INTELLIGENT HEALTHCARE MONITORING SYSTEM BASED ON SEMANTICALLY ENRICHED CLINICAL GUIDELINES

A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES OF MIDDLE EAST TECHNICAL UNIVERSITY

BY

GÖKÇE BANU LALECİ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN COMPUTER ENGINEERING

JUNE 2008

Approval of the thesis

"INTELLIGENT HEALTHCARE MONITORING SYSTEM BASED ON SEMANTICALLY ENRICHED CLINICAL GUIDELINES "

submitted by Gökçe Banu Laleci in partial fullfillment of the requirements for the degree of Doctor of Philosophy in Computer Engineering, Middle East Technical University by,

Prof. Dr. Canan Özgen _____ Dean, Graduate School of Natural and Applied Sciences

Prof. Dr. Volkan Atalay Head of Department, **Computer Engineering**

Prof. Dr. Asuman Doğaç Supervisor, **Department of Computer Engineering, METU**

Examining Committee Members:

Prof. Dr. İsmail Hakkı Toroslu Department of Computer Engineering, METU

Prof. Dr. Asuman Doğaç Department of Computer Engineering, METU

Prof. Dr. Mehmet Reşit Tolun Department of Computer Engineering, Çankaya University

Assoc. Prof. Dr. Nihan Kesim Çiçekli Department of Computer Engineering, METU

Assoc. Prof. Dr. Ahmet Coşar Department of Computer Engineering, METU

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

> Name, Last name : Gökçe Banu Laleci Signature :

ABSTRACT

INTELLIGENT HEALTHCARE MONITORING SYSTEM BASED ON SEMANTICALLY ENRICHED CLINICAL GUIDELINES

Laleci, Gökçe Banu Ph.D., Department of Computer Engineering Supervisor: Prof. Dr. Asuman Doğaç

June 2008, 175 pages

Clinical guidelines are developed to assist healthcare practitioners to make decisions on a patient's medical problems and as such they communicate with external applications to retrieve patient data, to initiate medical actions through clinical workflows and to transmit information to alert/reminder systems. The interoperability problems in the healthcare IT domain for interacting with heterogeneous clinical workflow systems and Electronic Healthcare Record (EHR) Systems prevent wider deployment of clinical guidelines because each deployment requires a tedious custom adaptation phase.

In this thesis, we provide machine processable mechanisms that express the semantics of clinical guideline interfaces so that automated processes can be used to access the clinical resources for guideline deployment and execution. For this purpose, we propose a semantically enriched clinical guideline representation formalism by extending one of the computer interpretable guideline representation languages, GuideLine Interchange Format (GLIF). To be able to deploy the semantically extended guidelines to healthcare settings semi-automatically, the underlying application's semantics must also be available. We describe how this can be achieved based on two prominent implementation technologies in use in the eHealth domain: Integrating Healthcare Enterprise (IHE) Cross Enterprise Document Sharing Integration Profile (XDS) for discovering and exchanging EHRs and Web service technology for interacting with the clinical workflows and wireless medical sensor devices. Since the deployment and execution architecture should be dynamic, and address the heterogeneity of underlying clinical environment, the deployment and execution is coordinated by a multi-agent system. The system described in this thesis is realized within the scope of the SAPHIRE Project.

Keywords: Semantics, Clinical Guidelines, Interoperability, Clinical Decision Support Systems

ÖΖ

ANLAMSAL OLARAK ZENGİNLEŞTİRİLMİŞ KLİNİK UYGULAMA KLAVUZ TABANLI AKILLI SAĞLIK TAKİP SİSTEMİ

Laleci, Gökçe Banu Doktora, Bilgisayar Mühendisliği Bölümü Tez Yöneticisi: Prof. Dr. Asuman Doğaç

Haziran 2008, 175 sayfa

Klinik yol haritaları hastaların sağlık problemleri konusundaki kararlarda doktorlara yardımcı olmak için geliştirilmişlerdir. Bunu gerçekleştirebilmek için hastanın sağlık kayıtlarına erişebilmeleri, klinik iş süreçlerindeki uygulamaları çalıştırabilmeleri ve alarm-hatırlatma sistemlerine bilgi gönderebilmeleri gerekmektedir. Sağlık bilişimindeki heterojen kilinik işsüreçlerine ve Elektonik Sağlık Kayıt Sistemlerine erişme aşamasındaki birlikte işlerlik problemleri, her hastaneye özgü farklı entegrasyon süreçleri gerektirdiğinden klinik yol haritalarının yaygın bir şekilde sahada kullanımını engellemektedir.

Bu tezde klinik yol haritalarının arayüzlerinin anlamsal olarak bilgisayarlar tarafından işlenebilir bir şekilde tanımlanmasını sağlayacak mekanizmalar önerilmektedir. Böylece klinik yol haitalarının hastane sistemleri üzerine otamatik olarak konuşlandırmak ve çalıştırmak mümkün kılınmaktadır. Bu amaçla, var olan klinik yol haritası betimleme dillerinden GLIF üzerine kurulu anlamsal bir model önerilmektedir. Anlamsal olarak zenginleştirilmiş bu klinik yol haritası tanımlarını yarı-otamatik olarak hastane sistemleri üzerine konuşlandırabilmek için, var olan hastane bilgi sistemleri altyapılarının da anlamsal olarak tanımlanması gerekmektedir. Bu tezde bu anlamsal tanımlama işleminin e-sağlık alanında yaygın olarak kullanılan iki uygulama teknolojisinin üzerine nasıl kurulabileceği de anlatılmaktadır. Bu teknolojiler sağlık kayıtlarına erişebilmek için "Sağlık Kurumlarını Entegrasyonu- Kurumlar Arası Doküman Paylaşım Profili (IHE-XDS)" ve klinik iş süreç sistemlerine ve kablosuz algılayıcı cihazlarına erişmek için Ağ Servisleridir. Konuşlandırma ve çalıştırma altyapısının dinamik olması ve heterojen bir altyapının ihtiyaçları ile başa çıkabiliyor olması gerektiğinden, konuşlandırma ve çalıştırma altyapısı bir çoklu-etmen platformu tarafında koordine edilmektedir. Bu tezde geliştirlen sistem SAPHIRE projesi tarafından desteklenmektedir.

Anahtar Kelimeler: Anlamsallık, Klinik Uygulama Klavuzu, Birlikte-İşlerlik, Klinik Karar Destek Sistemleri

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude to my supervisor Prof. Dr. Asuman Doğaç for all her guidance, encouragement, motivation and continuous support throughout my graduate studies. Without her great advice, help and understanding this work would never been possible.

I wish to express a lot of thanks to Prof. Dr. Uğur Halıcı and Assoc. Prof. Dr. Nihan Kesim Çiçekli for their valuable suggestions and comments throughout the steering meetings of this study.

Special thanks to Mehmet Olduz, İbrahim Taşyurt, Mustafa Yüksel, Yıldıray Kabak and all other members at the Software Research and Development Center for the cooperation and support they have provided.

I would also like to thank SAPHIRE End-users, SCUB Team, especially Dr. Ana Fruntelata for her support in understanding clinical guidelines.

I also want to express my gratefulness to my dear husband, Alpay, for all his patience, friendship and tolerance. Without his continuous support and encouragement, I would have never had the strength to complete this work.

Finally, I would like to thank to my parents and brother, for their support.

To my dear husband, Alpay..

TABLE OF CONTENTS

ABS	ГRAC	Τ	iv
ÖΖ.			vi
ACK	NOW	LEDGMENTS	viii
DED	ICAT	ON	ix
TAB	LE OI	F CONTENTS	x
LIST	OF F	IGURES	xiii
LIST	OF A	ABBREVIATIONS	cvii
CHA	PTEF	l de la companya de la companya de la companya de la companya de la companya de la companya de la companya de l	
1	INT	RODUCTION	1
2	BAG	CKGROUNG ON ENABLING TECHNOLOGIES	13
	2.1	Guideline Interchange Format, GLIF	13
	2.2	HL7 Clinical Document Architecture (CDA) and EN 13606-1 EHRcom	18
	2.3	Medical Terminologies and Terminonology Servers	20
	2.4	Integrating the Healthcare Enterprise - Cross Enterprise Document Shar-	
		ing Profile (IHE-XDS)	22
	2.5	OWLmt, An Ontology Mapping Tool	23
	2.6	Semantically Annotating Web services	26
	2.7	IEEE 11073 Standards Family	27
	2.8	JADE: Java Agent Development Environment	28
3	THI	E SEMANTICALLY ENRICHED GUIDELINE MODEL: THE MEDICAL	
Κ	NOW	LEDGE LAYER	30
	3.1	Semantically Extending GLIF to Facilitate Accessing the Electronic Health-	
		care Record Systems	32

3.2	Semantically Extending GLIF to Enhance Communication with Clinical	
	Workflows	37
3.3	Semantically Extending GLIF to Improve Communication with Alert and	
	Reminder Systems	42
4 TH	E ENABLING SEMANTIC ARCHITECTURE FOR THE DEPLOYMENT	١
AND E	XECUTION OF THE EXTENDED GLIF MODEL	44
4.1	Addressing the Semantic Interoperability for Accessing EHRs	47
2	4.1.1 Addressing the Interoperability of Patient Identifiers	50
2	4.1.2 Addressing the Interoperability of EHR Content	50
4.2	Addressing the Interoperability in Interacting with the Clinical Workflows	56
2	4.2.1 Addressing the Interoperability of Exchanged Messages used in the	
	Clinical Web Services	58
4.3	Addressing the Interaction with the Alert/Reminder Services $\ \ldots \ \ldots \ \ldots$	61
5 TH	E MULTI-AGENT SYSTEM RESPONSIBLE FOR DEPLOYMENT AND)
EXECU	JTION OF CLINICAL GUIDELINES	63
5.1	EHR Agent	67
5.2	Ontology Agent	70
5.3	Agent Factory Agent	76
5.4	Guideline Agent	79
Į	5.4.1 Action Specification Handlers	84
5.5	Alarm Distribution Agent	86
5.6	Monitoring Agent	88
6 A C	ASE STUDY IN THE ACUTE TREATMENT OF MYOCARDIAL INFARC-	
TION		94
7 REI	LATED WORK	104
7.1	Clinical Practice Guideline Representation Formalisms and Execution En-	
	vironments based on these Formalisms	104
,	7.1.1 Arden Syntax	104
,	7.1.2 Asbru Model	105
,	7.1.3 GUIDE	106
,	7.1.4 PROforma	108
,	7.1.5 GLEE	108

	7.1.6	GLARE	110
	7.1.7	SAGE: Standards-Based Sharable Active Guideline Environment $\ .$.	111
	7.1.8	An Analysis of the support of the Available Clinical Guideline Repre-	
		sentation and Execution Architectures for the interfaces with EHRs	
		and Clinical Workflows and Discussion of How this thesis comple-	
		ments these efforts	113
7.	2 HeC	ase2: Agent Based Management of Clinical Guidelines	115
	7.2.1	How this thesis complements HeCase2	118
• •	ONCLU	GLONG	110
8 C	ONCLU	SIONS	119
REFERI	ENCES		122
АТ	HE EXT	TENDED GLIF MODEL	132
ВТ	HE SER	VICE FUNCTIONALITY ONTOLOGY	136
СА	N EXAN	MPLE CDA USED IN THE PILOT APPLICATION	165
VITA .			171

LIST OF FIGURES

FIGURES

A Part of the Management of Diabetes Mellitus Guideline Text and	
Algorithm [59]	2
The main components of Thesis Study	5
A Part of the GLIF Model	13
An algorithm for the stable angina guideline in GLIF [29]	15
The Reference Information Model used by GLIF	16
A very simple guideline flow	17
An Example Get Data Action retrieving Aspirin Medication Status and	
an Example Medically Oriented Action Ordering Aspirin Prescription	
in GLIF Model	18
Logical Building Blocks of HL7 CDA	19
Logical Building Blocks of EHRcom [51]	20
IHE-XDS Architecture	23
Architecture of OWLmt [6]	24
A Part of the Formal GLIF Model	31
Extending GLIF with EHR Semantics	35
The Formal Description of GetDataAction and EHREntity in the Ex-	
tended GLIF Model	36
An example GetDataAction Instance in the Extended GLIF Model $\ .$.	36
The Formal Description of MedicallyOrientedAction and MedicalActio-	
nEntity in the Extended GLIF Model	38
Annotating "MedicalActionEntity" class with the Service Functionality	
Ontology	38
	Algorithm [59]

Figure 3.7	A Partial presentation of an example Service Functionality Ontology .	39
Figure 3.8	An example MedicallyOrientedAction Instance in the Extended GLIF	
	Model	41
Figure 3.9	An example MedicalActionEntity Instance for annotating interaction	
	with Sensor Devices in the Extended GLIF Model	41
Figure 3.10	The MessageAction and AlarmEntity in the Extended GLIF Model $\ .$.	42
Figure 3.11	Annotating "AlarmEntity" class with semantics	43
Figure 3.12	An Example MessageAction Instance in the Extended GLIF Model $\ .$.	43
Figure 4.1	Semantic Annotation of Guidelines and the Complementing Infrastruc-	
	ture	45
Figure 4.2	The Relationship between the Runtime Implementation Classes and	
	the Medical Knowledge Layer Classes	46
Figure 4.3	The Implementation Layer Specification in the Extended GLIF Model .	46
Figure 4.4	The Registration and Query of a HL7 CDA document to a XDS Reg-	
	istry/Repository Architecture	48
Figure 4.5	An example EHRImpl Instance in the Extended GLIF Model	48
Figure 4.6	Annotation and Discovery of Web Services Exposing EHRs	49
Figure 4.7	Patient Identifier Cross Referencing Architecture	51
Figure 4.8	An Example of Extracting GLIF Patient Entity from the HL7 CDA $\ .$.	52
Figure 4.9	The Discrete Data Import Algorithm	53
Figure 4.10	Mapping of terms from different Terminology System through UMLS	
	Concept Hierachy	54
Figure 4.11	Relating Service Functionality and Service Message Ontologies with	
	OWL-S	56
Figure 4.12	Publishing Medical Web Services with their Semantics to UDDI	57
Figure 4.13	An example ServiceImpl Instance in the Extended GLIF Model	58
Figure 4.14	An example mapping definition of WS parameters	60
Figure 4.15	Semantic Mediation of Clinical Web Service Messages	60
Figure 5.1	The Multi-Agent System responsible for Deployment and Execution of	
	Clinical Guidelines	64
Figure 5.2	The Multi-Agent System spanning across clinical affinity domains	67
Figure 5.3	Interaction of EHR Agent with other components for discovering EHR	
	References	68

Figure 5.4	An example queryEHR request and response	69
Figure 5.5	Interaction of EHR Agent with other components for Retrieving Clin-	
	ical Statements	70
Figure 5.6	An example retrieveEntry request and response	71
Figure 5.7	The Ontology Agent Interactions with other agents	72
Figure 5.8	The Ontology Service Reference Model proposed by FIPA [28] \ldots .	73
Figure 5.9	An example translation request and response	74
Figure 5.10	The Architecture of the Ontology Agent	75
Figure 5.11	Interaction of Agent Factory Agent with other components for deploy-	
	ing and Initializing a Clinical Guideline	76
Figure 5.12	The Agent Factory Agent Graphical Interface	77
Figure 5.13	Example MonitorGuideline and LoadGuideline request messages	79
Figure 5.14	The Guideline Agent Handlers	80
Figure 5.15	An example Java Script used as the "specification" of a Criterion	81
Figure 5.16	The Guideline Step Handlers	82
Figure 5.17	The Action Specification Handlers	84
Figure 5.18	Example Distribute Alarm Message request messages	86
Figure 5.19	The Architecture of Alarm Distibution Agent [1]	87
Figure 5.20	An example JESS Rule created by Alarm Distribution Agent	87
Figure 5.21	The Architecture of Monitoring Agent	88
Figure 5.22	Example Subscribe Monitoring Messages request message	89
Figure 5.23	The Graphical Interface of the Monitoring Agent	92
Figure 5.24	The Guideline History Window of Monitoring Agent	93
Figure 6.1	The Flowchart of "Management of acute myocardial infarction in pa-	
	tients presenting with ST-segment elevation" Guideline \ldots	95
Figure 6.2	The Flowchart of "FirstLine Medication"	96
Figure 6.3	The first part of the representation of the guideline in the extended	
	GLIF model	97
Figure 6.4	The "getBloodPressure" GetData Action representation in the extended $% \mathcal{A}^{(1)}$	
	GLIF	98
Figure 6.5	$The ``getAsthmaStatus'' \ GetData \ Action \ representation \ in \ the \ extended$	
	GLIF	98
Figure 6.6	The Script used in Morphine Treatment Decision Option	99

Figure 6.7	The "morphine Recommendation" Message Action representation in the
	extended GLIF
Figure 6.8	The second part of the representation of the guideline in the extended
	GLIF model
Figure 6.9	The "orderCoronaryAngiography" Medically Oriented Action represen-
	tation in the extended GLIF
Figure 7.1	The Architecture proposed in GUIDE [11]
Figure 7.2	The Architecture of the Interence Engine proposed in GUIDE $[11]$ 107
Figure 7.3	The internal structure of GLEE and its interactions with a local envi-
	ronment [100]
Figure 7.4	The GLARE's three layered architecture $[92]$
Figure 7.5	SAGE deployment architecture [77]
Figure 7.6	An example table that maps the patient data stored in institutional
	databases to the data model used in GLARE [72] $\ldots \ldots \ldots \ldots \ldots 114$
Figure 7.7	The Multi-agent System proposed by HeCase2 [42]

LIST OF ABBREVIATIONS

- ACC Agent Communication Channel
- ACL Agent Communication Language
- AMS Agent Management System
- CDA **Clinical Document Architecture**
- Domain Information Model DIM
- DF **Directory Facilitator**
- \mathbf{DL} **Description** Logics
- EHR Electronic Healthcare Record
- EMR Electronic Medical Record
- GUID Globally Unique Agent Identifier
- GLIF GuideLine Interchange Format
- ICD International Statistical Classification of Diseases
- IHE Integrating Healthcare Enterprise
- JADE Java Agent Development Framework
- JVM Java Virtual Machine
- KIF Knowledge Interchange Format
- LOINC Logical Observation Identifiers Names and Codes
- MeSH Medical Subject Headings
- MLM Medical Logic Module
- MTP Message Transport Protocol
- **OWL** Web Ontology Language
- **OWLmt** OWL Mapping Tool
- **OWL-QL** OWL Query Language

RIM	Reference Information Model
SAGE	Standards Based Sharable Active Guideline Environment
SFO	Service Functionality Ontology
SNOME	ED Systematized Nomenclature of Medicine
UDDI	Universal Description Discovery and Integration
UMLS	Unified Medical Language System
XDS	Cross Enterprise Document Shar- ing
XML	Extensible Markup Language
\mathbf{vMR}	Virtual Medical Record
WHO	World Health Organization
WSDL	Web Service Definition Language

CHAPTER 1

INTRODUCTION

Clinical guidelines are systematically developed statements to assist general practitioners in making clinical decisions and managing medical actions more effectively [41]. They usually include plans for treatment and are used in developing the "Clinical Decision Support" systems.

The scope of the clinical guidelines is ultimately to assist the physician in clinical decision, by providing evidence-based approach to certain conditions. The purpose of the guidelines is to provide standard, optimal healthcare based on the most recent evidence/ clinical trials in the field. The benefits of adopting clinical guidelines can be summarized as follows:

- Improved patient safety by reducing medical errors and improving selection of right medication and laboratory tests
- Improved quality of care by facilitating the use of up-to-date clinical evidence and by reducing inter-practice variations
- Improved efficiency in healthcare delivery by reducing test duplications and incidences of adverse events

Over the past 20 years, there has been an explosion in the availability of clinical practice guidelines. A variety of government and professional organizations are producing and disseminating clinical guidelines [97, 8, 14, 61, 46, 12]. Guidelines are usually developed by Task Forces of experts in the field such as "The Task Force on the Management of Acute Myocardial Infarction of the European Society of Cardiology" or by National and International Medical Societies such as "Infectious Diseases Society of America". They are regularly updated (at least once in 4-5 years) and usually represent standard of medical knowledge that is recommended to be followed by the medical practitioners in their clinical practice. Guide-

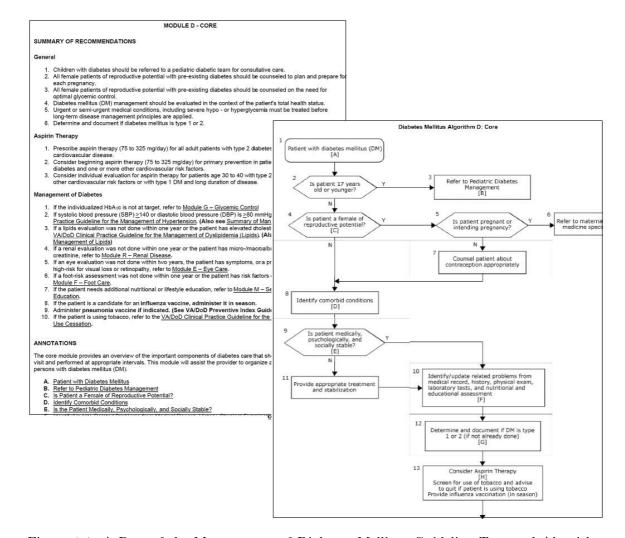


Figure 1.1: A Part of the Management of Diabetes Mellitus Guideline Text and Algorithm [59]

lines are usually provided as narrative descriptions accompanied by flowcharts, as presented in Figure 1.1.

It has been realized that clinician behavior is most effectively influenced through patientspecific advice, particularly if delivered during patient encounters. However, conventional narrative guidelines present population-based recommendations, and the information contained within such guidelines may be difficult to access and apply to a specific patient during patient encounters [72]. In order to address this problem, guideline-based point-of-care decision support systems are started to be built. A prerequisite for the development of such systems is the creation of computer interpretable representations of the clinical knowledge contained in clinical guidelines. For this purpose several computer interpretable models of clinical guidelines have been proposed such as GLIF [29], ASBRU [84], PROforma [90], AR- DEN [71] and EON [93]. Different approaches have been followed to model clinical guidelines, such as rule-based formalisms where the guidelines are represented as cause-effect relationships and network based models where algorithms, or hierarchical task networks are used. These computer interpretable formalisms increased the understandability and sharability of clinical guidelines by medical practitioners.

Several different guideline execution engines have been built processing these models, such as GLEE [100], GLARE [91], NewGuide [11] and DeGel [85] demonstrating that the guideline definitions can be executed to automate the decision making process.

However despite the benefits the clinical guidelines provide, it is a well accepted fact that wider adoption of computerized clinical practice guidelines by the healthcare community in real life healthcare setting is yet to be realized. The reasons for this failure can be summarized as follows:

- "The failure of integration of guideline implementations with clinical workflows" [25, 86]: The success of clinical decision-support systems requires that they are seamlessly integrated with the underlying clinical workflows of the healthcare organizations. The recommendations generated by the guideline should be reflected as events (institutional procedures) in the clinical workflow system of the hospitals such as a lab order, alerts or reminders to the respective healthcare staff.
- "The complexity of fully integrated decision support systems due to the nature of heterogeneous set of clinical applications need to be involved in the decision process" [22]: Lack of a common set of interoperable interfaces to proprietary hospital information systems hampers wide adoption of clinical guidelines, since developing generic clinical guideline execution engines that can be run on any set of clinical information systems based on the available guideline representation formalisms is not possible.
- "The failure of integration of guideline implementations with comprehensive computerized patient healthcare record data": In order to be able to provide patient-specific recomendations, clinical guideline execution environments need to access the electronic healthcare records of a patient. Lack of commonly agreed Electronic Healthcare Record standards is listed as another major problem that hinders wide adoption of clinical guidelines. This problem is usually cited as "Curly Braces Problem" in the literature. In one of the earliest guideline representation formalism, in Arden Syntax [71], the guideline is defined in term of Medical Logic Modules (MLMs). While the guidelines are being modeled, the references to clinical data are represented in curly braces in

MLMs. These references in curly braces should be mapped to queries to the institutional databases of healthcare organizations where actual patient electronic healthcare records are stored. Since then, the problem of this manual localization process for deploying guideline models is usually cited as "Curly Braces Problem".

These problems are cited as the main obstacles for achieving fully sharable and *deployable* clinical practice guideline implementations. The available clinical guideline execution engines often address the automation in a single homogeneous healthcare institute and either built on top of an already available clinical information system as an integrated add-on feature, or require custom adaptation phases to communicate with clinical applications such as for accessing the patient records or invoking medical services. These adaptation phases usually consist of manual mappings of the data models used in clinical guidelines to the data models used in local clinical repositories, and manual binding of clinical events supported by the underlying clinical information system to the action definitions in clinical guideline execution environment. This lack of integration support in the guideline representation languages and guideline execution environments is stated in GLIF specifications [29] as follows: "There is a need for an implementable specification that can be incorporated into an institutional system where the actions specified must be mapped to institutional procedures and the patient data references must be mapped to the electronic medical records of the underlying system". GLIF identified this requirement however latest GLIF specification left this level open as a possible future work.

In this thesis, we address this interoperability problem by developing a semantically enriched guideline model that enables the specification of enough level of semantics of the interfaces of the clinical guideline representation formalism to the underlying clinical applications and Electronic Healthcare Record (EHR) systems. The aim of this enrichment is to enable semi-automatic deployment of guidelines. Furthermore a semantic infrastructure based on widely accepted healthcare standards is described for the semi-automatic deployment and automated execution of guidelines using the semantics encoded in the guideline model proposed in heterogeneous healthcare settings. For this purpose we first describe how the semantics of underlying healthcare information systems can be enriched and annotated based on widely accepted industry standards. As a second step, we have built a multi-agent system which coordinates the semi-automatic deployment of clinical guidelines in heterogeneous healthcare settings using the semantically enriched clinical guideline definition and the semantically enriched healthcare information systems. The main components of this thesis is presented in Figure 1.2.

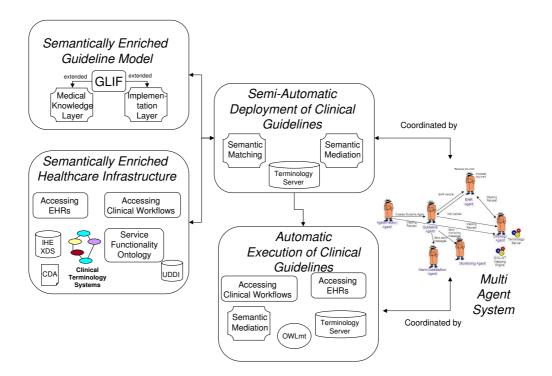


Figure 1.2: The main components of Thesis Study

The contributions of this thesis and details of these components can be introduced as follows:

• In this thesis, we propose a semantically enriched clinical guideline representation formalism. We decided to base our extensions on one of the available guideline representation formalism, in order to benefit from the already available substantial domain knowledge in these representation formalisms. We choose to semantically enrich the GuideLine Interchange Format (GLIF) [29] since GLIF model is formally expressed as an ontology.

GLIF in its current version, describes the clinical guidelines in two layers: first, at the conceptual level as a flowchart, second, by formally defining the medical concepts involved in the guideline definition as an ontology. A third layer is mentioned which aims to provide an implementable specification that can be deployed at a healthcare setting but this layer is left unspecified in the latest GLIF specifications. We argue that to have an implementable guideline specification, first the semantics of guideline's interfaces with clinical applications, and EHR systems must be described. In line with this requirement, we extend the GLIF model through a "Medical Knowledge Layer" to describe the semantics of GLIF execution steps based on the domain knowledge exposed by the related prominent healthcare IT standards as follows:

- Accessing the Content of Electronic Healthcare Records (EHRs): The most prominent EHR standards are the Health Level 7 (HL7) Clinical Document Architecture (CDA) [36] and the European Committee for Standardization (CEN) EN 13606-1 EHRcom [10]. Investigating these standards reveals that to locate an EHR document and to extract the requested patient clinical information from the EHR document, the semantics needs to be explicated at two levels: at the EHR document semantics level to discover the related EHR and at the entry semantics level to extract the clinical statement requested by the guideline. Based on this observation, we have created the "EHREntity" class as a subclass of the "Medical Knowledge Layer", where EHR Document Semantics and EHR Entity Semantics is formally represented. The "EHREntity" class is used to annotate GLIF's flowchart entities (namely the GetDataAction) while accessing EHRs.
- Accessing the medical services: The success of a clinical guideline execution system widely depends on how well it is integrated with the clinical workflow running in healthcare organizations. Previous efforts either did not attempt to define the semantics of clinical workflow interfaces of guidelines, or identified a number of fixed event types that should be manually bound to the events supported by the underlying clinical workflow. We choose to define the operational semantics of medical services through ontologies, and to use this semantics to semi-automatically discover and bind the services of the clinical workflows to guideline execution environment. For this purpose we defined an example Service Functionality Ontology based on well-established standards in the healthcare domain: for representing the semantics of clinical workflow services we exploited HL7 [34], which has categorized the events in healthcare domain by considering service functionality.

Furthermore during the execution of a guideline, there are decision points where the vital signs of a patient are needed. Currently wireless medical sensor devices are widely used both in hospital and in homecare settings. Representing the operational semantics of services through an ontology enables us to extend the types of events that can be described in a guideline definition: in our Service Functionality Ontology we define the semantics of sensor services through the nomenclature defined by the IEEE 11073-10101 standard [39] for medical devices. In order to annotate the operational semantics of GLIF's flowchart entities (namely the MedicallyOrientedAction), we created a "MedicalActionEntity" as a subclass of the "Medical Knowledge Layer".

- Accessing the Alert/Reminder Systems: The clinical guideline systems need to interact with the Alert/Reminder systems of the underlying clinical information systems. In order to seamlessly integrate guideline execution environment with alert/reminder systems, the required semantics is investigated and the healthcare roles to whom the alert messages should be delivered to and the urgency of the alarm are identified as necessary attributes of the "AlarmEntity" class of our "Medical Knowledge Layer". The "AlarmEntity" class is used to annotate the "MessageAction" flowchart entity of GLIF.
- Providing the semantics of guideline execution steps solves only one part of the problem: when deploying the extended guideline model over existing clinical applications, the underlying applications' semantics must also be available so that matches can be discovered. Therefore the extended guideline model is complemented with an "Implementation Layer" addressing both the technical interoperability challenge in accessing these underlying clinical applications and exposing the semantics of them. In this way we also demonstrate that through the extended guideline model, the deployment and execution of guidelines can be facilitated in heterogeneous healthcare environments. In this semantic infrastructure facilitating the semi-automatic deployment and execution of guidelines, latest trends and the most widely accepted medical standards are adopted for accessing EHRs and clinical workflow interfaces. This demonstrates that the proposed GLIF extensions can facilitate the deployment and execution of Clinical Guidelines in real life settings, i.e, what has been identified as a future research area in the latest GLIF specifications. Different facets of the "Implementation Layer" can be summarized as follows:
 - Addressing the Semantic Interoperability for Accessing EHRs: In our architecture we provide support for two different semantic mechanism for accessing the EHRs:
 - * The healthcare institutes may implement IHE Cross-Enterprise Document Sharing (XDS) profile [40] to share clinical records. IHE is an industry initiative, and XDS profile has been adopted for sharing EHRs in the National

eHealth System blueprints of Canada, USA, Italy, Norway and France. In IHE XDS, the healthcare institutes store the medical documents of the patients to repositories of their choice and access these repositories through the document metadata stored in the associated registries. In other words, the semantics of EHRs are expressed through metadata defined through terminology systems such as LOINC [58] stored at the "DocumentRegistry"s so that the documents of interest can be discovered and retrieved by using this metadata.

* The healthcare institutes may choose to expose the available Electronic Healthcare Records of the patients through Web Services. In this case, in order to find the EHR documents through their semantics, the functionality semantics of the Web services exposing them can be defined through LOINC document type codes [58].

In this thesis, we describe in detail how these two widely used industry standards can be exploited for enabling the discovery of the necessary Electronic Healthcare Records in the semi-automatic deployment of guidelines. It may be the case that different ontologies, or clinical terminology systems are used for the semantic annotation of clinical guidelines and EHR systems, in this case, we describe how this issue can be addressed through ontology mediation and terminology servers.

- Addressing the Interoperability of EHR Content: After the required EHR is located through the XDS Registries, or through the Web Services exposing EHRs, in order to execute the clinical guidelines, there is a need to extract a specific clinical statement from the EHR of the patient. For example, the guideline may need information on whether the patient has experienced any "asthma" previously. From the retrieved EHR, this specific clinical information must be extracted.

In this thesis, we describe how clinical statement data can be extracted from a machine processable EHR formats by using semantic mechanisms proposed in this work through an example based on HL7 Clinical Document Architecture (CDA) [36]. For this purpose we make use of a terminology server, the UMLS Knowledge Source Server Metathesaurus [96]. UMLS comprises over one million biomedical concepts and five million concept names, all of which are from over a hundred controlled vocabularies and classification systems such as ICD-9, SNOMED and LOINC, providing a mapping structure between them. In our architecture a

mapping functionality supporting subsumption is used based on the concepts and mapping provided by UMLS. In this way although different terminology systems (such as LOINC, SNOMED, ICD-10) are used to annotate the content of EHRs and the interfaces of clinical guidelines, the related clinical statement can be retrieved seamlessly from the CDA document.

- Addressing the Interoperability in Interacting with the Clinical Workflows: Considering that there could be several different applications running on different platforms that need to be invoked while the clinical guideline is being executed, Web services are used to address the technical interoperability problem in our implementation layer. Web service technology is already being used by the Healthcare Industry as a solution to technical interoperability problem, hence our implementation layer is realizable in real healthcare settings. The Dutch national infrastructure for healthcare messaging is implemented by wrapping HL7v3 messages as Web services [63] and Health Level Seven (HL7) has announced the approval of a Web Services Profile as Draft Standards for Trial Use (DSTUs) [35]. In another study, the integration of different medical systems in a configurable network of interconnected organizations in Canada is achieved by wrapping existing systems using Web services to provide a uniform and adaptable interface among the individual systems [65].

In this thesis, we presented how the operational semantics of the Web services exposing the functionality of clinical workflows can be described through our "Service Functionality Ontology" via their OWL-S [69] descriptions, and also how these services can be semantically published to UDDI registries [95], so that the discovery of these services can be achieved in the semi-automatic deployment phase [16].

Service functionality semantics enables us to discover the Web services based on their semantics. However, in order to invoke the discovered Web services while the guideline is executing, message level interoperability is also needed. Service functionality semantics may suffice to achieve interoperability only when all the Web services and guideline execution engines use the same message standards, and same reference information models. However, it is not realistic to assume that all the healthcare organizations comply with the same message structure and content. Hence, there is a need to transform one message content into another. In order to facilitate message transformation, our architecture utilizes ontology mapping. The OWLmt tool [6] is used for mapping the input and output parameters of Web services to the instances of the reference information model used in GLIF specification.

• The implementation layer described allows the semantically enriched guideline model to be semi-automatically bound to real life medical applications in the deployment phase and it becomes possible to specialize the guideline definition to a specific patient at run time for accessing his electronic healthcare records, and communicating with the healthcare institutes where he has been cured. The guideline needs deployed and executed in a distributed heterogeneous environment and the components involved must be capable of interacting with each other reactively. Therefore the semantically enriched clinical guidelines are deployed and executed on the semantic infrastructure through a multi-agent system. In this way it also becomes flexible to dynamically create and eliminate components based on the needs of the application when the components are implemented as software agents. The multi-agent system is implemented using JADE [45] agent development platform.

All the agents in our system has distinct roles that provides a level of abstraction to the other agents which is especially necessary in heterogeneous distributed healthcare settings, where different standards, ontologies, terminology systems and technologies may have been implemented to access Electronic Healthcare Records and clinical applications. The roles of the agents can be summarized as follows:

- The Agent Factory Agent is the coordinator of the deployment phase. It is capable of processing the generic guideline definition annotated with semantics and in cooperation with other agents in the system and by interacting with the resources of medical information systems, it discovers the real implementations of the medical services exposing hospital information system functionalities and sensor services and the document identifiers of the EHR documents of the patients, so that the guideline definition becomes ready to be executed.
- In our architecture, the healthcare institutes are organized as "Clinical Affinity Domains" to cooperate in the care of patients. The EHR Agent functions as the gateway to access and extract clinical data from the Electronic Healthcare records of the patient within a Clinical Affinity Domain. EHR Agent is modelled as a separate agent, to abstract the access to EHR from other agents. It has the

ability of communicating with the EHR Agents of other clinical affinity domains, to locate the EHRs of the patients wherever they are stored. Also each EHR Agent is capable of processing the EHR content format used within that affinity domain.

- The Ontology Agent is capable of reconciliation of semantic interoperability problems such as the use of different reference information models to represent clinical data or using different ontologies or clinical terminologies while accessing the resources of healthcare institutes. It implements the FIPA Ontology Service Specification [28].
- The Guideline Agent processes the guideline definition specialized to a patient and executes the activities specified in the guideline definition in co-operation with other agents. It can be thought as the enactment engine for the clinical guideline.
- The Monitoring Agent provides an interface to the clinical practitioners to visualize the execution of the guideline.
- The Alarm Distribution Agent is specifically designated to distribute the alertreminder messages to the necessary recipients in the most efficient and reliable way.

The system described in this thesis is realized within the scope of the SAPHIRE Project [82]. The aim of SAPHIRE Project is to provide an intelligent healthcare monitoring platform as a response to the challenge of providing high quality healthcare services with reasonable costs while the elderly population in Europe and the associated chronic diseases increase. The semantic clinical guideline description model and the semantically enriched guideline deployment and execution environment proposed in this thesis serves as one of the core components of SAPHIRE intelligent healthcare monitoring architecture: the intelligent clinical decision support system based on clinical guidelines. Through the extensions proposed in this thesis, the intelligent monitoring architecture is able to access seamlessly the medical history of a patient stored in medical information systems as well as the vital signs of the patients through wireless medical sensors. In this way, not only the observations received from wireless medical sensors but also the patient medical history can be used in the intelligent clinical clinical decision support system.

SAPHIRE is deployed through two pilot applications, one in a hospital environment and another for homecare. The hospital pilot application aims to demonstrate that the SAPHIRE system can provide bedside intelligent monitoring of patients with subacute coronary syndromes in a wireless fashion to provide computer-generated clinical decision in accordance to the latest European Cardiology Guidelines. In this pilot application, the guideline execution environment provides continuous feedback to the physicians that is patient-specific and guideline-oriented, to provide optimized medical care in accordance with medical standards.

