Resolving semantic conflicts through ontological layering

Pavandeep Kataria

School of Electronics and Computer Science

RESOLVING SEMANTIC CONFLICTS THROUGH ONTOLOGICAL LAYERING

PAVANDEEP KATARIA

A thesis submitted in partial fulfilment of
the requirements of the University of Westminster
for the degree of Doctor of Philosophy

July 2011

Dedication

The greatest pleasure for me is writing dedications and acknowledgements for this thesis. I would like to dedicate this thesis to my family, especially to my husband Mr Haresh Ahuja, my mother Ms. Krishna Devi, Papa Tom, Mama Renu and to my younger siblings, Sonia and Suraj. You gave me all the strength and love that brought me so far. It's simple; without you, I would be nothing.

Haresh, you are my soul mate and I am blessed to have you in my life. You have been my one constant rock and I deeply value our relationship and will treasure it forever. Mum, you are my best friend and I thank you for always being there whenever I have needed you. I will forever admire your strength as a woman and cherish the sacrifices you have made over the years. Papa and Mama, your support and patience is more than any daughter-in-law can ask for, I will be eternally grateful. Sonia, your positive words and confidence in me has given me the strength to complete this thesis. Suraj, you really are my little ray of "sunshine". You have constantly been there for me, and your hugs have never failed to make me feel better. Sonia and Suraj I love you with all my heart and I will always be there for you whenever you need me.

Acknowledgement

First of all, my deepest thanks goes out to my supervisor and friend Dr. Radmila Juric, who gave me this unique opportunity to write this thesis in her research group. As my academic supervisor, Dr Juric has offered me a lot, from determining the thesis topic, numerous constructive and intellectually stimulating discussions, to the careful examination and correction of every paper and thesis draft. Her priceless support and invaluable experience is highly appreciated and she has constantly been a role model in various aspects. As a respectable lecturer she has set an example with her devotion to her career and passion for research. As an individual, her energy and motivation is contagious and have been contributing factors to the completion of this thesis.Her integrity and boldness are worthy of admiration, and for me she is one of the most competent people I have met in my life, both on a personal level as well as professional level. I am deeply grateful and proud to be a student of hers.

Sincere thanks goes to Dr. Shamimabi Paurobally for agreeing to be the external examiner. Her detailed comments have been insightful and I would like to thank her for all the efforts she has made for the review of my thesis. Special thanks are extended to the financial support I received from Dr. Andrzej Tarczynski. Without this help my PhD thesis completion would not have been possible.

A very big thank you goes to all my research colleagues; Nigel Koay, Preya Syal and Reza Shojanoorie for their encouragement. Nigel and Preya your friendship, support, and encouragement will always be remembered.

I would also like to express my gratitude to the MSc. students I had the opportunity to work with, especially Anirudh Thotapalli, Tomas Jakimavicius, Emilian Adrian Filip, Raja Saadi and Nemanja Granatir. Your motivation and commitment to the task were invaluable for the research reported in this thesis. Furthermore, I'd like to thank all of my co-authors for their valuable contribution to the publications that this thesis uses. Sincere thanks goes to Sukanta Ganguly from Texas Tech University for his collaboration on a joint journal paper.

Finally, I give my heartfelt thanks to my husband, for his unfailing love, patience and proof reading. It is no exaggeration that the completion of this thesis is due largely to his understanding and efforts.

Abstract

We examine the problem of semantic interoperability in modern software systems, which exhibit pervasiveness, a range of heterogeneities and in particular, semantic heterogeneity of data models which are built upon ubiquitous data repositories. We investigate whether we can build ontologies upon heterogeneous data repositories in order to resolve semantic conflicts in them, and achieve their semantic interoperability. We propose a layered software architecture, which accommodates in its core, ontological layering, resulting in a **G**eneric **o**ntology for **C**ontext aware, **I**nteroperable and **D**ata sharing (**Go-CID**) software applications. The software architecture supports retrievals from various data repositories and resolves semantic conflicts which arise from heterogeneities inherent in them. It allows extendibility of heterogeneous data repositories through ontological layering, whilst preserving the autonomy of their individual elements.

Our specific ontological layering for interoperable data repositories is based on clearly defined reasoning mechanisms in order to perform ontology mappings. The reasoning mechanisms depend on the user's involvments in retrievals of and types of semantic conflicts, which we have to resolve after identifying semantically related data. Ontologies are described in terms of ontological concepts and their semantic roles that make the types of semantic conflicts explicit. We contextualise semantically related data through our own categorisation of semantic conflicts and their degrees of similarities.

Our software architecture has been tested through a case study of retrievals of semantically related data across repositories in pervasive healthcare and deployed with Semantic Web technology. The extensions to the research results include the applicability of our ontological layering and reasoning mechanisms in various problem domains and in environments where we need to (i) establish if and when we have overlapping "semantics", and (ii) infer/assert a correct set of "semantics" which can support any decision making in such domains.

Contents

List of Tables

List of Figures

List of Abbreviations

ADDED_VAL_ONT	Added Values Ontology
AI	Artificial Intelligence
CWA	Closed World Assumption
DB	Database
DAML	Defense Advanced Research Projects Agent Markup Language
DL	Descriptions Logic
eCTD	electronic Common Technical Document
ENV_ONT	Environment Ontology
ESM	Extending relational database Schema Model (ESM)
Go-CID	Generic ontology for Context aware, Interoperable and Data sharing
GUI	Graphical User Interface
HTML	Hyper Text Mark-up Language
KM	Knowledge Management
MA	Marketing Authorisations
OIL	Ontology Inference Layer
OWA	Open World Assumption
PCE	Pervasive Computational Environment
RDBO	Relational DataBase Ontology
RDF	Resource Description Framework language
RIF	Rule Interchange Format
SCO	Semantic Conflict Ontology
SCM	Semantic Conflict representation Model
SCROL	Semantic Conflict Resolution Ontology
SWRL	Semantic Web Rule Language
SE	Software Engineering
SPARQL	Protocol and RDF Query Language

SQL	Structure Query Language
UoD	Universe of Disclosure
URI	Uniform Resource Identifiers
USER_INP_ONT	User Input Ontology
OWL	Web Ontology Language
WWW	World Wide Web

Glossary

Alignment is the process of establishing a semantic relation between two semantically related ontological individuals, which in turn either (i) transfers duplicates of ontological individuals to a NEW ontological class, or (ii) infers a new ontological individual into a NEW ontological class.

Assertion is the selection of an ontological individual or creation of an axiom according to a particular context.

Atoms are ontological classes, properties, individuals or datatype property literal values declared as variables in a SWRL rule.

Axioms are constituent parts of OWL abstract syntax, which may include classes, individual, object/datatype property and literal values.

Classification of ontological concepts is the result of running SWRL rules upon them. Ontological concepts are being 'classified' in terms of having either their different characteristics or ontological individuals being aligned, integrated or merged into a different ontological class of types: NEW, CRADLE, COMMON or DISPLAY.

Class membership concerns ontological individuals that belong to a class because they meet the set criteria imposed on the class. We can also claim that class membership may be the result of classification of ontological concepts.

COMMON ontological class is an ontological class which initially, at the time of it's creation does not contain any ontological individuals, but may accommodate in future semantically equivalent ontological individuals that are asserted in this class because they have a semantic correspondence between them, as a consequence of ontology integration.

Comparison is a process which compares a pair of ontological individuals against a set of OWL restrictions/conditions.

Context is the result of grouping semantically related ontological individuals according to a user's request for retrievals, i.e. choice of repository or information type. We can also claim that a certain context pinpoints which ontological concepts might be semantically related.

Correlation is established between semantically equivalent ontological individuals that have the same meaning. A correlation can only be established through running SWRL rules in the Post-High-Level reasoning mechanism.

CRADLE ontological class is an ontological class which initially, at the time of it's creation does not contain any ontological individuals, but may accommodate in future semantically similar ontological individuals that are either (i) transferred in this class because they have a semantic relation between them, or (ii) inferred in this class as a consequence of ontology alignment.

Datatype property refers to a 'binary relation' between an ontological individual and a data literal value. Binary relations use 'domain' and range 'values' to specify a relationship between ontological individuals and a data literal. In a datatype property 'values' refer to the names of the ontological individuals and data literals of type XML schema datatype value or RDF literal.

DISPLAY ontological class is an ontological class which initially, at the time of it's creation does not contain any ontological individuals, but may accommodate in future ontological individuals that model the same 'real world' meaning; that are relocated as a consequence of ontology merge.

Domain and Range values are either (i) ontological classes that represent a set of ontological individuals or (ii) a set of literal values associated to ontological individuals.

Duplicate of an ontological individual is a copy of the ontological individual that has been transferred from the original class to a different ontological class of types CRADLE, COMMON or DISPLAY.

Grouping of ontological individuals is the process of acknowledging that these ontological individuals are semantically related. Grouping ensures that semantically related ontological individuals are moved into a NEW ontological class because they satisfy a set criteria for being a member of a NEW class.

High-Level reasoning mechanism is the process of executing ontological integration of semantically similar ontological individuals. High-Level reasoning mechanism ensures that semantically similar ontological individuals have a semantic correspondence between them and are transferred into an ontological class of type COMMON.

Inference is the creation of a "new" ontological concept.

Integration is the process of creating a semantic correspondence, as a consequence of a 'link' between two or more semantically related ontological individuals, and asserting ontological individuals into an ontological class of type COMMON.

Link is the establishment of a semantic correspondence between semantically similar ontological individuals. A link can only be established through running SWRL rules in the High-Level reasoning mechanism which may include the comparison between semantically similar ontological individuals.

Low-Level reasoning mechanism is the process of executing ontological alignment of semantically related ontological individuals. Low-Level reasoning mechanism ensures that semantically related ontological individuals have a semantic relation between them and are transferred into an ontological class of type CRADLE.

Match is the establishment of a semantic relation between semantically related ontological individuals. A match can only be established through running SWRL rules in the Low-Level reasoning mechanism which may include the comparison between semantically related ontological individuals.

Merge is the process of creating a semantic correlation between two or more semantically equivalent ontological individuals, and asserting ontological individuals into an ontological class of type DISPLAY.

Movement of ontological individuals is related to their groupings. It means that all ontological individuals from one ontological class are moved to a NEW ontological class because they all satisfy the set criteria for grouping. The criteria for grouping indicate that these ontological individuals are semantically related.

Post-High-Level reasoning mechanism is the process of executing ontological merge of semantically equivalent ontological individuals. Post-High-Level reasoning mechanism ensures that semantically equivalent ontological individuals have the same meaning in 'real world' concepts they model.

Named modeling concepts refer to modeling concepts that describe the view and interpretation a particular domain of discourse of a given application domain.

NEW ontological class is an ontological class which initially does not contain any ontological individuals, but may accommodate in future semantically related ontological individuals that generate semantic conflicts as a consequence of their grouping in a particular context.

NEW ontological individual is an ontological individual which did not initially exist as an instance in an ontological class, but is inferred as a result of a comparison between ontological individuals based on OWL conditions.

