ComRicResponse"/>
124         <fault message="srs_exception"/>
125     </operation>
126 </portType>
127
128
129 <!-- ===== -->
130 <!-- next: create binding for the PortType -->
131 <!-- ===== -->
132
133 <binding name="SRS_Binding" type="tns:SRS_PortType">
134     <soap:binding style="rpc"
135         transport="http://schemas.xmlsoap.org/soap/http"/>
136     <soap:operation name="getRecordIDs">
137         <input>
138             <soap:body use="encoded"
139                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
140         </input>
141         <output>
142             <soap:body use="encoded"
143                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
144         </output>
145     </soap:operation>
146
147     <soap:operation name="getRecordShape">
148         <input>
149             <soap:body use="encoded"
150                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
151         </input>
152         <output>
153             <soap:body use="encoded"
154                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
155         </output>
156     </soap:operation>
157
158     <soap:operation name="getComRIC">
159         <input>
160             <soap:body use="encoded"

```

```

161         encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
162     </input>
163     <output>
164         <soap:body use="encoded"
165             encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
166     </output>
167 </soap:operation>
168 </binding>
169
170
171 <!-- ===== -->
172 <!-- finally: the services -->
173 <!-- ===== -->
174
175 <service name="Synapses Record Server">
176     <port name="SRS" binding="tns:SRS_Binding"
177         <soap:address
178             location="http://www.cs.tcd.ie/CHI/synapses/srs/" />
179     </port>
180 </service>
181
182 </definitions>
183
184 <!-- ===== END OF WSDL for Synapses Record Server ===== -->
185

```

G. Synapses Consortium Members

The following list contains contact details of each member institution of the SynEx consortium [Syn01].

Belgium

- RAMIT, Ghent, Belgium.

Denmark

- KOMMUNEDATA Aarhus, Aarhus, Denmark
- KOMMUNEDATA Ballerup, Ballerup, Denmark.
- Roskilde Amts Syngæhus, Roskilde, Denmark.

Finland

- Valtion teknillinen tutkimuskeskus (VTT), Tampere, Finland.

France

- Association Claude Bernard-Laboratoire de Recherche en Informatique, Paris, France.

Germany

- University of Göttingen, Göttingen, Germany.

Greece

- Areteion University Hospital, Athens, Greece
- Intrasoft S.A., Athens, Greece

Ireland

- Decision Support Systems Ltd. (DSS), Dublin, Ireland.
- Dublin Institute of Technology, Dublin, Ireland.
- St. James's Hospital, Dublin, Ireland.
- The University of Dublin, Trinity College, Dublin, Ireland.

Italy

- Gestione Sistemi per d'Informatica srl (GESI), Roma, Italy.
- Università Cattolica del Sacro Cuore (UCSC), Roma, Italy.

Luxembourg

- Association des Medecins et Medecins Dentistes, Luxembourg, Luxembourg.

Netherlands

- Academic Medical Center Amsterdam (AMC), Amsterdam, Netherlands.
- BAZIS/HISCOM, Leiden, Netherlands.
- University of Nijmegen, Nijmegen, Netherlands.

Norway

- Central Hospital of Akershus, Nordbyhagen, Norway
- Siemens Nixdorf, Oslo, Norway.

Sweden

- BMSA Uppsala University, Uppsala, Sweden.

Switzerland

- Hôpitaux Universitaires de Genève, Geneva, Switzerland.

United Kingdom

- Health Data Protection Ltd. acting on behalf of NHS IMC, Malvern, United Kingdom.
- ISHTAR Project Manager, Burton on Trent, United Kingdom.
- NHS IMC, Birmingham, United Kingdom.
- Royal Marsden Hospital, Sutton, United Kingdom.
- University College London, London, United Kingdom.

H. SynEx Consortium Members

The following list contains contact details of each member institution of the SynEx consortium [SynEx01].

Denmark

- Danish Institute for Health and Nursing Research, København, Denmark.

France

- Association Claude Bernard-Laboratoire de Recherche en Informatique Médicale, Paris, France.

United Kingdom

- University College London, London, United Kingdom.
- Victoria University Manchester, Manchester, United Kingdom.

Ireland

- Decision Support Systems Ltd. (DSS), Dublin, Ireland.
- Dublin Institute of Technology, Dublin, Ireland.
- St. James's Hospital, Dublin, Ireland.
- The University of Dublin, Trinity College Dublin, Dublin, Ireland.

Italy

- GESI srl – Gestione Sistemi per l'Informatica, Roma, Italy.
- Università Cattolica del Sacro Cuore, Roma, Italy.

Netherlands

- Erasmus University Rotterdam, Rotterdam, Netherlands.
- Katholieke Universiteit Nijmegen, Nijmegen, Netherlands.
- University Hospital Dijkzigt Rotterdam, Netherlands.

Norway

- Rikshospitalet, Oslo, Norway.
- Sentralsykehuset, Akershus, Norway.
- Siemens Nixdorf Informasjonssystemer A/S, Oslo, Norway.

Sweden

- BMSA Uppsala University, Uppsala, Sweden.
- Enator Medical AB, Lund, Sweden.

Switzerland

- Association des Médecins de Genève, Geneva, Switzerland.
- Hôpitaux Universitaires de Genève – Belle Idée, Geneva, Switzerland.

- Hôpitaux Universitaires de Genève, Geneva, Switzerland.
- IBEX Knowledge System, Geneva, Switzerland.
- IBM Suisse SA, Geneva, Switzerland.
- Novasys, Lausanne, Switzerland.