The rest of this thesis is organized as follows: Chapter 2 briefly summarizes the enabling technologies and standards. In Chapter 3, the semantic extensions we propose to the GLIF model are presented. The enabling semantic architecture for the deployment and execution of the extended GLIF model is described in Chapter 4. In Chapter 5, the Multi-Agent System architecture that coordinates the deployment and execution of clinical guidelines is presented. The implementation details of SAPHIRE Hospital Pilot Application is presented as a case study in Chapter 6. Chapter 7 presents the related work in the literature in comparison with the architecture presented in this thesis. Finally, Chapter 8 concludes the thesis and suggests possible future research directions.

CHAPTER 2

BACKGROUNG ON ENABLING TECHNOLOGIES

2.1 Guideline Interchange Format, GLIF

The semantically extended guideline model we describe in this thesis is based on GuideLine Interchange Format (GLIF) [29]. GLIF is proposed as a standard computer interpretable representation model for sharing clinical guidelines among different healthcare institutes. It is presented by a research consortium including the Columbia University, the Harvard University and the Stanford University.

In the GLIF Model, clinical guidelines are represented as instances of a formal model called *guideline*. This formal model is represented as an ontology and editable by Protege [75]. GLIF uses a "task-based" paradigm for representing the guidelines, i.e., it decomposes the guideline definition hierarchically into networks of component tasks that unfold over time. A brief overview of the GLIF Model is given in Figure 2.1.

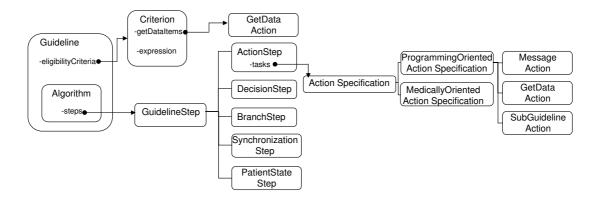


Figure 2.1: A Part of the GLIF Model

The *Eligibility Criteria* defines the features of the medical problem for which this guideline can be applied for. It is represented through a number of *Criterion* classes. The criteria is represented through the "expression" attribute of the *Criterion*. To be able to assess the eligibility, it may be necessary to check the current situation of the patient and his/her past medical history, hence these data gathering steps are added as "get data item" attributes to the *Criterion* class.

The clinical process is represented through an *algorithm*, which is a flowchart of guideline steps, including:

- An Action step is used for modeling actions to be performed which may include two types of tasks: Medically Oriented Actions, and Programming Oriented Actions. Through Medically Oriented Actions medical procedures usually run by medical staff such as recommendation for a particular course of treatment can be represented. The Programming Oriented Actions are divided into three: through the Get Data Action the actions that feeds data to guideline such as retrieving data from an electronic patient record are represented; through Message Action the messages that should be passed between medical systems, such as reminders, alerts are represented; finally through SubGuideline Actions the guideline definitions can be nested.
- A Decision step represents decision points in the guideline defined in terms of formal expressions. The GLIF specification exploits an expression language that is derived from the logical expression grammar of Arden Syntax [71]. Decision Steps include a number of Decision Options where decision expressions and destination steps that should be followed if this expression evaluates to true are presented.
- The *Branch step* is used to model concurrent guideline steps. Branch steps direct flow to multiple guideline steps. All of these guideline steps must occur in parallel. A branch step may link a guideline step to any other guideline step.
- The Synchronization steps are used in conjunction with branch steps. When multiple guideline steps follow a branch step, the flow of control can eventually converge in a single step. Each branch may lead to a series of steps, resulting in a set of branching paths. The step at which the paths converge is the synchronization step. When the flow of control reaches the synchronization step, a continuation attribute specifies whether all, some, or one of the preceding steps must have been completed before control can move to the next step.

• A *Patient state step* is usually used as a label that describes a patient state that is achieved by previous steps. This way, a guideline may be viewed as a state transition graph, where states are scenarios, or patient states, and transitions between these states are the networks of guideline steps (excluding patient state steps) that occur between two patient state step. A patient state step has a criterion that describes the state of the patient who is at that patient state.

An example of such guideline algorithm defined in GLIF is presented in Figure 2.2.

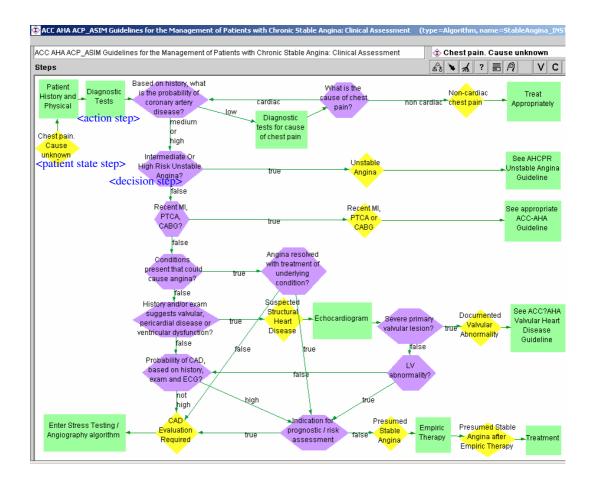


Figure 2.2: An algorithm for the stable angina guideline in GLIF [29]

GLIF Model proposed to represent the guidelines in three layers. The Core GLIF Model described above constitutes the "Level A". It enables the guideline author to concentrate on conceptualizing a guideline as a flowchart: the guideline is defined as a workflow of hierarchical steps defined in GLIF Formal Model. At this level of abstraction, the guideline author is not concerned with formally specifying details, such as decision criteria, relevant

patient data, and iteration information that must be provided to make the specification computable. GLIF proposes that in the second level of abstraction, that is, in "Level B", this workflow is detailed by formally expressing the patient data item definitions, the clinical concepts, and the logical criteria:

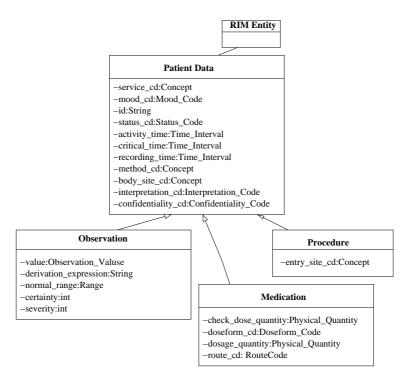


Figure 2.3: The Reference Information Model used by GLIF

- In "Level B" the *Clinical data* is represented as *data items*. For representing medical data items, GLIF supports the use of a Reference Information Model derived from HL7 RIM (USAM) [37] presented in Figure 2.3. This RIM defines a class hierarchy that organizes medical concepts into classes. For each class, it provides a data model that defines the attributes of the different classes. This enables the formal definition of high level *Patient data items* such as *Medication, Observation, and Procedure*.
- *Knowledge items* are used to define the *clinical concepts* which can be used to annotate the *data items* to relate them with well known medical terms. *Clinical concepts* are defined through the tuple <conceptName, conceptID, conceptSource>. For example, the "chronic cough" concept can be represented through the following tuple <chronic

cough, C0010201, UMLS> in reference to Unified Medical Language System (UMLS)[96] semantic network.

A formal expression language is used for representing decision criteria, triggering events, exceptional conditions, duration expression, and states in the guideline definition. For example a decision criteria can be defined as "selectAttribute(pq_value, selectAttribute(value, Current_LDL_Cholesterol)) >= 160" in this expression language.

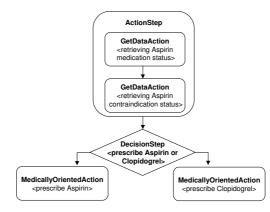


Figure 2.4: A very simple guideline flow

GLIF uses these two levels of abstractions to represent a guideline definition as follows: Assume an overly simplified guideline that decides whether to prescribe "Aspirin" or "Clopidogrel" as a firstline medication to a patient who is suffering from myocardial infarction as presented in Figure 2.4. The first guideline step in the guideline's "algorithm" can be an "action step" where it is necessary to gather data from patient's Electronic Healthcare Records (EHR). The tasks included in this "action step" are two "get data actions". The first "get data action" instance as shown in Figure 2.5 is to discover whether the patient is currently using the medication "Aspirin". This action states that the information should be retrieved from the "EHR" (also termed as Electronic Medical Record, EMR). The second "get data action" is for accessing the contraindications of the patient to the medication "Aspirin". After executing these actions, through a "decision step", it is checked whether it is advisable to prescribe Aspirin to the patient as presented in Figure 2.5, in the "OrderAspirin" "medically oriented action".

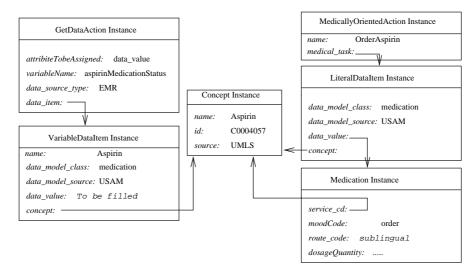


Figure 2.5: An Example Get Data Action retrieving Aspirin Medication Status and an Example Medically Oriented Action Ordering Aspirin Prescription in GLIF Model

As presented in the example, through GLIF Levels "A" and "B", a guideline is defined only at the conceptual level. In other words, how this model can be executed at a specific medical institute, how the EHR of the patient can be accessed, how the clinical procedures can be discovered and invoked are not specified. GLIF recognizes this requirement by defining a third level of abstraction, "Level C" or "Medical Knowledge Layer", which aims to provide an implementable specification. In this layer, the actions specified in the Level "A" and the patient data references presented in Level "B" must be mapped to institutional procedures and the electronic medical record systems of the underlying system of the medical institution where the guideline is to be deployed. For this third level of abstraction, the following needs have been identified:

- Interfaces to clinical repositories to retrieve Electronic Healthcare Records of a patient,
- Interfaces for interacting with applications such as clinical workflows and alert systems.

In the latest GLIF specification, these requirements have been noted but not addressed.

2.2 HL7 Clinical Document Architecture (CDA) and EN 13606-1 EHRcom

In the semantically enriched guideline model proposed in this thesis, the Electronic Healthcare Record semantics descriptions are based on the two prominent EHR standards, namely,

Clinic	calDocument
H	Ieader
S	tructuredBody
S	Section
	Entry
	Entry
	9
	Section
	Entry
	Entry

Figure 2.6: Logical Building Blocks of HL7 CDA

the HL7 Clinical Document Architecture (CDA) [36] and the EN 13606-1 EHRcom [10].

The HL7 CDA is a document markup standard that specifies the structure and semantics of "clinical documents" for the purpose of exchange. CDA documents are encoded in Extensible Markup Language (XML) and they derive their machine processable meaning from the HL7 Reference Information Model (RIM) [37] and use the HL7 Version 3 Data Types.

A CDA Document has two main components (Figure 2.6): the Header and the Body. The Header identifies and classifies the document and provides information on authentication, the encounter, the patient, and the involved providers. The Body of the CDA document contains the actual clinical report. The Body can be either an unstructured blob, or can be comprised of structured markup. The structured CDA Body is composed of nestable Sections. Each CDA Section can contain a single Narrative Block and a number of CDA Entries. These entries typically encode content presented in the narrative block of the same section in a machine processable manner through coded terms.

In the EHRcom specification [10], the *Extract* package defines the root class of the reference model ("EHR_EXTRACT") and the data structures for EHR content. Figure 2.7 shows the logical building blocks of EHR content. The top level is a directory of possibly nested *Folders* for a patient, allowing for a high-level organization of the EHR, for example, per episode or per clinical specialty. *Folders* contain zero or more *Compositions* by reference. A *Composition* (which roughly corresponds to a single clinical document) may contain *Sections* with section headers and *Entries* which consist of *Elements* or *Clusters* of elements. Each *Element* has a single value of a single data type.

In extending GLIF, we describe the patient clinical information needed at a guideline

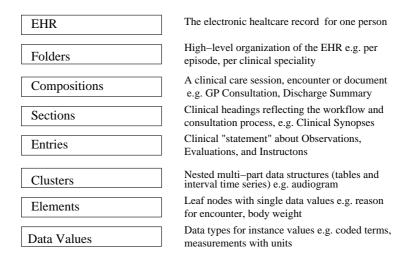


Figure 2.7: Logical Building Blocks of EHRcom [51]

execution step based on the way the structure and semantics are encoded in these two standards.

2.3 Medical Terminologies and Terminonology Servers

Terminologies are a collection of terms or concepts with subsumption relationships between them, which is used to build a classification hierachy of concepts. Terminologies provide a framework within which communities can communicate and express ideas in a consistent manner and facilitate unambiguous information share [3]. Medicine has a long tradition in structuring its domain knowledge through terminologies and coding schemes for diseases, medical procedures and anatomical terms such as SNOMED [87], LOINC [58], READ Codes [78], MeSH [60] and ICD-10 [38]. In the extended GLIF Model, clinical terms used are based on these terminology systems:

- SNOMED (Systematized Nomenclature of Medicine) [87], is a systematically organized computer processable collection of medical terminology covering most areas of clinical information such as diseases, findings, procedures, microorganisms, and pharmaceuticals. It allows a consistent way to index, store, retrieve, and aggregate clinical data across specialties and sites of care. SNOMED CT is a compositional concept system based on Description Logic [15], which means that concepts can be specialised by combinations with other concepts.
- Logical Observation Identifiers Names and Codes (LOINC) [58] is a universal standard for identifying laboratory observations and clinical results. Since its inception, it has

expanded to include not just medical and laboratory code names, but also nursing diagnosis, nursing interventions, outcomes classification, and patient care data set.

- The International Statistical Classification of Diseases and Related Health Problems 10th Revision (ICD-10) [38] is a coding of diseases and signs, symptoms, abnormal findings, complaints, social circumstances and external causes of injury or diseases, as classified by the World Health Organization (WHO) [102].
- READ CODES [78] are a coded thesaurus of clinical terms which enable clinicians to make effective use of computer systems. The codes facilitate the access of information within patient records to enable reporting, auditing, research, automation of repetitive tasks, electronic communication and decision support.
- Medical Subject Headings (MeSH) [60] is a huge controlled vocabulary for the purpose of indexing journal articles and books in the life sciences. MeSH vocabulary is created and updated by the United States National Library of Medicine (NLM) [98].

Since the available medical terminology systems are often developed for purposes with different requirements, the way the knowledge is organized differs between these heterogeneous terminologies; they exhibit considerable variability both in terms of coverage and granularity [83]. This created the need for homogeneous multi-purpose Clinical Terminology Servers that will allow the consistent and comparable entry of clinical data, e.g. patient observations, findings and events.

A Clinical Terminology Server provides the following services to client applications:

- management of external references to clinical concepts
- management of internal representations of clinical concepts
- mapping natural language to clinical concepts
- mapping clinical concepts to Medical classification schemes
- management of extrinsic information about clinical concepts

The Unified Medical Language System (UMLS) [96] is one of such clinical terminology servers. It is a controlled compendium of many medical vocabularies, also providing a mapping structure between them. The UMLS is composed of three main knowledge components: Metathesaurus, Semantic Network and SPECIALIST Lexicon:

- The Metathesaurus forms the base of the UMLS and it comprises over one million biomedical concepts and five million concept names, all of which are from over a hundred controlled vocabularies and classification systems such as ICD-9, SNOMED and LOINC. The purpose of the Metathesaurus is to provide a basis of context and intercontext relationships between these various coding systems and vocabularies to provide a common basis of information exchange between the variety of clinical systems.
- The UMLS Semantic network is designed to categorize concepts in the UMLS Metathesaurus and provide relationships among the concepts. It has 135 semantic types and 54 semantic relationships.
- The SPECIALIST Lexicon contains syntactic (how words are put together to create meaning), morphological (form and structure) and orthographic (spelling) information for biomedical terms.

GALEN [31] is another clinical terminology server which defines medical concepts in a descriptive logics model, called GRAIL, supporting subsumption.

In our architecture a mapping functionality supporting subsumption is used based on the concepts and mapping provided by UMLS.

2.4 Integrating the Healthcare Enterprise - Cross Enterprise Document Sharing Profile (IHE-XDS)

In thesis's architecture, an industry initiative called "Integrating the Healthcare Enterprise (IHE) Cross Enterprise Document Sharing (XDS)" Profile [40] is used for sharing Electronic Healthcare Records (EHRs) among different organizations in executing clinical guidelines. This profile has received considerable attention and appeared in the National eHealth System blueprints of Canada, Italy, Norway and France [80].

In the IHE XDS Profile, healthcare enterprises that agree to work together for clinical document sharing are called a "Clinical Affinity Domain" (Figure 2.8). Such institutes agree on a common set of policies such as how the patients are identified, the access is controlled, and the common set of coding terms to represent the metadata of the documents.

In each affinity domain there are a number of "Document Repositories": the healthcare institutes store the medical documents of the patients to these repositories in a transparent, secure, reliable and persistent way. It should be noted that these repositories can be the document stores that are already being used by the institutes, i.e., the profile does not force

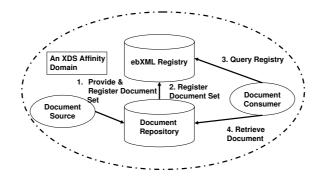


Figure 2.8: IHE-XDS Architecture

the institutes to create new dedicated repositories. There is a "Document Registry" which is responsible for storing metadata about those documents so that the documents of interest for the care of a patient may be easily found, selected and retrieved irrespective of the repository where they are actually stored.

The document repositories register the documents along with a set of metadata to the Document Registry. Whenever a "DocumentConsumer" wishes to discover a specific document of a patient, the "Query Registry" transaction is issued along with the specified query criteria, and as a response a list of document entries that contain metadata found to meet the specified criteria is returned including the locations and identifier of each corresponding document in one or more "Document Repositories". Using these document identifiers and the Document Repository URI's, the "Retrieve Document" transaction is issued to get the document content.

2.5 OWLmt, An Ontology Mapping Tool

Ontology Mapping is the process where two ontologies with an overlapping content are related at the conceptual level, and the source ontology instances are automatically transformed into the target ontology instances according to these relations. OWLmt tool [6] aims to handle ontology mediation by mapping the OWL ontologies in different structures and with an overlapping content one into other. The architecture of the system, as shown in Figure Figure 2.9, allows mapping patterns to be specified through a GUI tool based on a Mapping Schema. The Mapping Schema, is also defined in OWL.

Mapping patterns basically involve the following:

• Matching the source ontology classes to target ontology classes: In order to represent the

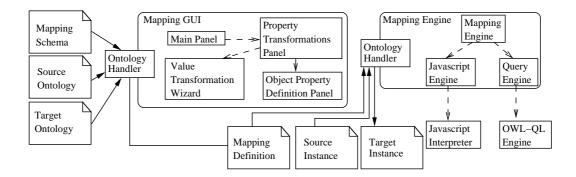


Figure 2.9: Architecture of OWLmt [6]

matching between the classes of source and target ontologies, four mapping patterns are available: EquivalentTo, SimilarTo, IntersectionOf and UnionOf. Two identical classes are mapped through EquivalentTo pattern. SimilarTo implies that the involved classes have overlapping content. How the similar classes are further related is detailed through their data type properties and object properties by using "property mapping patterns". The IntersectionOf pattern creates the corresponding instances of the target class as the intersection of the declared class instances. Similarly, the UnionOf pattern implies the union of the source classes' instances to create the corresponding instances of the target class. Furthermore, a class in a source ontology can be a more general (super class) of a class in the target ontology. In this case, which instances of the source ontology makes up the instances of the target ontology is defined through KIF (Knowledge Interchange Format) [52] conditions to be executed by the OWLmt mapping engine. When a source ontology class is a more specific (sub class) of a target ontology class, all the instances of the source ontology qualify as the instances of the target ontology.

• Matching the source ontology Object Properties to target ontology Object Properties: In addition to matching a single object property in the source ontology with a single object property in the target ontology, in some cases, more than one object property in the source ontology can be matched with one or more object properties in the target ontology. OWLmt allows defining "ObjectPropertyTransform" pattern which represents the path of classes connected through object properties such that whenever a path defined in the source ontology (inputPath) is encountered in the source ontology instance, the path defined for target ontology (outputPath) is created in the target ontology instance. Paths are defined as triples in KIF [52] format and executed through the OWL Query Language (OWL-QL) [68] engine. OWLmt constructs the specified paths among the instances of the target ontology in the execution step based on the paths defined among the instances of the source ontology.

• Matching source ontology Data Properties to target ontology Data Properties: Specifying the "DatatypePropertyTransform" helps to transform data type properties of an instance in the source ontology to the corresponding data type properties of instance in the target ontology. Since the data type properties may be structurally different in source and target ontologies, more complex transformation operations may be necessary than copying the data in source instance to the target instance. XPath specification defines a set of basic operators and functions which are used by the OWLmt such as "concat", "split", "substring", "abs", and "floor". In some cases, there is a further need for a programmatic approach to specify complex functions. For example, the use of conditional branches (e.g. if-then-else, switch-case) or iterations (e.g while, for-next) may be necessary in specifying the transformation functions. Therefore, JavaScript support is added to OWLmt. By specifying the JavaScript to be used in the "DatatypePropertyTransform" pattern, the complex functions can also be applied to the data as well as the basic functions and the operators provided by XPath.

The mapping definition is created in terms of these patterns through a graphical interface. The mapping engine is responsible for creating the target ontology instances using the mapping patterns given in the Mapping Definition and the instances of the source ontology. It uses OWL-QL to retrieve required data from the source ontology instances. OWL-QL is a query language for OWL developed at the Stanford University [68]. While executing the class and property mapping patterns, the query strings defined through the mapping GUI are send to the OWL-QL engine with the URL of the source ontology instances. The query engine executes the query strings and returns the query results. The use of the OWL-QL enables OWLmt to have reasoning capabilities. When querying the source ontology instances or while executing the KIF [52] patterns, OWL-QL reasons over the explicitly stated facts to infer new information.

After executing the class mapping patterns, the mapping engine executes the property mapping patterns. Similar to the class mapping patterns, OWL-QL queries are used to locate the data. In order to perform value transformations, the mapping engine uses the JavaScripts in the "DatatypePropertyTransform" pattern. To execute the JavaScripts, an interpreter is used. The engine prepares the JavaScript by providing the values for the input parameters and sends it to the interpreter. The interpreter returns the result, which is then inserted as the value of the data type property in the target ontology instance.

The OWLmt tool is publicly available in SourceForge [67].

2.6 Semantically Annotating Web services

In this thesis we use semantically enriched Web services for accessing the underlying clinical applications and sensor devices. In this way, the Clinical Web Services can be referenced with their semantics in the guideline definition, and the Web Service instances can be discovered automatically while the guideline is being deployed.

Several methodologies have been proposed for automated discovery of Web services through their semantics in the literature such as [99, 70, 104, 57]. In [57], the design and implementation of a service matchmaking prototype which uses a DAML-S based ontology and a Description Logic reasoner to compare ontology based service descriptions is given. In [70], another service matching algorithm based on DAML-S ontologies [13] are presented, where all the input and output parameters of the advertisements in UDDI registries are compared with those of the requested service. Alternatively, in [99], an extension to WSDL is proposed to semantically annotate and advertise Web services to UDDI registries in order to facilitate their discovery. In addition to these initiatives, WSMO [104] provides a formalism based on F-Logic to define goals and web service capabilities, and also a service discovery mechanism based on these. In our architecture we have chosen to use an OWL-S [69] (formerly DAML-S) based approach for semantically annotating the Web services.

OWL-S provides an upper ontology which defines a top level "Service" class with some generic properties common to most of the services. The "Service" class has the following three properties:

- *presents*: The range of this property is ServiceProfile class. That is, the class Service presents a ServiceProfile to specify what the service provides for its users as well as what the service requires from its users.
- *describedBy*: The range of this property is ServiceModel class. That is, the class Service is describedBy a ServiceModel to specify how it works.
- *supports*: The range of this property is ServiceGrounding. That is, the class Service supports a ServiceGrounding to specify how it is used.

In our architecture, after the service semantics is defined through OWL-S, the Web services are published to UDDI registries [95] in order to facilitate their discovery. UDDI itself does not provide support for storing ontologies to the Registry and for relating the Service definitions with these ontologies. Previously in [16], we have presented how UDDI registries can be extended to store functionality ontologies, and how this semantics can be used to discover Web services. In this thesis we have used the mechanisms presented in [16] to store ontologies, and advertise Web services to the UDDI registries through the nodes of these ontologies.

2.7 IEEE 11073 Standards Family

The IEEE 11073 Standards Family[39] aims to enable functional and semantic ad-hoc interoperability of medical devices such as medical sensors. For this purpose, the IEEE 11073 proposes an Object-oriented modeling of function and application area, the "Domain Information Model" (DIM). Through the DIM it is possible to define and represent devices, functionalities, measurement data, calibrations, alert information and so on. On top of the DIM, it provides standardized codes for naming all information elements in the DIM such as medical devices and device systems, units of measurements through the "Nomenclature" and "Data Dictionary". IEEE 11073 assumes that all device vendors to adopt this DIM to represent sensor data to achieve interoperability. In our architecture the Web Services exposing the data to be received wireless medical sensor devices are conforming to IEEE 11073 domain information model.

However for the time being the vendors still using proprietary formats or different standards can not be ignored. Also the guideline representation languages such as GLIF, may use different Reference Information Models such as HL7 RIM. In our architecture for the semantic mapping of the messages coming from medical sensor devices to GLIF RIM, we exploit a tool developed that is capable of definition of mapping relationships and also execution of these mapping definitions in order to automatically handle the translations [64].

In addition to the DIM, the IEEE 11073 standards family also provides a Nomenclature and a Data Dictionary to present standard classification of the medical devices. This Nomenclature is used in this thesis to create a Sensor Service Functionality Ontology to annotate the Sensor Web Services exposing sensor data.

2.8 JADE: Java Agent Development Environment

JADE (Java Agent Development Framework) is a software framework to aid the development of agent applications in compliance with the FIPA specifications for interoperable intelligent multi-agent systems. JADE is an Open Source project, and the complete system can be downloaded from JADE Home Page [45].

The main aim of JADE is to simplify development of agent-based systems and ensuring FIPA standard compliance [26] through a set of system services and agents. This goal is achieved by offering the following features to the agent programmer [4]:

- FIPA-compliant agent platform including the main components of FIPA, namely, AMS (Agent Management System), DF (Directory Facilitator), ACC (Agent Communication Channel). These three agents are by default inserted into the run-time system to manage multi-agent system coordination, agent communication and discovery and agent name service.
- JADE offers a distributed agent platform. That is, the platform can be split on several hosts. At each host only one Java Virtual Machine (JVM) is executed and agents are created as threads on each host and JVM to increase system performance. Java events are used for agent-to-agent communication on the same host and Java RMI across hosts. An agent may divide its tasks for parallel execution and JADE schedules these tasks more efficiently than JVM does for threads.
- More than one FIPA-compliant DFs (Directory Facilitator) can be started in order to implement multi-domain applications logically divided into domains.
- A programming interface to simplify registration of agent services with one, or more, domains (i.e. DF);
- Automatic conversion of messages to/from FIPA-compliant string format from/to encoded transferable encoded Java objects.
- A library of FIPA agent interaction protocols ready to be used by agents.
- A transport mechanism and interface to send/receive messages to/from other agents.
- A FIPA-compliant MTPs (Message Transport Protocol) to connect different agent platforms.

- A light-weight transport of Agent Communication Language (ACL) [27] messages inside the same agent platform, since messages are transferred encoded as Java objects, rather than strings, in order to avoid marshalling and unmarshalling procedures. When sender or receiver do not belong to the same platform, the message is automatically converted to/from the FIPA compliant string format. In this way, this conversion is hidden to the agent implementers that only need to deal with the same class of Java object.
- Automatic registration of agents with the AMS.
- A FIPA-compliant naming service: at start-up agents obtain their GUID (Globally Unique Agent Identifier) from the platform.
- A graphical user interface (GUI) is provided by JADE agent platform to remotely manage the life-cycle of agents and agent containers, as well as monitoring and control of agents. A user may stop, restart, create and start agents provided that an agent container is already running.
- Graphical tools for debugging the multi-agent system, like the dummy agent to send/ receive ACL messages and the sniffer agent to monitor the on-going communication between several agents. The Sniffer Agent, as the name implies, allows to track messages exchanged in a JADE agent platform. When the user decides to sniff an agent, or a group of agents, every message directed to or coming from that agent is tracked and displayed in the sniffer window. The user can view, save, and load, every message track for later analysis.

The JADE system can be described from two different points of view. On the one hand, JADE is a middleware for FIPA-compliant Multi Agent Systems, supporting application agents whenever they need to exploit some feature covered by the FIPA standard specification (message passing, agent life-cycle management, etc.). On the other hand, JADE is a Java framework for developing FIPA-compliant agent applications, making FIPA standard assets available to the programmer through object oriented abstractions.

In this thesis, JADE Agent development platform is used to develop the Multi Agent System that is responsible for deploying and executing clinical guidelines.

CHAPTER 3

THE SEMANTICALLY ENRICHED GUIDELINE MODEL: THE MEDICAL KNOWLEDGE LAYER

Computer interpretable models of clinical guidelines such as GLIF [29], ASBRU [84], PROforma [90], ARDEN [71] and EON [93] have been proposed to increase the understandability and sharability of clinical guidelines by medical practitioners. Several different guideline execution engines have been built processing these models, such as GLEE [100], GLARE [91], NewGuide [11] and DeGel [85] demonstrating that the guideline definitions can be executed to automate the decision making process.

When it comes to deploy and execute the guidelines in real life settings, it is a well accepted fact that, the current formalism to represent guidelines fall short in enabling the share of *computable* guidelines [76]. On the other hand the success of a clinical guideline execution system widely depends on how well it is integrated with the clinical workflow running in healthcare institutes [25, 86]. The complexity of fully integrated decision support systems is usually based on the nature of heterogeneous set of clinical applications need to be involved in the decision process" [22], and the lack of de-facto standards in accessing Electronic Healt-care Records and clinical applications served by clinical workflows. As discussed in Chapter 7, available guideline representation formalisms assume local reference information models to represent clinical data, and either not attacked to formally defining the interfaces to clinical workflows, or proposed some fixed event types to communicate with clinical workflows. As a result, in the deployment phase, custom manual adaptations becomes necessary: the events supported by the underlying clinical workflows should be manually bound to these events by patches to the clinical workflow engines and usually re-coding of the clinical data

represented in guideline definition is required to match the reference information model used by the clinical workflow.

```
Guideline \equiv (GuidelineModelEntity \cap (\exists algorithm.Algorithm))
Algorithm \equiv (GuidelineModelEntity \cap (\exists steps.GuidelineStep))
PatientStateStep \subseteq (GuidelineStep)
BranchStep \subseteq (GuidelineStep)
SynchronizationStep \subseteq (GuidelineStep)
DecisionStep \subseteq (GuidelineStep)
ActionStep \equiv (GuidelineStep \cap (\exists tasks.ActionSpecification))
ProgrammingOrientedActionSpecification \subset (ActionSpecification)
GetDataAction \equiv (ProgrammingOrientedActionSpecification \cap (\forall dataSourceType.({User}) \cup
{EMR})))
MedicallyOrientedAction \equiv (ActionSpecification \cap (\exists medicalTask.LiteralDataItem))
MessageAction \equiv (ProgrammingOrientedActionSpecification \cap (\exists message.xsd:String))
SubGuidelineAction \subseteq (ProgrammingOrientedActionSpecification)
Concept \equiv (KnowledgeItem \cap (\exists conceptID.xsd:String) \cap (\exists conceptName.xsd:String)
\cap (\exists conceptSource.xsd:String))
DataItem \equiv (BasicDataType \cap (\exists dataValue.DataItemValue))
PatientData \equiv (DataItemValue \cap (\exists serviceCD.Concept))
Observation \subseteq (PatientData)
Medication \subseteq (PatientData)
Procedure \subseteq (PatientData)
```

Figure 3.1: A Part of the Formal GLIF Model

To address this problem, we aim to extend one of the computer interpretable guideline representation formalisms, namely GLIF, so that a not only human-sharable, but also a semantically enriched machine processable clinical guideline representation is achieved that results in a deployment driven computable guideline representation formalism. The essence of extending GLIF is to provide a machine processable mechanism that can express the structure and the semantics of its references to clinical workflows and Electronic Healthcare Records so that automated processes can be used to access the clinical resources for guideline deployment and execution. This lack of support in the guideline representation languages and guideline execution environments is also stated in the latest GLIF specifications [29] as follows: "There is a need for an implementable specification that can be incorporated into an institutional system where the actions specified must be mapped to institutional procedures and the patient data references must be mapped to the electronic medical records of the underlying system". GLIF identified the requirement for a "Medical Knowledge Layer", that will enable the definition of the following interfaces:

- Interfaces to clinical repositories to retrieve Electronic Healthcare Records of a patient,
- Interfaces for interacting with applications such as clinical workflows
- Interfaces for interacting with alert systems.

In this chapter, we describe how these interfaces can be semantically defined formally in guideline definition. The formal GLIF model is defined as an ontology for representing guidelines referring to medical ontologies for representing medical data and concepts [30] as shown in Figure 3.1. In the following sections we present our extensions to this formal model. The aim of this formal extension is to describe the data needed at each guideline deployment and execution step semantically in a machine processable way. Furthermore, since we want to deploy the clinical guidelines to real healthcare settings, we base the semantics of the proposed extensions to the standards in use in the healthcare IT domain. The formal model of our extensions are also presented in Appendix A in an integrated format.

3.1 Semantically Extending GLIF to Facilitate Accessing the Electronic Healthcare Record Systems

At various steps in executing a guideline, there is a need for a specific patient clinical data. For example, the guideline may need information on whether the patient has previously suffered from "bronchial spasm". To be able to retrieve this information, first various EHR systems serving patient clinical data must be discovered in a heterogeneous environment and from these EHR Systems, the specific clinical information, such as "bronchial spasm", must be extracted.

The original GLIF model uses a "GetDataAction" class to represent data to be retrieved from EHR Systems. However, since in its current form GLIF aims to create a sharable guideline definition, rather than a deployable and directly executable one, the *dataSourceType* property of this object is either the String "EMR" (Electronic Medical Records) or the String "User" as presented in Figure 3.1, i.e. it does not have a mechanism to represent how to access the underlying EHR system.

Since the clinical databases vary in structure, access methods and vocabulary used to represent data, accessing the Electronic Healthcare Records of the patients has become an important problem for guideline execution. In one of the guideline representation languages, Arden [71], the database queries to access patient records are encoded in ad hoc ways and enclosed in "curly braces" in Medical Logic Modules (MLMs). Since then, the necessity to adapt these queries to the needs of site specific implementations to access EHRs are cited as "curly braces problem".

In many other clinical guideline execution engines such as GLEE [100] which uses GLIF as guideline representation format, a guideline data model is used in the guideline definition for representing patient data and a centralized repository is assumed to be supported by the local medical institutes where the patient data is forwarded to be stored using the guideline data model. However converting patient data to the model supported by the guideline and storing it in a centralized repository are not practical.

In some other studies, the "Virtual Medical Record" (vMR) [50] concept is used such as in EON [93], ProDIGY [49] and SAGE [94]. "Virtual Medical Record" (vMR) concept provides a structured data model for representing information related to individual patients where the mapping of local EHR to vMR is facilitated. In some other guideline models such as PROFORMA [90], ASBRU [84] and GUIDE [11], relational tables are used to store the mappings between the data model entities and the "columns of the database tables" where the patient data is actually stored.

Instead of hard-coding the mappings between the data model entities in guideline definitions and the "columns of the database tables" of the clinical workflow systems, or assuming that there is a commonly accepted "Virtual Medical Record" standard and a central repository to collect the patient data in this format, we chose to represent the semantics of the clinical data needed by the guideline, and in the deployment and execution phase, use this machine processable semantics to access the EHR of the patient wherever it is stored based on industry accepted standard mechanisms.

In order to create a sound and interoperable "Medical Knowledge Layer" to define the semantics of the patient data in order to support the deployment and execution of guidelines, it is necessary to follow the current standardization trends in exchanging Electronic Healthcare Records. As mentioned in Section 2.2, the most prominent EHR standards are the Health Level 7 (HL7) Clinical Document Architecture (CDA) [36] and the European Committee for Standardization (CEN) EN 13606-1 EHRcom [10]. Investigating these standards reveals that to locate the EHR document and to extract the piece of information from the EHR document, the semantics needs to be explicated at two levels: the EHR document semantics to discover the document and the entry semantics to extract the clinical statement requested by the guideline.

In HL7 CDA documents, the type of the EHR document is presented in the document heading in the *code* attribute such as "Discharge Summary Note". In the document itself the clinical statements represented as "Entries" that are grouped under "Sections", such as "Past Medical History", "Medications" and "History of Present Illnesses". In order to locate an EHR document that contains the requested clinical statement, both the document type semantics and the semantics of the relevant sections are needed.

Similarly in EHRCom [10], the "Compositions" represent a single EHR document, and the semantics of "Composition" presents the document type such as "Consultation Note". The "Sections" in EHRCom corresponds to the "Sections" in CDA, that is, the clinical statements are grouped under "Sections" such as "Adverse Reactions" and "Family Clinical History" based on their semantics. As in the CDA, in the EHRCom the "Composition" semantics together with the "Section" semantics represent the EHR semantics. The "Entry" semantics, on the other hand, is readily provided by both of these standards.

In order to embed this semantics into GLIF "GetDataAction" class, we have created an "EHREntity" class as a subclass of our "Medical Knowledge Layer". Then the range of the *dataSourceType* property is made to point to a newly defined "EHREntity" class as shown in Figure 3.2. Then the following properties are defined for the "EHREntity" class:

• *ehrSemantics* property is used for annotating the semantics of an EHR document with nodes from clinical terminologies or ontologies. As the range of this property, the "Concept" class (Figure 3.1) defined in GLIF is used. In this way, it becomes possible to specify the EHR document and section semantics with formally defined terms from terminology systems. For example, the Document Type Codes of LOINC [58] can be used to specify this semantics with the following instance of the "Concept" class: <Past Medical History, 11348-0, LOINC>. This property can have multiple values, that is, in an "EHREntity" instance definition, the document type semantics and the semantics of the sections it contains can be specified through multiple instances of *ehrSemantics* property as presented in Figure 3.2.