Object property refers to a 'binary relation' between ontological individuals. 'Binary relations' use 'domain' and range 'values' to specify a relationship between ontological individuals. In an object property 'values' refer to the names of ontological classes, in which the ontological individuals belong.

Ontological Class is a representation of a domain concept, which could be placed at the roots of various taxonomical hierarchies. An ontological class may or may not contain instances, i.e. ontological individuals. Ontological classes, which do not contain ontological individuals may be of class types: NEW, CRADLE, COMMON or DISPLAY.

Ontological concept is an ontology description, where an ontology description is an ontological class, ontological individual, or ontological property (relationship).

Ontological individuals refer to 'instances' of a particular class, i.e. they are members or the extension of a particular class (OWL).

Ontological property refers to 'binary relations' between ontological individuals. Ontological properties can be in the form of 'object' or 'datatype' properties. 'Binary relations' use 'domain' and range 'values' to specify a relationship between ontological individuals. 'Values' refer to ontological individuals or data literals of type XML schema datatype value or RDF literal.

Ontological relationship is a specification of a 'domain' and 'range' value between two ontological classes using an object property.

Original ontological individual is an ontological individual that is in its class of origin.

OWL Abstract Syntax is a sequence of axioms, facts, imports, and annotations.

OWL conditions are a set of 'boolean combinations**'** attached to an object property/datatype property that exactly express the criteria for ontological individuals to secure their membership into an ontological class of type CRADLE and COMMON.

OWL file is a flat file containing either an ontology (i.e. all ontological concepts, SWRL rules or both ontology and SWRL rules created in OWL/SWRL.

OWL restrictions are a set of additional ontological descriptions attached to an object property/datatype property that exactly express the criteria for ontological individuals to secure their membership into an ontological class of type NEW.

Rule chaining is the mechanism of running a new SWRL rule on the result set of reasoning done with an old (previously run) SWRL rule. We use moved/inferred/transferred ontological individuals/axioms as a result set created after running an old SWRL rule to become one or more atoms in the antecedent (body) and consequent (head) of a new SWRL rule.

Semantic conflicts are the differences in named concepts as a consequence of 'semantic heterogeneities' between named modeling concepts because of either: (i) the *interpretation* of related named concepts in respect to their meaning in a given context, (ii) the *intended use* of related named concepts within a given context or (iii) the way we have *modelled* related named concepts in a universe of disclosure. Semantic conflicts may differ in terms of "modeling the meaning behind the same concept". This can happen at many levels: from meta-data or data level to technology specific and model/view specifications.

Semantic correspondence exists between one or more ontological individuals in the COMMON class. This means that these ontological individuals have exhibit similarities to each other before they have been asserted (integrated) into the COMMON class. The level of their similarity, prior to their integration has been established as "semantic equivalence".

Semantically equivalent ontological concepts are concepts which have the exact same meaning, i.e. these concepts are either result of (i) resolved semantic conflicts or (ii) bear no semantic conflicts initially between them.

Semantically related concepts are concepts which have a semantic relation between them, and subsequently generate a number of semantic conflicts.

Semantic relation exists between one or more ontological individuals in the CRADLE class. This means that these ontological individuals exhibit overlapping semantics to each other before they have been

inferred (alignment) into the CRADLE class. These individuals, prior to their alignment, have been grouped into "semantically related" concepts (in this case, concepts are classes).

Semantically similar concepts are concepts which have semantic correspondence defined between them. In other words these ontological individuals share the same semantics.

Set criteria are a set of OWL restrictions/conditions upon object properties or datatype properties between ontological classes and their individuals.

Strength of a match is the degree of similarity between overlapping ontological concepts.

SWRL rule is the conjunction of atoms. Atoms are deifned as classes, individual properties, data valued properties, individuals, data ranges, and built-in functions. The conjunctions of atoms are grouped into an antecedent (the body), and a consequent part (head) in the SWRL syntax.

Transferring of ontological individuals is COPYING of ontological individuals ("originals") and creating DUPLICATES of these "originals" in order to move them into a different ontological class of types: CRADLE, COMMON or DISPLAY.

Declaration

The following list of published papers has resulted from this research:

2011

PATADIA, R.; KATARIA, P.; JURIC, R. (2011) "Building Semantic Software Applications upon Ontological Environments", *To appear in the proceedings of the 16th International Conference on Transformative Science, Engineering, and Business Innovation (SDPS 2011), (Jeju Island, South Korea, June 12-16).*

SUH, S.; AJIT, S.; SADANANDAN NAIR, J.; KATARIA, P.; JURIC, R. (2011) "Smart Preventive Healthcare System using Ontological Reasoning", *To appear in the proceedings of the 16th International Conference on Transformative Science, Engineering, and Business Innovation (SDPS 2011), (Jeju Island, South Korea, June 12-16).*

KATARIA, P.; JURIC, R. (2009) "Sharing Healthcare Data by Manipulating Ontological Individuals", to appear in a chapter of the working title Advances in Bio-medical Engineering, published by Springer Verlag as hard copy with an ISBN number.

SAAIDI, S.R.; KATARIA, P.; JURIC, R. (2009), "Semantic Management of the Submission Process for Medicinal Products Authorisation", to appear in a chapter of the working title Advances in Bio-medical Engineering, published by Springer Verlag as hard copy with an ISBN number.

KATARIA, P.; JURIC, R. (2010) "Creating Semantics from User Inputs through Ontological Reasoning", In: *Proceedings of the 15th International Conference on System Design and Process Science (SDPS 2010), (Texas, Dallas, US, June 6-11),* CD-ROM.

2010

KOAY, N.; KATARIA, P.; JURIC, R. (2010) "Semantic Management of Non-Functional Requirements in e-Health Systems", Telemedicine and e-Health Journal, 16(4), pp. 461-471.

KATARIA, P.; JURIC, R. (2010) "Creating Semantics from User Inputs through Ontological Reasoning", In: *Proceedings of the 15th International Conference on System Design and Process Science (SDPS 2010), (Texas, Dallas, US, June 6-11),* CD-ROM.

SYAL, P.; KATARIA, P.; JURIC, R.; ROBBINS, D.; TANIK, M.M.; HLUPIC-VIDJAK, V. (2010) "The Virtual Learning Environment Ontology for Smart Classrooms", In: *Proceedings of the 15th International Conference on System Design and Process Science (SDPS 2010), (Dallas, US, June 6-11),* CD-ROM.

KATARIA, P.; JURIC, R. (2010) "Automated Reasoning in Resolving Semantic Conflicts across Heterogeneous Repositories", In: *Proceedings of the 17th Automated Reasoning Workshop - – Bridging the gap between theory and practice, (University of Westminster, Harrow, Middlesex, UK, March 30-31),* available at http://www2.wmin.ac.uk/bolotoa/ARW/arw-2010.html, [accessed September 2010].

KATARIA, P.; JURIC, R. (2010) "Sharing Healthcare Data through Ontological Layering" In: *Proceedings of the 43rd Annual Hawaii International Conference on System Sciences (HICSS 43), (Kauai, Hawaii, January 5-8),* pp. 1-10.

2009

GANGULY, S.; KATARIA, P.; JURIC, R.; ERTAS, A.; TANIK, M. M. (2009) "Sharing Information and Data across Heterogeneous e-Health Systems", Tele-Medicine and e-Health Journal, 15(5), pp. 454-464.

KATARIA, P.; JURIC, R. (2009) "Sharing Healthcare Data by Manipulating Ontological Individuals", In: *Proceedings of the 12th International Conference System Design and Process Science (SDPS 2009), (Montgomery, Alabama, US, November 1-5)*, CD-ROM.

SAAIDI, S.R.; KATARIA, P.; JURIC, R. (2009), "Semantic Management of the Submission Process for Medicinal Products Authorisation", In: *Proceedings of the 12th International Conference on Integrated Design and Process Technology (SDPS 2009), (Alabama Montgomery, USA, November 1-5)*, CD-ROM.

JAKIMAVICIUS, T.R.; KATARIA, P.; JURIC, R. (2009) "Semantic Support for Dynamic Changes in Enterprise Business Models", In: *Proceedings of the 12th International Conference on Integrated Design and Process Technology (SDPS 2009), (Alabama Montgomery, USA, November 1-5)*, CD-ROM.

FILIP, E.A.; KATARIA, P.; JURIC, R. (2009) "Intelligent Business Process Improvement", In: *Proceedings of the 12th International Conference on Integrated Design and Process Technology (SDPS 2009), (Alabama Montgomery, USA, November 1-5)*, CD-ROM.

KATARIA, P.; MACFIE, A.; JURIC, K.; MADANI, K.; (2009) "Ontology for Supporting Context Aware Applications for the Intelligent Hospital Ward", Transaction of Integrated Design & Process Science Tran-Disciplinary International Journal, 12 (3), pp. 35-44.

KOAY, N.; KATARIA, P.; JURIC, R.; TERSTYANSKY, G.; OBERNDORF, P. (2009) "Ontological Support for Managing Non-Functional Requirements in Pervasive Healthcare", In: *Proceedings of the 42nd Hawaii International Conference on System Science (HICSS 42), (Waikoloa, Big Island, Hawaii, January 5-8)*, pp. 1-10.

2008

MACFIE, A.; KATARIA, P.; KOAY, N.; DAGDEVIREN, H.; JURIC, R.; MADANI, K. (2008) "Ontology Based Access Control Derived From Dynamic RBAC and its Context Constraints", In *Proceedings of the 11th International Conference on Integrated Design and Process Technology (IDPT 2008), (Taichung, Taiwan, June 1-6)*, CD-ROM.

KATARIA, P.; KOAY, N.; JURIC, R.; MADANI, K.; TESANOVIC, I. (2008) "Ontology for Interoperability and Data Sharing in Healthcare", In: *Proceedings of the 4th International Conference on Advances in Computer Science and Technology (Langkawi, Malaysia, April 2-4)*, CD-ROM.

KATARIA, P.; JURIC, R.; PAUROBALLY, S.; MADANI, K. (2008) "Implementation of Ontology for Intelligent Hospital Wards", In: *Proceedings of the 41st Hawaii International Conference on System Science (HICSS 41), (Hawaii, Big Island, January 7-10)*, pp.1-8.

2007

KATARIA, P.; JURIC, K.; MADANI, K.; (2007) "Go-CID: Generic Ontology for Context Aware, Interoperable and Data Sharing Applications", In: *Proceedings of the 11th International Conference on Software Engineering Applications (Cambridge, MA, US, November 19-21)*, CD-ROM.

KATARIA, P.; JURIC, K.; MADANI, K.; CROFT, J. (2007) "Building Ontology for Intelligent Software Applications in Hospitals", In: *Proceedings of the 10th International Conference on Integrated Design and Process Technology (IDPT 2007), (Antalya, Turkey, June 3-8)*, CD-ROM.

The following list of papers in preparation as a result from this research:

"Transferring Ontological Individuals in OWL/SWRL enabled Ontologies", to be submitted to the *8th Extended Semantic Web Conference 2011*, (*May 29 - June 2, Heraklion, Crete, Greece*).

"Application Architectures for Semantic Computational Spaces (SemCOS)", to be submitted to the Journal of Software Practices and Experiences (JSPE), John Wiley and Sons, Ltd.