- Swiss Medical Association – Fedaratio Medicorum Helveticorum FMH, Geneva, Switzerland.

I. Acronym Glossary

ACR	American College of Radiology	DICOM	Digital Imaging and Communications in Medicine
API	Application Programming Interface	DLL	Dynamic Link Library
ASCII	American Standard Code for Information Interchange	DML	Data Manipulation Language
ASN	Abstract Syntax Notation	DOB	Date of Birth
ASP	Active Server Pages	DOM	Document Object Model
ASTM	American Society for Testing and Materials	DSFS	Dublin Synapses EHR Server
B2B	Business to Business	DTD	Document Type Definition
BIH	Beth Israel Hospital	ebXML	Electronic Business XML
CBPR	Computer Based Patient Record	ECG	Electrocardiogram
CDA	Clinical Document Architecture	ecgML	ECG Markup Language
CDATA	Character Data	ECMA	European Computer Manufacturers Association
CED	Computer Environment details Code Qualifier	EDI	Electronic Data Interchange
CEN	Comité Européen de Normalisation	EDIFACT	Electronic Data Interchange for Administration, Commerce and Transport
CERN	Conseil Européen pour la Recherche Nucléaire (European Laboratory for Particle Physics)	EEG	Electroencephalogram
CGI	Common Gateway Interface	EHCR	Electronic Health Care Record
CHI	Centre for Health Informatics	EHR	Electronic Health Record
COM	Component Object Model	EIA	Electronics Industry Association
CORBA	Common Request Broker Architecture	EPR	Electronic Patient Record
CPRI	Computer-based Patient Record Institute	ERD	Entity Relationship Diagram
CSV	Comma Separated Values	EU	European Union
CXML	Canonical XML	FDA	U.S Food and Drug Administration
DAML	DARPA Agent Markup Language	FHCR	Federated Healthcare Record
DBMS	Database Management System	FO	Formatting Objects
DDL	Data Definition Language	FTP	File Transfer Protocol
DHE	Distributed Health Environment	GEHR	Good Electronic Health Record
DHIG	Dublin Health Informatics Group	GML	Generalized Markup Language
		GP	General Practitioner
		GSC	Generic Synapses Client
		GUI	Graphical User Interface

HISA	Healthcare Information System Architecture	PCHR	Personal/Consumer Health Record
HL7	Health Level Seven	PDA	Personal Digital Assistant
HTML	Hypertext Markup Language	PDF	Portable Document Format
HTTP	Hypertext Transfer Protocol	PEMR	Paediatric Electronic Medical Record
HTTPS	HTTP using Secure Socket Layer	PID	Patient Identifier
HW	Hardware	PIS	Patient Information System
ICD	International Classification of Diseases	POCT	Point Of Care Testing
ICT	Information and Communications Technology	POMR	Problem Oriented Medical Record
ICU	Intensive Care Unit	PRA	Patient Record Architecture
IEEE	Institute of Electrical and Electronics Engineers	prENV	European pre-standard
ISBN	International Standard Book Number	PRI	Patient Record Identification
ISO	International Organisation of Standards	QL	Query Language
IT	Information Technology	QUILT	XML Query Language for Heterogeneous Data Sources
JAXP	Java API for XML Processing	RDBMS	Relational Database Management System
JDOM	Java Document Object Model	RDF	Resource Description Format
JSP	Java Server Pages	RFC	Request For Comment
LIS	Laboratory Information System	RH	Rikshospitalet (Oslo)
MERC	Maintainability-Extensibility-Reusability-Consistency	RI	Record Item (SynOM)
MGML	Minimal Generalized Markup Language	RIC	Record Item Complex (SynOM)
MIT	Massachusetts Institute of Technology	RIM	Reference Information Model
MRT	Magnet Resonance Tomography	RNG	Relax NG
MTS	Microsoft Transaction Server	RSB	Record Structure Builder
MVM	Multi View Modelling	RSS	Rich Site Summary
NEMA	National Electrical Manufacturers Association	RVML	Rich Vector Markup Language
NPT	Near Patient Testing	SAX	Simple API for XML
ODBC	Open Database Connectivity	SDE	Structured Data Entry
OIL	Ontology Interference Layer	SDI	Schema Development Interface
OODBMS	Object Oriented Database Management System	SGML	Standard Generalized Markup Language
OWL	Ontology Language for the Web	SHR	Semantic Health Record
PACS	Picture Archiving and Communications Systems	SIG	Special Interest Group
PCDATA	Parsable Character Data	SIMPL	Simple Internet Markup Protocol Language
		SJH	St. James's Hospital (Dublin)
		SMIL	Synchronized Multimedia Integration Language
		SOAP	Simple Object Access Protocol
		SQL	Structured Query Language

SR	Structured Reporting	URN	Uniform Resource Name
SRS	Synapses Record Server	UTF	Universal Transformation Format
SSN	Social Security Number	VRML	Virtual Reality Markup Language
SVG	Scalable Vector Graphics	VXML	VoiceXML
SW	Software	WAV	Waveform Audio
SynEx	Synergy on the Extranet	WML	Wireless Markup Language
SynOD	Synapses Object Dictionary	WSDL	Web Services Description Language
SynOM	Synapses Object Model	WWW	World Wide Web
TCP/IP	Transmission Control Protocol/Internet Protocol	XML	Extensible Markup Language
UDDI	Universal Description, Discovery and Integration	XQL	XML Query Language
UML	Unified Modelling Language	XSD	XML Schema Definition Language
UN	United Nations	XSL	Extensible Stylesheet Language
URC	Uniform Resource Class	XSLT	XSL Transformation
URI	Uniform Resource Identifier	xSQL	eXtended Structured Query Language
URL	Uniform Resource Locator		