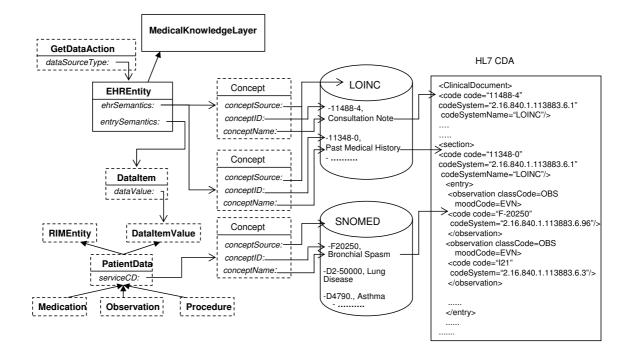


Figure 3.2: Extending GLIF with EHR Semantics

• The entrySemantics property is introduced for annotating the EHR entry semantics. The range of this property is the "DataItem" class as defined in GLIF (formally presented in Figure 3.1). The dataValue property of "DataItem" class in GLIF is specialized to the "PatientData" class which is further specialized to "Observation", "Medication", and "Procedure" classes. "PatientData" class has a property called service_cd inherited from HL7 RIM, whose range is the "Concept" class as shown in Figure 3.1. Through the instances of the "Concept" class, we represent the semantics of EHR entries, such as <BronchialSpasm, F-20250, SNOMED> to indicate that the guideline needs to know whether the patient previously experienced "Bronchial Spasm" or not.

A formal description of these extensions is given in Figure 3.3. An example instance of this description is depicted in Figure 3.4. Here, as the range of the *dataSourceType* property of the "getAsprinMedicationStatus : GetDataAction" instance, the "EHREntity" instance "medicationStatusMedicalDataEntity" is selected. For annotating the semantics of the EHR document needed, a "Concept" instance, "medicationUseHistoryConcept" is created and selected as the range of the *ehrSemantics* property of the "medicationStatusMedicalDataEntity" class instance. In the definition of the "medicationUseHistoryConcept" the UMLS

GetDataAction \equiv (ProgrammingOrientedActionSpecification \cap (\exists dataSourceType.EHREntity)) EHREntity \equiv (MedicalKnowledgeLayer \cap (\exists ehrSemantics.Concept) \cap (\exists entrySemantics.DataItem))

Figure 3.3: The Formal Description of GetDataAction and EHREntity in the Extended GLIF Model

concept "C0489536" is used [96]. Note that the code "C0489536" in UMLS indicates "History of Medication Use". The "aspirinMedicationDataItem" instance defines the entry semantics of the "medicationStatusMedicalDataEntity". For this purpose the "aspirinMedication" instance is specified as the range of the *dataValue* property of "aspirinMedicationDataItem". Finally the "aspirinConcept" instance pointing to a node in UMLS semantic network is used to specify the *serviceCD* semantics of the "aspirinMedication" instance.

getAspirinMedicationStatus : GetDataAction <getAspirinMedicationStatus, medicationStatusMedicalDataEntity> : dataSourceType medicationStatusMedicalDataEntity : EHREntity <medicationStatusMedicalDataEntity, medicationUseHistoryConcept> : ehrSemantics <medicationUseHistoryConcept,C0489536>: conceptID <medicationUseHistoryConcept,UMLS>: conceptSource <medicationUseHistoryConcept,History of Medication Use> : conceptName <medicationStatusMedicalDataEntity, dischargeSummaryConcept> : ehrSemantics <dischargeSummaryConcept,C0743221>: conceptID <dischargeSummaryConcept,UMLS> : conceptSource <dischargeSummaryConcept, DischargeSummary> : conceptName <medicationStatusMedicalDataEntity, aspirinMedicationDataItem> : entrySemantics < a spirinMedicationDataItem, a sprinMedication>: dataValueaspirinMedication : Medication <aspirinMedication,aspirinConcept>: serviceCD <aspirinConcept,C0004057>: conceptID <aspirinConcept,UMLS>: conceptSource

<aspirinConcept,aspirin> : conceptName

Figure 3.4: An example GetDataAction Instance in the Extended GLIF Model

It is clear that through these two levels of semantics, the data needed at each guideline execution step is described in a machine processable way.

3.2 Semantically Extending GLIF to Enhance Communication with Clinical Workflows

GLIF uses the "MedicallyOrientedAction" class to specify the medical actions or services provided by the underlying clinical workflow, for example, for prescribing medicine, giving lab orders or making referrals. Currently these tasks are specified with the *medical_task* property whose range is the *LiteralDataItem* class as presented in Figure 3.1. As presented in the example in Section 2.1, the *LiteralDataItem* is not capable of describing the semantics of a specific medical action.

The clinical workflow applications are complex and usually diverse software systems, therefore adoption of a single client software solution is not feasible to interact with them. In order to be able to define an implementable guideline specification, it is necessary to propose a solution that can address this technical interoperability problem to be able to interact with diverse clinical workflows. Web service technology is already being used by the Healthcare Industry as a solution to technical interoperability problem. The Dutch national infrastructure for healthcare messaging is implemented by wrapping HL7v3 messages as Web services [63]; Health Level Seven (HL7) has announced the approval of a Web Services Profile as Draft Standards for Trial Use (DSTUs) [35]. In another study, the integration of different medical systems in a configurable network of interconnected organizations in Canada is achieved by wrapping existing systems using Web services to provide a uniform and adaptable interface among the individual systems [65].

When the medical information systems expose their medical applications as Web services for interacting with the clinical workflows such as placing lab orders, the endpoints of these Web services (references to their Web Service Description Language (WSDL) definitions) can be used in clinical guideline definitions. However, in a generic guideline definition that has not been specialized for a specific patient and a healthcare institute, it is not feasible to directly refer to the end point of a Web service. Therefore we propose to specify the semantic definitions of the Web services in the guideline definition rather than the references to their concrete implementations. This also enables us to find alternative resources when exceptions are raised in accessing the specified Web Service in the execution phase.

To be able to define the semantics of the medical action to be interacted during the guide-

MedicallyOrientedAction \equiv (ActionSpecification \cap (\exists medicalTask.MedicalActionEntity)) MedicalActionEntity \equiv (MedicalKnowledgeLayer \cap (\exists functionality.Concept) \cap (\exists input.DataItem) \cap (\exists output.DataItem))

Figure 3.5: The Formal Description of MedicallyOrientedAction and MedicalActionEntity in the Extended GLIF Model

line execution, the GLIF model is extended as follows: First, the "MedicalActionEntity" class is created as a subclass of "MedicalKnowledgeLayer". The range of *medicalTask* property of the "MedicallyOrientedAction" class is set to be the "MedicalActionEntity" (Figure 3.5). The "MedicalActionEntity" class is used to specify the semantics of the guideline execution steps which correspond to clinical services.

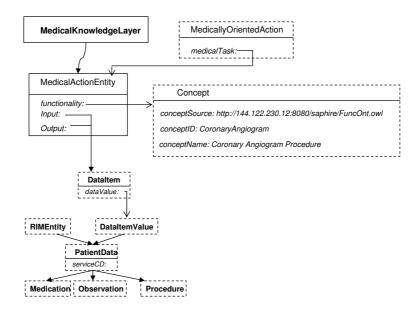


Figure 3.6: Annotating "MedicalActionEntity" class with the Service Functionality Ontology

To describe this semantics, we define a *functionality* property for the "MedicalActionEntity" class whose range is the "Concept" class defined in the GLIF Model. In this way, it becomes possible to annotate the clinical services with a node from a clinical terminology or ontology.

To be able to define the operational semantics of Healthcare services through this "func-

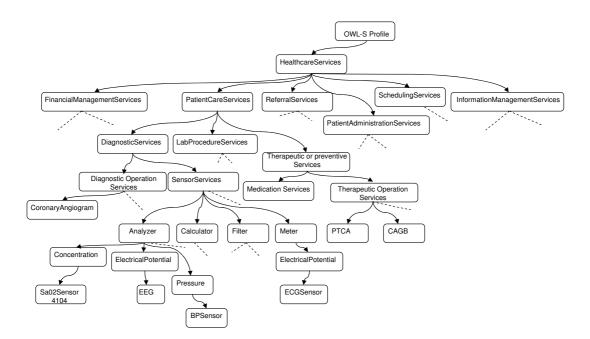


Figure 3.7: A Partial presentation of an example Service Functionality Ontology

tionality" property, we define a Service Functionality Ontology (SFO) based on the UMLS Semantic Network [96], HL7 event definitions [34], and IEEE 11073 Nomenclature [39] as presented in Figure 3.7. The HL7 standard groups the clinical events into the following clusters: Patient Administration, Order Entry, Query, Financial Management, Observation Reporting, Master Files, Medical Records/Information Management, Scheduling, Patient Referral, and Patient Care. On the other hand in UMLS Semantic Network, the Patient Care Services are categorized as: Diagnostic Services, Lab Procedure Services and Therapeutic or Preventive Services. We have used these major categorizations to create an ontology to be able to annotate the operational semantics of these services. It should be noted that the aim of this Service Functionality Ontology is to demonstrate if such a functionality ontology exist, how it can be exploited for annotation of clinical events to be able to deploy clinical guidelines semi-automatically. The subclasses of these major categories have been filled with sub-classes representing clinical services which are majorly used in clinical guideline-clinical workflow interaction, such as Medication Services, Therapeutic or Diagnostic Operation Services and so on. The Service Functionality Ontology is presented in Appendix B.

We use this Service Functionality Ontology to annotate the operational semantics of the clinical services. For example, Figure 3.6 shows how the "MedicalActionEntity" class can be annotated through a Service Functionality Ontology. The advantages of annotating the semantics of Medical Actions through ontologies rather than binding them to some predefined action types in the guideline model are as follows:

- The ontology is easily extensible allowing us to widen the scope of medical actions that the guideline needs to invoke in clinical workflows without the need to update the whole guideline model. For example, currently wireless medical sensor devices are widely used in clinical environments especially in homecare settings. During the execution of a guideline, there are decision points where the current vital signs of the patient are needed. For example, for a patient suffering from myocardial infarction with persistent ST-elevation, the decision on whether to start an "Oxygen Therapy" depends on the current value coming from the Pulse Oximeter sensor. In previous guideline definition formalisms the interaction with medical sensor devices to receive the vital signs of patients have not been covered. In our architecture the semantics of services exposing the data produced by wireless medical sensor devices are defined in the functionality ontology, under the node of "Diagnostic activities", where the "IEEE 11073 Nomenclature" are exploited to create a hierarchy of sensor device services. This capability moves the semantic support beyond what is currently available in guideline models; it becomes possible to easily extend the type of actions that the guideline deployment/execution engine can interact with.
- Medical Information systems can expose the services of the clinical workflows by annotating their semantics with any ontology of their choice. When the semantics of the actions in the guideline definition are also defined through ontologies, it becomes possible to use "ontology mapping/alignment" mechanisms for service matching. In the deployment phase, although different ontologies may be used to annotate the services of the clinical workflows and the actions in guideline definition, it becomes possible to locate the corresponding clinical services of the guideline actions through semantic mediation.

An example instance of Figure 3.5 is given in Figure 3.8 to show how a guideline execution step semantics can be expressed using this Service Functionality Ontology to express that the guideline execution engine needs an "Coronary Angiogram Procedure". This is expressed by selecting the <"Coronary Angiogram Procedure", CoronaryAngiogram, http://144.122.230. 12:8080/saphire/FuncOnt.owl> "Concept" class instance as the range of the *functionality* property.

In the example presented in Figure 3.9, we see how the IEEE 11073-10101 standard

orderAngiography : MedicallyOrientedAction <coronaryAngiography, coronaryAngiographyMedicalActionEntity> : medicalTask coronaryAngiographyMedicalActionEntity : MedicalActionEntity <coronaryAngiographyMedicalActionEntity, coronaryAngiogramConcept> : functionality <coronaryAngiogramConcept,CoronaryAngiogram> : conceptID <coronaryAngiogramConcept,Coronary Angiogram Procedure> : conceptName <coronaryAngiogramConcept,http://144.122.230.12:8080/saphire/FuncOnt.owl> : conceptSource

Figure 3.8: An example MedicallyOrientedAction Instance in the Extended GLIF Model

[39] nomenclature integrated to our Service Functionality Ontology can be used to denote that in some certain step of clinical guideline, the oxygen saturation of the patient should be retrieved from a sensor device. This semantics is expressed by selection the following "Concept" class instance: <Oxygen Saturation Sensor, SP02Monitor, http://144.122.230.
12:8080/saphire/FuncOnt.owl> as the range of the *functionality* property.

getOxygenSaturationStatus : MedicallyOrientedAction

< getOxygenSaturationStatus, getoxygenSaturationMedicalActionEntity>: dataSourceType

getoxygenSaturationMedicalActionEntity : MedicalActionEntity

<MedicalActionEntit, Sp02SensorConcept> : functionality

<Sp02SensorConcept,"Oxygen Saturation Sensor">: conceptName

<Sp02SensorConcept,SP02Monitor> : conceptID

<Sp02SensorConcept, http://144.122.230.12:8080/saphire/FuncOnt.owl>: conceptSource

Figure 3.9: An example MedicalActionEntity Instance for annotating interaction with Sensor Devices in the Extended GLIF Model

This semantics is used during the guideline deployment to describe the functionality of the service needed. Symmetrically, there is a need to expose the semantics of the underlying technical interoperability layer. How to map this layer to technical interoperability layer is detailed in Chapter 4.

On top of the functionality semantics of the clinical services, we need to define the semantics of the input and output messages of them. In guideline specification, the input and output parameter semantics are specified through GLIF RIM classes such as *Medication*, *Prescription* and *Observation* as presented in Figure 3.6.

3.3 Semantically Extending GLIF to Improve Communication with Alert and Reminder Systems

$MessageAction \equiv (ActionSpecification \cap (\exists message.AlarmEntity))$
$AlarmEntity \equiv (MedicalKnowledgeLayer \cap (\exists alarmMessage.xsd:String)$
\cap (\exists alarmUrgency.Concept) \cap (\exists roles.Concept)

Figure 3.10: The MessageAction and AlarmEntity in the Extended GLIF Model

As identified in the GLIF specification, another interface of the guideline definition with external applications is the alarm and reminder services. The basic information needed for delivering an alarm includes determining whom to send it, through which channels and also the priority of the alarm. At the conceptual level, there is a need to express this information with possible extensions as needed.

GLIF model uses the "MessageAction" class for sending messages to other entities. This "MessageAction" has an attribute named *message* whose range is the "String" class in the GLIF model as presented in Figure 3.1. Through this class only the content of the alarm message can be expressed.

To handle the requirements of message delivery including the ability to specify different roles and the different communication media, we extend the range of the *message* attribute to be the "AlarmEntity" class defined in the "Medical Knowledge" layer as presented in Figure 3.10 and Figure 3.11. Through the *roles* property of the "AlarmEntity" class, the healthcare professional roles to whom the message should be sent is specified in reference to a medical terminology. For example the <Cardiologist, J-0612B, SNOMED> "Concept" instance can be used to specify one of the roles in reference to SNOMED clinical terminology.

In addition to this, the urgency of the alarm is one of the factors that may affect the alert delivery mechanisms. The *alarmUrgency* property is also bound to a "Concept" class so that a node in an urgency level taxonomy can be referenced. Finally through the *message* property the alert message itself is represented. An example instance of Figure 3.10 is given

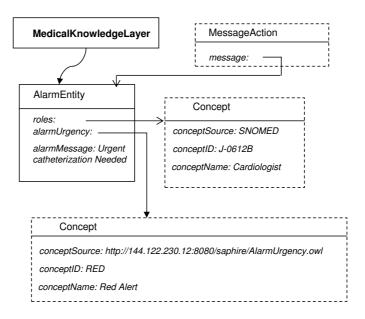


Figure 3.11: Annotating "AlarmEntity" class with semantics

in Figure 3.12 describing the message semantics in a machine processable way.

catheterizationRecommendation : MessageAction
$<\!\!{\rm catheterizationRecommendation,catheterizationAlarmEntity}\!>: \ {\rm message}$
catheterizationAlarmEntity : AlarmEntity
<catheterizationAlarmEntity, cardiologistConcept>: roles
<cardiologistConcept,J-0612B>: conceptID
<cardiologistConcept,SNOMED> : conceptSource
<cardiologistConcept,Cardilogist>: conceptName
<catheterizationAlarmEntity, redAlertConcept>: alarmUrgency
<redAlertConcept,RED> : conceptID
$<\!\!\mathrm{redAlertConcept, http://144.122.230.12:8080/saphire/AlarmUrgency.owl\!>: conceptSource}$
<redalertconcept,red alert=""> : conceptName</redalertconcept,red>
$<\!\!{\rm catheterizationAlarmEntity,} {\rm Urgent\ catheterization\ needed}\!>:\ alarmMessage$

Figure 3.12: An Example MessageAction Instance in the Extended GLIF Model

CHAPTER 4

THE ENABLING SEMANTIC ARCHITECTURE FOR THE DEPLOYMENT AND EXECUTION OF THE EXTENDED GLIF MODEL

Through the extended GLIF Model presented in Chapter 3, we are able express the detailed semantics of what the guideline execution engine expects from the underlying EHR systems, sensor systems or clinical workflows. To be able to deploy a guideline to a healthcare setting, the underlying applications' semantics also need to be available so that in the deployment phase the clinical applications that can address the needs of the clinical guideline's interfaces can be automatically discovered and bound to the specialized guideline definition to be used in the execution phase.

Our aim is to enable the semi-automatic deployment and execution of clinical guidelines in real life settings and this can be successful only if the implementation layer is based on open standards enabling interoperability [56]. Hence we based our deployment and execution architecture on widely accepted healthcare and IT standards supporting interoperability. There are two prominent technologies in use in the eHealth domain that provide implementation layer semantics: Integrating Healthcare Enterprise Cross Enterprise Document Sharing Integration Profile (IHE XDS) [40] for discovering and exchanging EHRs and the Web service technology for interacting with the clinical workflows and the sensor devices. IHE XDS exposes the semantics of clinical documents through document metadata and well-defined mechanisms exist for exposing Web service semantics such as OWL-S [69].

The implementation architecture based on these technologies consists of two layers: the deployment layer and the execution layer. When a semantically extended guideline is de-

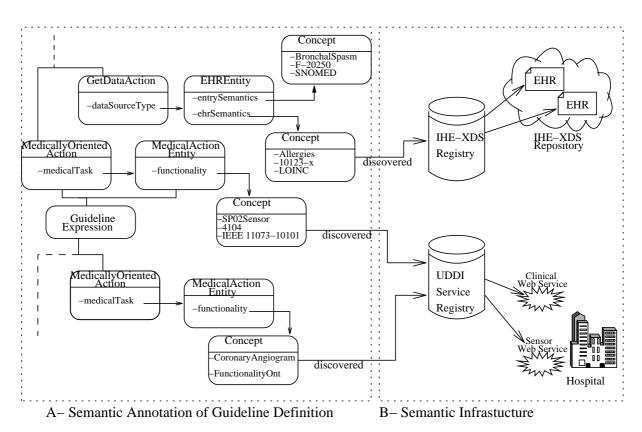


Figure 4.1: Semantic Annotation of Guidelines and the Complementing Infrastructure

ployed to a healthcare setting, the guideline execution steps are bound to the underlying applications by matching the semantics of both. In other words, the guideline instance parameters are initialized so that it becomes ready for execution when needed. Figure 4.1 depicts this process pictorially where Figure 4.1. A shows a part of a semantically extended guideline and Figure 4.1.B describes how the related EHR documents of the specific patient are located, the instances of the services are discovered.

The parameters initialized at the deployment phase, such as the EHR document references, the OWL-S [69] and WSDL (Web Service Description Language) [103] descriptions of the Web services are used at the execution level. To be able to store these parameters in a specialized clinical guideline definition after the deployment phase, we introduced a new layer in the extended GLIF model: "The Implementation Layer". As presented in Figure 4.2, the "Implementation Layer" sits on top of the "Medical Knowledge Layer" introduced in Chapter 3: new properties are also added to each of the *AlarmEntity*, *EHREntity*, and *MedicalActionEntity* classes to link the Entities introduced in "Medical Knowledge layer" to the classes introduced in "Implementation Layer". The formal model of the "Implementation Layer" is presented in Figure 4.3. In the following sections we provide details of the implementation

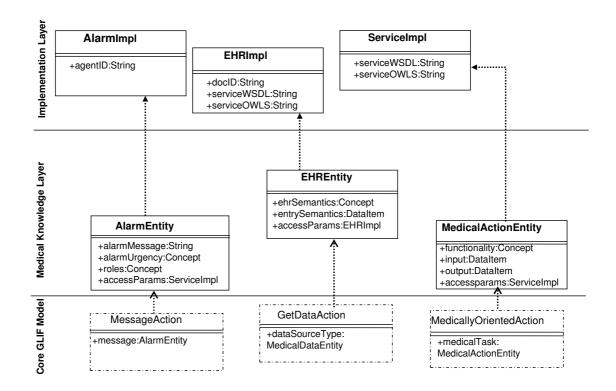


Figure 4.2: The Relationship between the Runtime Implementation Classes and the Medical Knowledge Layer Classes

architecture elaborating on how the "Implementation Layer" classes are initialized to be used in the execution phase.

$$\begin{split} & \text{EHREntity} \equiv (\text{MedicalKnowledgeLayer} \cap (\exists \text{ ehrSemantics.Concept}) \cap \\ & (\exists \text{ entrySemantics.DataItem}) \cap (\exists \text{ accessParams.EHRImpl})) \\ & \text{MedicalActionEntity} \equiv (\text{MedicalKnowledgeLayer} \cap (\exists \text{ functionality.Concept}) \cap \\ & (\exists \text{ input.DataItem}) \cap (\exists \text{ output.DataItem}) \cap (\exists \text{ accessParams.ServiceImpl})) \\ & \text{AlarmEntity} \equiv (\text{MedicalKnowledgeLayer} \cap (\exists \text{ alarmMessage.xsd:String}) \cap \\ & (\exists \text{ alarmUrgency.Concept}) \cap (\exists \text{ roles.Concept}) \cap (\exists \text{ accessParams.AlarmImpl})) \\ & \text{EHRImp} \equiv (\text{ImplementationLayer} \cap (\exists \text{ docID.String}) \cap (\exists \text{ serviceWSDL.String}) \cap \\ & (\exists \text{ serviceOWLS.String})) \\ & \text{ServiceImp} \equiv (\text{ImplementationLayer} \cap (\exists \text{ serviceWSDL.String}) \cap (\exists \text{ serviceOWLS.String})) \\ & \text{AlarmImp} \equiv (\text{ImplementationLayer} \cap (\exists \text{ agentID.String})) \\ \end{array}$$

Figure 4.3: The Implementation Layer Specification in the Extended GLIF Model

4.1 Addressing the Semantic Interoperability for Accessing EHRs

In our architecture, the healthcare institutes are organized as "Clinical Affinity Domains" to cooperate in the care of patients. For sharing electronic healthcare records of a patient, our architecture supports the following two widely adopted alternatives:

• The healthcare institutes may implement IHE Cross-Enterprise Document Sharing (XDS) profile [40] to share clinical records. IHE is an industry initiative, and XDS profile has been adopted for sharing EHRs in the National eHealth System blueprints of Canada, USA, Italy, Norway and France. In IHE XDS, the healthcare institutes store the medical documents of the patients to repositories of their choice and access these repositories through the document metadata stored in the associated registries. Although IHE XDS Profile is document content neutral, by recognizing the need to access the EHRs through their semantics, it has specified metadata mechanisms such as "document type codes" through standard terminologies. In other words, the semantics of EHRs are expressed through metadata defined through terminology systems such as LOINC [58] stored at the "DocumentRegistry"s so that the documents of interest can be discovered and retrieved by using this metadata. IHE XDS specifies well-defined transactions for these operations.

For example, as presented in Figure 4.4, when a medical institute stores an HL7 CDA document to an XDS Document Repository with ITI15- Provide & Register Document Set transaction [40], the Document Repository registers the CDA document to XDS Registry with its document type and section content semantics represented through LOINC Document Type codes: 11488-4 (Consultation Note) and 11348-0 (Past Medical History). ITI 14- Register Document Set transaction [40] is issued for this purpose. The extended GLIF Model is deployed on the IHE XDS layer as follows:

- The first step is to locate the relevant Electronic Healthcare Records of the patient. For this purpose the "ehrSemantics" property of the *EHREntity* class in the extended GLIF Model is processed. In the example given in Figure 3.4, the UMLS code "C0552487" is used to indicate that the metadata of the EHR is "PastMedicalHistory". To locate the relevant EHR document references, a "queryRegistry" transaction (ITI-16) is issued to IHE Document Registry. As a response, the list of document references is returned. These references are saved back to the guide-line definition as an instance of "EHRImpl" (Figure 4.2) whose formal definition

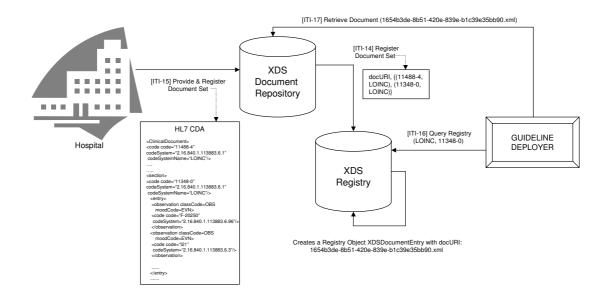


Figure 4.4: The Registration and Query of a HL7 CDA document to a XDS Registry/Repository Architecture

medicationEHRImp: EHRimp
$<\!\!\mathrm{medicationEHRImp},1654\mathrm{b}3\mathrm{de}\text{-}8\mathrm{b}51\text{-}420\mathrm{e}\text{-}839\mathrm{e}\text{-}b1\mathrm{c}39\mathrm{e}35\mathrm{b}\mathrm{b}90.\mathrm{xml}\!>:\mathrm{docID}$
$<\!\!{\rm medicationStatusMedicalDataEntity,\ medicationEHRImp}\!>:\ {\rm accessParams}$

Figure 4.5: An example EHRImpl Instance in the Extended GLIF Model

is given in Figure 4.3. EHRImpl is bound to the "EHREntity" as presented in Figure 4.5 through the "accessParams" property. In this way these documents can be retrieved from the Repository through the "retrieveDocument" (ITI-17) IHE transaction, while the guideline is executed.

- It may be the case that different clinical terminologies are used in the guideline definition, and in the XDS Registry to annotate the EHR document metadata. In this case, before the query transaction is issued, a translation request may be sent to a terminology server in order to map the terminology nodes to each other as described in Section 4.1.2.
- In our architecture, we do not expect every healthcare institute to implement the IHE-XDS Profile. Some healthcare institutes may choose to expose the available Electronic Healthcare Records of the patients through Web Services. In this case, in order to find

the EHR documents through their semantics, the functionality semantics of the Web services exposing them can be defined through LOINC document type codes [58]. For example, a Web service retrieving the "Past Medical History" of a given patient can be annotated with LOINC code "11348-0" to indicate this fact. In other words, since these Web services are used only passing the EHR Documents, it suffices to annotate their semantics with the semantics of the documents they carry.

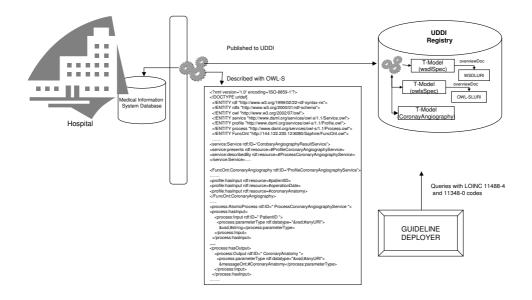


Figure 4.6: Annotation and Discovery of Web Services Exposing EHRs

In our architecture, HL7 CDA level three documents are used, hence the semantics at all the levels such as the document type, section and entry are annotated with the coded terms. In the CDA documents, the document metadata is already available in the machine processable CDA document header and sections in the "code" attribute. This metadata is extracted automatically and used as the metadata of the related Web services while publishing them to the UDDI registry [95] as presented in Figure 4.6.

When it comes to discover the EHR Web services in the deployment phase, similar to IHE-XDS case, the "ehrSemantics" property of the *EHREntity* class is extracted from the GLIF instance (Figure 3.4), a mapping is achieved if necessary and a Web Service Query transaction is issued to UDDI registry. As a result the WSDL and OWL-S references are saved back to the specialized GLIF instance such as <medicationStatusMedicalDataEntity, http://hospitalA.com:8080/services/ExEHRService?wsdl> : serviceWSDL.

In this way, in the deployment phase it is possible to locate the EHRs which are semantically referred in the guideline definition, by querying either the XDS or Web Service registries.

4.1.1 Addressing the Interoperability of Patient Identifiers

In the previous section, we have discussed how to discover the resources allowing us to access EHRs of the patients, once the semantics of the EHR documents are presented in extended guideline model. However, before using semantic mechanisms to locate patient EHRs, there is another important problem that needs to be addressed: Patient Identifiers. The use of patient identifiers is the accepted practice for locating patient EHRs. However, each organization may (and typically will) have a different patient identifier domain. Additionally, in the IHE XDS profile, the Document Registry also assigns a unique ID to the patient when a document of the patient is registered. To resolve the different patient identifiers assigned to a patient, we use IHE "Patient Identity Cross-referencing" (PIX) Profile [40]. PIX Managers facilitate the mappings between the local patient identifiers and the identifiers used in the DocumentRegistry.

To facilitate the identifier mapping process, the hospitals send a "Patient Identifier Feed" message to the PIX Manager whenever a new patient is admitted to the hospital (Figure 4.7). The PIX manager applies a probabilistic matching algorithm to check whether the new patient demographics information matches to one of the already existing patients. In this way, given a local patient identifier number in one domain, the PIX Manager can match it to a local patient identifier in another domain. Details of probabilistic record linkage algorithms are available in [7, 47]. In our architecture, there are also Cross Affinity Domain PIX managers, which facilitate the mapping of unique Patient Identifiers used in each clinical affinity domain to one another.

4.1.2 Addressing the Interoperability of EHR Content

After the required EHR is located through the XDS Registries, or through the Web Services exposing EHRs, in order to execute the clinical guidelines, there is a need to extract a specific clinical statement from the EHR of the patient. For example, the guideline may need information on whether the patient has experienced any "corticoadrenal insufficiency"

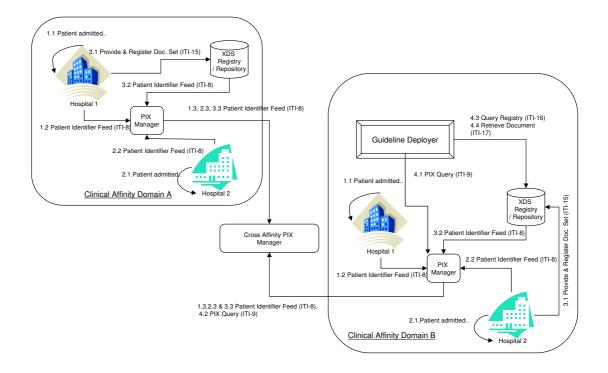


Figure 4.7: Patient Identifier Cross Referencing Architecture

previously. From the retrieved EHR, this specific clinical information must be extracted. It is clear that to achieve this, the EHR must be available in a machine processable content standard such as CEN EHRcom [10] or HL7 CDA [36]. In addition to this, since different EHR document formats and coding schemes can be used to represent the EHR content, there should be a semantic mediation mechanism available to use the data retrieved from the documents.

In the following, we describe how clinical statement data can be extracted from a machine processable EHR by using semantic mechanisms proposed in this work. First we give an insight to the problem through an example based on HL7 CDA [36].

In Figure 4.8, on the left hand side, a part of a sample HL7 CDA document is given. In this document, the semantics of a section is annotated with the "11348-0" code from LOINC to describe that this section is about the "Past Medical History". Additionally each entry is an instance of HL7 RIM classes such as the "Observation" class, and their content is also annotated with coded terms. For example, in one of the "Observation" class instances, the MeSH code "D000224" is used to indicate that the patient has suffered from "Addison Disease".

On the other hand, as presented in Figure 4.8 in the extended model through the

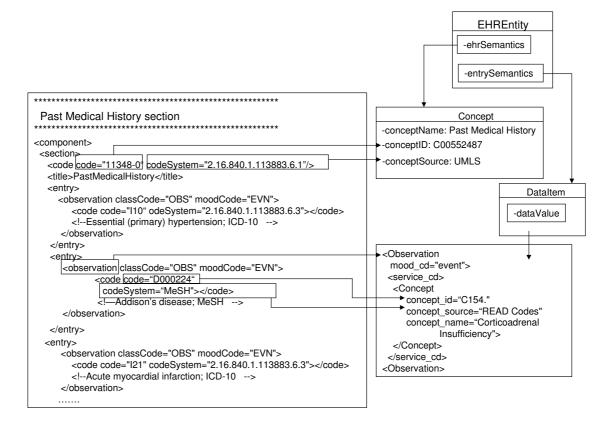


Figure 4.8: An Example of Extracting GLIF Patient Entity from the HL7 CDA

"EHREntity" class, the type of the data to be extracted from the EHR document is specified through the following mechanisms:

- First the semantics of the "Section" to be discovered is specified in the "ehrSemantics" attribute of the "EHREntity". As an example, <Past Medical History, C0552487, UMLS> expresses such a meaning.
- The "entrySemantics" attribute of the "EHREntity" specifies the exact semantics of the data needed as an instance of a *DataItem* class of GLIF. The range of the *dataValue* property of the "DataItem" class, specifies the semantics of the clinical statements in terms of the "PatientData" (Observation, Medication or Procedure) instances defined in GLIF RIM [29]. In terms of granularity, the data specified in these "DataItemValue" instances corresponds to one of the entries in the CDA documents. For example, on the right hand side of Figure 4.8, there is an example "DataItemValue" instance, representing the "Corticoadrenal Insufficiency" observation. Here, the meaning is given through the "GLIF RIM Observation class", and the "Concept" that are bound to this

observation. In the "Concept" class, the semantics of the data is specified with the code of "CorticoAdrenal Insufficiency" in READ Codes[78]. In this way, it is possible to represent the semantics of the clinical data required by the guideline execution formally in a machine processable manner.

```
if(glifCodingScheme! = DocCodingScheme)
```

then

```
target EHRS emantic Category = map (glif EHRS emantic Category, glif Coding Scheme, respectively) = map (glif EHRS emantic Category, glif Coding Scheme, respectively) = map (glif EHRS emantic Category, glif Coding Scheme, respectively) = map (glif EHRS emantic Category, glif Coding Scheme, respectively) = map (glif EHRS emantic Category, glif Coding Scheme, respectively) = map (glif EHRS emantic Category, glif Coding Scheme, respectively) = map (glif EHRS emantic Category, glif Coding Scheme, respectively) = map (glif EHRS emantic Category, glif Coding Scheme, respectively) = map (glif EHRS emantic Category, glif Coding Scheme, respectively) = map (glif EHRS emantic Category, glif Coding Scheme, respectively) = map (glif EHRS emantic Category, glif Coding Scheme, respectively) = map (glif EHRS emantic Category, glif Coding Scheme, respectively) = map (glif EHRS emantic Category, glif Coding Scheme, respectively) = map (glif EHRS emantic Category, glif Coding Scheme, respectively) = map (glif EHRS emantic Category, glif Coding Scheme, respectively) = map (glif EHRS emantic Category, glif Coding Scheme, respectively) = map (glif EHRS emantic Category, glif Coding Scheme, respectively) = map (glif EHRS emantic Category, glif Coding Scheme, respectively) = map (glif EHRS emantic Category, glif Coding Scheme, respectively) = map (glif EHRS emantic Category) = map (glif EHRS emantic Category) = map (glif EHRS emantic Category) = map (glif EHRS emantic Category) = map (glif EHRS emantic Category) = map (glif EHRS emantic Category) = map (glif EHRS emantic Category) = map (glif EHRS emantic Category) = map (glif EHRS emantic Category) = map (glif EHRS emantic Category) = map (glif EHRS emantic Category) = map (glif EHRS emantic Category) = map (glif EHRS emantic Category) = map (glif EHRS emantic Category) = map (glif EHRS emantic Category) = map (glif EHRS emantic Category) = map (glif EHRS emantic Category) = map (glif EHRS emantic Category) = map (glif EHRS emantic Category) = map (glif EHRS emantic Category) = ma
```

DocCodingScheme);

else targetEHRSemanticCategory=glifEHRSemanticCategory;

Section targetSection = sectionImport(EHRDoc,targetEHRSemanticCategory); Entry targetEntry = discreteDataImport(targetSection,glifEntrySemanticCategory, glifCodingScheme);

function discreteDataImport(targetSection,glifEntrySemanticCategory,glifCodingScheme) forall Entry e in targetSection

 $\{entryCodingScheme=(e.getCode()).getCodingScheme();$

```
if(entryCodingScheme! = glifCodingScheme)
```

then

targetSemanticCategory = map(entrySemanticCategory, entryCodingScheme, the semanticCategory) = map(entrySemanticCategory, entryCodingScheme, the semanticCategory) = map(entrySemanticCategory, entryCodingScheme, the semanticCategory) = map(entrySemanticCategory, entryCodingScheme, the semanticCategory) = map(entrySemanticCategory, entryCodingScheme, the semanticCategory) = map(entrySemanticCategory, entryCodingScheme, the semanticCategory) = map(entrySemanticCategory, entryCodingScheme, the semanticCategory) = map(entrySemanticCategory, entryCodingScheme, the semanticCategory) = map(entrySemanticCategory, entryCodingScheme, the semanticCategory) = map(entrySemanticCategory, entryCodingScheme, the semanticCategory) = map(entrySemanticCategory) cheme);

```
else\ {\tt targetSemanticCategory} = {\tt entrySemanticCategory};
```

```
if(targetSemanticCategory == glifEntrySemanticCategory)
```

then return e;

}

Figure 4.9: The Discrete Data Import Algorithm

The algorithm used in extracting EHR entry semantics is presented in Figure 4.9.