"Building Software Applications with Semantic Web Technologies, to be submitted to the ACM/SIG, Software Engineering Notes (SEN).

"Ontological Layering for Resolving Semantic Conflicts in Retrievals across Heterogeneous Sources", to be submitted to the ACM Transactions on Information Systems.

"Adding Values through Reasoning upon User's Requirements in Retrievals from Heterogeneous Sources", to be submitted to the Requirements Engineering Journal, Springer.

"Semantic Management of Submissions of Applications for Marketing Authorisations of Medicines" to be submitted to the IEEE Transactions on Information Technology in Biomedicine.

"Driving Business Values with Semantic Technologies: Supporting Dynamic Enterprise Models", to be submitted to Knowledge-based Systems, Elsevier.

Chapter 1

Introduction

1.1 The Domain

Wesier's [1] idea of "a collection of a wide variety of smart devices and services that react to their environments, coordinate with each other and network services to assist users in completing their tasks, and provide them with ubiquitous access to information" has given us a visionary direction towards Pervasive Computational Environments (PCEs) as the natural evolution of traditional standardised desktop computing to ubiquitous computing. Embedded and smart devices, which perform computations in such environments through direct communication with user centric applications [2], have led us to the pervasive computing paradigm, which we have embraced as the way forward in modern computing. The most cited author Satayanarayan [3] simplified in 2001, Weiser's views on what PCEs are, and subsequently described them as "*distributed information systems and mobile computing environments*". However, Satayanarayan's simplified definition does not really clarify the complexity and content of any PCE, therefore, Gupta's and Moitra's [4] views on pervasiveness in modern computational spaces are more precise. Gupta and Moitra emphasise that computers, including smart and mobile devices, are essential mechanisms for making PCEs possible, through the communication and exchange of data, using wireless connectivity and dynamic application development on an ad hoc basis, thus providing feasibility of anytime, anywhere computations. Therefore, if we agreed that PCEs are our reality, in terms of creating computational environments today, then we may argue that it is impossible to eliminate pervasiveness from software systems in the 21st century.

However, meaningful sharing of data, which allows immediate access to data/information, regardless of who owns it, is also typical of PCEs. Therefore, PCEs are not only characterised by a multitude of devices and software executions upon them, but rather by semantically rich data and information which they generate and store. The different nature, structure and complexity of data/information generated in PCEs very often include a number of data repositories that contain semantically related data, which are heterogeneous in their models and structures (some even lack structures). This is not only a consequence of the way we create PCEs and decide about their contents. Furthermore, different pervasive applications and software systems in PCEs are understood by different types of data models and utilised by different types of data repositories. Consequently, heterogeneities in

PCEs is one of their most important characteristics, which allows us to exercise all possible combinations of hardware, software, data, information, communication patterns, and the philosophy of the storage and retrievals of persistent data in any particular PCE. Without this freedom we will not be able create a space which secures *anytime, anywhere computations*.

Inevitably, pervasive software systems, like any other heterogeneous software systems, are required to interoperate seamlessly in order to allow the sharing of semantically related data from multiple repositories, which in turn makes human interaction and user involvements in such systems feasible. We users, are consumers of PCEs and are able to draw upon the most relevant information from our environment, i.e. we draw their content to clarify our ideas, to make decisions and even adapt our behavior according to semantics stored in PCEs. Communication and exchange of data is essential in PCEs, but at the same time it does not necessarily lead to the retrieval and sharing of meaningful data. This has been known as the semantic interoperability problem, which has existed across autonomous and heterogeneous software systems since the late 80s and was defined as *the ability to manage semantic heterogeneities between them* [5, 6, 7, 8, 9, 10, 11, 12, 13 and 14].

1.2 The Research Problem

Semantic heterogeneities make the semantic interoperation [10, 15, 16, 17 and 18] in software systems extremely difficult to achieve, since each of them may operate within different environments, may have a number of different data repositories which have been created according to different models, purposes, and may run on different platforms. Thus, the ability to manage semantic heterogeneities effectively and successfully would imply addressing the semantic interoperability problem in the first place. This will depend on how we manage the various interpretations of data (or data heterogeneities) and their overlapping meaning with regards to either [19, 20, 21 and 22]:

(i) the intended use of related data, or

(ii) when the same phenomena in an Universe of Disclosure (UoD) is modelled in different ways.

Obviously, our concerns on heterogeneities and interoperability have not changed since the late 80s, i.e. we are still dependent on various interpretations of data and manipulating their meaning (semantics?) when addressing heterogeneities in modern software systems.

The ad hoc creation, and dynamic nature of PCEs, suggests that the semantic heterogeneity problem will often arise when conflicting data and their interpretations occur. This is an important dimension of the heterogeneity problem. Whether we have semantic heterogeneity or not, will depend on the exact content of a PCE, and the way we build it. Furthermore, when choosing a set of data repositories essential for the functioning of a particular PCE, we will not nessecarily be aware immediately about possible heterogeneity at data level. These semantic heterogeneities become evident when we start retrieving data from such repositories. Therefore, user's interventions in terms of issuing request for retrievals in a particular instance of a PCE can trigger the first awareness of potential conflicts in data and their interpretations.

Managing semantic heterogeneities would mean *resolving conflicting data and interpretations*, i.e. resolving semantic conflicts[1] between semantically related data. Examples of semantic conflicts have been documented in [10, 12, 14, 16, 17, 18, 19, 20, 21 and 22]. However, one of the best and the oldest papers on semantic conflicts dates from the 80s. The work in [23] gives author's perception on "why semantic conflicts may appear", which is itemised in the bullet-points below:

- the existence of different perspectives as a result of the different viewpoints that groups of users and designers have about a certain 'real-world' concept when modeling systems;
- the appearance of equivalent constructs as a result of different data models;
- the incompatible design specifications as a result of different schemas.

This was one of the first attempts to understand what was happening in heterogeneous database systems in the 80s and why semantic conflicts were natural "results" of their heterogeneities. In addition to the perception of semantic conflicts from [23], we have the additional views on what further aggravates the problem of semantic interoperability created by the existence of semantic conflicts [13]:

- semantic (or other) data models in heterogeneous databases may be unable to sufficiently capture the semantics of real world concepts[2] in terms of their meaning and use, and
- there may be multiple views and interpretations of a given application domain which may change with time.

Therefore, we agree with the views from [24] that, if we do not resolve semantic conflicts, they may result in either (i) the wrong usage of semantically related data within a given context[3], or (ii) the incorrect understanding of the meaning of real life concepts they model, hence becoming an obstacle to achieving semantic interoperability. Semantic conflicts, if undetected, may produce disastrous results and may provide imprecise and incomplete sharing of data.

Subsequently, the software community has been interested in the heterogeneity problem since the early 80s, which primarily addressed heterogeneities in relational databases. The problem remained in the focus of the Database (DB) research community for more than 15 years, because of a huge demand for databases in business software applications, which were supposed to be autonomous and heterogeneous. There were many solutions which attempted to resolve heterogeneities in DB systems, and claimed that they delivered interoperable databases. Solutions ranged from migrations between various DB systems and federations amongst them [25, 26 and 27], to mediation and wrapping architectures which accommodate a variety of repositories and software applications built upon them [28, 29, 30, 31, 32, 33 and 34]. However, in 2010, we know that none of these solutions have provided a definitive answer to "how we manage semantic heterogeneities in databases" and how we can resolve semantic conflicts inherent in them [10, 35, 36, 37, 38, 39, 40].

The distributed and heterogeneous nature of PCEs [41] makes semantic heterogeneities an unavoidable outcome, if we attempt to share data and information across them. This is because the true nature of modern computational environments is in favoring heterogeneities over uniform representations of data, information and software applications which generate them. The continuous growth of the World

[1] See the Glossary for the definition of 'semantic conflicts'.
[2] The term "real world concept" distinguishes from the "(model) concepts" that can be captured using abstractions in the semantic data model [44].
[3] See the Glossary for the definition of a given 'context'.

Wide Web (WWW), and the need for pervasive software systems, has revitalised the research in interoperability. As more software systems are interrelated across domains, through pervasive software applications, we have started witnessing the demand for information retrievals, data sharing and meaningful exchange of semantically related data within and across PCEs. This has become essential in modern software systems and reinforces the need to address the semantic interoperability problem, i.e. to manage semantic heterogeneities.

However, if we attempt to address the semantic interoperability problem in 2010, we are in a much better position than the DB research community was in the 90s. Firstly, we are adopting heterogeneities as a desirable feature of modern software systems, i.e. we are not trying to eliminate it. Secondly, we are in a position of experimenting with and embracing the Semantic Web initiative [42], as a possible vehicle for providing "meaningful exchange of semantically related data across PCEs". This can be done through the commercialisation of Semantic Web technologies, such as Extensible Markup Language (XML)[4], the development of tools for supporting semantic interoperability on the WWW, standardisation of the Resource Description Framework (RDF) [5] language, plus the power of the Web Ontology Language (OWL)[6] and reasoning languages such as the Semantic Web Rule Language (SWRL) [43], which provide the additional expressivity for making the semantics of data explicit.

1.3 Research Approach

The complexities of semantic heterogeneities in modern software systems might not require addressing the interoperability problem in its entirety. Our view is that we should welcome and keep heterogeneities of software systems today, particularly in PCEs, maintain their autonomy and secure retrievals across them according to the semantics of data and applications stored in them. This will help us to:

- understand the environment where PCEs reside,
- exploit the power of users' involvements when creating and managing PCEs,
- manage semantically related data in such environments, and
- secure the correct results of retrievals across them.

We advocate the use of the Semantic Web technology, i.e. OWL ontologies and reasoning mechanisms upon their ontological concepts, in order to understand the environment where retrievals across heterogeneous repositories are made, and help us to resolve semantic conflicts triggered by the existence of semantically related data involved in such retrievals. OWL ontologies and reasoning mechanisms can be implemented through a software architectural model, which in turn can manage multiple ontologies through ontology mappings. The purpose of ontology mappings is to ultimately secure the correct results of retrievals across heterogeneous data repositories. The correctness of the results of retrievals will depend on how successfully we resolve semantic conflicts which may exist between semantically related data involved in such retrievals.

Our research approach has the following pathway:

[4] www.xml.com/
[5] http://www.w3.org/RDF/
[6] http://www.w3.org/OWL/

- we start by proposing a layered Software Architecture (SA), based on ontological layering, which supports retrievals across various data repositories in pervasive software systems and resolves semantic conflicts which arise from heterogeneities inherent in them;
- we design ontology mappings, and execute them through reasoning mechanisms, which create different ontological layers of our SA. Each layer resolves a set of semantic conflicts, triggered by the existence of semantically related data involved in a particular retrieval;
- we ensure that such ontology mappings are deployable with Semantic Web technologies.

However, there are four important issues that we must take into account in our research approach.