First, it is necessary to check whether the coding schemes for annotating the Sections (through the ehrSemantics attribute) used in GLIF specification match the coding schemes used in the CDA Document. If not, the coded terms used to annotate the section contents should be mapped to each other so that the relevant section in the CDA document can be imported. In our example, in the CDA document, the section is annotated with the LOINC

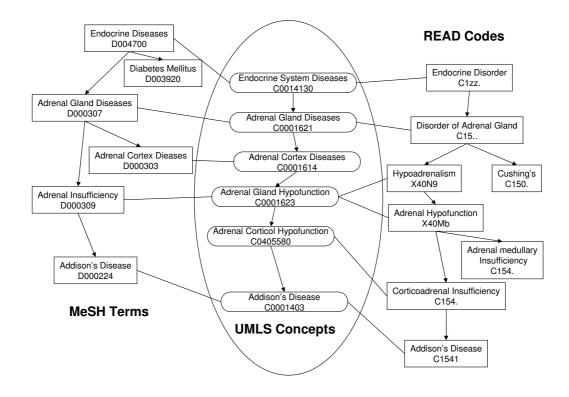


Figure 4.10: Mapping of terms from different Terminology System through UMLS Concept Hierachy

code "11348-0" to represent that it is about "Past Medical History". However in the guideline definition the same concept is represented with the UMLS code "C0552487" to specify that the Guideline needs the "Past Medical History" of the patient.

Once we have mapped the term used in guideline definition to the the term used to identify the section in the CDA Document, this particular section is imported by parsing the CDA document. For retrieving the particular clinical statement mapped the term used in guideline definition to the guideline specification, a similar approach is followed as outlined in the "discreteDataImport" function in Figure 4.9. The coded terms used to annotate the entries in the CDA document, and the entry semantics in guideline definition may be different. For example, in the example CDA document presented in Figure 4.8, the entry is annotated with a MesH code while a READ code is used to annotate the entry semantics in the extended guideline definition.

To find the correspondences among different coding schemes, it is possible to use a terminology server. A medical terminology server serves a controlled vocabulary for medical information systems by modeling medical concepts in a descriptive manner. Among other functionalities, medical terminology servers handles the mapping of these medical concepts to Medical classification schemes; through this facility they also provide translation facilities between the terms of different classification schemes.

GALEN [31] is one of the first implementations of such a medical terminology server which defines medical concepts in descriptive logics supporting subsumption. Another such terminology server is UMLS: the UMLS Knowledge Source Server Metathesaurus [96] comprises over one million biomedical concepts and five million concept names, all of which are from over a hundred controlled vocabularies and classification systems such as ICD-9, SNOMED and LOINC, providing a mapping structure between them. In our architecture a mapping functionality supporting subsumption is used based on the concepts and mapping provided by UMLS.

In Figure 4.10, a part of the UMLS Concept hierarchy and mappings of these concepts to different terminology systems is presented as defined by UMLS Knowledge Metathesaurus. Once such mapping definitions are stored in an ontology server supporting subsumption, it becomes possible to deduce implicit relationships and mapping between the terms of different coding schemes. Our mapping service queries the Ontology Server bootstrapped with the UMLS concepts and term mappings to different terminology systems, in order to discover the respective equivalent terms of the coded terms used in the extended guideline definition in several other medical terminology systems.

As an example, in Figure 4.8, the READ Code "C154." is specified as the "entrySemantics" in the GLIF definition, to be able to check whether the patient has previously suffered from "Corticoadrenal Insufficiency". However in the CDA document in Figure 4.8, the MeSH codes are used to represent this semantics.

As seen in Figure 4.10, there is a direct mapping between the term "C154." and UMLS Concept "Adrenal Corticol Hypofunction, C0405580", however this term does not have a direct correspondent in MeSH codes. On the other hand, from the UMLS Concept hierarchy we see that the concept "Addison's Disease, C0001403" is a subclass of the UMLS Concept "Adrenal Corticol Hypofunction, C0405580", and the concept "Addison's Disease, C0001403" has a direct mapping to the "D000224" term in MeSH. Using the subsumption relationship between these terms, the mapping service deduces that the term "Addison's Diseases, D000224" is in fact "IS-A" "Corticoadrenal Insufficiency, C154.". Hence a match in the CDA document can be found and used in the guideline execution.

In this way, using the semantics defined in the extended GLIF model, the related entry can be retrieved seamlessly from the CDA document, if such a clinical statement exists.

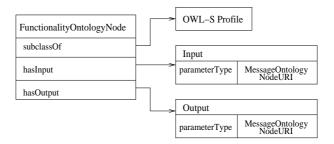


Figure 4.11: Relating Service Functionality and Service Message Ontologies with OWL-S

4.2 Addressing the Interoperability in Interacting with the Clinical Workflows

Considering that there could be several different applications running on different platforms that need to be invoked while the clinical guideline is being executed, in our architecture Web services are used to address the technical interoperability problem. In order to facilitate the discovery of these Web services, first the OWL-S descriptions of the Web Services are defined. As also presented in Figure 3.7, the Service Functionality Ontology is expressed through the "ServiceProfile" class of OWL-S [69]. For this purpose, as presented in Figure 4.11, the top-most class of our Service Functionality ontology is created as a subclass of OWL-S ServiceProfile class. OWL-S Release 1.1 also indicates that service characterization must effectively position a service within the broad array of services that exists within some domain, or perhaps in the world at large [74]. OWL-S proposes to construction of a "Service Profile hierarchy", with inheritance of properties by subclasses as a technique for this kind of service characterization. In the same manner, we have created our Service Functionality Ontology (Figure 3.7) as a "Profile Hierarchy".

To express the service functionality semantics of Web services, the ServiceProfile instance of the specific Web service is represented as an instance of one of the Functionality ontology nodes as presented in Figure 4.12. Similarly in order to be able to annotate the Web services exposing wireless sensor data, the nodes of our Functionality Ontology created based on IEEE 11073 standard nomenclature are exploited. In this way the semantic annotation of Web Services through one of the Functionality ontology nodes is achieved.

After the OWL-S service descriptions of the Web services are created they are published to UDDI registries through their semantics. For this purpose we have used the methodologies we have previously proposed in [16] for storing Functionality Ontologies to UDDI, and annotating the semantics of Web Services through this functionality ontology. The following

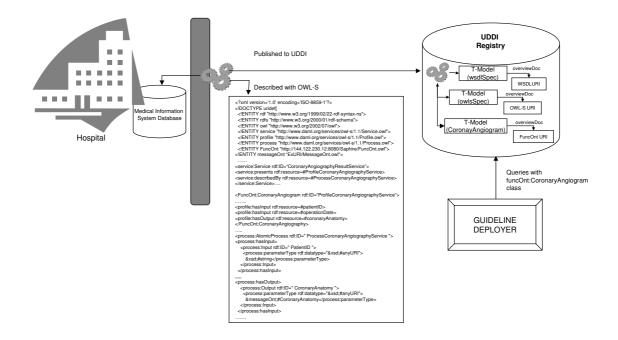


Figure 4.12: Publishing Medical Web Services with their Semantics to UDDI

steps are followed as presented in Figure 4.12:

- First of all, the functionality ontology is parsed and a TModelKey is created for each of these functionality ontology nodes. TModels provide the ability to describe compliance with a specification, a concept, or a shared design. When a particular specification is registered with the UDDI as a tModel, it is assigned a unique key, which is then used in the description of service instances to indicate compliance with the specification. The specification is not included in the tModel itself. The "OverviewDoc" and "OverviewURL" elements of tModels are used to point at the actual source of a specification.
- Two more TModels are created for representing the WSDL and OWL-S Specifications of Web services.
- When a Web Service is to be published to the UDDI Registry, first of all its WSDL and OWL-S file URIs are bound to the service by adding the Tmodels with "wsdlSpec" and "owlsSpec" keys and WSDL and OWL-S URIs as OverviewURLs to the category bag of the Service Definition.
- For annotating the functionality semantics, the OWL-S file is parsed and the TModel

${\rm coronaryAngiographyMedicalActionImp:\ ServiceImpl}$
$<\!\!\mathrm{coronaryAngiographyMedicalActionImp,http://81.180.27.130:8080/axis/services/}$
eq:coronaryAngiography?WSDL>: serviceWSDL
$<\!\!\mathrm{coronaryAngiographyMedicalActionImp,http://81.180.27.130:8080/serviceOWLS/}$
CoronaryAngiography.owl>: serviceOWLS
$<\!\!\mathrm{coronaryAngiographyMedicalDataEntity},\!\mathrm{coronaryAngiographyMedicalActionImp}\!>:\ \mathrm{accessParams}$

Figure 4.13: An example ServiceImpl Instance in the Extended GLIF Model

previously created for the Service Functionality Ontology node in Service Profile Hierarchy is found and this Tmodel is added to the category bag of the Service.

In the deployment phase of a generic guideline definition for a specific patient and an institute, the range of the functionality property of a *MedicalActionEntity* in the guideline instance (Figure 3.8) is used in querying the UDDI registry. The discovered service instances WSDL and OWL-S references are saved to the executable guideline instance as a property of a "ServiceImpl" (Figure 4.2) instance as presented in Figure 4.13.

4.2.1 Addressing the Interoperability of Exchanged Messages used in the Clinical Web Services

Service functionality semantics enables us to discover the Web services based on their semantics. However, in order to invoke the discovered Web services while the guideline is executing, message level interoperability is also needed. Service functionality semantics may suffice to achieve interoperability only when all the Web services and guideline execution engines use the same message standards, and same Reference Information Models. However, it is not realistic to assume that all the healthcare organizations comply with the same message structure and content. Hence, there is a need to transform one message content into another.

In order to facilitate message transformation, our architecture utilizes ontology mapping. The OWLmt tool [6] is used for mapping the input and output parameters of Web services to the instances of the reference information model used in GLIF specification.

OWLmt is an Web Ontology Language (OWL) [66] based ontology mapping tool to handle ontology mediation by mapping the OWL ontologies in different structures and with an overlapping content to each other. It aims to define a document called the "Mapping Definition" describing how the source ontology and the target ontology classes and properties relate. This document includes the units of information called the "Mapping Patterns", which are the matchings between the source ontology and the target ontology classes and properties. The "Mapping Definition" is then used to transform the source ontology instances to the target ontology instances automatically.

The semantic mediation in our architecture is enabled through the following steps:

- Service message ontologies are created to express the semantics of the Clinical Web service messages. In the healthcare domain, the Web services usually exchange XML messages. Through a normalization tool [32], we create the OWL ontology of the Web service messages from the XML Schema (XSD) [105] definitions of the service messages. It is clear that when an OWL ontology is created from an XML Schema (XSD), it is not possible to extract some of the OWL specific semantics such as class expressions or various types of properties. Yet, it is still possible to obtain the class hierarchies and the properties of classes and this information proves useful in ontology mapping.
- Based on the service functionality and service message ontologies, Web services are annotated in our architecture through OWL-S [69] as depicted in Figure 4.11. The OWL-S Profile class has properties called *hasInput* and *hasOutput* whose ranges are "Input" and "Output" classes. These classes, in return have a property, namely, *parameterType*. The value of this property is set to a node in the local message ontologies. In this way the service's input and output parameters are annotated with the service message ontologies.
- In the GLIF model, the data exchange with the external resources is realized through the instances of Procedure, Medication and Observation RIM classes. We have created an XML schema (XSD) of these classes. Then, the OWL ontology from the XSD file is created automatically through the Normalization Tool.
- The next step is to create the mapping definition between the GLIF RIM ontology and Clinical Web service Message Ontologies through the graphical interface of the OWLmt tool. An example mapping is illustrated in Figure 4.14. On the left hand side, the Medication concept in GLIF RIM ontology is presented, the parts of which are mapped to the Medication instance expected by the Web Service. Apart from the copy and the concatenation functionalities, the OWLmt tool also allows the user to define more advanced transformation functions in terms of java scripts.

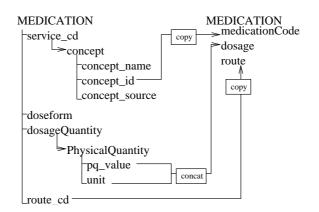


Figure 4.14: An example mapping definition of WS parameters

• Once such a "Mapping Definition" is created by the OWLmt tool, the mediation of message instances are achieved as presented in Figure 4.15:

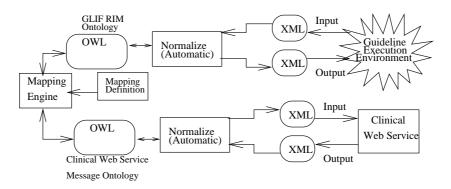


Figure 4.15: Semantic Mediation of Clinical Web Service Messages

- While the guideline is being executed, the Web service parameters are provided as the XML instances of the Reference Information Model used in GLIF.
- These XML instances are normalized to OWL instances of the Reference Information Model used in GLIF.
- The OWL instances are automatically transformed to the instances of the Clinical Web Service message ontologies through the OWLmt tool mapping engine using the "Mapping Definitions" previously created.
- The instances of the Clinical Web Service message ontologies are normalized to XML instances which are the messages the Clinical Web services are expecting to

receive.

• The same procedure is followed when a response is received from the Web Service.

In this way, the guideline execution environment always processes the data encoded as instances of the GLIF RIM, as proposed in the GLIF specification; the Clinical Web Services always process the input, output parameters as the instances of the Reference Information Model used internally in the hospitals.

4.3 Addressing the Interaction with the Alert/Reminder Services

In our architecture, a role based, publish/subscribe mechanism is implemented for delivering the alerts and reminders generated by guideline execution to the related recipients [1]. The alert/reminder system works as follows:

- The alert/reminder system has a Web Service Interface with the underlying Medical Information Systems. Through the Web Services exposed by the hospitals, the information about the Patients, the Guidelines assigned to them, the doctors responsible for their care can be retrieved.
- Graphical Interfaces are provided to the Healthcare Professionals to subscribe to the alerts/reminders about a patient/guideline pair. In addition to this, healthcare professionals may define rules to specify the transmission mechanism of the alerts/reminders (such as SMS, Pager, or email) based on the urgency of the alert/reminder.
- The alert system has a Transmission Layer. In this layer modular transmission mechanisms are provided. Currently message transmission through SMS, email and the Google messenger are supported. SMS transmission is achieved through a Web Service provided by a mobile service provider company.
- The alert system has an interface to receive the alert/reminder message delivery request. The message tuple has the following structure: (patientID, guidelineID, urgency, roles and message content). The alert/reminder message created as a result of the guideline execution is sent to the Alert/Reminder System, by specifying the "roles" to whom the message should be delivered. The roles currently available are: "doctor", "nurse" or "patient relative".

• The alert system has a rule engine which coordinates the publish/subscribe mechanism. The healthcare users who have the specified role and have subscribed to receive the alert/reminder messages of the given patient/guideline pair are located. Then by running the rules specified by the healthcare user, the delivery mechanism is decided and the delivery of the messages are initiated through the Transmission Layer.

This alert system is used in cooperation with the tools developed in this thesis in order to demonstrate how the semantic extensions proposed in this thesis to represent the semantics of alert messages can be exploited to deploy the guideline that can communicate with alert/reminder systems.

CHAPTER 5

THE MULTI-AGENT SYSTEM RESPONSIBLE FOR DEPLOYMENT AND EXECUTION OF CLINICAL GUIDELINES

The semantically enriched clinical guidelines described in Chapter 3 are deployed and executed on the semantic infrastructure described in Chapter 4 through a multi-agent system. Using the semantically enriched guideline definition, the multi-agent system is capable of specializing the clinical guideline definition to a specific patient and can locate the resources for accessing patient data in heterogeneous healthcare settings.

An overview of the subcomponents and their interaction is depicted in Figure 5.1. The system is implemented as a multi-agent system, since as a result of the conceptual design phase we have realized that in order to deploy and execute the clinical guidelines in a heterogeneous distributed environment, there should be a number of autonomous components that should be communicating with each other in a reactive manner, and some of these components should be instantiated and eliminated dynamically based on the demand.

The roles of each agent in our multi-agent can be introduced as follows:

• Agent Factory Agent: The Agent Factory Agent is mainly responsible for specializing the guideline definition to a patient, and creating the Guideline Agent which will execute the clinical guideline. The Agent Factory Agent is capable of processing the generic guideline definition annotated with semantics. By interacting with the resources of medical information systems, it discovers the real implementations of the medical services exposing hospital information system functionalities and sensor services and the document identifiers of the EHR documents of the patients, so that the guideline definition becomes ready to be executed. As a result, a specialized guideline

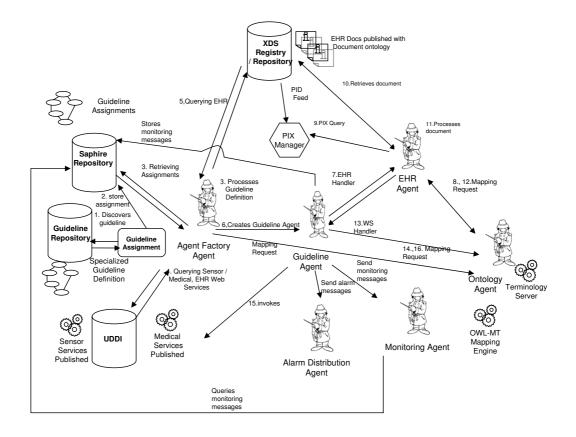


Figure 5.1: The Multi-Agent System responsible for Deployment and Execution of Clinical Guidelines

definition whose "Implementation Layer" classes are initialized is produced. The Agent Factory Agent achieves these by interacting with a number of different components, namely, the XDS Registry, UDDI Registry, Saphire Repository, Guideline Repository, Ontology Agent and Guideline Agent.

• *EHR Agent*: In our architecture, the EHR Agent functions as the gateway to access and extract clinical data from the Electronic Healthcare Records of the patient within a Clinical Affinity Domain. EHR Agent is modelled as a separate agent, to abstract the access to EHR from other agents. Currently in our architecture for sharing EHR documents two different mechanisms are supported: the IHE XDS Registry/Repository architecture and a Web Service based architecture. The EHR Agent is capable of communicating with either the IHE XDS Registry/Repository or Web Services to retrieve the EHR documents. On top of this, in each affinity domain, the EHR Content standard conformed, such as HL7 CDA or EHRCom, may differ. The EHR Agent also abstracts how to process these machine processable EHR Content standards to extract the clinical statements sought by the clinical guideline.

- Ontology Agent: Our architecture is capable of reconciliation of semantic interoperability problems while accessing the resources of healthcare institutes. In the guideline definition, patient data references are modelled in a reference information model based on HL7 RIM. It is possible that the medical Web services, the sensor data, and the EHR documents use different reference information models, and clinical terminologies. Through Ontology Agent this semantic interoperability problem is solved. For this purpose the Ontology Agent exploits the OWLmt ontology mapping tool, and the UMLS terminology server.
- *Guideline Agent*: The Guideline Agent is the main entity which executes the Clinical practice guidelines. It is created by the Agent Factory Agent for each patient-guideline assignment. The Guideline Agent processes the guideline definition specialized to a patient and executes the activities specified in the guideline definition. It can be thought as the enactment engine for the clinical guideline. The Guideline Agent exploits several modular handlers to achieve this responsibility. The Guideline Agent also communicates with other entities in the architecture such as EHR Agent, Ontology Agent, Monitoring Agent and Alarm Distribution Agent.
- *Monitoring Agent*: While the guideline is executed, the current status of the guideline execution is sent to a specific agent which we call Monitoring Agent. Monitoring Agent provides an interface to the clinical practitioners to visualize the execution of the guideline. Monitoring Agent is directly in contact with the Guideline Agent.
- Alarm Distribution Agent: While the guideline is executed, several alarms, notifications, reminders may need to be issued to medical practitioners, and when necessary to the patient relatives. In such cases the alarm message and the role to whom the message should be delivered is informed to an agent, the Alarm Distribution Agent, which is specifically designated to distribute these messages to the necessary recipients in the most efficient and reliable way. The Alarm Distribution Agent is directly in contact with the Guideline Agent.

Apart from these agent the roles of the other entities in the architecture can be summarized as follows:

- Guideline Repository: In our architecture the clinical guideline definitions modelled in the extended guideline model are stored in a Guideline Repository so that they can be accessed by healthcare institutes which want to deploy such guidelines to their institutes. These generic guideline definitions are stored to this Guideline Repository by annotating them with a number of ICD 10 codes. ICD 10 codes presents a categorization of diseases and signs, symptoms, abnormal findings, complaints, social circumstances and external causes of injury or diseases, as classified by the World Health Organization (WHO). The healthcare institutes can discover the relevant clinical guidelines from the Guideline Repository given a problem definition coded in ICD 10 codes.
- Saphire Repository: In order to share information entities among the components of the architecture, we use the Saphire Repository. It is a persistent object store, enabling the storage and retrieval of internal information entities such as Patient-Guideline Assignments, Monitoring messages and so on.
- Patient Identifier Cross-Referencing Manager: As discussed in Section 4.1.1, Patient Identifier Cross-Referencing Managers (PIX Managers), enables the mapping of different Patient identifiers assigned to the same patient by different healthcare institutes. The EHR Agent uses this component to be able to access Electronic Healthcare Records of the patients.
- UDDI Registry: As discussed in Section 4.2, the Web services exposing the functionalities of Clinical Workflows and the data produced by medical sensor devices are published in UDDI registries through their functionality semantics.
- XDS Registry: As discussed in Section 4.1, the Electronic Healthcare Documents are registered in XDS registries through their semantics, so that their URI's can be discovered to retrieve them from the respective EHR repositories.

The Multi-agent System supported with additional components briefly described enables the deployment and execution of clinical guidelines not only within a single clinical affinity domain, but also across clinical affinity domains: i.e. the resources necessary for the execution of the clinical guideline such as Electronic Healthcare Records and medical services of clinical workflows can be discovered and accessed across affinity domains. For this purpose, besides the entities that should be available in each clinical affinity domain, some supplementary components should also be hosted to enable the communication across affinity domain

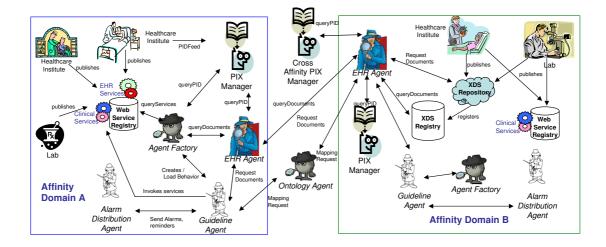


Figure 5.2: The Multi-Agent System spanning across clinical affinity domains

as depicted in Figure 5.2. These are the Cross-Affinity PIX Managers to resolve different patient identifiers used in different clinical affinity domains, and the Ontology Agents to resolve different clinical terminology systems used by different clinical affinity domains.

For implementing the Multi-agent System we have utilized the JADE [45] agent development platform. The agents communicate with each other using JADE Ontologies implemented conforming to FIPA Agent Communication Language specifications [27]. In the following sections the functionalities of these agents and their interaction with each other will be detailed.

5.1 EHR Agent

As presented in Chapter 3, accessing the Electronic Healthcare Records of the patient is an indispensable requirement for executing the clinical guidelines. However the EHR's of a patient may be stored separately in each healthcare institute s/he has been previously hospitalized. In our architecture, the healthcare institutes that cooperate for the care of a patient are grouped as clinical affinity domains. These clinical affinity domains may have agreed on different platforms for sharing the EHRs of the patient that are not interoperable with each other. This is in fact a real life situation: in U.K as the national health infrastructure, a central architecture called SPINE [89] will be used for sharing medical summaries of patients, while in Canada, an IHE-XDS based infrastructure is being built for the same purpose [9]. To abstract the access to the EHR from the Clinical Guideline Execution Environment, we

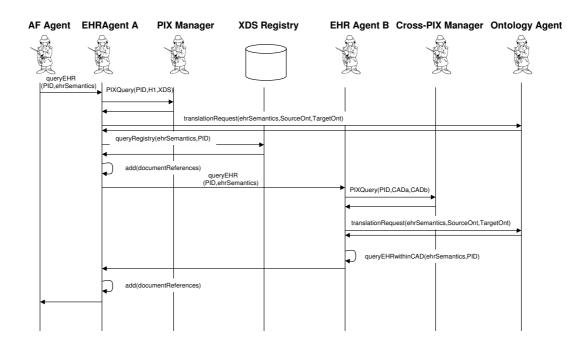


Figure 5.3: Interaction of EHR Agent with other components for discovering EHR References

have created a dedicated agent, the EHR Agent for each such affinity domain as presented in Figure 5.2.

The EHR Agent can be thought as a gateway for locating and accessing EHRs of the patients. Each EHR Agent is specialized in the platform agreed in that affinity domain for sharing EHRs. When a request for discovering and requesting an EHR document is received by an EHR Agent, the EHR Agent both tries to locate the EHR document within its affinity domain through the methodology agreed by the clinical affinity domain such as IHE-XDS, and also forwards the request to the EHR Agents of the other clinical affinity domains as presented in Figure 5.3. In this way, the EHR documents will be available to the requesting entity, although heterogeneous systems are used by different affinity domains.

In our architecture, we have implemented EHR Agents accessing the IHE-XDS EHR Registry/Repositories: When a specific EHR of a specific patient is sought, a "queryEHR" message is sent to the EHR Agent as presented in Figure 5.4. In this example message, the patient identifier is presented and the document type metadata is specified with "LOINC Document Type Codes" such as "11450-4" for "Active Problems". Using this metadata, and the patient identifier, a "QueryRegistry" transaction is issued to the XDS Registry, and as a response a set of Document Identifiers pointing to document stored in EHR Repositories is presented (Figure 5.4). These document identifiers are used to access the document

An example queryEHR request	An example response to a queryEHR request
(request	(inform
:sender	:sender
(agent-identifier	(agent-identifier
:name agentfactory-agent@foo.com	:name ehr-agent@foo.com
:addresses (sequence iiop://foo.com/acc))	:addresses (sequence iiop://foo.com/acc))
:receiver (set	:receiver (set
(agent-identifier	(agent-identifier
:name ehr-agent@foo.com	:name agentfactory-agent@foo.com
:addresses (sequence iiop://foo.com/acc)))	:addresses (sequence iiop://foo.com/acc)))
:protocol FIPA-Request	:language FIPA-SL2
:language FIPA-SL2	:ontology (set AF-EHR-Ontology)
:ontology AF-EHR-Ontology	:content
:content	(= (iota ?i
(action	(result
(agent-identifier	(action
:name ehr-agent@foo.co	(agent-identifier
:addresses (sequence iiop://foo.com/acc))	:name ehr-agent@foo.com
(queryEHR	:addresses (sequence iiop://foo.com/acc))
(EHR-description	(queryEHR
:pid 12345678	(EHR-description
:ehrSemantics (concept	:pid 12345678
conceptname : Past MedicalHistory	:ehrSemantics (concept
conceptSource:LOINC	conceptname : Past MedicalHistory
conceptID :11348-0))))	conceptSource:LOINC
:reply-with queryEHR-1123234)	conceptID :11348-0))))
	?i))
	(sequence (1654b3de-8b51-420e-839e-b1c39e35bb90.xml
))))
	:in-reply-to queryEHR-1123234)

Figure 5.4: An example queryEHR request and response

content from the Repositories by issuing a "RetrieveDocument" transaction as presented in fig-queryEHR.

Apart from locating and retrieving EHR documents, EHR Agents also serve another important feature: retrieving a specific piece of information from the EHR content. The EHR content standard agreed by each clinical affinity domain may be different, however the EHR Agent of that domain, is capable of processing the document format agreed and extract the requested piece of information in the format requested by the clinical guideline execution environment. As presented in section 4.1.2, in our architecture, we are using HL7 CDA documents as EHR documents, and in our implementation, we have implemented an EHR Agent that is capable of processing the CDA document, locate the requested piece of information among the CDA Entries, and present it to the requesting entity. The algorithm presented in Figure 4.9 is collaboratively executed by the EHR Agent and the Ontology Agent in the Multi-agent System. In the RetrieveEntry request sent to the EHR Agent,

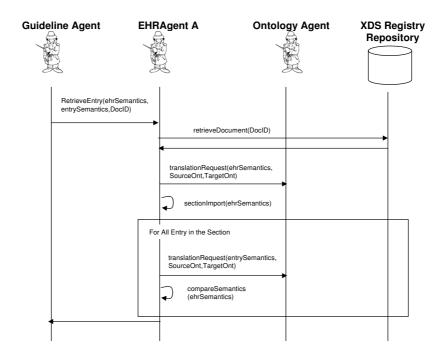


Figure 5.5: Interaction of EHR Agent with other components for Retrieving Clinical Statements

the semantics of the piece of information requested is also specified with coded terms. For example, the clinical guideline execution environment may be in need of discovering whether the patient has previously experienced "bronchial spasm". In the request sent to the EHR Agent (Figure 5.6), besides the document type code for "Past Medical History", the coded term representing "bronchial spasm" is also specified for example as "C0006266" in UMLS medical terminology. In the CDA document all the entries are also annotated with coded terms, however another code from a different terminology may have been used for identifying the same entry in the CDA document which could be the "F-20250" term from SNOMED terminology [87]. To solve this interoperability problem, the EHR Agent consults to the Ontology Agent, and receives an answer to its translation request. In this way although different medical terminologies may have been used, the requested part of the EHR can be extracted from the whole EHR document as presented in Figure 5.5.

5.2 Ontology Agent

The Ontology Agent in our architecture is responsible for handling the semantic mediation of the clinical content used in guideline deployment and execution architecture. It is used for the following purposes as presented in Figure 5.7:

An example retrieveEntry request	An example response to a retrieveEntry request
(request	(inform
:sender	:sender
(agent-identifier	(agent-identifier
:name guideline-agent@foo.com	:name ehr-agent@foo.com
:addresses (sequence iiop://foo.com/acc))	:addresses (sequence iiop://foo.com/acc))
:receiver (set	:receiver (set
(agent-identifier	(agent-identifier
:name ehr-agent@foo.com	:name guideline-agent@foo.com
:addresses (sequence iiop://foo.com/acc)))	:addresses (sequence iiop://foo.com/acc)))
:protocol FIPA-Request	:language FIPA-SL2
:language FIPA-SL2	:ontology (set GA-EHR-Ontology)
:ontology GA-EHR-Ontology	:content
:content	(= (iota ?i
(action	(result
(agent-identifier	(action
:name ehr-agent@foo.co	(agent-identifier
:addresses (sequence iiop://foo.com/acc))	:name ehr-agent@foo.com
(retrieveEntry	:addresses (sequence iiop://foo.com/acc))
(Entry-description	(retrieveEntry
:ehrSemantics (concept	(Entry-description
conceptname : Past MedicalHistory	:ehrSemantics (concept
conceptSource:LOINC	conceptname : Past MedicalHistory
conceptID :11348-0)	conceptSource:LOINC
:entrySemantics (concept	conceptID :11348-0)
conceptname : BronchialSpasm	: entrySemantics (concept
conceptSource:UMLS	conceptname : BronchialSpasm
conceptID :C0006266)	conceptSource:UMLS
:docID : 1654b3de-8b51-420e-839e-b1c39e35bb90.xml	conceptID :C0006266)
)))	:docID : 1654b3de-8b51-420e-839e-b1c39e35bb90.xml
:reply-with retrieveEntry-1123234)))) ?i))
	<pre>(<observation> <service_cd> <concept conceptid="`F-<br">20250' conceptSource=`Snomed' conceptName=`Bronchial Spasm'/> </concept></service_cd></observation>))))</pre>
	:in-reply-to retrieveEntry-1123234)

Figure 5.6: An example retrieveEntry request and response

• Mapping the parameters of Medical Web Services: In our architecture, the guideline execution environment uses a reference information model based on HL7 RIM subset to represent the clinical information. However, it is a fact that several other standards or even propriety formats may be used by the healthcare institutes to represent clinical information. The guideline execution environment needs to communicate with the hospital information systems to reflect the results of guideline execution. For example, the guideline execution can result with a proposal of prescription of a medication to the patient; in this case this information may need to be stored to the hospital information system to affect the clinical workflow. In our architecture, these kinds of interactions are handled through the Web services exposed by the healthcare institutes.

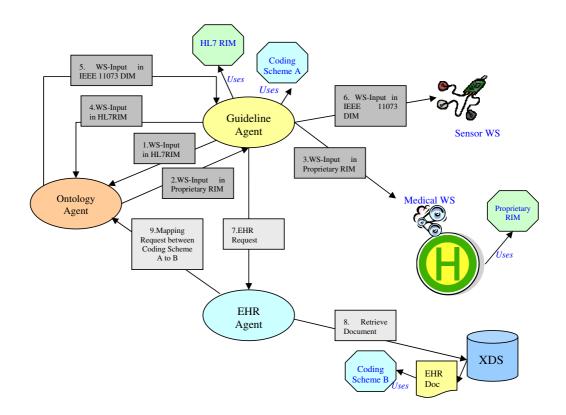


Figure 5.7: The Ontology Agent Interactions with other agents

However it is natural that the parameters of these Web services are conforming to the messaging and content standards used within the hospital, not to the one used in the guideline execution environment. Whenever the Guideline Agent needs to invoke a Medical Web Service, it consults with the Ontology Agent and the input parameters are automatically mediated to the messaging and content standards used by the hospital. The same mechanism is used for mapping the output parameters.

- Mapping the parameters of Sensor Web Services: In our architecture, the guideline execution environment represents the sensor data to be used in guideline execution in the same reference information model based on HL7 RIM. Currently in our architecture the sensor data are exposed as Web services which represent the data in IEEE 11073 Domain Information Model [39]. Whenever a data is received form a Sensor Web Service, the Guideline Agent consults with the Ontology Agent to mediate the sensor data to the reference information model used in the guideline execution environment.
- Mapping the content of the Electronic Healthcare Records of the Patient: In our architecture the Electronic Healthcare Records of the patients are represented as HL7

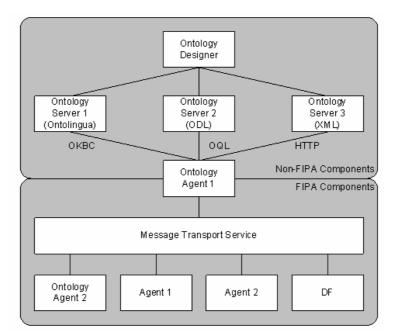


Figure 5.8: The Ontology Service Reference Model proposed by FIPA [28]

CDA documents. In HL7 CDA, the document sections and entities can be coded with coded terms from different coding schemes. In the extended guideline definition model the EHR data can also be annotated with concepts from ontologies or coding schemes. Whenever different coding scheme standards are used, the Ontology Agent is consulted for mediation. Since the Guideline Agent cooperates with the EHR Agent whenever an EHR content is necessary, the mediation request to Ontology Agent is sent by the EHR Agent.

The Ontology Agent is compliant with the FIPA Ontology Service Specifications [28]. According to FIPA Specifications, an Ontology Agent is an agent that provides access to one or more ontology servers and which provide ontology services to an agent community. The Ontology Agent (OA) is responsible for the one or some of these services:

- maintain (for example, register with the DF, upload, download, and modify) a set of public ontologies,
- translate expressions between different ontologies and/or different content languages,
- respond to query for relationships between terms or between ontologies,

The FIPA Specification deals with a standard way to serve the ontology services; it does not mandate any mechanism on how to map the ontologies to one another as presented in

An example translation request	An example response to a translation request
(request	(inform
:sender	:sender
(agent-identifier	(agent-identifier
:name client-agent@foo.com	:name ontology-agent@foo.com
:addresses (sequence iiop://foo.com/acc))	:addresses (sequence iiop://foo.com/acc))
:receiver (set	:receiver (set
(agent-identifier	(agent-identifier
:name ontology-agent@foo.com	:name client-agent@foo.com
:addresses (sequence iiop://foo.com/acc)))	:addresses (sequence iiop://foo.com/acc)))
:protocol FIPA-Request	:language FIPA-SL2
:language FIPA-SL2	:ontology (set FIPA-Ontol-Service-Ontology)
:ontology FIPA-Ontol-Service-Ontology	:content
:content	(= (iota ?i
(action	(result
(agent-identifier	(action
:name ontology-agent@foo.co	(agent-identifier
:addresses (sequence iiop://foo.com/acc))	:name ontology-agent@foo.com
(translate (C0262926))	:addresses (sequence iiop://foo.com/acc))
(translation-description	(translation-description
:from UMLSDocTypeOntology	:from UMLSDocTypeOntology
:to LOINCDocTypeOntology)))	:to LOINCDocTypeOntology))) ?i))
:reply-with translation-query-1123234)	(11348-0))
	:in-reply-to translation-query-1123234)

Figure 5.9: An example translation request and response

Figure 5.8. As well as all the other agents, the OA registers its service with the Directory Facilitator (DF) and it also registers the list of maintained ontologies and their translation capabilities in order to allow agents to query the DF for the specific OA that manages a specific ontology. Being compliant with the FIPA Ontology Service Specification necessitates the Ontology Agent to be able to accept and respond to the ontology service requests in FIPA-Ontol-Service-Ontology ontology. An example translation request and response is presented in Figure 5.9.

As presented, the FIPA Ontology Service Specification does not deal with how the mapping is facilitated. In our architecture, the mapping is facilitated through three different mediation mechanisms (Figure 5.7 and Figure 5.10):

• Mapping the parameters of Medical Web Services: In one of our previous projects, Artemis [18], we have developed an OWL Ontology Mapping Tool, the OWLmt [6], to mediate the input and output parameters of medical Web services between different standards. The Ontology Agent handles such mapping requests through the OWLmt tool. The OWLmt tool provides a graphical interface to define the mapping patterns between OWL ontologies in different structures but with an overlapping content. This

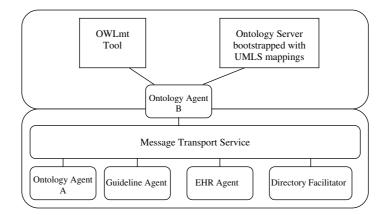
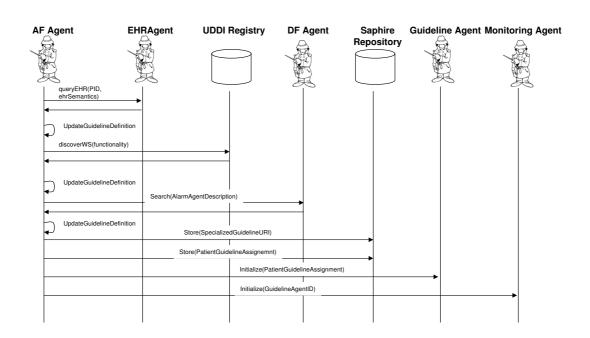


Figure 5.10: The Architecture of the Ontology Agent

mapping definition is used to automatically translate ontology instances to one another. In our architecture, the schemas of Web service messages, and the schema of the Reference Information Model used by the clinical guideline execution environment are lifted to metamodel level and represented as OWL ontologies. Then through the OWLmt GUI, the mapping relationships between them is defined graphically once, which will be used by the OWLmt Mapping engine to mediate the Web service parameters to the reference information model understood by the clinical guideline execution environment. For the details of the OWLmt tool, please refer to [6], where detailed examples of mapping definitions from medical domain are presented.

• Mapping the terminologies used in Clinical Document Content: The Ontology Agent handles such requests by communicating our Ontology Server bootstrapped by the UMLS concepts and their mappings to the nodes of medical terminology systems [96]. The UMLS Metathesaurus contains information about over one million biomedical concepts defined based on the terms from many controlled vocabularies and classifications used in patient records, administrative health data, bibliographic and fulltext databases, and expert system. These controlled vocabularies and classifications (i.e. medical terminology systems) are referred to as the "source vocabularies" of the Metathesaurus. The Metathesaurus reflects and preserves the meanings, concept names, and relationships from its source vocabularies. It provides mapping of the terms in these "source vocabularies" to the concepts in UMLS concept hierarchy. The UMLS Knowledge Sources are downloadable as databases in UMLS Site. In our architecture, we have bootstrapped an Ontology Server, saved the UMLS concept hierarchy and the mappings of the terms of medical terminology systems to these concepts as presented in Figure 4.10. The Ontology Agent queries this Ontology server which supports subsumption for finding the synonyms of clinical terms in different medical terminology systems such as ICD10, LOINC and SNOMED CT if there are any.

• Mapping the parameters of Sensor Web Services: As presented in Section 2.7 in our architecture the sensor data is exposed as Web services in IEEE 11073 DIM. However this information in DIM, should be translated to HL7 RIM which is used by the clinical guideline execution environment. The IEEE 11073 Standards family names this level as "Observation Reporting Interface", and provides guidelines to map the IEEE 11073 DIM to the HL7 observation reporting messages, segments, and fields. The Ontology Agent uses a tool that implements these guidelines to handle the mediation of messages in DIM to HL7 RIM used by GLIF [64].



5.3 Agent Factory Agent

Figure 5.11: Interaction of Agent Factory Agent with other components for deploying and Initializing a Clinical Guideline

In our architecture the agent that is responsible for leading the deployment of a generic

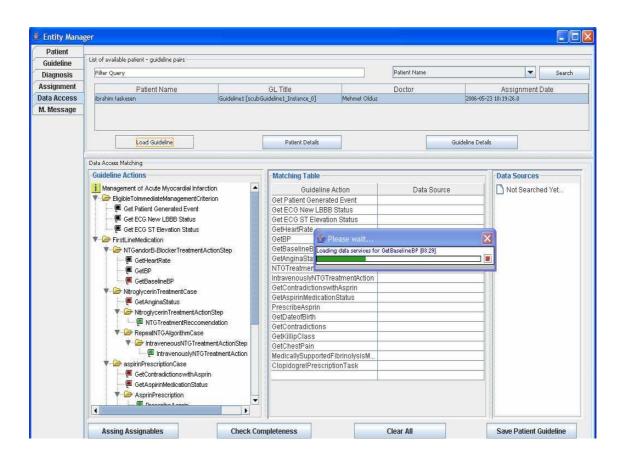


Figure 5.12: The Agent Factory Agent Graphical Interface

clinical guideline definition to a specific patient in a healthcare institution is the Agent Factory Agent. The Agent Factory Agent processes the clinical guideline definitions represented in our extended model, and based on the semantic annotations of the external resources, discovers the instances of the specified resources that are relevant for our specific patient. This process can be summarized as follows (Figure 5.11):

- In our architecture, the medical Web services exposing functionalities of healthcare information systems, and also the sensor Web services exposing the sensor data retrieved from wireless medical sensor devices are published to a UDDI registry by annotating them with their functionality semantics as described in Section 4.2. Whenever the Agent Factory Agent encounters a reference to a medical procedure represented as an instance of "MedicallyOrientedAction" class, it locates the medical services from UDDI service registries by their functionality semantics which has been specified in the extended GLIF model.
- Whenever the Agent Factory encounters a reference to a clinical data of patient to be retrieved from an EHR document (represented through an instance of "EHREntity"

linked to a "GetDataAction" task), it sends a message to the EHR Agent presenting the Document type, and Entry type semantics (the ehrSemantic and entrySemantics properties of "EHREntity" class) presented in the semantically extended GLIF model. As a response a set of document identifiers are received pointing to relevant EHR documents as explained in Section 5.1.

When the Agent Factory Agent encounters a "MessageAction" in the guideline definition, the Agent Factory Agent sends a discovery message to the "Directory Facilitator" (DF) Agent to locate the Alarm Distribution Agent in our multi-agent system. The DF Agent is a centralized registry of entries which associate service descriptions to agent IDs. The agents that wish to advertise their services to other agents register their services to the the DF Agent, so that they can be discovered by the other agents.

This process is supported with a graphical interface as presented in Figure 5.12; whenever more than one possible resource is located by the Agent Factory Agent, those are presented to the user, so that one of the alternative can be selected.

In addition to that, as explained in Chapter 4, in the semantically extended GLIF model in the Implementation Layer (Figure 4.2), we have also reserved slots for storing the pointers to the discovered resources, for example, document identifiers in EHR repositories, the WSDL and OWL-S files of Web services, and Agent Identifier of the Alarm Distribution Agent. As a result of the deployment phase briefly presented, the Agent Factory Agent specializes the generic guideline definition to a patient by creating the instances of the Implementation Layer classes. The specialized clinical guideline definition is saved to SAPHIRE Repository, so that other components such as Guideline Agent can access it.

Whenever the clinical guideline is wished to be executed for remote monitoring of a specific patient, the Agent Factory Agent instantiates a dedicated Guideline Agent for a specific guideline patient pair. Then a request message is sent to the newly created guideline agent to load the guideline-patient assignment, presenting the identifier of the specialized guideline's identifier and patient identifier as presented in Figure 5.13. In addition to this, the Agent Factory Agent informs the Monitoring Agent, about this instantiation, so that the execution of the remote monitoring process can be traced by clinical practitioners as presented in Figure 5.13.

An example Load Patient-Guideline Assignment request	An example Monitor Guideline request
(request	(request
:sender	:sender
(agent-identifier	(agent-identifier
:name agent-factory-agent@foo.com	:name agent-factory-agent@foo.com
:addresses (sequence iiop://foo.com/acc))	:addresses (sequence iiop://foo.com/acc))
:receiver (set	:receiver (set
(agent-identifier	(agent-identifier
:name guideline-agent@foo.com	:name monitoring-agent@foo.com
:addresses (sequence iiop://foo.com/acc)))	:addresses (sequence iiop://foo.com/acc)))
:protocol FIPA-Request	:protocol FIPA-Request
:language FIPA-SL2	:language FIPA-SL2
:ontology AF-GA-Ontology	:ontology AF-MA-Ontology
:content	:content
(action	(action
(agent-identifier	(agent-identifier
:name guideline-agent@foo.co	:name monitoring-agent@foo.co
:addresses (sequence iiop://foo.com/acc))	:addresses (sequence iiop://foo.com/acc))
(loadGuideline	(monitorGuideline
(Assignment-description	(Guideline-description
:patientID 12345678	:patientID 12345678
:guidelineID 1654b3de	:guidelineID 1654b3de
)))	:guidelineagent (agent-identifier
:reply-with loadGuideline-1123234)	:name guideline-agent@foo.co
	:addresses (sequence iiop://foo.com/acc))
)))
	:reply-with monitorGuideline-1123234)

Figure 5.13: Example MonitorGuideline and LoadGuideline request messages

5.4 Guideline Agent

Guideline Agent is the leading agent that coordinates the execution of the clinical guideline definition for remote monitoring of the patients. The Guideline Agent is capable of processing any guideline definition represented in the extended GLIF model, and execute the guideline in cooperation with the other entities of the Multi-agent System. As presented in Figure 5.14, the guideline definition is composed of a number of building blocks. For each building block we have implemented modular handlers. The data sharing among these modular handlers is facilitated through a "Global Variable Pool", where each handler can access to store or retrieve the values of variables used in extended guideline definition. Global Variable Pool is a hash table located in Guideline Agent. Keys of the hash table are the names of the variables as strings. The values of these keys are in GLIF *PatientData* type which can be

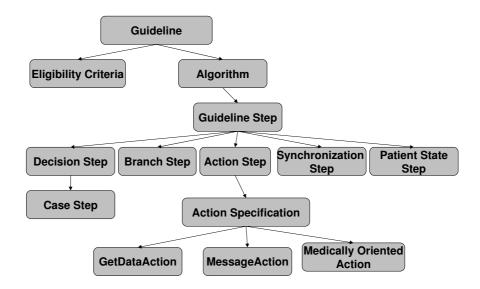


Figure 5.14: The Guideline Agent Handlers

either Medication, Observation or Procedure classes defined in GLIF RIM (Figure 2.3).

The Guideline Agent behavior is implemented to process the extended guideline definition and instantiate these modular handlers as follows:

• Expression Handler: The first step to be executed before a guideline algorithm is initiated is the Eligibility Criteria. The longitudinal records and vital signs of a patient has to be eligible in order to perform the guideline execution. Eligibility Criteria consists of a number of Criterion all of which have to be eligible to start the execution of clinical guideline. A Criterion includes a number of GetDataActions and MedicallyOrientedActions, which define the needed data to be acquired from the EHR records or Sensor devices respectively. The assessment of eligibility criteria is done via evaluating the "expression" component of the Criterion. The Criterion expressions are scripts that specifies whether the Criterion is eligible or not. In our extended guideline model, these Criterion expressions are represented as JavaScript expressions an example of which is presented in Figure 5.15. JavaScript Expressions are executed in our architecture by the Rhino JavaScript Execution Engine [79].

In this execution process, first of all the *GetDataActions* and *MedicallyOrientedActions* specified in the *Eligibility Criteria* Step are executed through the respective handlers. As a result of this, the results of these data gathering actions are stored to Global Variable Pool through their variable names. Then the parameters of the JavaScript expression are fetched from the Global Variable Pool. After each variable is fetched

 $function\ {\tt Eligible TolmmediateManagement Criterion} ({\tt ECGSTElevationStatus},$

ECGNewLBBBStatus, PatientGeneratedEvent)

{

C.	
	if ((ECGSTElevationStatus.getValue().getText() == "true")
	((ECGNewLBBBStatus.getValue().getText() == "true") &&
	$((PatientGeneratedEvent.getService_cd().getConcept_name() == "Angina") $
	$(PatientGeneratedEvent.getService_cd().getConcept_name() == "Dyspnea"))))$
	then return true;
	else return false;
,	

}

Figure 5.15: An example Java Script used as the "specification" of a Criterion

from the Global Variable Pool, they are stored in another hash table named "parameter table", during JavaScript execution, expression variables are accessed through this table by the Rhino Java Script Handler.

If an expression evaluates to false, it means that the guideline has failed to pass the eligibility test and execution is stopped; otherwise the "Eligibility Criteria Handler" continues with the next criteria in the guideline specification. If all of the criteria evaluate to "true", it means that the guideline algorithm needs to be executed. At this point the functionality of "Eligibility Criteria handler" is completed. The execution continues with "Algorithm Handler".

• Algorithm Handler: After the eligibility test yields to "true", the execution of the guideline algorithm starts. Guideline Algorithm is handled by "Algorithm Handler". The functionality of the "Algorithm Handler" is quite simple as fetching the first step from the Guideline Algorithm and calling the "Guideline Step Handler" with the proper parameters. The "Guideline Step Handler" is a generic handler that takes care of every kind of guideline step and calls the appropriate step handler. After step handlers complete their jobs, they call the guideline step handler again. As a result, the algorithm steps can be followed in a mutual recursive procedure. Then the "Guideline Step Handler" calls the next guideline step in the algorithm till "Final" state is reached.

As described in Section 2.1, there are five basic types of guideline steps, for each such step a handler is implemented as presented in Figure 5.16.

- Patient Step Handler: Patient State Step is the simplest guideline step that takes

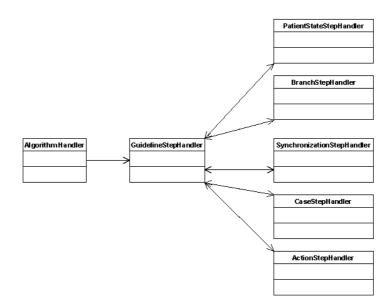


Figure 5.16: The Guideline Step Handlers

place in the GLIF specification. There is no medical or computerized functionality associated with it. The functionality of *Patient State Step* is serving as a milestone in guideline execution. They are used to inform the healthcare users who are monitoring the Guideline Agent about the phases of the guideline execution. Typically in GLIF, the initial and final steps of every clinical guideline are modeled as a Patient State Step. These steps are displayed separately in the graphical interface of the Monitoring Agent.

Branch and Synchronization Step Handlers: Branch and Synchronization Steps are required to execute multiple guideline steps at a point of time and rejoin them when their execution is finished. In Branch Step, the steps that the guideline execution will branch are placed under the "branches" attribute. This attribute refers to a set of guideline steps (branches). Another attribute in the Branch Step definition, declares the order constraint of the guideline executed in parallel. Otherwise order constraint is "parallel", multiple branches are executed in parallel. Otherwise order constraint is "any order" in this case branches are executed one after another. If "parallel" constraint is selected, for each branch step, a new thread is created providing the parameters of the parent thread.

Synchronization of the branches is performed through "Synchronization Step Handler". To keep the track of the branch executions, a "synchronization table" is implemented within the Guideline Agent. Synchronization Table is a hash table the keys of which are the names of "Synchronization Steps", the values corresponding to these keys are integers determining the continuity parameter of the synchronization step. Continuity attribute is specified in the Synchronization Step definition, and it is the number of branches that are needed to be completed in order to proceed to the next step. For example if continuity is equal to 4, it means that 4 different branches must arrive at that synchronization step. The execution mechanism of "Synchronization Step Handler" is briefly as follows: When a synchronization step is reached, the synchronization table of the guideline step is checked. If the step is not present in the table, it denotes that this is the first visit to the step. The step is added to the table with its continuity parameter. In the case that the step is currently in the table, then the continuity value is fetched and decremented by 1. If new value is equal to 1, it means sufficient number of branches has reached Synchronization Step. The step is then deleted from the table. Since the synchronization table is accessed by multiple threads, the table itself needs synchronization. In the implementation, the methods dealing with synchronization table are encapsulated and synchronized so that no two threads access the table at the same time. This prevents the inconsistencies which may occur when different threads reading and updating the table at the same time. When enough number of branches reaches the synchronization step, "Guideline Step Handler" is called by the last thread reaching the Synchronization Step.

- Case Step Handler: In Decision steps one of the steps is selected among the "options" listed in the option list of the Decision Step. The option list is a set of DecisionOptions each of which contains an attribute referring to a Condition Value class. ConditionValue class has an attribute named "caseValue" which is nothing but a Criterion. As the Eligibility Criteria Step, Criterion has a list containing GetDataItems and MedicallyOrientedActions and an expression to be executed. As in Eligibility Criteria Step execution, first the data items are retrieved from the related sources and put in the Global Variable Pool of the Guideline Agent, parameter list formed and JavaScript is executed. This process is carried out by "Expression Handler" classes.

If a *Criterion* yields to "false", the option encapsulating the Criterion is aborted and the next option in the option list is evaluated in the same way. In this way

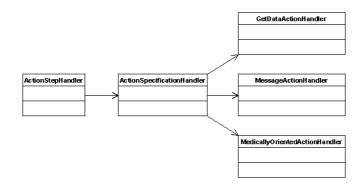


Figure 5.17: The Action Specification Handlers

the execution is analogous to the "if-then-else" statements execution. The options have priority according to their position in the option list. If the *Criterion* bound to an option returns true, the destination of that option (which is a guideline step) is selected directly and guideline continues its execution from that step. The other steps are directly bypassed. If all of the options in the option list fail to result "true" then the "default next step" of the *Decision Step* is compulsorily selected. The guideline execution continues from that default next step through the "Guideline Step Handler".

Action Step Handler: Action steps are the major group of the guideline steps by means of which the Guideline Agent interacts with the other components of the guideline deployment and execution architecture including the Multi-agent System. Through the functionality of action steps, the Guideline Agent gets data from EHR sources (XDS) and sensor Web services, generates alarm messages, and conducts medically oriented actions such as prescription recommendations. The "Action Step Handler" executes the Action Specifications listed in the "tasks" attribute one by one. For this purpose three different "Action Specification Handlers" are created as presented in Figure 5.17. These are detailed in Subsection 5.4.1.

5.4.1 Action Specification Handlers

As presented, three different types of action specifications may occur as a"task" in an Action Step in the guideline model: Get Data Action, Message Action, and Medically Oriented Action. In our implementation each of these action specification types are dealt by means of separate handler classes which can be detailed as follows:

• Get Data Action is the action specification where the required patient data is retrieved from the related data source. After the data is retrieved from the source, they are stored into the Global Variable Pool with the "variable name". In our extended GLIF model, the nature of the data required in the guideline specification is mentioned in "data source type" attribute of the GetDataAction building block of GLIF through an instance of an EHREntity class.

In order to retrieve the required clinical statement from an EHR document, through the "GetDataAction Handler", the Guideline Agent as a first step extracts the entrySemantics and ehrSemantics properties of *EHREntity*, and the docID of the EHR document which has been previously discovered by the Agent Factory Agent. The Guideline Agent creates a "RetrieveEntry" message and sends it to EHR Agent of that clinical affinity domain in an ACL Message (Figure 5.6). After the message is sent, the Guideline Agent waits for the response from the EHR Agent. EHR Agent queries the XDS repository, makes necessary transformations and sends the result back to the Guideline Agent. Guideline Agent which waits busy until the response from the Guideline Agent arrives then extracts the data which comes in XML format. First the data is unmarshalled into a *PatientData* class and then is stored into Global Variable Pool of the Guideline Agent.

- The Medically Oriented Actions represent the medical Web services in the extended GLIF definition. Through the "MedicallyOrientedAction Handler" the Guideline Agent extracts the WSDL and OWL-S of the Web service from the guideline definition specialized to a patient by the Agent Factory Agent. The Guideline Agent prepares the input parameters in HL7 RIM, since GLIF uses this RIM for representing clinical data. While the Web services are discovered from the UDDI registry by the Agent Factory, the OWL-S files of the Web services are also retrieved and saved to the specialized guideline definition. Using this OWL-S file, the Guideline Agent checks the semantics of the input/output parameters, and sends a translation request to the Ontology Agent to translate the input messages from the HL7 RIM to the message ontology specified in the OWL-S file. The same procedure is repeated when the output is received from the Web service. The Sensor Web services are also invoked as the Medical Web service, by contacting with the Ontology Agent to mediate the input and output parameters.
- The Message Actions are used to generate alarm messages within the clinical guideline

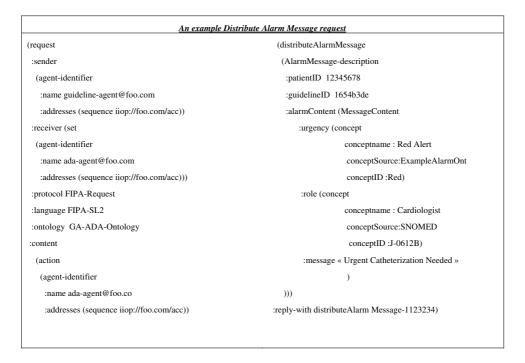


Figure 5.18: Example Distribute Alarm Message request messages

execution. When the "ActionStep Handler" encounters a *Message Action* in the tasks list the "MessageAction Handler" is called which immediately constructs an Alarm Message by combining information coming through guideline definition and Guideline Agent properties. Alarm message, healthcare roles to whom the message is to be delivered and alarm urgency parameters are retrieved from guideline definition whereas patient and guideline IDs are retrieved from Guideline Agent properties. The constructed alarm messages are transmitted to Alarm Distribution Agent as presented in Figure 5.18.

5.5 Alarm Distribution Agent

Alarm Distribution Agent is responsible from accurate and punctual delivery of alarm messages to the healthcare users. It triggers the distribution of the alarms when it receives such a request from the Guideline Agent as presented in Figure 5.18.

Alarm Distribution Agent employs a role based delivery mechanism, in which the real responsible healthcare users for a patient-guideline pair are determined based to the roles indicated by the alarm message. Through a web based interface, the healthcare users can

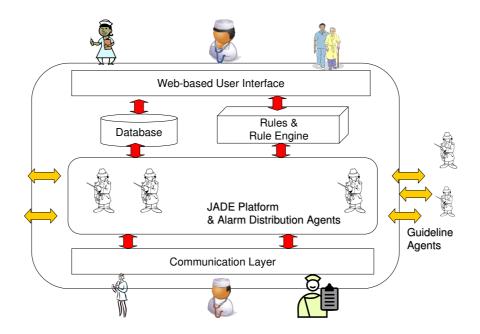


Figure 5.19: The Architecture of Alarm Distibution Agent [1]

subscribe to receive alarm messages related with a specific patient guideline pair. Alarm messages are delivered to the users through three different mediums: SMS, GoogleTalk Instant Messaging and secure e-mail. The users can customize their preferences for receiving alarm messages in different urgencies (medium type, number of deliveries, acknowledgement requirement, routing option etc.) through a web based user interface. User preferences are stored as JESS [48] rules. An example such rule is presented in Figure 5.20. Based on the information presented in the DistributeAlarm Message, these rules are executed in delivery time and the delivery terms are determined [1].

$(defrule \ decide-urgency-Contact1_1_3$
(userid 1)(assignmentid 1)(urgency RED)=>
(store mediumid 3)
(store needack true)
(store numberoftry 4)
(store waitduration 3)
(store mustsend true)
)

Figure 5.20: An example JESS Rule created by Alarm Distribution Agent

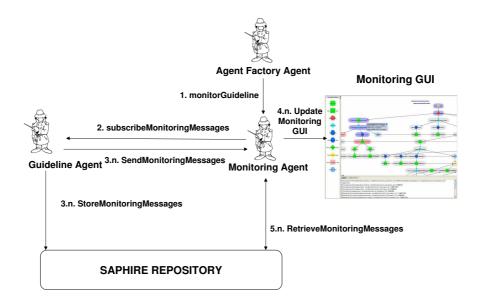


Figure 5.21: The Architecture of Monitoring Agent

5.6 Monitoring Agent

Whenever a guideline execution is initiated for a specific agent through the Agent Factory Agent graphical interface, a Monitoring Agent is created besides a Guideline Agent, and the Monitoring Agent is informed about this new Guideline Agent which will execute the guideline as presented in Figure 5.13. As a response to this message the Monitoring Agent subscribes to Guideline Agent to be able to be informed about the steps of guideline execution. This is achieved through a "SubscribeMonitoringMessages" ACL message sent to the Guideline Agent as presented in Figure 5.22.

As a result of this request, the Guideline Agent, periodically sends Monitoring Messages to each of the subscribed Monitoring Agents while the guideline execution continues. For this purpose while the guideline is being executed, each "Guideline Handler" prepares Monitoring Messages to be ready to sent to the Monitoring Agent. As explained in Section 5.4, the "Guideline Step Handler" calls its subclasses such as "Case Step Handler", "Action Step Handler", and "Patient Step Handler" for the execution of guideline algorithm. Each of these subclasses prepare their Monitoring Messages, based on their execution semantics, and pass this message to the "Guideline Step Handler". The "Guideline Step Handler" sends these Monitoring Messages to the subscribed agents. Among the subclasses of "Guideline Step Handler", the "Action Step Handler" is the most information rich one since it may include a number of tasks defined as Action Specification classes. The "Action Specification

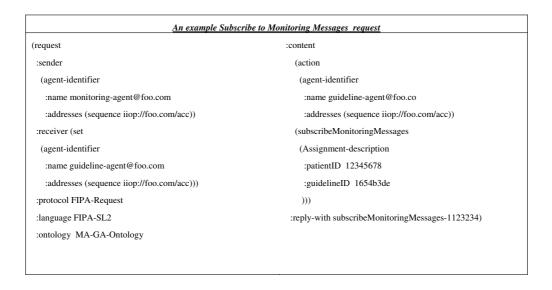


Figure 5.22: Example Subscribe Monitoring Messages request message

Handlers" such as "GetDataAction or MedicallyOrientedAction Handlers", prepare their own monitoring messages and these are collected by the "Action Step Handler" and passed to the "Guideline Step Handler" as an integrated message. In the following paragraphs, the semantics of each monitoring message will be presented:

- Script Execution Message: In Eligibility Criteria evaluation phase and Case Step handling, scripts need to be executed to take appropriate decisions. The clinician has to be informed about these decisions and their reasons. The functionality of the "Script Execution Message" is to inform the clinician about the JavaScripts executed and their results. "Script Execution Message" structure is composed of the following parts:
 - Step Name: Step name is a string which indicates the name of the *Guideline Step*.
 Step name enables the user to view, in which phase the execution is; furthermore it transmits the necessary information to the Monitoring GUI to update the status of the *Guideline Steps*. If a message is sent in an *Eligibility Step*, the Step Name is "Eligibility" by default.
 - Script Expression: It is the JavaScript Expression String that has been executed.
 It is the complete script as a function with parameters and body.
 - Script Result: It is the return value of the function.
 - Time Stamp: Every message has a timestamp which indicates the sending time from the Monitoring Agent. The timestamp facility has two functionalities: to

inform the healthcare user about the times of the actions and to archive the monitoring messages in the SAPHIRE repository to enable later monitoring of the execution.

- Action Step Execution Message: Action Step has a number of tasks that should be executed. For each of such task, an information message is prepared and appended to each other. This appended message consists of the following parts:
 - Step Name
 - Time Stamp
 - Executed Tasks: This part is either a "Web Service Message" or "EHR Message" or "Alarm Message" as described below.
- *EHR Message*: "EHR Messages" are generated after the "RetrieveEntry Result" messages are received from the EHR Agent. Its functionality is to report the retrieved clinical statement. "EHR Message" contains the following parts:
 - Step Name: The name of the GetData Action Specification Class
 - Variable: The variable name of the EHR data in the Global Variable Pool of the Guideline Agent. It is retrieved from the variable_name attribute of the *GetData* Action.
 - Value: The clinical statement retrieved from the EHR Agent. Value is an XML formatted string that corresponds to the *PatientData* instance of the retrieved data.
 - Time Stamp
- Web Service Message: The "Web Service message" is sent to the user before and after invoking a Web service in a Medically Oriented Action. The viewer is informed about the Web service called and its response. The structure of the message is as follows:
 - Step Name: The name of the Medically Oriented Action Specification Class
 - Web Service Name
 - Input: The input parameter which is used to invoke the Web service. It is an XML formatted string that corresponds to the *PatientData* instance
 - Output: The out parameter which is received as a response the invoked the Web service. It is an XML formatted string that corresponds to the *PatientData* instance

- Time Stamp
- Alarm Message: "Alarm Messages" are sent to the user after a Message Action task is executed and an Alarm Message is sent to the Alarm Distribution Agent. In this message the alarm message content and its attributes are presented to the user as follows:
 - Step Name: The name of the Message Action Class
 - Roles: The Healthcare roles to whom this message should be sent
 - Alarm Message content as a string
 - Alarm Urgency
 - Time Stamp

Apart from these, the "Guideline Step Handler" send a "Step Status Message" after guideline execution process enters or exits one of the *Guideline Steps*. The Step Status Message is composed of three parts:

- Step Name
- Step Status: Step Status field denotes the latest status of the steps:
 - RUNNING denotes that the guideline started execution of the step but has not completed yet
 - ABORTED denotes the execution of the step has aborted unexpectedly
 - COMMITTED denotes successfully completed Guideline Steps.
- Time Stamp

Monitoring Agent presents a graphical user interface to the healthcare users for monitoring the execution of the clinical guidelines. Through the Monitoring Agent Interface, healthcare users can start/stop and monitor the execution of clinical guidelines by interacting with the Guideline Agent. Guideline execution is monitored on a user friendly interface which is composed of three parts as presented in Figure 5.23. The main part of the interface depicts the flowchart of the clinical guideline model, whereas the others are for presenting a brief history of executing guideline steps and the legend of the flowchart.

Guideline execution can be traced on the flowchart model. The status of the guideline steps (committed/ongoing/not visited) are identified with different colors. User can click

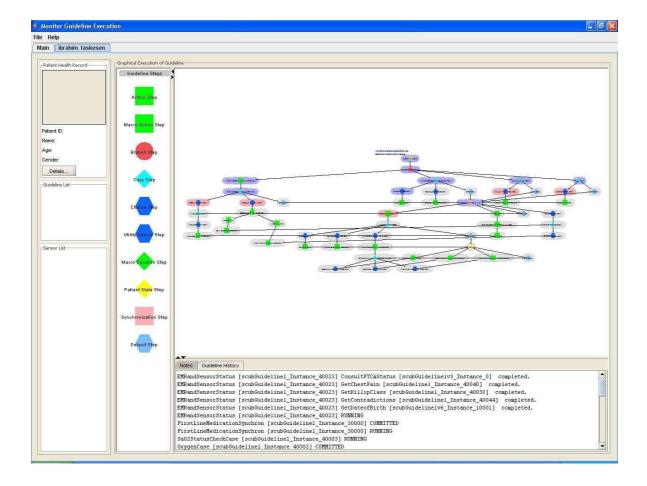


Figure 5.23: The Graphical Interface of the Monitoring Agent

on the steps to get detailed information about the step. In the detailed information screen, user can view the tasks, retrieved patient data (sensor, EHR etc.) and the invoked medical services within these tasks as presented in Figure 5.24.

Whenever requested, the flowchart of previously executed guidelines can also be monitored. For this purpose, the Guideline Agent stores all of the Monitoring Messages sent to Monitoring Agent also to the SAPHIRE Repository. Whenever necessary, the Monitoring Agent retrieves these messages from the SAPHIRE Repository and presents in the Monitoring GUI. Apart from these, an important outcome of the Monitoring Agent is the visual model that it provides for clinical guidelines. This visual flow-chart model can be utilized as an educative medium in training healthcare professionals.

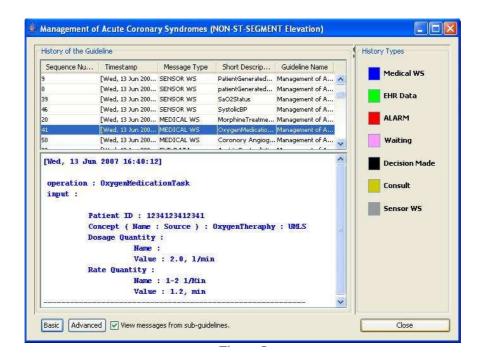


Figure 5.24: The Guideline History Window of Monitoring Agent

CHAPTER 6

A CASE STUDY IN THE ACUTE TREATMENT OF MYOCARDIAL INFARCTION

In this chapter, we present a case study where we define a clinical guideline model for the management of acute myocardial infarction in our semantically extended model.

This case study is one of the pilot applications of SAPHIRE project which is currently being developed for the Emergency Hospital of Bucharest (SCUB). This pilot application aims to demonstrate that the SAPHIRE system can provide bedside intelligent monitoring of patients with subacute coronary syndromes in a wireless fashion to provide computergenerated clinical decision in accordance to the latest European Cardiology Guidelines. In this pilot application, the guideline execution environment provides continuous feedback to the physicians that is patient-specific and guideline-oriented, to provide optimized medical care in accordance with medical standards.

In our pilot application we are using the "Management of acute myocardial infarction in patients presenting with ST-segment elevation" guideline defined by the European Society of Cardiology [101]. It should be noted that the patients with acute myocardial infarction on admission is not our target population. We are addressing patients who are in subacute phase who can still have acute Myocardial Infarction during hospital stay.

A very brief overview of the guideline is presented in Figure 6.1. The guideline document is examined and comprehended by our medical doctor colleagues in SAPHIRE project, who then provided us the flowcharts as presented in Figure 6.1. Then we modelled these flowcharts in the semantically extended GLIF Model that we propose.

The guideline is triggered by a sensor alarm indicating "Persistent ST Elevation¹ and/or

¹ST-segment elevation is usually associated with looming infarction, but can also be due to pericarditis

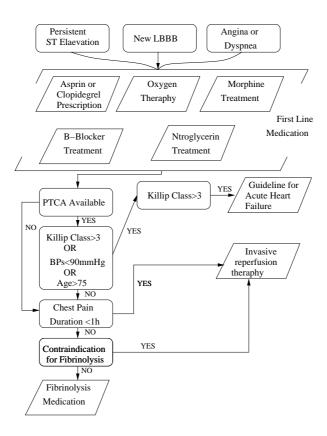


Figure 6.1: The Flowchart of "Management of acute myocardial infarction in patients presenting with ST-segment elevation" Guideline

new LBBB²" coming from the ECG sensor, or by a sensor alarm indicating a "patient generated event" such as dyspnea or angina by pushing a button on one of the sensor devices. This initiates the "First Line Medication" step (detailed in Figure 6.2), where depending on the current medication and medical history of the patient coming from Electronic Healthcare Records (EHR), and also vital signs coming from sensor devices, medications and therapies are proposed by the guideline. After the first-line medication (such as giving aspirin, applying nitroglycerin therapy), based on the vital signs coming from sensors and patient's medical history, either another Guideline for Acute Heart Failure is followed, or a Fibrinolysis therapy³ or an appropriate "invasive reperfusion therapy" is applied to patient. The parallelograms in Figure 6.1 indicate sub-flows that are not detailed in the figure.

or variant angina and it directly comes as an alert from the ECG

 $^{^{2}}$ LBBB means left bundle branch block and usually indicates widespread cardiac disease. When the left bundle is blocked, activation of the left ventricle proceeds through the muscle tissue, resulting in a wide (0.12 msec) QRS complex

³Fibrinolytic drugs are given after a heart attack to dissolve the thrombus blocking the coronary artery, experimentally in stroke to reperfuse the affected part of the brain, and in massive pulmonary embolism

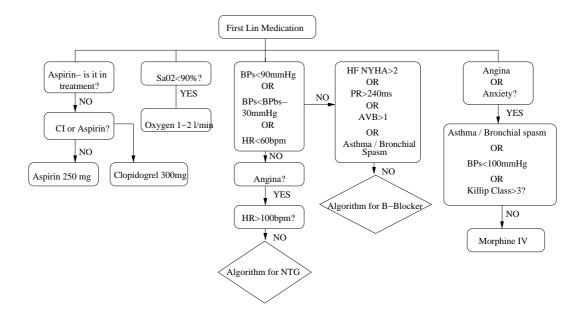


Figure 6.2: The Flowchart of "FirstLine Medication"

The triggering conditions initiating "First Line Medication" are modelled as "Eligibility-Criteria" as presented in Figure 6.3. When such sensor alarms are fired, the steps in the "Algorithm" are initiated. The first item in the "Algorithm" is the "First Line Medication" steps. To be able to model the treatment options for "First Line Medication" (flow of which is detailed in Figure 6.2) a "Branch Step" is created. Four "Case Steps" to represent each treatment option are created and bound to this "Branch Step". This graph model conforming to extended GLIF model is presented in Figure 6.3.

In Figure 6.3 the "Morphine Treatment" "Case Step" is detailed. As presented in the figure, each "Case Step" may have a number of *options*, each of which are instances of "Decision Options". Each "Decision Option" has a "Case Condition" which is an instance of "Criterion" and a "Case Destination" which is an instance of "Action Step". In the "Criterion", the "Expression" to be evaluated to decide whether this option will be followed is defined. It is also possible to add a number of "GetData Actions" and "MedicallyOrientedActions" that will be executed before evaluating the "Expression" in order to collect the data necessary to decide whether this "Case Step" is followed or not. For example, in order to decide whether this patient has previously experienced angina or anxiety, whether asthma or bronchial spasm are medical problems he has previously encountered in his/her medical history, the stage of his/her heart problem condition (Killip Class) and his/her current blood pressure.

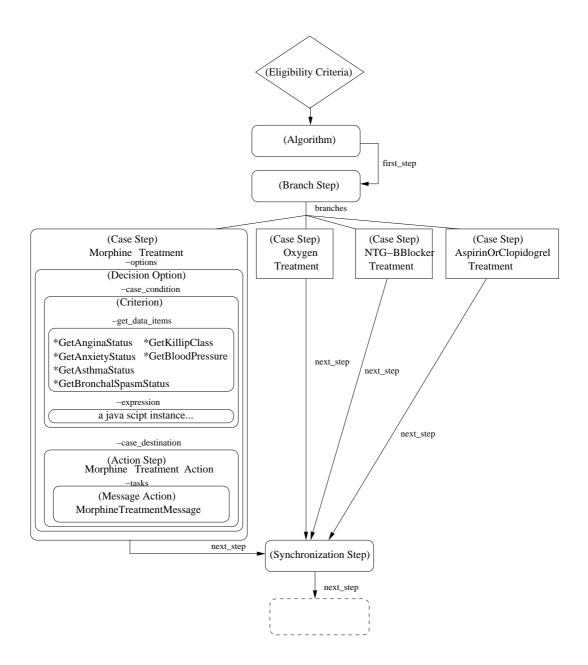


Figure 6.3: The first part of the representation of the guideline in the extended GLIF model

GetAsthmaStatus, GetBronchialSpasmStatus, GetKillipClass) i.e. from the EHR documents and modelled as "GetDataActions", while some of the data should be gathered from medical sensor devices such as GetBloodPressure and GetAnxietyStatus, and should be modelled as "MedicallyOrientedActions". In Figure 6.4, one of these "MedicallyOrientedActions" is presented. As a requirement of our extentions, as the range of the *medicalTask* property of a "MedicallyOrientedAction", a "MedicalActionEntity" instance should be created. Since the recent "Blood Pressure" should be retrieved from a wireless sensor device through a Web Service, we have created "MedicalActionEntity" instance, the BPSensorEntity. As seen in the

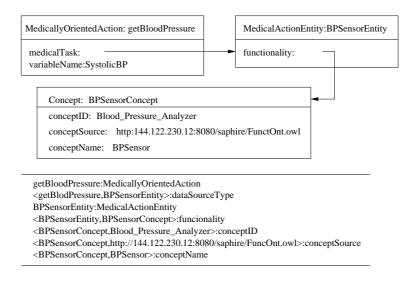


Figure 6.4: The "getBloodPressure" GetData Action representation in the extended GLIF

Figure 6.4, the *functionality* of this instance is selected as an instance of Service Functionality Ontology which is also a class IEEE 11073-10101 Nomenclature defining a blood pressure sensor device.