The first relates to the complex nature of semantic heterogeneities and semantic conflicts which we may have in PCEs. Semantic conflicts in heterogeneous data repositories are usually of interest only when ***semantically related data*** exists, i.e. where data may have similar meaning according to a particular UoD [19, 20, 21, 22 and 44]. Thus, resolving semantic conflicts in such situations inevitably depends on the understanding of the semantics stored in heterogeneous data repositories (i.e. computational models of database schema or ontology elements) and the identification of that particular semantically related data. However, the context within which data is ***semantically related*** also implies the understanding of the overlapping ***meaning of data*** in terms of their similarities and differences [45]. These similarities and differences can point towards the different types of semantic conflicts and help us to understand the different relationships between them. Thus, the context within which data is semantically related also needs to be taken into account, when resolving semantic conflicts, as it can indicate when a particular semantic similarity holds true between semantically related data.

The second relates to the power of user's involvement in retrievals across heterogeneous repositories. It is often the case that user's involvements are essential in verifying what users expect and in determining which information is relevant to user's particular retrieval [24 and 46]. Therefore, the need to understand and exploit the semantics of user's involvements in pervasive software systems can prove to be essential in creating a particular context within which semantic conflicts occur. This will ultimately secure the correct results of retrievals across heterogeneous data repositories.

The third relates to the deployment of our ontological and layered SA using Semantic Web technologies. We know that ontologies have appeared to be the answer to resolving semantic conflicts between heterogeneous multi-databases [47, 48, 49, 50, 51 and 52]. We also know that there are a variety of ontological approaches that are commonly used to deal with mismatches between heterogeneous ontologies available for the Semantic Web [53 and 54], and that they are also used in tools which relate different ontological concepts to each other, such as ASCO [55], SAT [56], GLUE [57], Cupid [58], QOM [59], Anchor-PROMPT [60] and Chimaera [61]. Thus, if we wanted to build an ontological and layered SA, for retrievals across heterogeneous data repositories, then these previous works should be taken into account in our research pathway.

The fourth relates to the design of the environment for deploying ontologies and reasoning in our SA. This will be dependent on the choice of tools and languages available in the Semantic Web technology stack[7], thus having an impact on our ontological models and reasoning mechanisms.

[7] http://www.w3.org/2004/Talks/0412-RDF-functions/slide4-0.html

1.4 Research Objectives

The aim of this research is to propose an ontological and layered SA, which generates software applications for supporting retrievals across heterogeneous data repositories in pervasive software systems, and resolves semantic conflicts which arise from heterogeneities inherent in them. We also aim to examine whether ontology mappings and reasoning mechanisms can be designed and deployed within the proposed architectural layers, using Semantic Web technologies. Consequently, ontology mapping will be the main contributing factor when creating SA for resolving semantic conflicts in PCEs.

Thus, the first research objective is to investigate traditional ontology based approaches to resolving semantic conflicts across various repositories, databases and in information systems in general. We should also examine whether ontological engineering and its existing mappings between ontologies, would be sufficient to guide us in our own approach to creating mappings between ontological layers in our SA. We have to pay attention to the latest standardisation of Semantic Web tools and languages and their impact on the way we use them to exploit and manipulate semantics of ontological models.

The second objective is to propose the SA and prove that the deployment of its components through ontological layering will create software applications which support retrievals in pervasive software systems and resolves semantic conflicts which arise from heterogeneities inherent in them. More specifically, we build specific ontologies for each layer of the SA, with clearly defined reasoning mechanisms. We perform ontology mappings and resolve a particular set of semantic conflicts at each layer. Thus, our SA should be described through its building blocks: ontologies, ontological concepts and their roles needed for resolving semantic conflicts. We also use reasoning rules for adding more expressivity to our ontological models, which in turn will specify which type of reasoning would be responsible for building a particular ontological layer within our proposal.

The third objective is to illustrate and evaluate the proposed ontological and layered SA through a specific example of retrievals of semantically related data across repositories in pervasive healthcare. However, our SA and its ontological layering, which is based on reasoning rules, has proved to be an excellent vehicle for manipulating semantically related concepts in a range of of other problem domains. We include a case study which uses our ontological layering and reasoning in the domain of applications for Marketing Authorisations (MAs) of medicines. We also list several other case studies where our ontological layering and reasoning have been re-used, such as supporting dynamic changes of business models and business processes, managing non-functional requirements in pervasive healthcare and creating virtual learning environments. All these examples are extensions to our research results when exploiting ontological layering for the purpose of resolving semantic conflicts in retrievals of heterogeneous data.

The difficulty of manipulating semantically related concepts, and addressing the semantic interoperability problem in PCEs, while using Semantic Web technologies, may raise questions on how we can approach this problem in 2010. Thus, our final objective is to put forward any suggestions we may have when resolving semantic conflicts in modern software applications of the 21st century.

1.5 Research Methods

The research methods include primarily the reading and surveying of literature, which involves examining the existence of semantic heterogeneities through generations of software systems: from the research in heterogeneous databases which started in the late 80s, to the current research in PCEs, which may use ontological engineering when addressing the semantic interoperability problem. We have also conducted a separate research into the definitions of semantic conflicts and the role of semantic similarities between data, which trigger these conflicts. This has helped us to create our own classification of semantic similarities, their degrees and the types of semantic conflicts they trigger.

Our investigation of Semantic Web technologies, as the vehicle for manipulating the semantics stored in any type of data repositories, including WWW, has helped us to:

i. exploit the enormous power of OWL and SWRL enabled ontologies in modern computing, and
ii. create inferences from existing semantics, in order to satisfy our research objectives, without being dependent on traditional forms of inferences known in the Artificial Intelligence (AI) and Description Logic (DL) communities.

We have demonstrated that OWL and SWRL enabled ontologies do guarantee the deployment of our SA and prove the feasibility of our proposal.

1.6 Outline of Forthcoming Chapters

This work is divided into 7 chapters. We start with the analysis of semantic heterogeneities and the problem of semantic interoperability, applications of ontologies in modern computational environments and related works in chapters 2 and 3. Chapter 4 proposes an ontological and layered SA, which generates a **G**eneric **o**ntology for **C**ontext aware, **I**nteroperable and **D**ata sharing (**Go-CID**) that ensures correct results of retrievals across heterogeneous data repositories, and secures inference mechanisms for resolving semantic conflicts. We illustrate and test the SA through case studies in chapters 5 and 6. Conclusions and future works are in chapter 7. The content of the forthcoming chapters are detailed below.

In chapter 2 we review semantic heterogeneities through generations of software systems: from traditional heterogeneities in database systems, to heterogeneities inherent in modern pervasive software systems. We give an insight into how technologies and research advances have changed the nature of computing and we look at the problem of semantic heterogeneities in PCEs in terms of enabling data sharing, which raises the question of semantic interoperability. We also review the evolution and application of ontologies as a software engineering solution for semantic interoperability in today's modern computational environments. Meaning based computing is discussed as a step towards managing the problem of semantic heterogeneities today.

In chapter 3 we analyse approaches related works to resolving semantic conflicts in heterogeneous databases, information systems and ontologies. The related work has been discussed through the various classifications of semantic conflicts, understanding when/where they can occur and in which type of data repositories we may find them. We also discuss methods and approaches for resolving them, such as well-known federations, global schemas and mediations in heterogeneous databases, demonstrating their benefits and drawbacks. We also look at the use of ontologies as vocabularies in heterogeneous databases

and information systems, which happened to be the first attempt for using ontologies when resolving semantic conflicts. However, we pay more attention to the use of Semantic Web technologies, ontological modelling and reasoning in resolving semantic conflicts. As our proposal is dependent on ontological layering and ontology mapping; therefore we also review research on ontology mapping and open/closed world reasoning.

In chapter 4 we propose a generic SA which accommodates ontological layering and Go-CID, which supports retrievals from various data repositories and resolves semantic conflicts. The architecture focuses on resolving semantic conflicts and achieving data sharing and interoperability in any heterogeneous environment through unique ontological layering which is based on a set of specific ontological mappings and reasoning performed upon ontological concepts. We illustrate the way ontological layers are created, and when we resolve semantic conflicts through ontological *alignment*, *integration* and *merge*. We also layout our theoretical foundations for classifying similarities between semantically related concepts in heterogeneous data repositories and introduce the process for resolving semantic conflicts.

In chapter 5 we illustrate the implementation of SA components through a case study in pervasive healthcare. We conduct three activities, as a part of our process for resolving semantic conflicts:

I. Detailing the preparation of semantics essential for creating core ontological layers by:
   a. translating the content and structure of heterogeneous Data Repositories $\{Rep_i \mid i = 1, ... m\}$ into Local Ontologies $\{LO_j \mid j = 1, ... n\}$ and the Environment Ontology (ENV_ONT), and
   b. storing user's involvements in terms of capturing user's inputs, while requesting retrievals across heterogeneous data repositories, in the User Input Ontology (USER_INP_ONT). We interpret the meaning of user's involvement through reasoning upon concepts of Local Ontologies LO_j and USER_INP_ONT. The results of reasoning are stored in concepts of the Added Value Ontology (ADDED_VAL_ONT).

II. Creating core ontological layers according to the semantic stored in ADDED_VAL_ONT. We illustrate how we perform ontological *alignment*, *integration* and *merge* to resolve different types of semantic conflicts. We have three different types of SWRL rules which support mappings: *Low-Level* rules that align Local Ontologies LO_j into Target Ontologies $\{TO_k \mid k = 1, ... p\}$, *High-Level* rules that integrate Target Ontologies TO_k into Derived Ontologies $\{DO_g \mid g = 1, ... q\}$ and *Post-High-Level* rules that merges Derived Ontologies DO_g into the final Go-CID ontology.

III. Describing our full scale implementation of a software application built upon ontological layering that successfully retrieves ontological concepts stored in Go-CID, thus ensuring the correct results of retrievals.

In chapter 6 we cover a separate case study which illustrates how ontologies ENV_ONT, USER__INP_ONT and ADDED_VAL_ONT from our proposal, can be used in a completely different domain of applications for Marketing Authorizations (MAs) of medicines. We reuse the reasoning mechanism from (I) above to ensure that, when applying for MA of medicines, correct data has been submitted, as a part of MAs procedure and according to the requirements specified in their electronic Common Technical Document (eCTD). In this section we also list a number of case studies that illustrates and tests ontological layering for solving the problem of manipulating semantically related

concepts in a range of sub-domains such as dynamic changes of business models and business processes, managing non-functional requirements in pervasive healthcare and creating virtual learning environments.

In chapter 7 we summarise our research and evaluate it by looking at the research objectives and our results. We have proved that it is feasible to use the ontological and layered SA to resolve semantic conflicts, which appear in retrievals across heterogeneous data repositories. Our approach of creating and using ontology mappings is unique because it is done according to:

- the understanding of the environment where heterogeneous data repositories reside;
- the power of user's involvement in retrievals across these repositories;
- our own classification of semantically related data and the degrees of similarities between them which trigger semantic conflicts in retrievals across heterogonous data repositories;
- specific SWRL rule chaining which automatically creates core ontological layers though reasoning upon ontological concepts.

We close chapter 7 by outlining contributions of the research, reflecting upon certain concerns and views, as the result of this research, and by listing future works.

Chapter 2

Semantic Heterogeneities and Ontologies

In this chapter we chronologically review the impact of semantic heterogeneities through various generations of development of software systems, and how we perceive them today, when ubiquity and pervasiveness are at the core of modern computational environments.