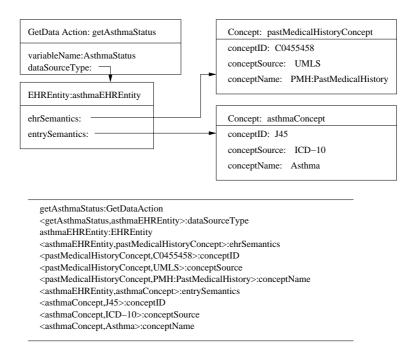


Figure 6.5: The "getAsthmaStatus" GetData Action representation in the extended GLIF

Similarly, for representing the need to retrieve the information from patient's previous

medical history related with "asthma", another "GetData Action" instance is created as presented in Figure 6.5. Here as the range of the *dataSourceType* property, an "EHREntity" instance (*asthmaEHREntity*) is created. The *entrySemantics* property of the *asthmaEHREntity* instance is set to be the *pastMedialHistoryConcept* "Concept" instance. This will guide the guideline deployment engine, i.e. the Agent Factory Agent in our architecture, to locate the Electronic Healthcare Records of the patient, that contain a section related with "Past Medical History". The *entrySemantics* property of *asthmaEHREntity* instance is on the other hand selected to be the *asthmaConcept* "Concept" instance. Here in the "Concept" instance the semantics of the entry to be located is represented through referencing the "J45" node in ICD-10 medical terminology. This indicates that, once the EHR that contain a "Past Medical History" section is located, the "EHR Agent" should look for the related entry presenting previous observations asthma in the EHR document. The other "GetData Action" and "MedicallyOriented Action" instances instances are created similarly to retrieve the required information either from Sensor devices, or EHR documents.

In our architecture the "Expressions" are defined as Java Scripts. The "Expression" for "Morphine Treatment" Case Step is defined in Figure 6.6.

$function\ {\rm morphine Treatment Decision Criterion (Systolic BP,\ Asthma Status,$
Bronchial Spasm Status, Killip Class, Angina Status, Anxiety Status)
{
$if~(!(({ m Systolic BP.getValue}().get{ m Index}() < 100) \mid \mid$
(AsthmaStatus.getValue().getText() == "true")
(BronchialSpasmStatus.getValue().getText() == "true")
${ m (KillipClass.getValue().getIndex()>3))}\$ &&
((AnginaStatus.getValue().getText() == "true")
(AnxietyStatus.getValue().getText() == "true")))
return true;
else return false;

}

Figure 6.6: The Script used in Morphine Treatment Decision Option

As a "Case Destination" of the "MorphineTreatment" Case Step's only option (Figure 6.3),

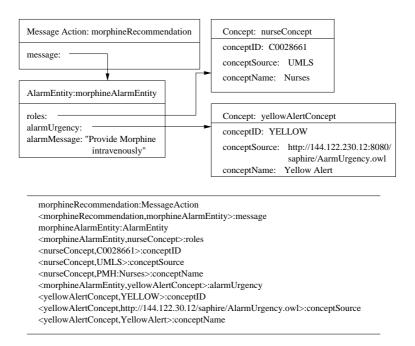


Figure 6.7: The "morphineRecommendation" Message Action representation in the extended GLIF

we need to define an "Action Step" that will alarm a nurse to provide "Morphine" intravenously to the patient to relieve him/her. For this purpose a "MessageAction" instance, *morphineRecommendation* is created as presented in Figure 6.7. We create an "AlarmEntity" as the range of the *message* property, where we define the range of the *roles* property as the *nurseConcept* "Concept" instance, and the range of *alarmUrgency* property as the *yellowAlertConcept* "Concept" instance.

The "Case Steps" are synchronized through a "Synchronization Step" as presented in Figure 6.3, and the guideline continues with an "Action Step", *Get EHR and Sensor Data Status Action*, that aims to gather the information that will be required by the forthcoming decision steps. This second part of the guideline representation in the extended model is presented in Figure 6.8. In this "Action Step", a number of "GetData Actions" and "MedicallyOriented Actions" are added as *tasks*, to know the current blood pressure of the patient, whether *Chest Pain* was among the admission diagnoses, the type of medications that the patient has contraindication and so on. As the *next step* of this "Action Step" a "Case Step" is added. This "Case Step" has three *options*. Each of these *options* are implemented as "Decision Options" where certain conditions expressed through "Criterion" classes are checked to decide whether it is appropriate to provide "Fibrinolysis Treatment" or a "Invasive Reperfusion Therapy" or whether it is necessary to follow the steps of the "Guideline for Acute Heart Failure".

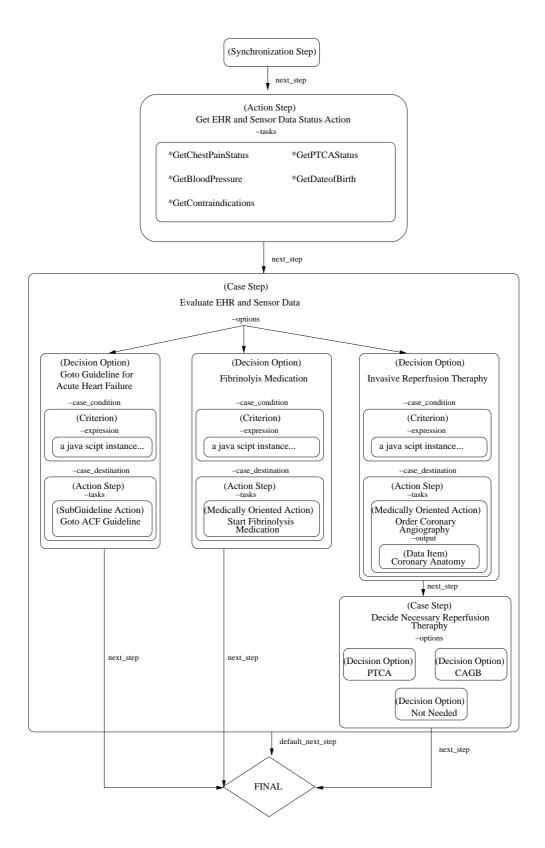


Figure 6.8: The second part of the representation of the guideline in the extended GLIF model

In the third option of the Evaluate EHR and Sensor Data Case step, the guideline checks whether it is necessary to apply an invasive reperfusion therapy. However to decide the type of the invasive reperfusion therapy (i.e. that is whether to perform a PTCA operation (Percutaneous transluminal coronary angioplasty), or to perform CAGB operation (Coronary artery by-pass graft)), the type of the "Vessel Disease" the patient is suffering from should be known. It is only possible to know the condition of the vessels through a "Coronary Angiograpgy" diagnostic operation. Hence as the case destination of Invasive Reperfusion Therapy "Decision Option", a "Medically Oriented Action" which describes a procedure for ordering a Coronary Angiography operation to the necessary department of the hospital is specified. A detailed presentation of how this "Medically Oriented Action" can be represented in the extended GLIF model as presented in Figure 6.9.

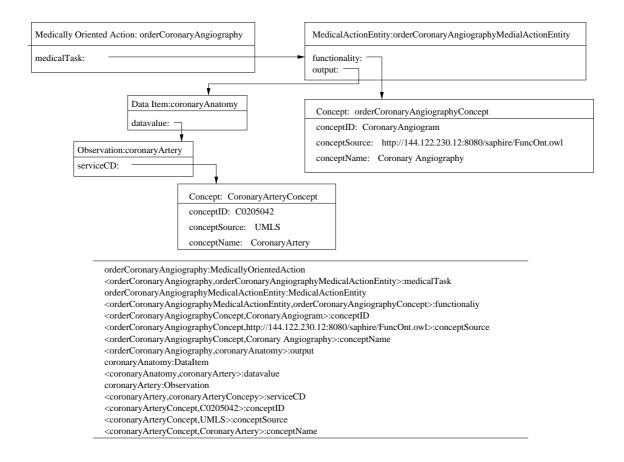


Figure 6.9: The "orderCoronaryAngiography" Medically Oriented Action representation in the extended GLIF

As can be seen in Figure 6.9, the functionality of the "Medical Action Entity" is set to

be a "Concept" class representing a node in the Service Functionality Ontology that we have created. In this way, the guideline deployer, the Agent Factory Agent in our case will be able to locate the required service of the hospital from a service registry for ordering an *Coronary Angiography* operation in the hospital worklist. When this Web service is found and invoked in the guideline execution phase, the output of the Web service should be set to a variable in the guideline definition, so that the *type of the vessel disease that the patient is suffering from* can be exploited to decide the type of the *invasive reperfusion theraphy* to be applied to the patient. As presented in Figure 6.9, the *output* of the *orderCoronaryAngiographyMedicalActionEntity* is set to be the *coronaryAnatomy* "Data Item". The *dataValue* of this "Data Item" is a subclass of "HL7 Reference Information Model", the "Observation" class. In this *coronaryArtery* "Observation" instance, the semantics is defined through the *serviceCD* property, which is set to a "Concept" representing a node in UMLS ontology.

CHAPTER 7

RELATED WORK

7.1 Clinical Practice Guideline Representation Formalisms and Execution Environments based on these Formalisms

In this section a brief overview of computer interpretable clinical guideline representation formalisms, and the guideline execution architectures built based on these formalisms will be presented. While analyzing these formalisms especially their support for formalizing the interfaces with the underlying medical information system functionalities and Electronic Healthcare Record systems will be surveyed and compared with the contribution of this thesis.

7.1.1 Arden Syntax

Arden Syntax [71] is cited as one of the best-known language for representing clinical knowledge needed to create patient-specific decision-support systems. It is a rule-based formalism that encodes medical knowledge in knowledge base form as Medical Logic Modules (MLMs). An MLM is a hybrid representation formalism between a production rule (i.e. an "if-then" rule) and a procedural formalism. Each MLM is invoked as if it is a single-step "if-then" rule, but then it executes serially as a sequence of instructions, including queries, calculations, logic statements and write statements.

Arden was developed for embedding MLMs into proprietary clinical information systems. It was designed to support clinical decision making, each MLM contains sufficient logic to make a single medical decision. Sequencing tasks can be modelled by chaining a sequence of MLMs. MLMs have been used to generate clinical alerts and reminders, interpretations, diagnoses, screening for clinical research studies, quality assurance functions, and administrative support. With an appropriate computer program (known as an event monitor), MLMs run automatically, generating advice where and when it is needed, e.g. to warn when a patient develops new or worsening kidney failure.

One of the deficiencies of the Arden Syntax that it does not provide full support for conceptualizing a multi-step guideline that unfolds over time [72]. It has been presented that the Task Network Model (TNM) approach has arisen in response to this problem. TNM languages typically provide modeling primitives specifically designed for the representation of complex, multi-step clinical guidelines, and for describing temporal and other relationships between component tasks. Unlike rule - based systems, alternative pathways or sequences of tasks (i.e., control flow) can be explicitly modeled, and tools for the visual representation of plans and the organization of tasks within them are provided.

In Arden syntax the references to clinical data is represented in curly braces in MLMs. This is because, these data references must be adapted to the local institution in order to use the local clinical repository. When a clinical guideline model is to be deployed to a local institution, these references in curly braces are mapped to the data model of the local clinical repository. After Arden, this localization problem of deploying guideline models is usually cited as "Curly Braces Problem" in the literature.

7.1.2 Asbru Model

Asbru is a collaboratively effort of Ben Gurion University and the Vienna University of Technology within the scope of Asgaard project [2]. In Asbru formalism, clinical guidelines are viewed as generic skeletal-plan schemata that represent clinical procedural knowledge and that are instantiated and refined dynamically by care providers over significant time periods. As a reflection of this idea, Asbru is designed a task-specific and intention-based plan representation language to embody clinical guidelines and protocols as time-oriented skeletal plans. Skeletal plans provide a powerful way to reuse existing domain-specific procedural knowledge, while leaving room for execution-time flexibility to achieve particular goals [84]. The skeletal plans have been enriched by adding plan attributes such as intentions, conditions and effects; adding formalisms to support rich set of ordering plans; and defining temporal dimension of states and plans [2]:

- Arguments are values passed from the invoking or calling plan (called parent) to the invoked or called plan (called child).
- Preferences describe the costs, resource constraints, and responsible actor.

- Intentions are high-level goals of the plan an annotation specified by the designer independently of the plan body. Intentions are represented by temporal patterns of actions and states that should be maintained, achieved or avoided.
- Conditions mediate the changes between plan states. Each plan is initially considered. After the filter precondition is fulfilled, it becomes activated. When it is activated and the complete condition is fulfilled, the plan is completed. When in the same situation the abort condition is fulfilled first, the plan becomes aborted.
- Effects describe the relationship between plan arguments and measurable parameters by means of mathematical functions or in a qualitative way. A probability of occurrence can be denoted.
- The plan body contains set of plans to be executed in a particular way. Four different types of plans are available: in sequence, in parallel, in any-order, and unordered. The difference between any-order and unordered is that for any-order only one child plan may be active at a time while for unordered there is not any restriction.

7.1.3 GUIDE

GUIDE is part of a guideline modeling and execution framework being developed at the University of Pavia. One of the important properties of GUIDE and its re-engineered execution environment NewGuide, is that it is a component-based multi-level architecture designed to integrate a formalized model of the medical knowledge contained in clinical guidelines and protocols with both workflow management systems and Electronic Patient Record technologies. It proposes an architecture that aims to integrate:

- Guideline Management System (GlMS) (providing clinical decision support)
- Electronic Patient Record (EPR)
- Careflow Management System (CfMS) (providing organisational support).

The architecture proposed by NewGuide is presented in Figure 7.1. After the guideline is formalized through an editor it is stored to a guideline repository to be shared with other organizations. Then a healthcare organization that aim to run a guideline in its environment selects a guideline from the repository. The final user invokes the inference engine and creates an instance of the GL for the management of an individual patient. This requires data from a Virtual Medical Record (VMR). VMR is the NewGuide middle layer that stores every

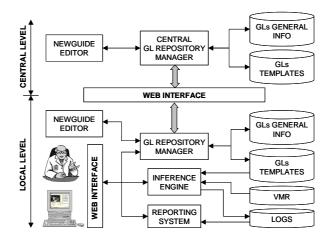


Figure 7.1: The Architecture proposed in GUIDE [11]

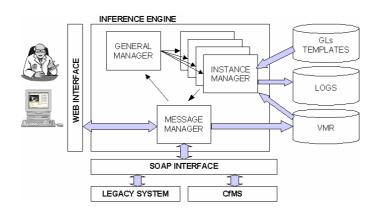


Figure 7.2: The Architecture of the Interence Engine proposed in GUIDE [11]

kind of patient information either acquired through a legacy system (HIS) or entered by the Guideline user. Each inference engine step implies both producing recommendations, such as a drug prescriptions or laboratory tests, and updating a logs database. The latter contains care process information such as progress status of each GL task with relative time stamps. In other words, the VMR and Logs database is the interface of the Guideline Management System with Careflow Management System. In the architecture, the communication between NewGuide and the external world is managed by the message manager, which delegates requests and responses to the web user interface or to a SOAP interface on the basis of the system configuration as presented in Figure 7.2.

7.1.4 PROforma

PROforma was developed at Cancer Research UK for the general purpose of building decision support and intelligent agents. The technology includes the PROforma language, which is a formal knowledge representation language capable of capturing the structure and content of a clinical guideline in a form that can be interpreted by a computer [90]. The language forms the basis of a method and a technology for developing and publishing executable clinical guidelines. In PROforma, a guideline application is modelled as a set of tasks and data items. The notion of a task is central - the PROforma task model divides from the keystone (generic task) into four types: plans, decisions, actions and enquiries:

- Plans are the basic building blocks of a guideline and may contain any number of tasks of any type, including other plans.
- Decisions are taken at points where options are presented, e.g. whether to treat a patient or carry out further investigations.
- Actions are typically clinical procedures (such as the administration of an injection) which need to be carried out.
- Enquiries are typically requests for further information or data, required before the guideline can proceed.

PROforma software consists of a graphical editor to support the authoring process, and an engine to execute the guideline specification. The engine can also be used as a tester during the application development phase. Tallis is one of such software which is a Java implementation of PROforma-based authoring and execution tools developed by Cancer Research UK. Tallis is based on a later version of the PROforma language model. It consists of a Composer (to support creation, editing, and graphical visualisation of guidelines), Tester and Engine (to enact guidelines and allow them to be manipulated by other applications).

7.1.5 GLEE

GLIF Guideline Execution Engine (GLEE) [100] is developed as a tool for executing guidelines encoded in the GLIF format [29]. It is built as middleware that is intended to be integrated with the clinical information system at a local institution through defined interfaces to its electronic medical records (EMRs) and clinical applications. GLEE provides interfaces intended to support integration with the host clinical information system at a

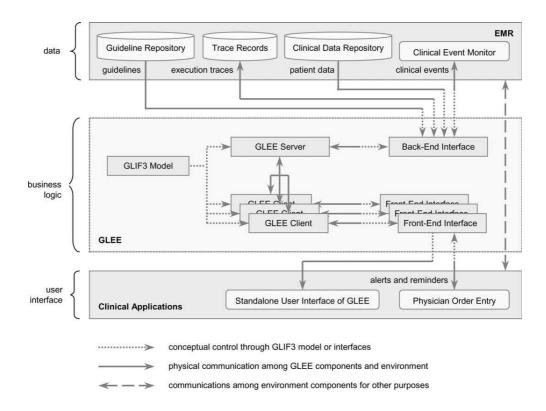


Figure 7.3: The internal structure of GLEE and its interactions with a local environment [100]

local institution. These interfaces are used to link GLEE to a local EMR at the back-end and associated clinical applications (e.g., a physician order-entry system) at the front-end. The communication between GLEE and the EMR at the back-end enables GLEE's access to various resources in the local environment, such as retrieval of patient data and monitoring of clinical events in case the local institution needs to trigger a guideline through specific clinical events. The communication between GLEE and associated clinical applications at the front-end is intended to enable smooth integration of the decision support services provided by GLEE, such as alerts and reminders, within a clinician's workflow. In other words, GLEE defines the business logic of a guideline application, the local EMR will provide data, and the associated clinical application will support the interactions between users and a guideline implementation system. The overall system architecture is shown in Figure 7.3.

GLEE's components can be classified into three conceptual layers: (1) the GLIF guideline representation model, (2) the core components of GLEE, and (3) the interfaces to a host clinical information system. The GLIF guideline representation model specifies a set of generic functions, such as recommendations for specific clinical actions and assistance in medical decision-making, which should be supported by any tool executing guidelines encoded in the GLIF format. The core components of GLEE, as an execution environment for GLIF, define an execution model to realize the generic functions that are required by the GLIF representation model. The interfaces to a host clinical information system reflect GLEE's assumptions on the interactions between GLEE and its host environment during guideline execution. In the GLEE architecture, the system interface between the GLEE server and a host clinical information system is maintained at the back-end based on these assumptions.

GLEE believes that for guidelines to be shared across different institutions, a standard data encoding system and a generic patient data model are two prerequisites. This standard data encoding system plus a generic patient data model will enable references to patient data in an encoded guideline such as in a specification of decision criteria without the need to know the implementation details. It is believed that once such a standard data model exists, at a local institution, the standard definition of patient data are then mapped to the implementation-specific data schema and access methods of the local EMR. GLEE's implementation is based on this assumption, and it is assumed that using the patient data required by GLEE during guideline execution can then be retrieved from the local clinical data repository where clinical data is stored in this standard data model. GLEE itself accepts that it has not yet solved the curly braces problem which refers to the hindrance of medical knowledge sharing caused by incompatible approaches to patient data representation.

Registration of clinical events and notification of clinical actions are implemented in GLEE using a similar approach, through a standard controlled terminology corresponding to the events or the clinical actions that will enable the communication between GLEE and the local environment.

Finally it has been stated that wide acceptance of a guideline system in clinical practice depends on the development of a widely-accepted standard patient-data model and the indepth understanding of local adaptation of guidelines. These issues are not addressed in GLEE but it helped to define the challenges for the future.

7.1.6 GLARE

GLARE is a domain-independent system for acquiring, representing and executing clinical guidelines [91]. The system is based on a modular architecture, which includes an acquisition tool and an execution tool. The acquisition tool is used when a guideline is introduced in the system, e.g., by a committee of expert, and the execution tool is used when a guideline

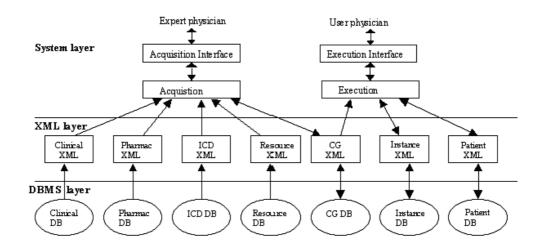


Figure 7.4: The GLARE's three layered architecture [92]

is applied by physicians to a specific situation.

The GLARE representation language is designed to achieve a balance between expressiveness and complexity. The formalism consists of a limited, but very focused and clearly understandable set of primitives. It is made up of different types of actions: plans (i.e. composite actions, hierarchically decomposable in their sub-actions) and atomic actions. Atomic actions can be queries, decisions, work actions and conclusions. All actions are linked by control relations (e.g. sequence, alternative, repetition), defining their order of execution.

The architecture is based on three layers as presented in Figure 7.4: The System Layer contains the two modules called acquisition and execution; The DBMS Layer contains several databases with all data required for both create and execute guidelines. There is stored data about available resources, terminology used in guidelines, information about drugs, information about all open instances of guidelines, a repository of guidelines, and a patient's medical record. Finally The XML Layer allows to represent/manage/exchange data between DBMS layer and System Layer in a structured way.

7.1.7 SAGE: Standards-Based Sharable Active Guideline Environment

The SAGE (Standards-Based Sharable Active Guideline Environment) [94] project is a collaborative research and development project among research groups at IDX Systems Corporation, the University of Nebraska Medical Center, Mayo Clinic-Rochester, Intermountain Health Care, Apelon, Inc., and Stanford University to develop a standards-based comprehensive technology infrastructure that will enable encoding and dissemination of computable

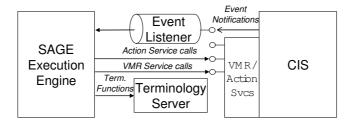


Figure 7.5: SAGE deployment architecture [77]

clinical practice guidelines. Key objectives of the SAGE project are: interoperability of encoded guideline content across disparate Clinical Information System (CIS) platforms and active rendering of guideline content via real-time interaction with existing CIS application functions.

The SAGE guideline model is designed to encode guideline content at the level of detail required for execution within the context of a specific care workflow supported by existing functions of the CIS. To this end, guideline content is represented as detailed clinical "recommendation sets" comprising action specifications, decision logic, and the clinical context in which the recommendations are to be active. The SAGE guideline model uses standard information models, constructs, and data-types to express medical and decision-making concepts. A virtual medical record (VMR) information model [50] has been employed and extended for representation of patient data and guideline-driven actions and all medical concepts are referenced to standard medical terminologies (e.g., SNOMED CT, LOINC).

SAGE aims to develop a deployment driven guideline environment. For this purpose SAGE proposes a methodology for representing clinical content in a standard way through vMR's supported with "Clinical Expression Models (CEMs) which place constraints on the attributes of vMR classes. A compositional method is proposed to express new complex concepts.

For technical interoperability with the underlying clinical information systems, SAGE proposes a set of fixed Action Types for abstracting Clinical Information System's (CIS) functions (such as Notify, Inquire, Recommend_OrderSet) inside the guideline model [81]. SAGE engine supports communication with clinical workflows through events based on these action types as presented in Figure 7.5.

In SAGE methodology, before a formalized guideline can be installed and used in a local institution, its medical content must be reviewed and revised (localization process) and its data models, terminologies, and organization assumptions (roles, events, and resources) must be mapped to those of the local institution by the medical staff (the binding process). The semantics represented in the guideline model is presented to the medical staff, who should manually choose and bind the most appropriate clinical workflow interfaces to the "action types" defined in the guideline definition in order to be able to interact the guideline execution environment. SAGE assumes a set of standard vMR/Action Service Interfaces will be set by standard bodies [77], and clinical information systems will be using these standard messages, vMR interfaces to interact with guideline execution systems.

7.1.8 An Analysis of the support of the Available Clinical Guideline Representation and Execution Architectures for the interfaces with EHRs and Clinical Workflows and Discussion of How this thesis complements these efforts

The difficulty of deploying guideline implementations to healthcare institutes has been addressed in the literature where achieving interoperability with the underlying systems is highlighted as the key challenge.

Some clinical guideline execution engines such as GLEE [100] expect the local medical institutes to store their data in a centralized repository conforming to the guideline data model. However converting patient data to the model supported by the guideline and storing it in a centralized repository are not practical approaches.

In some other guideline models such as PROFORMA [90], ASBRU [84] and GUIDE [11], the relational database tables are used to store the mappings between the guideline model entities and the "columns of the database tables" where the patient data is actually stored. For example in [72], and example table that aims to store mapping guideline data items to EMRs in the GUIDE model is presented (Figure 7.6). The description column gives the name of the data item that guidelines use (Guidelines column). The three other columns are related to the EMR; code is a unique code for that attribute. If possible, it is the SNOMED code. Datatype is the data type of the attribute, and tablename is the name of the EMR table where the attribute value is stored.

In EON [93], ProDIGY [49] and SAGE [94], the availability of "Virtual Medical Record" (vMR) [50] concept is assumed. A vMR is a concept that is claimed to support (1) a structured data model for representing information related to individual patients (2) domains for values of attributes in the data model and (3) queries through which guideline decision

Guideline_Outputs					
Code	Description	Table name	Data Type	Guidelines	
111	Patient_id	Patient_anagraphic	Number	GERD	
112	Pregnancy	Patient_table2	Boolean	GERD	

Figure 7.6: An example table that maps the patient data stored in institutional databases to the data model used in GLARE [72]

support system can test the states of the patient. It is assumed that vMRs allows guideline authors to encode clinical guidelines in a standard way, and the clinical information system developers to accept this standard to communicate with the systems that execute clinical guidelines. The HL7 Clinical Decision Support Technical Committee is undertaking the task of developing a virtual medical record that transforms HL7 RIM [37] to a view of patient data that can be utilized by decision support systems since year 2001, however no standard model has been agreed upon yet.

EON [93], ProDIGY [49] and SAGE [94] assumes that the mapping of local EHR to the vMR is facilitated. For example, in SAGE [94] which addresses the integration of guidelinebased decision support systems with the workflow of care process, a guideline is encoded using the Virtual Medical Record. As in GLIF, the clinical concepts are defined through referencing clinical terminologies such as SNOMED CT where complex concepts are defined as boolean combinations of existing terms. The guideline itself is defined as an activity graph referring to the vMR instances.

It is clear that, these approaches either propose hard coded solutions to the problem like manual deployment procedures to map database tables to variables, or a single common model is assumed to solve the problem. However, even when such a common model is finally agreed upon, the local adaptations of this model are very likely. In other words, even when the vision of vMR is achieved, it is very likely that there will be more than one vMR that the clinical guideline execution engine should be able to handle due to local adaptations.

Our work complements these efforts as follows:

• The extended GLIF Model that we propose aims to facilitate guideline deployment through semantic mediation rather than hard coding the underlying systems to the guideline, or assuming a central repository that provides all the data in the format needed by the guideline. None of the previous approaches addresses automatic or semi-automatic deployment of clinical guidelines. We describe both the semantics of the guideline steps accessing external resources and the semantics of the interfaces of the underlying applications. Our work complements these effort such that we facilitate the semantic matching process by defining both the machine processable semantics of the guideline interfaces and the underlying clinical applications. When different standards, or proprietary formats are used for representing and accessing data, we provide the ability of mapping the clinical content from one standard to the other.

- We describe the semantics of the actions that will interact with Clinical Information System services through a Service Functionality Ontology. This ontology is easily extensible, and also semantic mediation is possible when different ontologies are used to represent clinical services and the actions in the guideline definitions. Additionally we facilitate a semi-automatic semantic matching, discovering and binding process based on the this machine processable semantics of the guideline interfaces and the underlying clinical applications.
- When different message schemas are used by clinical information systems and guideline execution systems, we support a semantic mediation mechanism.

7.2 HeCase2: Agent Based Management of Clinical Guidelines

One of the important works that uses a multi-agent system for the execution of clinical practice guidelines is HeCase2 System [44]. It proposes an agent-based system, where Clinical Practice Guidelines are automatically incorporated into the workflow of doctors and hospital services. The system works on top of HeCase system [43]. The system allows the doctors to be reminded about the steps that should be followed in the treatment of a certain disease, and in this way it reduces the possibility of making errors or forgetting tasks to be done. In addition to this, agents representing patients, doctors and hospital services can automatically coordinate their activities to provide a fast care (e.g. they can arrange the dates for different tests to be performed on the patient easily, without the patient having to visit personally different units of the hospital to arrange those tests).

The multi-agent system proposed is outlined in Figure 7.7. This multi-agent system maps different entities in a healthcare organization (e.g. medical centres, departments, services,

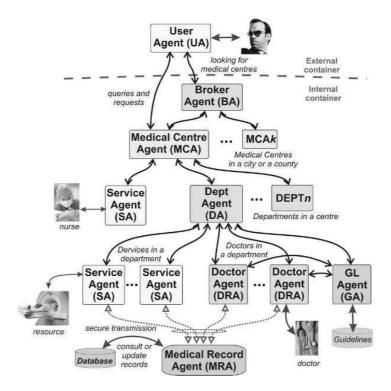


Figure 7.7: The Multi-agent System proposed by HeCase2 [42]

doctors, patients) as agents with different roles.

The functionalities of different agents can be summarized as follows:

- The users interacts with the system through User Agents (UA). These agents stores static data related to the user (e.g. national healthcare number, name, address, access information -login, password, and keys-) and dynamic data (the timetable and the preferences of the user).
- The Broker Agent (BA) is an agent that knows about all the medical centres located in a certain area.
- A Medical Centre Agent (MCA) centralises and monitors the outsiders accesses to the agents that manage the information of a medical centre.
- The departments of Medical Centres are represented by Department Agents (DAs).
- Service Agents (SAs) represent a set of general services linked to human or physical resources (e.g. a blood test service).
- Each department has a staff of several doctors, modelled through Doctor Agents (DRAs), and offers more specific services, also modelled as SAs (e.g., a nurse that

can take different observations in situ). Both MCAs and DAs are aware of the services they can provide (when a SA enters the system, it sends a message detailing its services to the associated MCA or DA).

- Each department contains a Guideline Agent (GA) that performs all actions involved with guidelines (e.g. it can retrieve the CPG associated to a specific illness). The guidelines are represented in PROforma language [90].
- Each department also contains an Ontology Agent (OA) that provides access to the designed medical ontology and complements the information provided by the GA.
- Medical Record Agent (MRA) controls the access to a database that stores all medical records of the patients of the medical centre.
- A clinical practice guideline is executed by this multi agent system as follows:
- When the doctor diagnoses that the patient has a certain disease, its associated DRA requests from GA the associated guideline and it starts to execute it.
- The DRA has a partial knowledge of the system and it does not know all the agents or its responsibilities. The OA is intended to provide that information. Concretely, which agents perform a task or who is the responsible of a required parameter contained in an enquiry. Once the required source/agent/service has been identified, the DRA knows exactly where it can be found.
- The DRA accesses the PHR through the MRA to look for the required parameters.
- Sometimes an action has to be performed on the patient, or the guideline needs data that is not included in the Healthcare Record (e.g. the level of glucose in blood, which can be known with a blood analysis). In these cases, the DRA has to contact with the appropriate SA (it is known by way of the OA) which performs the required action, from the same medical centre or another medical centre.
- The BA allows to exchange information between different medical centres and the UA.

In HeCase2, a medical ontology is defined which describes all the relations established in the multi-agent system associated to a healthcare organisation. It has three main groups of concepts: organisational information of agents, all possible semantic types (entities and events) of the used concepts, and a set of medical concepts related to the clinical guidelines. The ontology agent manages this ontology, and supports the execution of the clinical practice guideline by providing domain knowledge.

7.2.1 How this thesis complements HeCase2

The architecture described in this thesis complements HeCaSe2 architecture in the following aspects:

- In our architecture, the guideline definition is semantically extended so that semiautomatic deployment can be feasible. HeCase2 architecture does not address how a clinical guideline is deployed to a healthcare setting.
- Our architecture enables to retrieve patient records from disparate information sources not only from a database within the institute as proposed in HeCase2. This is achieved by exploiting an industry initiative, namely, IHE XDS.
- In our architecture healthcare services are exposed as Web services and we show that the semantic interoperability problem of guideline automation for accessing disparate clinical workflows can be handled through semantically enriched Web services. Web services have already started to be used in the Healthcare Industry as a solution to technical interoperability problem. In HeCase2 architecture it has been assumed that each healthcare institute has a Service Agent, which enables communication with the underlying medical information system. In our architecture the multiagent system is loosely coupled with the underlying systems of healthcare institutes. We assumed that the healthcare institutes use healthcare industry standards to communicate with Clinical guideline execution systems. The multi agent system in our architecture can be easily deployed on top of a regular healthcare institute's information system infrastructure. However in HeCase2 all the internal applications are modelled as agents, which may not be practically easy to deploy. It needs every other healthcare institute to be re-designed in terms of a multi-agent system, which may not be practical in real life cases.
- In our architecture, the guideline definition as well as the various healthcare institutes may use different reference information models and EHR standards for representing the clinical information about a patient. These are resolved through the semantic mediation capabilities of our Ontology Agent. The Ontology Agent used in HeCase2 architecture serves a medical ontology, does not facilitate semantic mediation.
- In our architecture, it is possible to access and process not only Electronic Healthcare records but also the vital signs coming from sensor devices.

CHAPTER 8

CONCLUSIONS

The World is facing the challenge of delivering high-quality healthcare at affordable cost while the greying population continues to grow at an increasing pace. Due to aging population, chronic diseases and their management costs are also on the rise. In parallel with these, the load of medical practitioners continues to increase. Intelligent healthcare monitoring systems supported with clinical decision support systems are seen as promising tools to address this problem.

Clinical decision support systems are in need of formally expressed domain knowledge, and in healthcare domain, the domain knowledge in clinical practice is usually represented as clinical guidelines which provide evidence-based diagnostic and therapeutic guidelines given a certain clinical condition. Clinical guideline aim to reduce inter-practice variations and cost of the medical services, improve the quality of care [24]. A variety of government and professional organizations are producing and disseminating clinical guidelines [97, 8, 14, 61, 46, 12] for this purpose in the form of narrative documents accompanied by flowcharts.

In order to increase sharability and understandability of these narrative guidelines, several different computer interpretable modeling mechanisms have been proposed such as GLIF [29], ASBRU [84], PROforma [90], ARDEN [71] and EON [93]. Based on these a number of different clinical decision support systems have been built processing these models, such as GLEE [100], GLARE [91], NewGuide [11] and DeGel [85] demonstrating that the guideline definitions can be executed to automate the decision making process.

Although clinical decision support systems based on clinical guidelines are seem to be promising supportive tools for medical practitioners, there are some critical challenges that should be addressed to achieve wide adoption of such tools. First of all the clinical guideline execution environments need to seamlessly access the medical histories of the patients so that personalized guidance for a specific patient rather than text book solutions can be provided to the medical practitioners. In addition to this, these clinical decision support systems need to seamlessly interact with the underlying clinical workflow applications so that the suggestions of the clinical guidelines can be reflected as clinical workflow actions. The interoperability challenges of interacting with diverse clinical workflow systems and Electronic Healthcare Record systems are usually cited as one of the important reasons why clinical guidelines failed to be adopted widely [25, 86]. For this reason the available clinical guideline models aimed to create a "sharable" models rather than "directly" deployable ones, and available clinical guideline execution engines address the automation in a single homogeneous healthcare institute and they are either built on top of an already available clinical information system as an integrated add-on feature, or require custom adaptation phases to communicate with clinical applications.

In this thesis we propose a semantically enriched clinical guideline representation formalism to address this problem. This model is a deployment-driven model that enables the specification of enough level of semantics of the interfaces of the clinical guideline representation model to the underlying clinical applications and Electronic Healthcare Record (EHR) systems [53, 54]. These semantic enrichments are based on widely adopted industry initiatives in healthcare domain. For this purpose we have surveyed the available Electronic Healthcare Record standards[21], and based the semantic enrichments on these widely accepted standards. Through this semantic model we believe that we provide a solution for the "curly-braces" problem cited in the clinical guideline literature, which is the problem of manual localization and deployment of clinical guidelines to healthcare settings.

On top of this semantically enriched clinical guideline model, we have addressed the semantic enrichment of the underlying clinical applications so that semi-automatic deployment of clinical guidelines can be made possible. We described how the semantics of Electronic Healthcare Records systems and Clinical Workflows can be annotated based on widely accepted healthcare standards, and how this semantics can be exploited to facilitate semiautomatic deployment and automatic execution of clinical guidelines: We presented how semantically enriched Web services [18] and IHE-XDS architecture [17] can be exploited for the semantic enrichment of the underlying clinical applications. We demonstrate that it becomes possible to exploit semantic mediation mechanisms (such as ontology mapping tools [6] and terminology servers) to discover and exploit the clinical resources in guideline deployment and execution phases, once machine processable semantics of both clinical guideline interfaces and also the clinical applications are made available. Since the deployment and execution environment is highly distributed and heterogeneous, we have designed and implemented a multi-agent system to coordinate the deployment and execution of clinical guidelines [55]. The system is composed of different agents that can be dynamically instantiated whenever necessary, that can reactively interact with the other entities, and that can abstract the complexity of their roles from the other agents.

The implementation of this thesis is being deployed in real life clinical settings in the Emergency Hospital of Bucharest, for the management of patients with acute myocardial infarction [62, 19]. This deployment aims to increase adherence to the guidelines, hence provide standardization to care processes, to reduce costs of care with optimal benefit for the patient and doctor, to reduce human error in hospital events/complications and finally to provide a feedback system for medical staff in training.

Clinical guidelines usually address a single phase in the patients' treatment, such as screening, diagnosis or rehabilitation. The challange of aging population all over the World can be better addressed by long term care rather than acute care where the citizens take a more informed role in his lifestyle management rather than hospital centered treatments. As a future work, we plan to build an Adaptive Care Planner which aims to ensure continuity and coordination of care between the nursing care, primary, secondary healthcare and homecare based on the results of this thesis. The Adaptive Care Planner will be based on care pathways and will use clinical guidelines as building blocks to define the medical processes for diagnosis, cure, care and monitoring. The control flow of the whole process will be based on the citizen's context such as vital signs, behavioral patterns gathred through environmental sensors, and his electronic healthcare records.