In section 2.1 we classify a variety of heterogeneities which has existed in software systems since the late 80s and the way they influence semantic interoperability in today's software systems. In section 2.2 we give an insight into how technologies and research advances have changed the nature of our computing environments and software systems developed in them. Ubiquitous computing and pervasive software systems have created new computing directions: we talk about pervasive or "smart" spaces, pervasive software applications, pervasive middleware, sensor driven computing, etc. In section 2.3 we look at the problem of semantic heterogeneities in PCEs and emphasise the importance of data sharing in, and autonomy of data repositories in various domains.

The need for data sharing across heterogeneous data repositories, whilst preserving their autonomy, and the emergence of ubiquity and pervasiveness in modern computing, has once again brought the problem of semantic interoperability to our attention. Today, we are not talking solely about semantic interoperability of data intensive software applications, as we did in the late 80s, which singled out database and information systems semantic heterogeneities. We are now raising the same question of semantic interoperability in different computational environments characterised by ubiquity and pervasiveness, because they are heterogeneous and autonomous by their nature, and they heavily depend on sharing of data and information stored within them.

In section 2.4 we review the role and application of ontologies as a possible solution in addressing semantic interoperability in modern computational environments. The power of modern technologies, such as the Semantic Web technologies brings us towards a new era of meaning-based computing, which we discuss in section 2.5. This might be a possible fundamental step towards better management of semantic heterogeneities inherent in modern computational environments. We summarise in section 2.6.

2.1 Semantic Heterogeneities through Generations of Software Systems

We perceive software systems as heterogeneous if they have different characteristics such as data/information resources/services, computation/communication devices, application/service interfaces, access methods, protocols etc., which are commonly referred to as heterogeneities. These heterogeneities can range from differences in organisational autonomy of data repositories and applications built upon them, hardware/software platforms which are used for their implementations, to similar/identical data which exists in different sets of requirements for and solutions in data-intensive software applications. The understanding of heterogeneities has evolved so much during the past two decades, and it has always been closely related to the problem of interoperability. However, the issue of heterogeneities in software systems originated in the DB community of the 80's, when they wanted to address the problem of DB systems interoperability. Heterogeneities in those days were considered an unwelcome feature because they proved to be an obstacle in achieving DB system's communication and exchange of their operational data. This was before the era of the WWW when we were concerned about heterogeneties in [17, 18, 35, 36, 37, 38, 62, 63, 64, 65, 66, 67, 68 and 69]:

- platforms, hardware/operating systems, the different database management systems, access methods, protocols, legacy applications and similar;
- interface expressiveness, query facilities and their possible restrictions in heterogeneous DB systems;
- software applications: *structural* (e.g. data structures and meta-data specifications), *behavioral* (e.g. languages which deal with data management such as Structure Query Language (SQL[8]) and *implementation information* (e.g. creating schemas and supporting DB engines);
- data modalities in the different kinds of data stored in DB systems (e.g. records, text, multimedia);
- data structures that may overlap and may run autonomously on different computers, and may be designed to meet different organisational needs.

The 1990's saw the advent of the WWW and the increasing requirements for the interoperation of systems [45 and 70]. As information and knowledge proliferated across the web, heterogeneities of systems became a critical problem for achieving the interoperability. The lack of standards and appropriate solutions of that time proved to make exchange of data between heterogeneous systems difficult. As a result several classifications of heterogeneities existed at different levels of details and for different purposes. The best sources which elaborate on both are [17, 18, 65, 68 and 69].

Furthermore, this was the first time that the research community classified the types of heterogeneities that can become an obstacle during the interchange of information between software systems [10, 32, 71 and 72]:

- *system* heterogeneities in hardware and operating systems;
- *syntactic* heterogeneities in different representation languages and data formats;
- *structural* heterogeneities in different model representations;
- *semantic* heterogeneities in different meaning of terms used in the interchange.

[8] www.sql.org/

Subsequently, in the 90's we also witnessed 'federations' [13 and 73] and 'mediations' [17], as two dominant approaches to DB and information systems interoperability. Federations relied on the construction of mappings between heterogeneous DB, and were usually accomplished by constructing a federated (or global) schema [25, 26 and 27]. Mediations on the other hand, relied on intermediary mechanisms such as mediators, agents and ontologies. They usually required the use of domain specific knowledge, mapping knowledge, or rules for specifically coordinating various autonomous databases [28, 29 and 31]. Both federations and mediations proved to be successful in addressing system, syntactic and structural heterogeneities that enabled multiple DBs to cooperate and interoperate even though their implementation languages, interfaces and execution platforms were different. They indicated that we can address application-level of interoperability though standards such as (i) XML, (ii) Web Services based on the Simple Object Access Protocol[9], and (iii) the Web Services Description Language[10], demonstrating that system and syntactic interoperability problems were more easily dealt with [10, 74 and 75].

However, at the same time when the DB research communities were using federations and mediations in heterogeneous DB systems, they did not explicitly deal with "semantic heterogeneity". This was because there was no clear definition on what exactly "semantic heterogeneity" might mean. The solutions in the forms of federations and mediations provided users with a single unified interface, sometimes called query interfaces, supported by algorithms/rules which manipulated the content of databases, in order to address their heterogeneities and did not explicitly give their way of resolving "semantic heterogeneity". There was also no clear evidence that data was being shared without integrating heterogeneous databases, and hence interoperable DB systems from the 90s sacrificed DB autonomy and integrations compromised the semantics stored in original DBs. Consequently, the problem of semantic interoperability became more evident as autonomous, distributed and evolving data repositories started being deployed across a globally interconnected WWW environment. The combination of structured DB, semi-structured and unstructured Web data exuberated the problem of semantic heterogeneity. In 2000 we started talking about semantic heterogeneity of multiple XML documents, web services [76] and ontologies, or more broadly, whenever there was more than one way to structure a body of data [77].

Therefore, it has become more evident that *the meaning of the information* that is supposed to be interchanged across software systems is heterogeneous, and any disagreement about the meaning, interpretation or intended use of the same or related data, had to be resolved. This change in focus led to a shift in our thinking on how semantic heterogeneities affect semantic interoperability. Semantic heterogeneities that formerly referred to the different meaning of terms used in the interchange of information was now more concerned with the *disagreements* in the *implicit meanings*, *perspectives* and *assumptions* made during the creation of computational models of data repositories, i.e. the nature of semantic heterogeneities were now concerned with the various interpretations of data (or data heterogeneities) regarding intended use of related data or when the same phenomena in an UoD is modeled in different ways [19, 20, 21 and 22]. In particular, the works of [21, 26, 78, 79, 80, 81, 82 and 83] singled out semantic heterogeneities in the form of the differences in meta-data, data value, and

[9] www.w3.org/TR/soap/
[10] www.w3.org/TR/wsdl

model/view specifications in contrast to the syntax heterogeneities that refers to the structure of the schema items (e.g., classes and attributes).

Today, semantic heterogeneities are inherent in modern computational environments, i.e. the true nature of modern computational environments is in favoring heterogeneities over uniform representations of data, information and software applications which generate them. The continuous growth of the WWW and the need for pervasive software systems have revitalised research in the problem of interoperability and has consequently created a modern division of semantic heterogeneities into different [5, 6, 7, 8, 9 and 11]:

- data models, which occurs when semantically-related data exists, and where the semantics behind them is important for data sharing across heterogeneous pervasive systems;
- context models, which occurs when a variety of sensors or devices produce semantics necessary to pervasive environment;
- sensors derived information, which are often the consequence of synthesising multiple sources of sensor derived information;
- application/interfaces and data they use, which may include user preference/profile models for designing interfaces or reasoning mechanisms for adjusting/adapting them to users/locations;
- user generated data, which includes task/domain related data, role of users in a particular context where data is generated, user personal preferences etc.

The bullet-points above highlight the complexity of the heterogeneities in modern computational environments and signal that we might not be able to solve this problem in its entirety. As more pervasive software systems in ad hoc and dynamic situations are interrelated across domains through pervasive smart spaces/software applications/semantic information retrievals; data sharing and the exchange of data increases, and become increasingly necessary to the functioning of modern software systems. This reinforces the need for semantic interoperability and the issue of managing semantic heterogeneities in pervasive computational environments.

2.2 Modern Software Systems

2.2.1 The Vision of Ubiquitous Computing

In 1991, the term *ubiquitous computing* was coined by Mark Weiser, who described the future of modern computational environments as “a world in which computers and associated technologies become invisible and thus, indistinguishable from everyday life” [84]. An emphasis on technologies that would disappear and weave themselves into everyday lives through embodied virtuality required the process of drawing computers into the physical world; where they would no longer be seen as silicon technology but as a part of the natural environment [1]. Contrary to virtual reality, where humans are placed inside a computer-generated world, Weiser’s vision sparked changes in the direction of the way humans would participate and interact within PCEs. Technologies that would lead to true calm and comfort within the PCEs, entailed new interfaces which would allow users to experience an unobtrusive computing experience without the distractions made by technology [85].

The importance of computation and communication capabilities of small embedded computers and devices, that would seamlessly recede into the background of everyday lives, would utilise wireless connectivity on an 'always' and 'everywhere' basis; where the dynamicity of PCEs would dictate the level of computation and communication available [86]. This required a surge in technological advancements that not only changed the way we, as consumers of ubiquitous computing, would perceive PCEs, but also the way in which new computational spaces would be created.

The significance of computer readable data, including all the different ways in which it can be retrieved, altered, processed and analysed was made to be an intrinsic part of PCEs to overcome the problem of information overload [87]. User-selective representations of information without redundant/un-useful data would transform the way in which we valued the importance of computer readable data. Ubiquitous use and retrieval of adequate information highlighted the role for the effective management of information exchange between users and their surroundings in semantic information retrievals.

Hence, Mark Weiser envisioned that PCEs would be pervaded with computing capabilities that invisibly enhanced the world, through being 'present, appearing or found everywhere'. Letting humans focus on their daily tasks, rather than on the underlying technologies surrounding them, underlines Weiser's vision to transform the human psyche to embody ubiquitous computing by being unaware of the computer, technologies and software systems that entailed them. The benefit of realising this vision was obvious in the way 21st century computing would progress, and most importantly, shape computing directions over the next decade.

2.2.2 Current Trends in Pervasive Computing

A decade later, we see Weiser's vision of ubiquitous computing steadily taking place. The evolution of mobile computing, embedded and wireless communication and computing networks have transformed the intersection of personal computers and computational/communication mechanisms. The high bandwidth and low error rate of wireless local area networks has made it possible to connect two or more computers by a network - whether mobile, static, wired or wireless, sparse or pervasive [3]. The technological maturity of portable and wearable computers, such as laptops and hand held devices, and sophisticated embeddable sensors, have moved traditional standardised desktop computing towards a less obtrusive computing experience, where embedded computing devices are capable of performing computations through direct communications with user centric applications [2]. This has consequently made it easier for contributing towards computational and communication abilities that provide users with universal information that can be transparently transported into answering their user requests on an 'anywhere' and 'anytime' basis [88].