REFERENCES

- Bugrahan Akcay. An Agent-Based Alert Distribution System For Intelligent Healthcare Monitoring. Master's thesis, Middle Eact Technical University, Computer Engineering Department, Ankara, Turkey, 2006.
- [2] The ASGAARD Project: Plan Representation, The Asbru Language. http://www.asgaard.tuwien.ac.at/plan_representation/asbru_doc.html, last visited on May 2008.
- [3] S. K. Bechhofer, C. A. Goble, A. L. Rector, and W. D. Solomon. Terminologies and terminology servers for information environments. In *Proceedings of the 8th International Workshop on Software Technology and Engineering Practice*, pages 484–497, 1997.
- [4] F. Bellifemine, A. Poggi, and G. Rimassa. JADE: A FIPA-compliant Agent Framework. Technical report, CSELT. http://sharon.cselt.it/projects/jade/papers.PAAM.pdf, last visited on May 2008.
- [5] B. Benatallah, M. Hacid, A. Leger, C. Rey, and F. Toumani. On automating web services discovery. VLDB Journal, 14:84–96, 2005.
- [6] V. Bicer, G. Laleci, A. Dogac, and Y. Kabak. Artemis Message Exchange Framework: Semantic Interoperability of Exchanged Messages in the Healthcare Domain. ACM Sigmod Record., 34(3), September 2005.
- [7] T Blakely and C Salmond. Probabilistic record linkage and a method to calculate the positive predictive value. *International Journal of Epidemiology*, 31:1246–1252, 2002.
- [8] Canadian Medical Association Clinical Practice Guideline Infobase. http://mdm.ca/cpgsnew/cpgs/index.asp, last visited on May 2008.

- Canada Health Infoway. http://www.infoway-inforoute.ca/en/home/home.aspx, last visited on May 2008.
- [10] CEN prEN 13606-1. Health informatics Electronic health record communication Part 1: Reference model. Draft European Standard for CEN Enquiry prEN 13606-1, European Committee for Standardization, Brussels, Belgium, 2004.
- [11] P. Ciccarese, E. Caffi, L. Boiocchi, S. Quaglini, and M. Stefanelli. A Guideline Management System. In *Proceedings of MedInfo 2004*, pages 28–32, 2004.
- [12] Clinical Guidelines in Russian. http://www.klinrek.ru/, last visited on May 2008.
- [13] DAML-S 0.7 Draft Release. http://www.daml.org/services/daml-s/0.7/, last visited on May 2008.
- [14] Danish Secretariat for Clinical Guidelines. http://www.sst.dk/Planlaegning-_og_behandling/SfR.aspx?lang=en, last visited on May 2008.
- [15] Description Logics. http://dl.kr.org/, last visited on May 2008.
- [16] A. Dogac, I. Cingil, G.B. Laleci, and Y. Kabak. Improving the Functionality of UDDI Registries through Web Service Semantics. In Proceedings of 3rd VLDB Workshop on Technologies for E-Services (TES-02), August 23-24, 2002.
- [17] A. Dogac, G. Laleci, M. Eichelberg, and T. Aden. Enhancing IHE XDS for Federated Clinical Affinity Domain Support. IEEE Transactions on Information Technology in Biomedicine, 11(2):213-221, 2007.
- [18] A. Dogac, G. Laleci, S. Kirbas, Y. Kabak, S. Sinir, A. Yildiz, and Y. Gurcan. Artemis: Deploying Semantically Enriched Web Services in the Healthcare Domain. Information Systems Journal, special issue on Semantic Web and Web Services, 31(4-5):321-339, 2006.
- [19] M. Dorobantu, A. N. Fruntelata, S. Ghiorghe, L. Ghilencea, G. B. Laleci, and A. Dogac. Intelligent Wireless Monitoring of Acute Cardiac Patients in a Platform Using Computerized Guidelines Models: the SAPHIRE Project. In *The American College of Cardiology 57th Annual Scientific Session*, March 2008.
- [20] ebXML Registry/Repository Architecture. http://www.ebxml.org/, last visited on May 2008.

- [21] M. Eichelberg, T. Aden, J. Riesmeier, A. Dogac A, and G. Laleci. A Survey and Analysis of Electronic Healthcare Record Standards . ACM Computing Surveys, 37(4), 2005.
- [22] M Entwiste and R. N. Shiffman. Turning Guidelines into Practice: Making It Happen With Standards - Part 1. Healthcare and Informatics Review Online, March 2005, March 2005.
- [23] Federal Health IT Initiatives. http://www.hhs.gov/healthit/, last visited on May 2008.
- [24] M J Field and K N Lohr. Guidelines for Clinical Practice: From development to use. Institute of Medicine, National Academy Press, Washington DC, 1992.
- [25] M Fieschi et al. Medical decision support systems: Old dilemmas and new paradigms? Methods Inf Med., 42(3):190–198, 2003.
- [26] Foundation for Intelligent Physical Agents (FIPA). http://www.fipa.org/, last visited on May 2008.
- [27] FIPA Agent Communication Language. http://www.fipa.org/specs/fipa00061/-SC00061G.html, last visited on May 2008.
- [28] FIPA Ontology Service Specification. http://www.fipa.org/specs/fipa00086/-XC00086C.html, last visited on May 2008.
- [29] Guideline Interchange Format (GLIF) 3. Technical report, InterMed Collaboratory, 2004.
- [30] The GLIF Ontology. http://www.smi.stanford.edu/projects/ intermed-web/guidelines/Protege_Ontology.htm, last visited on May 2008.
- [31] Carole A. Goble, Peter Crowther, and Danny Solomon. A medical terminology server. In Database and Expert Systems Applications, pages 661–670, 1994.
- [32] Harmonise, IST-200029329, Tourism Harmonisation Network, Deliverable 3.2 Semantic mapping and Reconciliation Engine subsystems.
- [33] A. Hein, O. Nee, D. Willemsen, T. Scheffold, A. Dogac, and G.B. Laleci. SAPHIRE
 Intelligent Healthcare Monitoring based on Semantic Interoperability Platform -The Homecare Scenario. In Proceedings of the 1st European Conference on eHealth (ECEH), Geneva, Switzerland, 2006.

- [34] Health Level Seven (HL7). http://hl7.org, last visited on May 2008.
- [35] HL7 Approves Web Services Profile and ebXML as 24-Month DSTUs for Messaging Standard. http://xml.coverpages.org/ni2004-05-06-a.html, last visited on May 2008.
- [36] HL7 Clinical Document Architecture, Release 2.0. http://hl7.org/library/ Committees/structure/CDA.ReleaseTwo.CommitteeBallot03.Aug.2004.zip, last visited on May 2008.
- [37] HL7 Reference Information Model. http://www.hl7.org/ v3ballot8/html/foundationdocuments/welcome/index.htm, last visited on May 2008.
- [38] International Statistical Classification of Diseases and Related Health Problems, 10th Revision (ICD-10), Second Edition. Technical report, World Health Organization, Geneva, Switzerland, 2005. http://www.who.int/whosis/icd10/, last visited on May 2008.
- [39] ISO/IEEE 11073-10101:2004:Health informatics, Point-of-care medical device communication, Part 10101: Nomenclature. http://www.iso.org/iso/en/ CatalogueDetail-Page.CatalogueDetail?CSNUMBER=37890, last visited on May 2008.
- [40] IHE IT Infrastructure Technical Framework. http://www.ihe.net/ Technical-_Framework/upload/ihe_iti_tf_2.0_vol1_FT_2005-08-15.pdf, last visited on May 2008.
- [41] Institute of Medicine. Crossing the Quality Chasm: A New Health System for the 21st Century. National Academy Press, Washington DC, 2001.
- [42] D. Isern and A. Moreno. Agent-based careflow using CPGS. In Proceedings of Sete Congres Catala d'Intelligencia Artificial (CCIA'04), pages 293–300, Barcelona, Catalunya, 2004.
- [43] D. Isern, A. Moreno, and D. Sanchez. Provision of agent-based health care services. Journal of Biomedical Informatics, 16(3):167–178, 2003.
- [44] Sr. David Isern. Agent-Based Management of Clinical Guidelines. PhD thesis, Universitat Politecnica de Catalunya, Catalunya, 2005.
- [45] JAVA Agent Development Platform, JADE. http://jade.tilab.com/, last visited on May 2008.

- [46] Japan Council for Quality Health Care. http://jcqhc.or.jp/html/, last visited on May 2008.
- [47] M A Jaro. Advances in Record Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida. Journal of the American Statistical Association,, 84(406):414-420, 1989.
- [48] Jess, the Rule Engine for the Java Platform. http://herzberg.ca.sandia.gov/jess/, last visited on May 2008.
- [49] P. Johnson, S. Tu, and N. Jones. Achieving Reuse of Computable Guideline Systems. In *Proceedings of MedInfo 2001*, pages 99–103, London, UK, 2001.
- [50] P.D Johnson, S.W. Tu, M.A Musen, and I. Purves. A Virtual Medical Record for Guideline-based Decision Support. In *Proceedings of AMIA Annual Symposium*, Washington, DC: Hanley and Belfus, 2001.
- [51] Dipak Kalra. CEN prEN 13606, draft standard for Electronic Health Record Communication, and its introduction to ISO TC/215. Document CEN/TC 251/WG I/N04-52, CEN/TC 251 Health Informatics, Brussels, Belgium, 2004.
- [52] Knowledge Interchange Format (KIF). http://logic.stanford.edu/kif/kif.html, last visited on May 2008.
- [53] G. Laleci and A. Dogac. A Semantically Enriched Clinical Guideline Model Enabling Deployment in Heterogeneous Healthcare Environments. submitted for publication to IEEE Transactions on Information Technology in Biomedicine, 2008.
- [54] G. Laleci, A. Dogac, B. Akcay, M. Olduz, M. Yuksel, U. Orhan, I. Tasyurt, T. Sen, Y. Kabak, T. Namli, O. Gulderen O, and A. Okcan. SAPHIRE: A semantic Web service based Clinical guideline deployment infrastructure exploiting IHE XDS. In *eChallenges Conference*, October 2006.
- [55] G. Laleci, A. Dogac, M. Olduz M, I. Tasyurt, M. Yuksel, and A. Okcan. SAPHIRE: A Multi-Agent System for Remote Healthcare Monitoring through Computerized Clinical Guidelines. In Roberta Annicchiarico, Ulises Cortés, and Cristina Urdiales, editors, Agent Technology and e-Health, Whitestein Series in Software Agent Technologies and Autonomic Computing. Birkhäuser Basel, 2008.

- [56] T Leong, K Kaiser, and S Miksch. Free and Open Source Enabling Technologies for Patient-Centric, Guideline-Based Clinical Decision Support: A Survey. IMIA Yearbook of Medical Informatics, Methods of Information in Medicine, 46-1:74-86, 2007.
- [57] L. Li and I. Horrocks. A Software Framework for Matchmaking Based on Semantic Web Technology. In *Proceedings of WWW03*, Budapest, Hungary, 2003.
- [58] Logical Observation Identifiers Names and Codes (LOINC). http://www.regenstrief.org/loinc/, last visited on May 2008.
- [59] Management of Diabetes Mellitus in Primary Care, Clinical Practice Guideline. http://www.oqp.med.va.gov/cpg/DM/DM3_cpg/frameset.htm, last visited on May 2008.
- [60] Medical Subject Heading Home provided by National Library of Medicine, National Institutes of Health (U.S.). http://www.nlm.nih.gov/mesh/, last visited on May 2008.
- [61] National Institute for Clinical Excellence- England/Wales (NICE) Published Guidelines. http://www.nice.org.uk/page.aspx?o=guidelines.completed, last visited on May 2008.
- [62] O. Nee, A. Hein, T. Gorath, N. Hulsmann, G. Laleci, M. Yuksel, M. Olduz, I. Tasyurt, U. Orhan, A. Dogac, A. Fruntelata, S. Ghiorghe, and R. Ludwig. SAPHIRE: Intelligent Healthcare Monitoring based on Semantic Interoperability Platform - Pilot Applications. *IEE Proceedings Communications-Special Issue on Telemedicine and* e-Health Communication System, June, 2007.
- [63] Nationaal ICT Instituut in de Zorg. http://www.nictiz.nl, last visited on May 2008.
- [64] Alper Okcan. Automatic Acquisition and Use of Multimodal Medical Device Observations based on CEN/ISO/IEEE 11073 and HL7 standards. Master's thesis, Middle Eact Technical University, Computer Engineering Department, Ankara, Turkey, 2007.
- [65] A. Onabajo, I. Bilykh, and J. Jahnke. Wrapping Legacy Medical Systems for Integrated Health Network. In Proceedings of Migration and Evolvability of Long-life Software Systems (MELLS-03) Workshop at the Conference NetObjectDays, Erfurt, Germany, 2003.
- [66] OWL Web Ontology Language Overview. http://www.w3.org/ TR/owl-features/, last visited on May 2008.

- [67] OWL Mapping Tool (OWLmt). http://sourceforge.net/projects/owlmt/, last visited on May 2008.
- [68] OWL Query Language. http://www-ksl.stanford.edu/projects/owl-ql/, last visited on May 2008.
- [69] OWL-S 1.0 Release. http://www.daml.org/services/owl-s/1.0, last visited on May 2008.
- [70] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Matching of Web Service Capabilities. In *Proceedings of ISWC 2002*, pages 333–347, 2002.
- [71] M. Peleg, O. Ogunyemi, and S. Tu. Using features of Arden Syntax with object-oriented medical data models for guideline modeling. In *Proceedings of AMIA Symposium*, pages 523–527, 2001.
- [72] M. Peleg, S. Tu, J. Bury, P. Ciccarese, J. Fox, R.A. Greenes, R. Hall, P.D. Johnson, N. Jones, A. Kumar, S. Miksch, S. Quaglini, A. Seyfang, E.H. Shortliffe, and M. Stefanelli. Comparing computer-interpretable guideline models: a case-study approach. J Am Med Inform Assoc., Jan-Feb;10(1):52–68, 2003.
- [73] M. Peleg, S.W. Tu, R. Haux, and C. Kulikowski. Decision Support, Knowledge Representation and Management in Medicine. IMIA, Schattauer, 2006.
- [74] Profile-based Class Hierarchies, Explanatory remarks for ProfileHierarchy.owl. http://www.daml.org/services/owl-s/1.1/ProfileHierarchy.html, last visited on May 2008.
- [75] Protégé Ontology Editor and Knowledge Acquisition System . http://protege.stanford.edu/, last visited on May 2008.
- [76] T. Pryor and G. Hripcsak. Sharing MLMs: An experiment between Columbia-Presbyterian and LDS Hospital. In Proceedings of the Seventeenth Annual Symposium on Computer Applications in Medical Care, pages 399–403, 1993.
- [77] P. Ram, D. Berg, S. Tu, G. Mansfield, Q. Ye, R. Abarbanel, and N. Beard. Executing clinical practice guidelines using the SAGE execution engine. In *MedInfo*, pages 251– 255, 2004.
- [78] READ Codes. http://www.connectingforhealth.nhs.uk/systemsandservices/data/readcodes/% -introduction, last visited on May 2008.

- [79] Rhino: JavaScript for Java. http://www.mozilla.org/rhino/, last visited on May 2008.
- [80] RIDE Deliverable 2.1.1 European Best practices. http://www.srdc.metu.edu.tr/webpage/projects/ride/modules.php?name=Calendar, last visited on May 2008.
- [81] SAGE Guideline Model Specification Document. http://sage.wherever.org/references/docs/SAGEGuidelineModelSpec.pdf, last visited on May 2008.
- [82] SAPHIRE: Intelligent Healthcare Monitoring based on Semantic Interoperability Platform. http://www.srdc.metu.edu.tr/webpage/projects/saphire/, last visited on May 2008.
- [83] S. Schulz and U. Hahn. Medical knowledge reengineering converting major portions of the UMLS into a terminological knowledge base. International Journal of Medical Informatics,, 64:207–221, 2001.
- [84] A. Seyfang, S. Miksch, and M. Marcos. Combining Diagnosis and Treatment using Asbru. International Journal of Medical Informatics, 68 (1-3):49–57, 2002.
- [85] Y. Shahar, O. Young, E. Shalom, M. Galperin, A. Mayaffit, R. Moskovitch, and A. Hessing. A Framework for a Distributed, Hybrid, Multiple-Ontology Clinical-Guideline Library and Automated Guideline-Support Tools. *Journal of Biomedical Informatics*, 37(5):325–344, 2004.
- [86] R N Shiffman, Y Liaw, C A Brandt, and G J Corb. Computer-based Guideline Implementation Systems: A Systematic Review of Functionality and Effectiveness. Journal of American Medical Informatics Assoc., 6(2):104–114, Mar-Apr 1999.
- [87] SNOMED (The Systematized Nomenclature of Medicine) Clinical Terms. http://www.snomed.org/snomedct_txt.html, last visited on May 2008.
- [88] Simple Object Access Protocol (SOAP) 1.1. http://www.w3.org/TR/soap/, last visited on May 2008.
- [89] SPINE- NHS Connecting for Health. http://www.connectingforhealth.nhs.uk/systems andservices/spine, last visited on May 2008.
- [90] D.R. Sutton and J. Fox. The Syntax and Semantics of the PROforma guideline modelling language. J Am Med Inform Assoc., Sep-Oct;10(5):433-443, 2003.

- [91] P. Terenziani, G. Molino, and M. Torchio. A modular approach for representing and executing clinical guidelines. Artificial Intelligence in Medicine,, 23(3):249–276, 2001.
- [92] P. Terenziani, S. Montani, A. Bottrighi, M. Torchio, G. Molino, L. Anselma, and G. Correndo. Applying Artificial Intelligence to Clinical Guidelines: The GLARE Approach. AI*IA 2003: Advances in Artificial Intelligence, 2829/2003:536-547, 2003.
- [93] S.W. Tu and M.A. Musen. Modeling Data and Knowledge in the EON Guideline Architecture. In Proceedings of MedInfo 2001, pages 280–284, London, UK, 2001.
- [94] S.W. Tu, M.A. Musen, and R. Shankar. Modeling Guidelines for Integration into Clinical Workflow. In *Proceedings of MedInfo 2004*, pages 174–178, 2004.
- [95] Universal Description, Discovery and Integration (UDDI). http://www.uddi.org/, last visited on May 2008.
- [96] Unified Medical Language System (UMLS). http://www.nlm.nih.gov/ research/umls/, last visited on May 2008.
- [97] US National Guideline Clearinghouse. http://www.guideline.gov/, last visited on May 2008.
- [98] United States National Library of Medicine. http://www.nlm.nih.gov/, last visited on May 2008.
- [99] K. Verma, K. Sivashanmugam, A.P. Sheth, A. Patil, S. Oundhakar, and J.A. Miller. METEOR-S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Services. Journal of Information Technology and Management, Special Issue on Universal Global Integration, 6(1):17–3, 2005.
- [100] D. Wang, M. Peleg, S.W. Tu, A. Boxwala, O. Ogunyemi, Q. Zeng, A.R. Greenes, V.L. Patel, and E.H. Shortliffe. Design and Implementation of GLIF3 guideline execution engine. Journal of Biomedical Informatics,, 37:305-318, 2004.
- [101] F. V. Werf, D. Ardissino, A. Betriu, D. Cokkinos, E. Falk, K. Fox, D. Julian, M. Lengyel, F. Neumann, W. Ruzyllo, C. Thygesen, S. Underwood, A. Vahanian, F. Verheugt, and William Wijns. Management of acute myocardial infarction in patients presenting with ST-segment elevation. *European Heart Journal*, 24:28–66, 2003.
- [102] World Health Organization. http://www.who.int/en/, last visited on May 2008.

- [103] Web Services Description Language (WSDL) 1.1. http://www.w3.org/ TR/wsdl, last visited on May 2008.
- [104] Web Service Modeling Ontology (WSMO), WSMO Working Draft 20 September 2004. http://www.wsmo.org/2004/d2/v1.0/, last visited on May 2008.
- [105] XML Scheme Definition (XSD). http://www.w3.org/XML/Schema, last visited on May 2008.

APPENDIX A

THE EXTENDED GLIF MODEL

```
<!DOCTYPE uridef[
 <!ENTITY glifModel "http://www.owl-ontologies.com/GLIF3_5.owl">
]>
<rdf:RDF
    xmlns="http://www.owl-ontologies.com/Ontology1210275825.owl#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:glifModel="&glifModel;#"
  xml:base="http://www.owl-ontologies.comExtendedGLIF.owl">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://www.owl-ontologies.com/GLIF3_5.owl"/>
  </owl:Ontology>
  <owl:Class rdf:ID="MedicalActionEntity">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:ID="accessParams"/>
        </owl:onProperty>
        <owl:allValuesFrom>
          <owl:Class rdf:ID="ServiceImpl"/>
        </owl:allValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="MedicalKnowledgeLayer"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#ServiceImpl">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="ImplementationLayer"/>
```

<?xml version="1.0"?>

```
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="EHRImpl">
 <rdfs:subClassOf rdf:resource="#ImplementationLayer"/>
</owl:Class>
<owl:Class rdf:ID="AlarmEntity">
 <rdfs:subClassOf>
   <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="AlarmImpl"/>
     </owl:allValuesFrom>
     <owl:onProperty>
        <owl:FunctionalProperty rdf:about="#accessParams"/>
      </owl:onProperty>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf rdf:resource="#MedicalKnowledgeLayer"/>
</owl:Class>
<owl:Class rdf:about="#AlarmImpl">
 <rdfs:subClassOf rdf:resource="#ImplementationLayer"/>
</owl:Class>
<owl:Class rdf:ID="EHREntity">
 <rdfs:subClassOf>
   <owl:Restriction>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:about="#accessParams"/>
     </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#EHRImpl"/>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf rdf:resource="#MedicalKnowledgeLayer"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="input">
 <rdfs:domain rdf:resource="#MedicalActionEntity"/>
 <rdfs:range rdf:resource=&glifModel;#Data_Item"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ehrSemantics">
 <rdfs:range rdf:resource="&glifModel;#Concept"/>
 <rdfs:domain rdf:resource="#EHREntity"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="functionality">
 <rdfs:domain rdf:resource="#MedicalActionEntity"/>
 <rdfs:range rdf:resource="&glifModel;#Concept"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="output">
 <rdfs:range rdf:resource="&glifModel;#Data_Item"/>
 <rdfs:domain rdf:resource="#MedicalActionEntity"/>
```

```
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID="entitySemantics">
  <rdfs:domain rdf:resource="#EHREntity"/>
  <rdfs:range rdf:resource="&glifModel;#Data_Item"/>
</owl:ObjectProperty>
<owl:FunctionalProperty rdf:ID="serviceOWSL">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain>
    <owl:Class>
       <owl:unionOf rdf:parseType="Collection">
         <owl:Class rdf:about="#ServiceImpl"/>
         <owl:Class rdf:about="#EHRImpl"/>
       </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="medicalTask">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#MedicalActionEntity"/>
  <rdfs:domain rdf:resource="&glifModel;#Medically_Oriented_Action_Specification"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="agentID">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#AlarmImpl"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="dataSourceType">
  <rdfs:range rdf:resource="#EHREntity"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:domain rdf:resource="&glifModel;#Get_Data_Object_Action"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="message">
  <rdfs:range rdf:resource="#AlarmEntity"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:domain rdf:resource="&glifModel;#Message_Action"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#accessParams">
  <rdfs:domain rdf:resource="#MedicalKnowledgeLayer"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#ImplementationLayer"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="docID">
  <rdfs:domain rdf:resource="#EHRImpl"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="serviceWSDL">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
```

```
134
```

```
<rdfs:domain>

<owl:Class>
<owl:unionOf rdf:parseType="Collection">
<owl:Class rdf:about="#ServiceImpl"/></owl:Class rdf:about="#EHRImpl"/>
</owl:unionOf>
</owl:Class>
</rdfs:domain>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty><//rdf:RDF>
```

<!-- Created with Protege (with OWL Plugin 3.2.1, Build 365) http://protege.stanford.edu -->

APPENDIX B

THE SERVICE FUNCTIONALITY ONTOLOGY

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:expr="http://www.daml.org/services/owl-s/1.1/generic/Expression.owl#"
   xmlns:process="http://www.daml.org/services/owl-s/1.1/Process.owl#"
   xmlns="http://144.122.230.12:8080/saphire/FuncOnt.owl#"
   xmlns:service="http://www.daml.org/services/owl-s/1.1/Service.owl#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:profile="http://www.daml.org/services/owl-s/1.1/Profile.owl#"
 xml:base="http://144.122.230.12:8080/saphire/FuncOnt.owl">
 <owl:Ontology rdf:about=""/>
 <rdfs:Class rdf:about="http://www.daml.org/services/owl-s/1.1/Profile.owl#Profile"/>
 <owl:Class rdf:ID="HealthcareServices">
   <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-s/1.1/Profile.owl#Profile"/>
 </owl:Class>
 <owl:Class rdf:ID="Lung_analyzer">
   <rdfs:subClassOf>
     <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="analyzesSubstance"/>
        </owl:onProperty>
        <owl:allValuesFrom>
          <owl:Class rdf:ID="Lung"/>
        </owl:allValuesFrom>
     </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
     <owl:Class rdf:ID="Multi-ParameterAnalyzer"/>
   </rdfs:subClassOf>
 </owl:Class>
```

```
<owl:Class rdf:ID="Skin_generator">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Skin"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="ElectricalPotentialGenerator"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="QueryServices">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="SchedulingServices"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="RequestNewAppointmentBookingServices">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="NoChangeOnAppointmentPlacerServices"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="AdmitVisitProcess">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="AdmitVisitService"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="RequestForPatientClinicalInformationServices">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="ClinicalInformationServices"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="DemographicDataServices">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="PatientInformationRequestServices"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Haemodynamics_calculator">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Heart"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
```

```
</owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="MultiParameterCalculator"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Heart_output_device">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Blood"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="FlowAnalyzer"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Generator">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="SensorServices"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="PatientProblemServices">
  <owl:disjointWith>
    <owl:Class rdf:ID="TherapeuticorPreventiveServices"/>
 </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="PatientPathwayServices"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="PatientGoalServices"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="LabProcedureServices"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="DiagnosticServices"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="PatientCareServices"/>
  </rdfs:subClassOf>
</owl:Class>
```

```
<owl:Class rdf:ID="ClinicalObservationReportingServices">
```

```
<rdfs:subClassOf>
    <owl:Class rdf:ID="ObservationReportingServices"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Pneumotacograph">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Airway"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="FlowMeter"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ConcentrationGenerator">
  <rdfs:subClassOf rdf:resource="#Generator"/>
</owl:Class>
<owl:Class rdf:ID="PatientGoalUpdateServices">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#PatientGoalServices"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Electromyograph">
 <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Muscle"/>
     </owl:allValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="ElectricalPotentialMeter"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Skin_analyzer">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
```

```
<owl:allValuesFrom>
        <owl:Class rdf:about="#Skin"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Multi-ParameterAnalyzer"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="UrineChemistryAnalyzer">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Urine"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="ConcentrationAnalyzer"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Airway">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Substance"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Heart">
 <rdfs:subClassOf>
    <owl:Class rdf:ID="BodyPart"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="NoChangeOnAppointmentFillerServices">
 <rdfs:subClassOf>
    <owl:Class rdf:ID="FillerServices"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="DischargeProcess">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="DischargeService"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="LabResultServices">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#PatientInformationRequestServices"/>
 </rdfs:subClassOf>
```

```
</owl:Class>
<owl:Class rdf:ID="PatientTreatmentAuthorizationRequestServices">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="PatientReferralServices"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="PatientPathwayResponseServices">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#PatientPathwayServices"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Recommendation">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="PlacerServices"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="RequestForPatientDemographicDataServices">
  <rdfs:subClassOf rdf:resource="#DemographicDataServices"/>
</owl:Class>
<owl:Class rdf:ID="ChangeService">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="PatientAdministrationServices"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Consult">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#PlacerServices"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Airway_volume_meter">
 <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Airway"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="VolumeMeter"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Urine_output_transducer">
  <rdfs:subClassOf>
    <owl:Restriction>
     <owl:allValuesFrom>
        <owl:Class rdf:about="#Urine"/>
      </owl:allValuesFrom>
```

```
<owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#FlowMeter"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Meter">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SensorServices"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="RequestReceiptOfPatientSelectionDisplayListServices">
 <rdfs:subClassOf>
    <owl:Class rdf:ID="PatientNameListServices"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="UpdateRegistrationToClinicalTrialForObservationReporting">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="ObservationReportingOfClinicalTrialsServices"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="GoOffRegistrationToClinicalTrialForObservationReporting">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#ObservationReportingOfClinicalTrialsServices"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ElectricalPotentialAnalyzer">
 <rdfs:subClassOf>
    <owl:Class rdf:ID="Analyzer"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="VolumeAnalyzer">
 <rdfs:subClassOf>
    <owl:Class rdf:about="#Analyzer"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="RegisterProcessService">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="RegisterService"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Muscle">
 <rdfs:subClassOf>
    <owl:Class rdf:about="#BodyPart"/>
 </rdfs:subClassOf>
```

```
</owl:Class>
<owl:Class rdf:about="#NoChangeOnAppointmentPlacerServices">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#PlacerServices"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ECG">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Heart"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#ElectricalPotentialMeter"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="TemperatureGenerator">
  <rdfs:subClassOf rdf:resource="#Generator"/>
</owl:Class>
<owl:Class rdf:about="#PatientInformationRequestServices">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="InformationManagementServices"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="PatientPathwayUpdateServices">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#PatientPathwayServices"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#DischargeService">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#PatientAdministrationServices"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Urine_flow_analyzer">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#Urine"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
```

```
<rdfs:subClassOf>
    <owl:Class rdf:about="#FlowAnalyzer"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="CAGB">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="TherapeuticOperationServices"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Heart_pressure_transducer">
  <rdfs:subClassOf>
   <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Heart"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="PressureMeter"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="CO2_scrubber">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Airway"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="ConcentrationFilter"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Blood_pressure_analyzer">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#Blood"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="PressureAnalyzer"/>
```

```
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#FlowAnalyzer">
 <rdfs:subClassOf>
   <owl:Class rdf:about="#Analyzer"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Lung_resistance_meter">
 <rdfs:subClassOf>
   <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#Lung"/>
     </owl:allValuesFrom>
     <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
     </owl:onProperty>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
   <owl:Class rdf:ID="ResistanceMeter"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="RequestPatientReferralStatusServices">
 <rdfs:subClassOf>
   <owl:Class rdf:about="#PatientReferralServices"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="UnsolicitedInsuranceInformationServices">
 <rdfs:subClassOf>
   <owl:Class rdf:ID="InsuranceInformationServices"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="AirwayFlowAnalyzer">
 <rdfs:subClassOf>
   <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Airway"/>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf rdf:resource="#FlowAnalyzer"/>
</owl:Class>
<owl:Class rdf:ID="Spirometry_analyzer">
 <rdfs:subClassOf>
   <owl:Restriction>
     <owl:allValuesFrom rdf:resource="#Airway"/>
     <owl:onProperty>
```

```
<owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
   <owl:Class rdf:about="#PressureAnalyzer"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#ResistanceMeter">
 <rdfs:subClassOf rdf:resource="#Meter"/>
</owl:Class>
<owl:Class rdf:ID="RequestForCncellationToAnAuthorizationServices">
 <rdfs:subClassOf rdf:resource="#PatientTreatmentAuthorizationRequestServices"/>
</owl:Class>
<owl:Class rdf:ID="UnsolicitedResponseForClinicalObservationReporting">
 <rdfs:subClassOf>
   <owl:Class rdf:ID="ResponseForClinicalObservationReportingService"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="RequestForTreatmentAuthorizationInformationServices">
 <rdfs:subClassOf rdf:resource="#PatientTreatmentAuthorizationRequestServices"/>
</owl:Class>
<owl:Class rdf:about="#DiagnosticServices">
 <rdfs:subClassOf>
   <owl:Class rdf:about="#PatientCareServices"/>
 </rdfs:subClassOf>
 <owl:disjointWith>
   <owl:Class rdf:about="#TherapeuticorPreventiveServices"/>
 </owl:disjointWith>
 <owl:disjointWith rdf:resource="#PatientProblemServices"/>
 <owl:disjointWith>
   <owl:Class rdf:about="#PatientPathwayServices"/>
 </owl:disjointWith>
 <owl:disjointWith>
   <owl:Class rdf:about="#PatientGoalServices"/>
 </owl:disjointWith>
 <owl:disjointWith>
   <owl:Class rdf:about="#LabProcedureServices"/>
 </owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="Skin_transducer">
 <rdfs:subClassOf>
   <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#Skin"/>
     </owl:allValuesFrom>
     <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
```

```
</owl:onProperty>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
   <owl:Class rdf:ID="Multi-ParameterMeter"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Filter">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#SensorServices"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#ElectricalPotentialGenerator">
 <rdfs:subClassOf rdf:resource="#Generator"/>
</owl:Class>
<owl:Class rdf:ID="DischargePending">
 <rdfs:subClassOf rdf:resource="#DischargeService"/>
</owl:Class>
<owl:Class rdf:ID="Heart_output_transducer">
 <rdfs:subClassOf>
   <owl:Restriction>
     <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Heart"/>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
   <owl:Class rdf:about="#FlowMeter"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#PatientGoalServices">
 <owl:disjointWith>
   <owl:Class rdf:about="#TherapeuticorPreventiveServices"/>
 </owl:disjointWith>
 <owl:disjointWith rdf:resource="#PatientProblemServices"/>
 <owl:disjointWith>
   <owl:Class rdf:about="#PatientPathwayServices"/>
 </owl:disjointWith>
 <owl:disjointWith>
   <owl:Class rdf:about="#LabProcedureServices"/>
 </owl:disjointWith>
 <owl:disjointWith rdf:resource="#DiagnosticServices"/>
 <rdfs:subClassOf>
   <owl:Class rdf:about="#PatientCareServices"/>
 </rdfs:subClassOf>
</owl:Class>
```

```
<owl:Class rdf:ID="Blood_chemistry_analyzer">
```

```
<rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#Blood"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Multi-ParameterAnalyzer"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#ElectricalPotentialMeter">
  <rdfs:subClassOf rdf:resource="#Meter"/>
</owl:Class>
<owl:Class rdf:ID="MultigasAnalyzer">
 <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Multi-Gas"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#ConcentrationAnalyzer"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="EEGAnalyzer">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Brain"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#ElectricalPotentialAnalyzer"/>
</owl:Class>
<owl:Class rdf:about="#TherapeuticOperationServices">
 <owl:disjointWith>
    <owl:Class rdf:ID="MedicationServices"/>
```

```
</owl:disjointWith>
 <rdfs:subClassOf>
   <owl:Class rdf:about="#TherapeuticorPreventiveServices"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="GoOffRegistrationToPhaseOfClinicalTrialForObservationReporting">
 <rdfs:subClassOf>
   <owl:Class rdf:ID="PhaseOfClinicalTrialRelatedServices"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#TherapeuticorPreventiveServices">
 <rdfs:subClassOf>
   <owl:Class rdf:about="#PatientCareServices"/>
 </rdfs:subClassOf>
 <owl:disjointWith rdf:resource="#PatientProblemServices"/>
 <owl:disjointWith>
   <owl:Class rdf:about="#PatientPathwayServices"/>
 </owl:disjointWith>
 <owl:disjointWith rdf:resource="#PatientGoalServices"/>
 <owl:disjointWith>
   <owl:Class rdf:about="#LabProcedureServices"/>
 </owl:disjointWith>
 <owl:disjointWith rdf:resource="#DiagnosticServices"/>
</owl:Class>
<owl:Class rdf:ID="Pacemaker">
 <rdfs:subClassOf>
   <owl:Restriction>
     <owl:allValuesFrom rdf:resource="#Heart"/>
     <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
   <owl:Class rdf:ID="RateGenerator"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="PatientProblemAddServices">
 <rdfs:subClassOf rdf:resource="#PatientProblemServices"/>
</owl:Class>
<owl:Class rdf:about="#ObservationReportingOfClinicalTrialsServices">
 <rdfs:subClassOf>
   <owl:Class rdf:about="#ObservationReportingServices"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#MedicationServices">
 <rdfs:subClassOf rdf:resource="#TherapeuticorPreventiveServices"/>
 <owl:disjointWith rdf:resource="#TherapeuticOperationServices"/>
```

```
</owl:Class>
<owl:Class rdf:about="#InformationManagementServices">
 <rdfs:subClassOf rdf:resource="#HealthcareServices"/>
</owl:Class>
<owl:Class rdf:ID="RequestDisplayForClinicalObservationReportingService">
 <rdfs:subClassOf>
   <owl:Class rdf:ID="RequestForClinicalObservationReportingService"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Muscle_analyzer">
 <rdfs:subClassOf>
   <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Muscle"/>
     <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
   <owl:Class rdf:about="#Multi-ParameterAnalyzer"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="PatientGoalQueryServices">
 <rdfs:subClassOf rdf:resource="#PatientGoalServices"/>
</owl:Class>
<owl:Class rdf:ID="Muscle_generator">
 <rdfs:subClassOf>
   <owl:Restriction>
     <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
     </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Muscle"/>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf rdf:resource="#ElectricalPotentialGenerator"/>
</owl:Class>
<owl:Class rdf:about="#ClinicalInformationServices">
 <rdfs:subClassOf rdf:resource="#PatientInformationRequestServices"/>
</owl:Class>
<owl:Class rdf:ID="RequestForResubmissionToAnAuthorizationServices">
  <rdfs:subClassOf rdf:resource="#PatientTreatmentAuthorizationRequestServices"/>
</owl:Class>
<owl:Class rdf:ID="Air_temperature_meter">
 <rdfs:subClassOf>
   <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Airway"/>
     <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
```