The advances in the development of embedded wireless network technology, and mobile computing represent the major trends for the future of pervasive software systems, which are significantly changing the nature of conventional software systems, we find in finance, healthcare, public sector, etc. We outline below some directions in pervasive computing applications, which give better insight on what the future of pervasive software systems hold.

Pervasive smart spaces aim to provide a mechanism for achieving an 'anywhere-anytime' computing experience with users accessing computational and computer controlled devices [89, 90, and 91]. Computers in everyday life, including personal digital assistants, smart phones and other mobile devices have made this possible through allowing users immediate access to situational information that may be embodied in smart spaces, i.e. a smart home, office, or automobile [92, 93, 94 and 95]. Moreover, smart spaces are capable of interacting with user devices and adapting according to user preference/profile. Thus, smart spaces utilize context-aware systems for fusing information from location-aware systems and computational world models to make contextual inferences about its location of use, the collection of nearby people and objects, as well as the changes to those objects over time [96].

The dynamic nature of a smart space creates great challenges for developing context-aware and location-aware systems. Some of the critical research issues are context modelling and reasoning, knowledge sharing, and user privacy protection [6, 7, 97 and 98]. With ad hoc networking technologies used as the backbone of applications pushing mobile computing and interconnectivity to connect to various devices, the issue of sharing, storing and managing data produced by devices requires the explicit representation of context meanings (or semantics) so that independently developed applications can easily understand them.

Pervasive software applications aim to endorse the practicality of PCEs across a variety of domains [99, 100, 101, 102 and 103]. Pervasive software applications are spontaneously established and terminated due to changing contexts, device mobility, and resource fluctuation in PCEs [104]. Smart mobile phones and rapid pervasive prototyping can be seen as some key indicatives to developing pervasive software applications [105 and 106]. Smart mobile phones act as mini personal computers, which include accessibility to operating systems and are engaged in many forms of computational methods including interactions of nearby users [107] and social interactions through distributed pervasive software applications [108]. However, the ad hoc engagement in spontaneous information exchange requires computational power and reasoning mechanisms for effective management of data across a variety of applications and devices [6, 109, 110, 111 and 112].

Pervasive middleware frameworks and software architectures handle the orchestration of wireless networks and distributed mobile applications into a functioning pervasive software system. The use of middleware frameworks and software architectures provide pervasive software applications with rich interfaces that support ad hoc communication among application software in various pervasive software systems [89, 113 and 114]. The research in middleware frameworks and software architectures has been in progress for a few years, both in academia and industry. Examples include the One.World [115], Event Heap [116], GAIA [117] and Aura [118]. However, issues in the scalability introduced by the potential number of users, heterogeneity of the different environments, devices/network diversity, changes in resource availability and user movements, all raise challenges in designing mobile applications and dealing with uncertainty and operating under unpredictable conditions [119, 120, 121, 122 and 123].

Sensor driven computing augments the physical environment where sensors are embedded ubiquitously and transparently into everyday objects and living spaces [124]. Often linked to smart spaces, sensor driven computing uses embedded computations and sensor capabilities to form computational intelligence that is capable of perceiving, reasoning, learning, and reacting to their

environments [125, 126 and 127]. The urbanet revolution allows the integration of computing and sensor technologies through system/human participation [120]. The use of location sensors and resource information are gathered by software and presented in a form suitable for supporting context aware applications.

As sensor derived computing technology matures, raw context data obtained from various sensors and sources comes in heterogeneous formats. Furthermore, without prior knowledge of the context representation of raw data applications are unable to use it. Thus, new issues into sharing, storing and managing of data produced by sensors are arising. The research into sharing, storing and managing of data produced by sensors is vast and ranges from the management of data in worldwide sensors web [128], making sense of sensors data [125 and 129] to the emergence of a highly heterogeneous spectrum of sensor technologies [130].

Semantic information retrievals aim to allow users immediate access to information that may be available in data repositories storing sensor derived data to XML and Hyper Text Mark-up Language (HTML)[11] documents, Web services, information systems, DBs, etc. They do not overlap with traditional Information Retrieval Systems in terms of organizing the content of data repositories as a collection of unstructured documents represented through relatively simple data models [131 and 132]. Instead, semantic information retrievals are likely to retrieve data direct from heterogeneous data repositories. Furthermore, semantic information retrievals in PCEs also support the derivation of semantics behind user requests that are naturally expressed in the form of their intention to retrieve, manage or manipulate the content of data repositories [133]. Therefore, semantic information retrievals use the power of the user's involvement to ensure correct retrievals across data repositories, as opposed to similarity-based retrievals using approximate keyword searches and retrieving documents based on the degree of relevance to a query, e.g. semantic annotations supporting semantic indexing, keyword matching and agent crawlers [134, 135, 136 and 137].

The ad hoc and dynamic nature of PCEs means that semantic information retrievals will be scattered across a number of data repositories. They will be required to perform intelligent task-directed information gathering whilst retrieving semantically related data needed for satisfying user requests. The heterogeneity of data repositories will have to be managed where the semantics behind them is important for ensuring the correct semantic information retrievals across them. Semantic information retrievals in PCEs will acquire the support of software applications which provide semantically rich interfaces that support the information richness behind PCEs.

2.3 The Problem of Semantic Heterogeneities in Pervasive Computing

The inherently distributed and heterogeneous nature of PCEs leads to enormous amounts of data and information generated in them. Data and information are semantically rich, semantically related, and heterogeneous in their structures (some even lack structures). Thus, PCEs are not only characterised by a multitude of devices and software executions. They also accommodate different natures, structures and

[11] www.w3.org/MarkUp/

complexities of data and information generated in them. This inevitably triggers the problem of semantic interoperability. The old fashioned semantic heterogeneity problem across structured DBs and information systems still persists, but is now aggravated with the surplus of data contained within PCEs, and the growing abundance of individual autonomous data repositories which is often needed to be shared, combined and interrelated [88]. Therefore we are concerned with the autonomy of data repositories and meaningful sharing of their content in PCEs.

Autonomy of Data Repositories

Pervasive software systems must be open to a wide range of data repositories in order to exchange the data produced by sensors and devices, user's generated data and any other task-related or domain-specific data within interrelated domains. However, the high possibility of these data repositories being independently designed and implemented leads to a very old division of autonomy which existed almost 15 years ago [13, 14 and 138] and which has been adapted to PCE and summarised in the bullet-points below:

- *communication autonomy*, which refers to the capability of a pervasive software system to decide with what other systems to communicate and what information to exchange with them;
- *execution autonomy*, which refers to the ability of a pervasive software system to decide how and when to execute requests received from another system;
- *design autonomy*, which includes the presentation of data elements, their naming and format, and the choice of UoD.

Specifically, design autonomy may result in semantic conflicts that arise from heterogeneities in the form of the different structure/format used for data or the semantic interpretations of data. Therefore, design autonomy and its resulting semantic conflicts constitute a major obstacle in accomplishing semantic interoperability. This is very much similar to what is desired for a Semantic Web [139]: supporting autonomous machine readable data exchange over the WWW [140]. However, communication and execution autonomy, on the other hand, pose many new challenges for query processing and optimization [141], which is outside the scope of this thesis.

Meaningful Data Sharing

Pervasive software systems must allow meaningful data sharing across other disparate software systems. Associated data repositories may be related to the same task or domain, hence having a similar set of semantics. This may result in semantic conflicts, which arise from heterogeneities in the form of (i) their different naming/terms used for data and (ii) their semantic interpretations of data. Thus, the data being shared will be semantically related and exhibit a number of semantic similarities, because they model concepts which may have identical or overlapping meaning. This implies an explosion in the number and type of domains that need to be involved in achieving semantic interoperability in PCEs.

Hence, managing semantic heterogeneities is to guarantee meaningful data sharing across heterogeneous data repositories in PCEs and to ensure that pervasive software systems interoperate seamlessly whilst preserving the autonomy of underlying data repositories, i.e. data sharing across

heterogeneous data repositories and the preservation of their autonomy leads to achieving semantic interoperability.

The ad hoc and dynamic nature of PCEs would imply that the semantic heterogeneity problem will arise only when conflicting data and interpretations occur. Thus, in the pervasive nature of today's computing, we must take into account unanticipated data sharing across data repositories and we should think about addressing semantic heterogeneities at the particular time it arises, i.e. to resolve semantic conflicts if and when they occur. At the same time, we should also think about the adaptation to the changing conditions of the environment (i.e. the task at hand or the domain in which it is built in) and particularly to the current needs of the user.

Subsequently, the future of PCEs lies in their computational power and reasoning mechanisms for managing semantic heterogeneities and to resolve semantic conflicts between an array of heterogeneous data repositories. Therefore, PCEs must contain sophisticated models of the environment they reside in, which means that, software systems that use them are aware of their users and the semantics stored in their heterogeneous data repositories. The complexity of PCEs and their ad hoc and dynamic nature should benefit from reasoning upon the semantics stored in relationships between (i) user's context, (ii) user's information needs, and (iii) the physical environment at runtime, i.e. available data repositories. Semantically related views upon available data repositories should be at the heart of PCEs, where the explication of implicit and hidden semantics in PCEs should support the identification and resolution of semantic conflicts. Furthermore, user's involvements which contribute to enhancing the physical surroundings of a PCE should make human interaction easy in PCEs, because they are consumers of a PCE, they can draw upon the most relevant data/information from our environment and decide on how to use them.

2.4 Ontologies and Semantic Heterogeneities

Using ontologies to address interoperability is proving to be successful in providing a broader range of information and contexts through shared semantics and syntax. The authors in Uschold and Gruninger [142] mention interoperability as a key application of ontologies, and many ontology-based approaches are used across a variety of business and scientific communities as a way to share, reuse and process domain knowledge, which is central to many applications such as information management, electronic commerce, semantic web services, scientific knowledge portals, etc.

By using ontologies which behave like reference trees, we can mask semantic and syntactic conflicts through mapping techniques to resolve heterogeneities [143, 144 and 145]. Furthermore, allowing ontologies to behave like enriched data-centric models provide a means to deal explicitly with semantic interoperability challenges. The examples of using ontologies in such a manner have been elaborated in various works, such as [146, 147, 148, 149, 150, 151, 152, 153, 154 and 155].

2.4.1 The Role and Application of Ontologies

The term 'ontology' as an idea originates from the field of philosophy that is concerned with the study of being or existence. Interpreting ontology as "the philosophical science behind the types and structures of the entities, properties and relationships behind every conceptualised reality" [156] has traditionally led

philosophers' to use ontologies as a way of a semantic "grounding", to either represent a regimentation of their scientific theories or a clarification of their foundations. Over the last decade the evolution of ontology and its application has spanned across a variety of computing disciplines. In computer science, Borst [157] and Gruber [158] introduced a precise definition of an ontology as 'a formal explicit specification of a shared conceptualisation'. Explicit means that the concepts used in the ontology and any other constraints upon them are explicitly defined. Formal means that the ontology describes exactly what each concept is meant to represent, and specifies formal axioms that constrain their interpretations. Shared conceptualisation means that the ontology captures a common understanding of a particular domain of interest. Hence, this definition described how the philosophical nature of ontologies would be incorporated into computer science.

In the 80s ontologies were adopted by AI researchers, who used ontology like a formal description of mathematical logic [159]. The notion of formalised ontologies created new computational models that automated certain types of decision making, i.e. the generation of inferences from a given set of facts about the world. Hence, various ontologies were promoted for representing knowledge in forms that can be exploited by computational procedures and heuristics. To add, using 'objects'[12] of a common interest between two or more agents, ontologies became components of knowledge systems that formed content theories that would provide forms of intelligence in terms of describing the theory of a modeled world [160]. Drawing inspiration from philosophical ontologies, computational ontologies were created and viewed as a kind of applied philosophy [161].

In the early 90s, ontologies were employed by Knowledge Management (KM) researchers, who used ontology as a mechanism of providing a common set of terms that would support the exchange of data. The diversity of application domains of that time and their contributing set of terms raised the need for knowledge organisation and shared conceptualisations of domain models. Various methods of that time, such as controlled vocabularies, only provided a list of terms for use during indexing or document retrievals. Data dictionaries that organised specific relations to form taxonomies or thesauri failed to specify the semantics of a domain in terms of conceptual relationships and logical theories [162, 163 and 164]. Thus, the employment of ontologies as replacement for more complete and precise domain models transformed and enhanced interoperability standards of that time. The CyC project [165], the ARPA Knowledge Sharing effort [166], the Knowledge Interchange Format effort [29], and the National Library of Medicine which works on the Unified Medical Language System [167] are examples of an effort to create interoperability standards through identifying technology stacks that called out the ontology layer as a core component of knowledge systems.

In the post 90s, ontologies were exploited by Software Engineering (SE) researchers, who used ontology to represent the explicit specification of a shared domain and to serve as a backbone for providing and searching for information sources [160, 163 and 168]. Using ontologies as a common structure of information transformed the way in which SE researchers achieved data sharing amongst disperse sources of information in heterogeneous environments [144, 169 and 170]. With respect to DB management systems, the key role of ontologies was to specify a data model representation at the level of abstraction above the specific database design (logical or physical), so that data could be exported,

[12] where an object represents the notion of entities describing some desired communication, interaction or event.

translated, queried and unified across independently developed systems and services [153 and 171]. Hence, ontologies proved to provide similar functionalities as that of a middleware layer/federated schema between heterogeneous databases but without the need to be dependent on local schemas/applications [151 and 172]. This was the first time that ontologies proved to mask semantic heterogeneities by providing a means to deal explicitly with semantic conflicts [151, 153, 170, 172, 173, 174 and 175].

In the 21st century, ontologies were breaking free from their past restrictive rigid formalised nature and traditional role for modeling semantics, i.e. ontologies were breaking free from the specification of formal axioms that constrain their interpretations. As advised in [70] and [176], ontologies promised success towards semantic interoperability between data repositories and devices embedded in modern compuataional environments, and in this manner the application of ontologies (including their associated knowledge bases) extended across a number of domains, ranging from:

- data integration and information analysis in biology systems [177],
- integration of multi-lateral data for disaster planning [178],
- integration of healthcare information systems [147],
- data integration for pharmaceutical based research and development processes [179],
- the European project for Standardized Transparent Representations in order to Extend Legal Accessibility: ESTRELLA, IST-2004-027655 [180], to
- data integration over multiple structured and semi-structured biological data sources [181].

At the same time ontologies were also being employed in context-aware computing and became a popular tool for modeling and reasoning upon contextual information, i.e. using ontology to either model elements of information conforming to a specific context [182 and 183], arrangements of sensor derived data or powerful predicate expressions/logical constructs for computational procedures and heuristics[13]. Works by [184, 185, 186, 187, 188 and 189] explicitly deal with modeling elements of context information and reasoning upon them through Description Logic (DL) [14]. To add, ontology usage also proved to be successful in Service Oriented Architectures (SOA)[15], supporting the formal specification of service descriptions, functions and the context in which resources will be used by services [113 and 190].

2.4.2 Ontologies for the Semantic Web

From 2000 to the present day, we notably identify the Semantic Web [139] as a key application of ontologies today. The Semantic Web is an evolving development of the WWW in order to make sense of semantics stored on the WWW, whether it is in web documents, pages or any other form of web data. Prior to the application of ontologies the WWW had been extremely successful in enabling information sharing among a seemingly unlimited number of people worldwide. However, the exponential growth of web documents, pages and data on the WWW had resulted in information overload which often made discovering the meaningful association between distinct pieces of information difficult. This was because semantics stored within underlying web data on the WWW are only accessed, understandable and

[13] notably different to representing knowledge in forms that can be exploited by computational procedures and heuristics.
[14] dl.kr.org/
[15] www.service-architecture.com/

manipulated by humans and not very machine friendly. Hence, web documents, pages and data are rarely exploited in terms of their semantic relationship to each other. To that end, the Semantic Web is initiated and put forth as the 'Web of Data', where machines were able to "connect the dots" through a common framework of ontologies. The aim was to use ontology and their explicit descriptions to make it easier for machines (as opposed to humans) to find, access and process data from web repositories [191, 192 and 193], hence reducing information overload and exploiting semantics for correct and most relevant retrievals.

To date, the Semantic Web consists of several phases, commonly known as Web 1.0, Web 2.0, Web 3.0. The Web 1.0 phase marked the early WWW days and was categorised by the static HTML pages, web forms, directories and homepages. The content of Web 1.0 was reflected by the passive consumption of information and was mainly aimed towards business, advertising and publishing domains. The Web 2.0 phase, more commonly referred to as the 'social web', was and continues to be aimed at facilitating the collaboration and information sharing among users through blogs, twitters, widgets/tagging and folksonomies. The content of Web 2.0 is reflected by the dynamic nature of web documents/pages and is mainly aimed at 'us' the end users, and our social interactions using the web [145].

The next phase, Web 3.0 (the 'real' Semantic Web) is aimed to be focused on meanings, connecting knowledge in ways that make the users experience of the WWW more relevant, useful, and enjoyable. The content of Web 3.0 will be reflected by mazes of web applications working together homogenously, where online searching and requests will be tailored specifically to users request and needs. The Web 3.0 will use machine-based learning and reasoning to provide a notion of 'anytime-anywhere' internet experience leading to 'pervasiveness' in the form of an integrated and interconnected world of the web [194].

Subsequently, ontologies are an integral part of the Semantic Web because ontologies are seen as the backbone of structuring the semantics of not only web documents or pages, but also information and services on the web, making it possible for the machines to use web content and to satisfy the requests of the users. With the high interconnectivity and access to many underlying data repositories, the primary issues of the Semantic Web 3.0 and beyond will not be how to efficiently process the data, but which data is relevant and semantically related.

2.4.3 Ontologies for Pervasive Computing

Ontologies have no doubt become an essential tool and product of engineering, hence becoming one of the most powerful software engineering solutions today [195]. Subsequently, ontologies could be the answer to the semantic interoperability problem in PCEs because they can provide:

- pervasive software applications with shared application or domain specific ontological knowledge that supports information exchange and data sharing across a variety of applications, devices and data repositories in PCEs [195, 196 and 197]. Examples can be seen in [188, 198, 199 and 200];
- pervasive smart spaces supported by sensor driven computing with the aid of ontological context models conforming to either a specific computational procedure or heuristic that describes data produced by embedded devices and sensors in PCEs. Examples can be seen in [92, 184, 186, 189, 190, 201, 202, 203 and 204];

- pervasive smart spaces with the support of ontological context models describing user intention/preference/profile. Examples can be seen in [199, 205, 206, 207, 208 and 209];
- pervasive middleware frameworks and software architectures the aid of machine readable descriptions which describe the explicit representation of the functionality of mobile applications and their logical inferences of uncertain and unpredictable situations in which they may be used in PCEs [210]. Examples can be seen in [185, 187 and 211];
- semantic information retrievals with ontological semantic models for facilitating semantic interoperability across multiple heterogeneous data repositories in PCEs [195]. An example can be seen in [205].

An important factor in the success of ontologies has been the availability of sophisticated Semantic Web tools with built in reasoning support. This is because reasoning is essential in supporting both the design of high quality ontologies, and the deployment of ontologies in their applications. Hence, in order to manage semantic heterogeneities, we must exploit Semantic Web technologies in order to support explicit representation, expressive querying, and flexible reasoning for the understanding and resolution of semantic conflicts. We should use a combination of semantic web tools, which may give us new hope in providing a basis upon which we can start to resolve semantic conflicts and provide a reasonable balance between power, expressivity and manipulation of semantics.

2.4.4 Towards Meaning-based Computing

As introduced by Berners-Lee in [139], "*for the Semantic Web to function, computers must have access to structure collections of information and sets of inference rules that they can use to conduct automated reasoning*". The core design principles and collaborative working groups of the World Wide Web Consortium (W3C)[16] have initiated a variety of technologies that express the implementation and future realization of the Semantic Web. Figure 2.1 outlines an architecture for the Semantic Web that is multi-layered and machine processable.

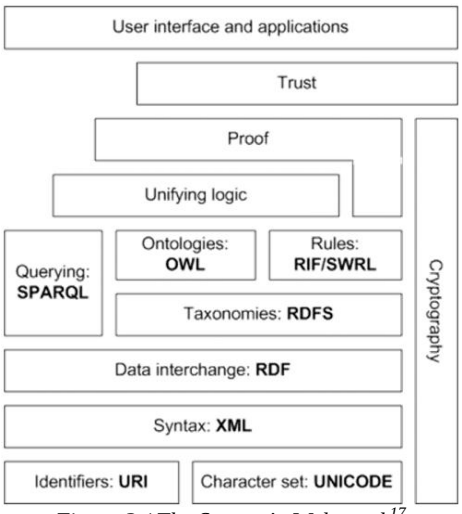

Figure 2.1*The Semantic Web stack*[17]

[16] www.w3.org/

[17] http://www.w3.org/2004/Talks/0412-RDF-functions/slide4-0.html

The various technologies in Figure 2.1 are supposed to be used on top of each other in order to provide standardised support for the vocabulary with which to make assertions about semantic relationships between web documents or pages in order to facilitate greater machine interoperability and in turn may be used as inputs or outputs of web applications [197]. These technologies are intended to provide a formal description of concepts, terms, and relationships within a given knowledge domain. The Web Ontology Language Overview[18] describes each of these technologies in more detail; however, we briefly outline their purposes below.

We are particularly interested in the OWL, because it supports formal semantics and efficient reasoning. It models RDF triples into high level concepts that form a standard semantic integration language. It adds more vocabulary in the form of an ontology for describing properties and classes: relations between classes (e.g. disjointness), cardinality (i.e. exactly one), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes. The OWL language provides three increasingly expressive sub-languages designed for use by specific communities of implementers and users. It consists of OWL Lite, OWL DL, and OWL Full in a layered approach; that is, OWL Lite is a subset of OWL DL, OWL DL is a subset of OWL Full [42]. It must be noted: OWL originates from the Ontology Inference Layer (OIL)[19] and the Defense Advanced Research Projects Agent Markup Language (DAML)[20] . The OIL is from the On-To-Knowledge Project and is notably the first ontology representation language that extends Resource Description Framework Schema (RDFS)[21] with additional language primitives. The DAML is aimed at developing a language to facilitate the semantic concepts and relationships understood by machines. DAML+OIL [212] is the latest extension of DAML which has some important features of OIL imported and is currently evolving as OWL. OWL is almost the same as DAML+OIL, but some primitives of DAML+OIL are renamed in OWL for easier understanding.