```
150
```

```
</owl:onProperty>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
   <owl:Class rdf:ID="TemperatureMeter"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#PatientPathwayServices">
 <rdfs:subClassOf>
   <owl:Class rdf:about="#PatientCareServices"/>
 </rdfs:subClassOf>
 <owl:disjointWith rdf:resource="#TherapeuticorPreventiveServices"/>
 <owl:disjointWith rdf:resource="#PatientProblemServices"/>
 <owl:disjointWith rdf:resource="#PatientGoalServices"/>
 <owl:disjointWith>
   <owl:Class rdf:about="#LabProcedureServices"/>
 </owl:disjointWith>
 <owl:disjointWith rdf:resource="#DiagnosticServices"/>
</owl:Class>
<owl:Class rdf:ID="Intracranial_pressure_analyzer">
 <rdfs:subClassOf>
   <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#Brain"/>
     </owl:allValuesFrom>
     <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
   <owl:Class rdf:about="#PressureAnalyzer"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="PatientProblemUpdateServices">
 <rdfs:subClassOf rdf:resource="#PatientProblemServices"/>
</owl:Class>
<owl:Class rdf:about="#FlowMeter">
 <rdfs:subClassOf rdf:resource="#Meter"/>
</owl:Class>
<owl:Class rdf:ID="PatientProblemResponseServices">
 <rdfs:subClassOf rdf:resource="#PatientProblemServices"/>
</owl:Class>
<owl:Class rdf:ID="PatientPathwayQueryServices">
 <rdfs:subClassOf rdf:resource="#PatientPathwayServices"/>
</owl:Class>
<owl:Class rdf:ID="AdmitVisitCancel">
 <rdfs:subClassOf>
```

```
<owl:Class rdf:about="#AdmitVisitService"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#RegisterService">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#PatientAdministrationServices"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Multi-Gas">
  <rdfs:subClassOf rdf:resource="#Substance"/>
</owl:Class>
<owl:Class rdf:about="#MultiParameterCalculator">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Calculator"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Intracranial_pressure_meter">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#Brain"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#PressureMeter"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="UpdateRegistrationToPhaseOfClinicalTrialForObservationReporting">
 <rdfs:subClassOf>
    <owl:Class rdf:about="#PhaseOfClinicalTrialRelatedServices"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#InsuranceInformationServices">
  <rdfs:subClassOf rdf:resource="#PatientInformationRequestServices"/>
</owl:Class>
<owl:Class rdf:ID="TransferPending">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="TransferService"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#ConcentrationFilter">
 <rdfs:subClassOf rdf:resource="#Filter"/>
</owl:Class>
```

```
<owl:Class rdf:about="#PatientCareServices">
```

```
<rdfs:subClassOf rdf:resource="#HealthcareServices"/>
</owl:Class>
\verb"<owl:Class rdf:ID="CancelRegistrationToPhaseOfClinicalTrialForObservationReporting">Cowl:Class rdf:ID="CancelRegistrationToPhaseOfClinicalTrialForObservationReporting">Cowl:Class rdf:ID="CancelRegistrationToPhaseOfClinicalTrialForObservationReporting">Cowl:Class rdf:ID="CancelRegistrationToPhaseOfClinicalTrialForObservationReporting">Cowl:Class rdf:ID="CancelRegistrationToPhaseOfClinicalTrialForObservationReporting">Cowl:Class rdf:ID="CancelRegistrationToPhaseOfClinicalTrialForObservationReporting">Cowl:Class rdf:ID="CancelRegistrationToPhaseOfClinicalTrialForObservationReporting">Cowl:Class rdf:ID="CancelRegistrationToPhaseOfClinicalTrialForObservationReporting">Cowl:Class rdf:ID="CancelRegistrationToPhaseOfClinicalTrialForObservationReporting">Cowl:Class rdf:ID="CancelRegistrationToPhaseOfClinicalTrialForObservationReporting">Cowl:Class rdf:ID="CancelRegistrationToPhaseOfClinicalTrialForObservationReporting">Cowl:Class rdf:ID="CancelRegistrationToPhaseOfClinicalTrialForObservationReporting">Cowl:Class rdf:ID="CancelRegistrationToPhaseOfClinicalTrialForObservationReporting">Cowl:Class rdf:ID="CancelRegistrationToPhaseOfClinicalTrialForObservationReporting">Cowl:Class rdf:ID="CancelRegistrationToPhaseOfClinicalTrialForObservationReporting">Cowl:Class rdf:ID="CancelRegistrationToPhaseOfClinicalTrialForObservationReporting">Cowl:Class rdf:ID="CancelRegistrationToPhaseOfClinicalTrialForObservationReporting">Cowl:Class rdf:ID="CancelRegistrationToPhaseOfClinicalTrialForObservationReporting">Cowl:Class rdf:ID="CancelRegistrationToPhaseOfClinicalTrialForObservationReporting">CancelRegistrationToPhaseOfClinicalTrialForObservationReporting</a>
   <rdfs:subClassOf>
       <owl:Class rdf:about="#PhaseOfClinicalTrialRelatedServices"/>
   </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ChangeOnAppointmentPlacerServices">
   <rdfs:subClassOf>
       <owl:Class rdf:about="#PlacerServices"/>
   </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="CancelRegistrationToClinicalTrialForObservationReporting">
   <rdfs:subClassOf rdf:resource="#ObservationReportingOfClinicalTrialsServices"/>
</owl:Class>
<owl:Class rdf:ID="PatientGoalDeleteServices">
   <rdfs:subClassOf rdf:resource="#PatientGoalServices"/>
</owl:Class>
<owl:Class rdf:ID="FinancialManagementServices">
   <rdfs:subClassOf rdf:resource="#HealthcareServices"/>
</owl:Class>
<owl:Class rdf:ID="OtherServices">
   <rdfs:subClassOf>
       <owl:Class rdf:about="#SchedulingServices"/>
   </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#PatientNameListServices">
   <rdfs:subClassOf rdf:resource="#PatientInformationRequestServices"/>
</owl:Class>
<owl:Class rdf:ID="ChangeOnAppointmentFillerServices">
   <rdfs:subClassOf>
       <owl:Class rdf:about="#FillerServices"/>
   </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Lung_resistance_analyzer">
   <rdfs:subClassOf>
       <owl:Restriction>
            <owl:onProperty>
                <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
            </owl:onProperty>
            <owl:allValuesFrom>
                <owl:Class rdf:about="#Lung"/>
            </owl:allValuesFrom>
       </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
       <owl:Class rdf:ID="ResistanceAlayzer"/>
   </rdfs:subClassOf>
```

```
</owl:Class>
<owl:Class rdf:about="#TransferService">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#PatientAdministrationServices"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#AdmitVisitService">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#PatientAdministrationServices"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#PlacerServices">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SchedulingServices"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Lung_flow_meter">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#Lung"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#FlowMeter"/>
</owl:Class>
<owl:Class rdf:ID="CancelPatientReferralServices">
 <rdfs:subClassOf>
    <owl:Class rdf:about="#PatientReferralServices"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="RequestReceiptOfClinicalDataListingServices">
 <rdfs:subClassOf rdf:resource="#ClinicalInformationServices"/>
</owl:Class>
<owl:Class rdf:about="#Calculator">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SensorServices"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#RateGenerator">
  <rdfs:subClassOf rdf:resource="#Generator"/>
</owl:Class>
<owl:Class rdf:ID="RequestForModificationToAnAuthorizationServices">
  <rdfs:subClassOf rdf:resource="#PatientTreatmentAuthorizationRequestServices"/>
</owl:Class>
```

```
<owl:Class rdf:ID="PatientProblemDeleteServices">
  <rdfs:subClassOf rdf:resource="#PatientProblemServices"/>
</owl:Class>
<owl:Class rdf:about="#LabProcedureServices">
  <owl:disjointWith rdf:resource="#TherapeuticorPreventiveServices"/>
  <owl:disjointWith rdf:resource="#PatientProblemServices"/>
  <owl:disjointWith rdf:resource="#PatientPathwayServices"/>
  <owl:disjointWith rdf:resource="#PatientGoalServices"/>
  <owl:disjointWith rdf:resource="#DiagnosticServices"/>
  <rdfs:subClassOf rdf:resource="#PatientCareServices"/>
</owl:Class>
<owl:Class rdf:about="#PhaseOfClinicalTrialRelatedServices">
  <rdfs:subClassOf rdf:resource="#ObservationReportingOfClinicalTrialsServices"/>
</owl:Class>
<owl:Class rdf:about="#SensorServices">
  <rdfs:subClassOf rdf:resource="#DiagnosticServices"/>
</owl:Class>
<owl:Class rdf:about="#Skin">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#BodyPart"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Multi-ParameterAnalyzer">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Analyzer"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="PTCA">
  <rdfs:subClassOf rdf:resource="#TherapeuticOperationServices"/>
</owl:Class>
<owl:Class rdf:ID="PatientPathwayDeleteServices">
  <rdfs:subClassOf rdf:resource="#PatientPathwayServices"/>
</owl:Class>
<owl:Class rdf:about="#ResistanceAlayzer">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Analyzer"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="TemperatureAnalyzer">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Analyzer"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="TransferCancel">
  <rdfs:subClassOf rdf:resource="#TransferService"/>
</owl:Class>
<owl:Class rdf:ID="CompleteClinicalTrialForObservationReporting">
  <rdfs:subClassOf rdf:resource="#ObservationReportingOfClinicalTrialsServices"/>
```

```
</owl:Class>
<owl:Class rdf:ID="GetLabResultServices">
 <rdfs:subClassOf rdf:resource="#LabResultServices"/>
</owl:Class>
<owl:Class rdf:ID="Spirometer">
 <rdfs:subClassOf>
   <owl:Restriction>
     <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Airway"/>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf rdf:resource="#Multi-ParameterAnalyzer"/>
</owl:Class>
<owl:Class rdf:about="#Brain">
 <rdfs:subClassOf>
   <owl:Class rdf:about="#BodyPart"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="TransferProcess">
 <rdfs:subClassOf rdf:resource="#TransferService"/>
</owl:Class>
<owl:Class rdf:ID="Vaporizer">
 <rdfs:subClassOf>
   <owl:Restriction>
     <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
     </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Airway"/>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf rdf:resource="#ConcentrationGenerator"/>
</owl:Class>
<owl:Class rdf:about="#FillerServices">
 <rdfs:subClassOf>
   <owl:Class rdf:about="#SchedulingServices"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="AdmitVisitPending">
 <rdfs:subClassOf rdf:resource="#AdmitVisitService"/>
</owl:Class>
<owl:Class rdf:about="#TemperatureMeter">
 <rdfs:subClassOf rdf:resource="#Meter"/>
</owl:Class>
<owl:Class rdf:about="#PressureMeter">
 <rdfs:subClassOf rdf:resource="#Meter"/>
</owl:Class>
```

```
<owl:Class rdf:ID="ModifyPatientReferralServices">
 <rdfs:subClassOf>
   <owl:Class rdf:about="#PatientReferralServices"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Heart_volume_transducer">
 <rdfs:subClassOf>
   <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
     </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Heart"/>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
   <owl:Class rdf:about="#VolumeMeter"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ChangeInToOut">
 <rdfs:subClassOf rdf:resource="#ChangeService"/>
</owl:Class>
<owl:Class rdf:ID="RegisterToPhaseOfClinicalTrialForObservationReporting">
 <rdfs:subClassOf rdf:resource="#PhaseOfClinicalTrialRelatedServices"/>
</owl:Class>
<owl:Class rdf:about="#Blood">
 <rdfs:subClassOf rdf:resource="#Substance"/>
</owl:Class>
<owl:Class rdf:ID="RegisterToClinicalTrialForObservationReporting">
 <rdfs:subClassOf rdf:resource="#ObservationReportingOfClinicalTrialsServices"/>
</owl:Class>
<owl:Class rdf:about="#VolumeMeter">
 <rdfs:subClassOf rdf:resource="#Meter"/>
</owl:Class>
<owl:Class rdf:ID="Diathermy_device">
 <rdfs:subClassOf>
   <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Muscle"/>
     <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
   </owl:Restriction>
 </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#TemperatureGenerator"/>
</owl:Class>
<owl:Class rdf:about="#ResponseForClinicalObservationReportingService">
 <rdfs:subClassOf rdf:resource="#ClinicalObservationReportingServices"/>
</owl:Class>
<owl:Class rdf:about="#Urine">
```

```
<rdfs:subClassOf rdf:resource="#Substance"/>
</owl:Class>
<owl:Class rdf:ID="PatientReferralRequestServices">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#PatientReferralServices"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="PatientPathwayAddServices">
  <rdfs:subClassOf rdf:resource="#PatientPathwayServices"/>
</owl:Class>
<owl:Class rdf:ID="Lung_pressure_meter">
 <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#Lung"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#PressureMeter"/>
</owl:Class>
<owl:Class rdf:ID="SolicitedResponseForClinicalObservationReporting">
  <rdfs:subClassOf rdf:resource="#ResponseForClinicalObservationReportingService"/>
</owl:Class>
<owl:Class rdf:about="#ObservationReportingServices">
  <rdfs:subClassOf rdf:resource="#HealthcareServices"/>
</owl:Class>
<owl:Class rdf:about="#BodyPart">
 <rdfs:subClassOf rdf:resource="#Substance"/>
</owl:Class>
<owl:Class rdf:ID="Lung_rate_analyzer">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#Lung"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="RateAnalyzer"/>
  </rdfs:subClassOf>
</owl:Class>
```

```
<owl:Class rdf:ID="Urine_chemistry_transducer">
```

```
<rdfs:subClassOf>
   <owl:Restriction>
     <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Urine"/>
   </owl:Restriction>
 </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Class rdf:ID="ConcentrationMeter"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="DiagnosticOperationServices">
 <rdfs:subClassOf rdf:resource="#DiagnosticServices"/>
</owl:Class>
<owl:Class rdf:ID="Brain_temp._meter">
 <rdfs:subClassOf rdf:resource="#TemperatureMeter"/>
 <rdfs:subClassOf>
   <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Brain"/>
   </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Heart_output_analyzer">
 <rdfs:subClassOf>
   <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Blood"/>
     <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
     </owl:onProperty>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf rdf:resource="#TemperatureAnalyzer"/>
</owl:Class>
<owl:Class rdf:ID="Kidney">
 <rdfs:subClassOf rdf:resource="#BodyPart"/>
</owl:Class>
<owl:Class rdf:ID="Lung_pressure_analyzer">
 <rdfs:subClassOf>
   <owl:Restriction>
     <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
     </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#Lung"/>
```

```
</owl:allValuesFrom>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
   <owl:Class rdf:about="#PressureAnalyzer"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#SchedulingServices">
 <rdfs:subClassOf rdf:resource="#HealthcareServices"/>
</owl:Class>
<owl:Class rdf:about="#RateAnalyzer">
 <rdfs:subClassOf>
   <owl:Class rdf:about="#Analyzer"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#RequestForClinicalObservationReportingService">
 <rdfs:subClassOf rdf:resource="#ClinicalObservationReportingServices"/>
</owl:Class>
<owl:Class rdf:ID="Defibrillator">
 <rdfs:subClassOf>
   <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Heart"/>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf rdf:resource="#ElectricalPotentialGenerator"/>
</owl:Class>
<owl:Class rdf:ID="Catheter_tip_temp._probe">
 <rdfs:subClassOf>
   <owl:Restriction>
     <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Blood"/>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf rdf:resource="#TemperatureMeter"/>
</owl:Class>
<owl:Class rdf:ID="Wait">
 <rdfs:subClassOf rdf:resource="#PlacerServices"/>
</owl:Class>
<owl:Class rdf:ID="DisplayResponseForClinicalObservationReporting">
 <rdfs:subClassOf rdf:resource="#ResponseForClinicalObservationReportingService"/>
</owl:Class>
<owl:Class rdf:about="#Analyzer">
 <rdfs:subClassOf rdf:resource="#SensorServices"/>
```

```
</owl:Class>
<owl:Class rdf:about="#Multi-ParameterMeter">
 <rdfs:subClassOf rdf:resource="#Meter"/>
</owl:Class>
<owl:Class rdf:ID="Electrobraingraph">
 <rdfs:subClassOf>
   <owl:Restriction>
     <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Brain"/>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf rdf:resource="#ElectricalPotentialMeter"/>
</owl:Class>
<owl:Class rdf:ID="Transcutaneous_gas_meter">
 <rdfs:subClassOf>
   <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Multi-Gas"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
   <owl:Class rdf:about="#ConcentrationMeter"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="CompletePhaseOfClinicalTrialForObservationReporting">
 <rdfs:subClassOf rdf:resource="#PhaseOfClinicalTrialRelatedServices"/>
</owl:Class>
<owl:Class rdf:ID="HeartActivityAnalyzer">
 <rdfs:subClassOf>
   <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Heart"/>
     <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf rdf:resource="#ElectricalPotentialAnalyzer"/>
</owl:Class>
<owl:Class rdf:ID="DischargeCancel">
 <rdfs:subClassOf rdf:resource="#DischargeService"/>
</owl:Class>
<owl:Class rdf:ID="Lung_flow_analyzer">
 <rdfs:subClassOf>
   <owl:Restriction>
```

```
<owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#Lung"/>
      </owl:allValuesFrom>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf rdf:resource="#FlowAnalyzer"/>
</owl:Class>
<owl:Class rdf:ID="PushLabResultServices">
 <rdfs:subClassOf rdf:resource="#LabResultServices"/>
</owl:Class>
<owl:Class rdf:ID="PatientGoalResponseServices">
 <rdfs:subClassOf rdf:resource="#PatientGoalServices"/>
</owl:Class>
<owl:Class rdf:about="#ConcentrationMeter">
 <rdfs:subClassOf rdf:resource="#Meter"/>
</owl:Class>
<owl:Class rdf:ID="PatientGoalAddServices">
 <rdfs:subClassOf rdf:resource="#PatientGoalServices"/>
</owl:Class>
<owl:Class rdf:ID="Sp02Monitor">
 <rdfs:subClassOf>
   <owl:Restriction>
     <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
     </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Blood"/>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
   <owl:Class rdf:about="#ConcentrationAnalyzer"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Blood_flow_meter">
 <rdfs:subClassOf>
   <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Blood"/>
     <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf rdf:resource="#FlowMeter"/>
</owl:Class>
<owl:Class rdf:ID="PatientProblemQueryServices">
 <rdfs:subClassOf rdf:resource="#PatientProblemServices"/>
```

```
</owl:Class>
<owl:Class rdf:about="#PatientReferralServices">
 <rdfs:subClassOf rdf:resource="#HealthcareServices"/>
</owl:Class>
<owl:Class rdf:ID="MultiGasIdentifier">
 <rdfs:subClassOf>
   <owl:Restriction>
     <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Airway"/>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
   <owl:Class rdf:about="#ConcentrationAnalyzer"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#PressureAnalyzer">
 <rdfs:subClassOf rdf:resource="#Analyzer"/>
</owl:Class>
<owl:Class rdf:about="#ConcentrationAnalyzer">
 <rdfs:subClassOf rdf:resource="#Analyzer"/>
</owl:Class>
<owl:Class rdf:ID="ChangeOutToIn">
 <rdfs:subClassOf rdf:resource="#ChangeService"/>
</owl:Class>
<owl:Class rdf:ID="RequestForInsuranceInformationServices">
 <rdfs:subClassOf rdf:resource="#InsuranceInformationServices"/>
</owl:Class>
<owl:Class rdf:about="#Lung">
 <rdfs:subClassOf rdf:resource="#BodyPart"/>
</owl:Class>
<owl:Class rdf:ID="Blood_chemistry_meter">
 <rdfs:subClassOf>
   <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Blood"/>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf rdf:resource="#Multi-ParameterMeter"/>
</owl:Class>
<owl:Class rdf:ID="CoronaryAngiogram">
 <rdfs:subClassOf rdf:resource="#DiagnosticOperationServices"/>
</owl:Class>
<owl:Class rdf:ID="RequestReceiptOfPatientSelectionListServices">
 <rdfs:subClassOf rdf:resource="#PatientNameListServices"/>
```

```
</owl:Class>
 <owl:Class rdf:about="#PatientAdministrationServices">
   <rdfs:subClassOf rdf:resource="#HealthcareServices"/>
 </owl:Class>
 <owl:Class rdf:ID="Renal_function_calculator">
   <rdfs:subClassOf>
     <owl:Restriction>
        <owl:allValuesFrom rdf:resource="#Kidney"/>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
        </owl:onProperty>
     </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf rdf:resource="#MultiParameterCalculator"/>
 </owl:Class>
 <owl:Class rdf:ID="Air_pressure_meter">
   <rdfs:subClassOf>
     <owl:Restriction>
        <owl:allValuesFrom rdf:resource="#Airway"/>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
        </owl:onProperty>
     </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf rdf:resource="#PressureMeter"/>
 </owl:Class>
 <owl:Class rdf:ID="Blood_pressure_strain_gauge">
   <rdfs:subClassOf>
     <owl:Restriction>
        <owl:allValuesFrom rdf:resource="#Blood"/>
       <owl:onProperty>
          <owl:ObjectProperty rdf:about="#analyzesSubstance"/>
       </owl:onProperty>
     </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf rdf:resource="#PressureMeter"/>
 </owl:Class>
 <owl:ObjectProperty rdf:about="#analyzesSubstance">
   <rdfs:domain rdf:resource="#SensorServices"/>
   <rdfs:range rdf:resource="#Substance"/>
 </owl:ObjectProperty>
 <owl:FunctionalProperty rdf:ID="code">
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
   <rdfs:domain rdf:resource="#SensorServices"/>
   <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
 </owl:FunctionalProperty>
</rdf:RDF>
```

```
<!-- Created with Protege (with OWL Plugin 3.2.1, Build 365) http://protege.stanford.edu -->
```

APPENDIX C

AN EXAMPLE CDA USED IN THE PILOT APPLICATION

<?xml version="1.0" encoding="UTF-8"?>

```
<ClinicalDocument xmlns="urn:hl7-org:v3"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<id root="111" extension="01" assigningAuthorityName="SAPHIRE"/>
<code code="34133-9" codeSystem="2.16.840.1.113883.6.1" displayName='SUMMARIZATION OF EPISODE NOTE'/>
<effectiveTime value="20060413"/> <!--yyyymmdd -->
<author>
<time value="20060413"/><!--yyyymmdd -->
<assignedAuthor>
<assignedAuthoringDevice>
<code code="111" codeSystem="222" codeSystemVersion="1"></code>
</assignedAuthoringDevice>
</assignedAuthor>
</author>
<custodian>
<assignedCustodian>
<representedCustodianOrganization>
<name>Dr Ana Fruntela</name>
</representedCustodianOrganization>
</assignedCustodian>
</custodian>
<!--
********
Demographics
**********
-->
<recordTarget>
<patient>
<patientPatient>
<administrativeGenderCode code="F" codeSystem="2.16.840.1.113883.5.1"></administrativeGenderCode>
```


<birthTime value="19790623"/> <maritalStatusCode code="S" codeSystem="2.16.840.1.113883.5.2"></maritalStatusCode> </patientPatient> </patient> </recordTarget> <component> <structuredBody> <!--******** Active Problems section ******** --> <component> <section> <code code="11450-4" codeSystem="2.16.840.1.113883.6.1"></code> <title>Active Problems</title> <entry> <observation classCode="OBS" moodCode="EVN"> <code code="R07.4" codeSystem="2.16.840.1.113883.6.3"/> <!-- Chest pain, unspecified; ICD-10 --> <effectiveTime> <low value="20060413100000"/> <high value="20060413103500"/> </effectiveTime> <value>46666</value> </observation> </entry> <entry> <observation classCode="OBS" moodCode="EVN"> <code code="134434004" codeSystem="2.16.840.1.113883.6.96"/> <!-- NYHA Class; SNOMED CT --> <value> 4 </value> </observation> </entry> </section> </component> <! __ Past Medical History section ******** --> <component> <section> <code code="11348-0" codeSystem="2.16.840.1.113883.6.1"></code> <title>PastMedicalHistory</title> <entry> <observation classCode="OBS" moodCode="EVN"> <code code="I10" codeSystem="2.16.840.1.113883.6.3"></code> <!--Essential (primary) hypertension; ICD-10 -->

</observation> </entry> <entry> <observation classCode="OBS" moodCode="EVN"> <code code="I21" codeSystem="2.16.840.1.113883.6.3"></code> <!--Acute myocardial infarction; ICD-10 --> <value>2 </value> </observation> </entry> <entry> <procedure classCode="PROC" moodCode="EVN"> <code code="297183000" codeSystem="2.16.840.1.113883.6.96"></code> <!--myocardial revascularization; SNOMED CT --> </procedure> </entry> <entry> <observation classCode="OBS" moodCode="EVN"> <code code="Z95.5" codeSystem="2.16.840.1.113883.6.3"></code> <!--Presence of coronary angioplasty implant and graft; ICD-10 --> <value>true</value> </observation> </entry> <entry> <observation classCode="OBS" moodCode="EVN"> <code code="I20.0" codeSystem="2.16.840.1.113883.6.3"></code> <!--Angina pectoris; ICD-10 --> <value>true</value> </observation> </entry> <entry> <observation classCode="OBS" moodCode="EVN"> <code code="I50.0" codeSystem="2.16.840.1.113883.6.3"></code> <!--Heart failure,ICD10 --> <value>true</value> </observation> </entry> <entry> <observation classCode="OBS" moodCode="EVN"> <code code="I05.0" codeSystem="2.16.840.1.113883.6.3"></code> <!--Rheumatic mitral valve diseases, ICD10 --> <value>1</value> </observation> </entry> <entry> <observation classCode="OBS" moodCode="EVN"> <code code="F41.9" codeSystem="2.16.840.1.113883.6.3"></code> <!--Anxiety disorder, unspecified, ICD10 --> <value>true</value> </observation> <!-- pulmunary, bleeding.... to be added --> </entry> <entry> <observation classCode="OBS" moodCode="EVN"> <code code="271649006" codeSystem="2.16.840.1.113883.6.96"></code> <!--BP, Snomed CT -->

<value>100</value> </observation> </entry> <entry> <observation classCode="OBS" moodCode="EVN"> <code code="J45" codeSystem="2.16.840.1.113883.6.3"/> <!-- Asthma; ICD10 --> <value> true </value> </observation> </entry> <entry> <observation classCode="OBS" moodCode="EVN"> <code code="4386001" codeSystem="2.16.840.1.113883.6.96"/> <!-- Bronchial Spasm; SNOMED CT --> <value> true </value> </observation> </entry> </section> </component> <!--******** Medications section *********** --> <component> <section> <code code="19789-7" codeSystem="2.16.840.1.113883.6.1"></code> <title>Medications</title> <entry> <substanceAdministration classCode="ActClass" moodCode="EVN"> <consumable> <manufacturedProduct> <manufacturedLabeledDrug> <code code="33252009" codeSystem="2.16.840.1.113883.6.96"/> <!--Beta blocker. SNOMED CT --> </manufacturedLabeledDrug> </manufacturedProduct> </consumable> <statusCode code="active"/> </substanceAdministration> </entry> <entry> <substanceAdministration classCode="ActClass" moodCode="EVN"> <consumable> <manufacturedProduct> <manufacturedLabeledDrug> <code code="387458008" codeSystem="2.16.840.1.113883.6.96"/> <!--Asprin. SNOMED CT --> </manufacturedLabeledDrug> </manufacturedProduct> </consumable>

```
<statusCode code="completed"/>
</substanceAdministration>
</entry>
<entry>
<substanceAdministration classCode="ActClass" moodCode="EVN">
<consumable>
<manufacturedProduct>
<manufacturedLabeledDrug>
<code code="108979001" codeSystem="2.16.840.1.113883.6.96"/> <!--Clopidogrel. SNOMED CT -->
</manufacturedLabeledDrug>
</manufacturedProduct>
</consumable>
<statusCode code="suspended"/>
</substanceAdministration>
</entry>
</section>
</component>
<! __
********
Allergies and Adverse Reactions section
********
-->
<component>
<section>
<code code="10123-x" codeSystem="2.16.840.1.113883.6.1"></code>
<title>Allergies and Adverse Reactions</title>
<entry>
<observation classCode='COND' moodCode='EVN' negationInd='false'>
<statusCode code='active'/>
<effectiveTime> <low value='20041100000000'/> </effectiveTime>
<value xsi:type='CD' code='387458008' displayName='asprin'
codeSystem='2.16.840.1.113883.6.96' codeSystemName='SNOMED CT'>
<originalText><reference value='#allergen-1'/></originalText>
</value>
<entryRelationship typeCode='SPRT'>
<observation classCode='OBS' moodCode='EVN'>
<code code='RXNASSESS'/>
<statusCode code='completed'/>
<!-- reactions -->
<entryRelationship typeCode='SUBJ'>
<observation classCode='OBS' moodCode='EVN'>
<id root='369bb3eb'/>
<statusCode code='completed'/>
<value xsi:type='CD' code='247472004' displayName='weal'
codeSystem='2.16.840.1.113883.6.96' codeSystemName='SNOMED CT'>
<originalText><reference value='#reaction-1'/></originalText>
</value>
</observation>
```

</entryRelationship> </observation> </entryRelationship> </observation> </entry> </section> </component> <!--********* Resting ECG Features ********* --> <component> <section> <code code="34752-6" codeSystem="2.16.840.1.113883.6.1"></code> <title> EVALUATION AND MANAGEMENT NOTE, Cardiology</title> <entry> <observation classCode="OBS" moodCode="EVN"> <code code="9874-9" codeSystem="2.16.840.1.113883.6.1"></code> <!--ECG Rythm Segment, LOINC --> <value>2</value> </observation> </entry> <entry> <observation classCode="OBS" moodCode="EVN"> <code code="250908004" codeSystem="2.16.840.1.113883.6.96"></code> <!--Left ventricular ejection fraction Snomed --> <value> 38</value> </observation> </entry> </section> </component> </structuredBody> </component> </ClinicalDocument>

VITA

PERSONAL INFORMATION

Surname, Name:	Laleci Ertürkmen, Gökçe Banu
Nationality:	Turkish (TC)
Date and Place of Birth:	23 June 1979, Ankara
Marital Status:	Married
Phone:	$+90 \ 312 \ 210 \ 2076$
e-Mail:	banu@srdc.metu.edu.tr

EDUCATION

Degree	Institution	Year of Graduation
MS	METU-Computer Engineering	2003
BS	METU-Computer Engineering	2001
High School	Yıldırım Bayezit Anatolian High School, Ankara	1997

WORK EXPERIENCE

Year	Place	Enrollment
2008-present	SRDC, Reseach, Development and Consultancy Ltd. Şti	Researcher
2001-2008	Software Research and Development Center, METU	Researcher
2000-2001	Software Research and Development Center, METU	Part Time
		Software Developer

FOREIGN LANGUAGES

Advanced English

PUBLICATIONS

- Laleci G., Dogac A., "A Semantically Enriched Clinical Guideline Model Enabling Deployment in Heterogeneous Healthcare Environments", submitted for publication to IEEE Transactions on Information Technology in Biomedicine
- Laleci G., Dogac A., Olduz M., Tasyurt I., Yuksel M., Okcan A., "SAPHIRE: A Multi-Agent System for Remote Healthcare Monitoring through Computerized Clinical Guidelines", Book Chapter in "Agent Technology and e-Health, Whitestein Series in Software Agent Technologies and Autonomic Computing", 2008
- Nee O., Hein A., Gorath T., Hulsmann N., Laleci G., Yuksel M., Olduz M., Tasyurt I., Orhan U., Dogac A., Fruntelata A., Ghiorghe S., Ludwig R., "SAPHIRE: Intelligent Healthcare Monitoring based on Semantic Interoperability Platform - Pilot Applications", Communications, IET, Special Issue on Telemedicine and e-Health Communication Systems, Volume 2, Issue 2, February 2008 Page(s):192 - 201
- Dogac A., Namli T., Okcan A., Laleci G., Kabak Y., Eichelberg M., "Key Issues of Technical Interoperability Solutions in eHealth and the RIDE Project", eChallenges Conference, The Hague, The Netherlands, October 2007
- Della Valle E., Cerizza D., Celino I., Dogac A., Laleci G., Kabak Y., Okcan A., Gulderen O., Namli T., Bicer V., "An eHealth Case Study", Book Chapter in "Semantic Web Services: Concepts, Technologies, and Applications", Studer, Rudi; Grimm, Stephan; Abecker, Andreas (Eds.), 2007, Approx. 15 p., 100 illus., Hardcover, ISBN: 978-3-540-70893-3, Due: April 5, 2007, Springer.
- Dogac A., Laleci G., Aden T., Eichelberg M., "Enhancing IHE XDS for Federated Clinical Affinity Domain Support", IEEE Transactions on Information Technology in Biomedicine, Vol. 11, No. 2, March 2007, pp. 213-221.
- Eichelberg M., Aden T., Riesmeier J., Dogac A., Laleci G., "ELECTRONIC HEALTH RECORD STANDARDS - A BRIEF OVERVIEW", 4th International Conference on Information and Communications Technology (ICICT 2006), Cairo, Egypt, December 2006, pp. 7-19.
- Hein A., Nee O., Willemsen D., Scheffold T., Dogac A., Laleci G., "SAPHIRE Intelligent Healthcare Monitoring based on Semantic Interoperability Platform - The Homecare Scenario", European Conference on eHealth (ECEH06), Fribourg, Switzerland, October 2006, pp 179-190 (2006).

- Laleci G., Dogac A., Akcay B., Olduz M., Yuksel M., Orhan U., Tasyurt I., Sen T., Kabak Y., Namli T., Gulderen O., Okcan A., " SAPHIRE: A semantic Web service based Clinical guideline deployment infrastructure exploiting IHE XDS", eChallenges Conference, Barcelona, Spain, October 2006.
- Dogac A., Laleci G., Kirbas S., Kabak Y., Sinir S., Yildiz A., Gurcan Y., "Artemis: Deploying Semantically Enriched Web Services in the Healthcare Domain", Information Systems Journal (Elsevier), Volume 31, Issues 4-5, June-July 2006, pp.321-339
- Dogac A., Laleci G., Kabak Y., Unal S., Beale T., Heard S., Elkin P., Najmi F., Mattocks C., Webber D., Kernberg M., "Exploiting ebXML Registry Semantic Constructs for Handling Archetype Metadata in Healthcare Informatics", International Journal of Metadata, Semantics and Ontologies, Volume 1, No. 1, 2006.
- Della Valle E., Cerizza D., Bicer V., Kabak Y., Laleci G., Lausen H., "The Need for Semantic Web Service in the eHealth", W3C workshop on Frameworks for Semantics in Web Services, 2005.
- Eichelberg M., Aden T., Riesmeier J., Dogac A., Laleci G., "A Survey and Analysis of Electronic Healthcare Record Standards", ACM Computing Surveys, Vol. 37, No:4, December 2005.
- Bicer V., Kilic O., Dogac A., Laleci G., "Archetype-based Semantic Interoperability of Web Service Messages in the Healthcare Domain", Int'l Journal on Semantic Web and Information Systems, Vol. 1, No.4, October 2005, pp. 1-22.
- Bicer V., Laleci G., Dogac A., Kabak Y., "Providing Semantic Interoperability in the Healthcare Domain through Ontology Mapping", eChallenges 2005, October 2005, Ljubljana, Slovenia.
- Bicer V., Laleci G., Dogac A., Kabak Y., "Artemis Message Exchange Framework: Semantic Interoperability of Exchanged Messages in the Healthcare Domain", ACM Sigmod Record, Vol. 34, No. 3, September 2005
- Dogac A., Kabak Y., Laleci G. C. Mattocks, F. Najmi, J. Pollock, "Enhancing ebXML Registries to Make them OWL Aware", Distributed and Parallel Databases Journal, Springer-Verlag, Vol. 18, No. 1, July 2005, pp. 9-36

- Dogac A., Kabak Y., Laleci G., Sinir S., Yildiz A., Tumer A., "SATINE Project: Exploiting Web Services in the Travel Industry", eChallenges 2004 (e-2004), 27 - 29 October 2004, Vienna, Austria.
- Dogac A., Kabak Y., Laleci G., Sinir S., Yildiz A., Kirbas S., Gurcan Y., "Semantically Enriched Web Services for Travel Industry", ACM Sigmod Record, Vol. 33, No. 3, September 2004.
- 20. Laleci G., Kabak Y., Dogac A., Cingil I., Kirbas S., Yildiz A., Sinir S., Ozdikis O., Ozturk O., " A Platform for Agent Behavior Design and Multi Agent Orchestration", Agent-Oriented Software Engineering (AOSE-2004) Workshop, the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004), New York City, New York - July 19, 2004
- Dogac A., Kabak Y., Laleci G., "Enriching ebXML Registries with OWL Ontologies for Efficient Service Discovery", 14th International Workshop on Research Issues on Data Engineering, Boston, USA, March 28-29, 2004.
- 22. Dogac A., Laleci G., Kabak Y., "Context Frameworks for Ambient Intelligence", eChallenges 2003, October 2003, Bologna, Italy.
- Athanasiadis I. N., Mitkas P. A., Laleci G., Kabak Y., "Embedding Data-driven Decision Strategies on Software Agents", 10th International Conference on Concurrent Engineering, Madeira Island - Portugal, 26-30 July, 2003.
- Dogac A., Kabak Y., Laleci G., " A Semantic-Based Web Service Composition Facility for ebXML Registries", 9th International Conference of Concurrent Enterprising, Espoo, Finland, June 2003.
- 25. Dogac A., Tambag Y., Pektas S., Laleci G., Kurt G., Toprak S., Kabak Y., "An ebXML Infrastructure Implementation through UDDI Registries and RosettaNet PIPs", ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA, June 2002.
- Dogac A., Laleci G., Kabak Y., Cingil I., "Exploiting Web Service Semantics: Taxonomies vs. Ontologies", IEEE Data Engineering Bulletin, Vol. 25, No. 4, December 2002,

- 27. Mitkas P., Symeonidis A., Kechagias D., Athanasiadis I., Laleci G., Kurt G., Kabak Y., Acar A., Dogac A., " An Agent Framework for Dynamic Agent Retraining: Agent Academy", e2002: European Commission's e-Business and e-Work Annual Conference, Czech Republic, October 2002.
- Dogac A., Cingil I., Laleci G., Kabak Y., "Improving the Functionality of UDDI Registries through Web Service Semantics", 3rd VLDB Workshop on Technologies for E-Services (TES-02), Hong Kong, China, August 23-24, 2002
- Dogac A., Laleci G., Kurt G., Kabak Y., Acar A., "A Platform for Semantically Enriched Mobile Services", in Proc. of the First International Conference on Mobile Business, Athens, Greece, July 2002.
- 30. Dogac A., Laleci G., Kurt G., Kabak Y., Acar A., " An ebXML Infrastructure Implementation through UDDI Registries and RosettaNet PIPs", ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA, June 2002

Number of Citations Received: 13 (Based on ISI Web of Science)