In our research we also use the **SWRL**, which provides additional expressivity to OWL concepts. It combines sublanguages of OWL (OWL DL and Lite) with those of the Rule Markup Language. SWRL rules are in the form of an implication between an antecedent (body) and consequent (head). The intended meaning can be read as: whenever the conditions specified in the antecedent hold true, then the conditions specified in the consequent must also hold true [213]. SWRL is needed to allow describing relations that cannot be directly described using description logic used in OWL. However, the **Rule Interchange Format** (RIF)[22] is the part of the rule layer for the semantic web. The design of RIF is based on the observation that there are many rule languages around which deal with Semantic Web data in one or another way, hence the exchange of rules between them is required [214]. RIF includes three dialects, a Core dialect which is extended into a Basic Logic Dialect and Production Rule Dialect [215].

The **XML** is the surface syntax for content structure within web documents and pages. XML syntax does not imply the semantic of web documents or pages, i.e. there is no meaning associated to the content of web documents or pages [216]. The **RDF** is a language for expressing metadata data models. RDF

[18] http://www.w3.org/TR/owl-features/
[19] http://www.ontoknowledge.org/oil/TR/oil.long.html
[20] http://daml.semanticweb.org/
[21] www.w3.org/TR/rdf-schema/
[22] www.w3.org/2005/rules/

models relationships between web pages (referenced by their Universal Resource Identifier (URI)) in simple statements in the form <subject, predicate and object>, where:

- a subject denotes a resource represented as a URI or International Resource Identifiers (IRI) that can be identified uniquely and globally,
- an object can be either a literal (such as a string or number) or a URI reference to another resource, and
- a predicate is a relationship between a subject and an object [216].

The **RDFS** is a vocabulary description language for describing semantics for generalized-hierarchies of RDF triples that indicate which application specific classes and properties are expected to be used together [217].

Finally, the **Protocol and RDF Query Language** (SPARQL)[23] is the query language for RDF, OWL and semantic web sources. SPARQL allows for a query to consist of triple patterns, conjuctions, disconjuctions and optional patterns [218].

Besides ontology languages, other Semantic Web tools and reasoning systems include modeling and development toolkit environments, inference engines, annotation tools, ontology based crawlers and mining tools [219]. Modeling and development toolkit environments such as TopBraid Composer[24] (which is commercially available), Protégé Knowledge Editor[25] and WSMO Studio[26] (which are open source tools) provide a standard way of creating and sharing ontologies, which can be reused, extended and translated into machine interpretable definitions. Inference engines such as Jess[27], Pellet[28], Racer [220] and Owler [221] are used to check the consistency and integrity of ontologies. They also are responsible for the deduction of new knowledge based on the rules or relationships of the concepts specified by ontologies without the explicit need for algorithms advocated by the Distributed Systems and AI communities etc. A detailed analysis of these modeling and development toolkit environments and can be found in [222].

However, during research on Semantic Web technologies the demands for combined formalisms which integrate ontology and rule languages have emerged as a consequence to supply advanced reasoning capabilities and the need for new inferences from existing data on the web. The wealth of mature Semantic Web tools and their reasoning systems available today have opened new doors towards meaning-based computing that does not necessarily follow the original Semantic Web Stack, but instead uses a combination of ontology technologies in which new inferences from existing data can be made. . At the heart of the inferences lies the dependency on DL as a decidable fragment of first order predicate logic as the formal foundation of OWL (as well as its predecessor DAML+OIL [212]). Subsequently, DLs has turned out to be an adequate formalism for representing and decidable reasoning about expressive ontologies.

Furthermore, OWL Lite and OWL DL represent decidable fragment of OWL with complete reasoning support, in which a subset of first order logic that allows description of complex concepts from

[23] www.w3.org/TR/rdf-sparql-query/
[24] http://www.topbraidcomposer.com/
[25] http://protege.stanford.edu/
[26] http://www.wsmostudio.org/
[27] http://www.jessrules.com/
[28] http://clarkparsia.com/pellet/

simpler ones with an emphasis on decidability of reasoning tasks. For example, using Protégé and TopBraid Composer in combination with:

- logic-based rule languages such as RuleML[29], F-Logic [223] or SWRL;
- information visualisation tools such as OWL viz [224], or Jamabalya [225];
- ontology translations techniques such as DR2Q mapping [226] or Datamaster [227];

allow OWL based ontology usage for not only structuring the WWW but also for the acquisition, structuring and reasoning upon XML documents, database management systems, context aware databases, sensor derived data etc.

The availability of Semantic Web tools and their reasoning systems such as those mentioned above has contributed to the increasingly widespread use of OWL, not only in the Semantic Web per se, but as a popular language for ontology development and reasoning in fields as diverse as medicine [228], astronomy [229], geography [230], defence [231], biology [232], agriculture [233] and geology[30].

In essence, Semantic Web tools and their reasoning systems have ultimately paved the way towards meaning-based computing that can bring extraordinary opportunities for semantic interoperability and the exploitation of semantics for dealing with the heterogeneity of software systems, data, their representational levels and terminological differences [234]. Thus, shouldn't we use Semantic Web tools and their reasoning systems in dealing with semantic heterogeneities in PCEs? Should we not assume that our meaning-based computing captures the content of heterogeneous data repositories in terms of understanding their semantically related data and resolving the semantic conflicts contained within them?

2.5 Summary

In this chapter we have reviewed the semantic heterogeneities through various generations of software systems. We have discovered that the nature of semantic heterogeneities has evolved over the years and remained the major obstacle to semantic interoperability. Today semantic heterogeneity are in the form of semantic conflicts that are concerned with the *disagreements* in the *implicit meanings*, *perspectives* and *assumptions* made during the creation of computational models of data repositories, i.e. the nature of semantic heterogeneities are concerned with the various interpretations of data. Consequently, we have also revealed how difficult it is to manage such semantic heterogeneities since past solutions have failed or proven to have a short life. The trade off between the loss of semantics and the preservation of 'original' semantics make managing semantic heterogeneities harder and hence still very much prevalent in modern computational environments today.

We have given an insight into how technologies and research advances have changed the nature of computing into ubiquitous computing in pervasive software systems. The inherently distributed and heterogeneous nature of PCEs has led to multiple computing directions that suggest that the future of PCEs will further aggravate the problem of semantic heterogeneities and trigger the problem of semantic interoperability as a result for the need to guarantee meaningful data sharing whilst preserving the autonomy of participating data repositories. The growing abundance of semantics generated in

[29] http://ruleml.org/
[30] http://sweet.jpl.nasa.gov/

heterogeneous data repositories in PCEs once again reminds us of the complexity of dealing with the semantic heterogeneity problem.

As a promising approach to addressing semantic interoperability in PCEs and managing semantic heterogeneities, we have pointed towards the use of ontologies as a powerful software engineering solution for dealing with semantic heterogeneities inherent in PCEs. We have reviewed the evolution and application of ontologies, where ontologies have been increasingly applied in computer science to areas that require the exchange of information with significant structure and diverse semantics. The multiple examples of exploiting and manipulating semantics has implied meaning-based computing as a new way forward in supporting the explicit representation, expressive querying, and flexible reasoning for the understanding and the resolution of semantic conflicts.

Chapter 3

Related Works

In this chapter we review research which is related to our proposal of resolving semantic conflicts through ontological layering. We have not found research which explicitly uses layering of OWL/SWRL enabled ontologies for the purpose of resolving semantic conflicts. However, we do relate our works across three main areas of research: (i) understanding and classification of semantic conflicts (ii) methods and approaches to resolving them and (iii) the use of semantic web technologies, ontology mapping and reasoning in resolving semantic conflicts, i.e ontology mismatches.

In section 3.1 we review various classifications of semantic conflicts, as in (i) above, in terms of understanding where they can occur, in which type of data repositories they may exist and with which level of data granularity they are concerned with, e.g. are they dealing at the meta-data or data value levels. In section 3.2 we give an overview of the most common methods and approaches used to resolve semantic conflicts, as in (ii) above. We divide them into three groups. Thus, section 3.2.1 reviews methods and approaches to resolving semantic conflicts through federations, global schemas and mediations in heterogeneous databases. Section 3.2.2 reviews methods and approaches which use ontologies as vocabularies in heterogeneous databases and information systems. Last but not least, section 3.2.3 reviews the use of Semantic Web technologies, ontological modelling and reasoning as in (iii) above, in modern computational environments, which naturally create semantic conflicts due to their heterogeneous nature. Finally, as our proposal is based on ontological layering and reasoning mechanisms using the Semantic Web technology, section 3.3 reviews research on ontology mapping (section 3.3.1), reasoning mechanisms (section 3.3.2) and any other issues that are relevant to our work (section 3.3.3). We end the chapter by summarising current research outcomes which single out limitations of existing solutions for resolving semantic conflicts in general (section 3.4.).

3.1 Semantic Conflicts - Types and Classifications

Semantic conflicts arise from semantic heterogeneities. Therefore, the issue of resolving semantic conflicts is closely related to the semantic heterogeneity problem in the database and information system research communities (see chapter 2, section 2.1). In general, the various types of semantic conflicts can be classified into two broad categories: structural differences and semantic differences. As the name

suggests, structural differences are concerned with how data is logically organised and structured. Semantic differences are concerned with the interpretation of data and what data means. Both structure and semantic differences have been well documented in literature over the last decade.

3.1.1 Structural Differences in Data Modelling

Structural differences were of major interest in the DB community, where the focus was on resolving structure conflicts [235], constraint conflicts [44] and schematic conflicts [236, 237, 238 and 239] at the syntactic level in the federation, mediation and integration of databases. This was when the database community was interested in the structural differences between database schemas at the following levels of data granularity [18 and 80]:

- *concept level*, i.e. the use of different semantics for the concepts in data models, e.g. relational model does not have the inheritance concept as in object orientation models;
- *schema level*, i.e. the use of different schemas of DB models, e.g. when data under one schema corresponds to database or schema labels in the other;
- *attribute level*, i.e. the structure of a set of attributes and their values belonging to an entity class in one database is organized to form a different structure in another database.

In particular, the works of [21, 50, 66, 78, 82 and 83] singled out heterogeneities in the form of structural conflicts in meta-data, data value, and model/view specifications as:

- *naming/representation* conflicts, which occur when different models use different names to represent the same concepts, i.e. labels of schema elements such as entity, relationships and attributes are arbitrarily different;
- *domain-specific/scaling/scope* conflicts, which occur when different models use different values to represent the same concepts, i.e. data values can have multiple representations and interpretations;
- *structural/granularity/precision* conflicts, which occur when different models use different data organization to represent the same concepts (for example, the same concept can have a number of attributes);
- *meta-data/identifier* conflicts, which occur when the same concepts are represented at different levels of the model (for example, at the schema level in one database and at the instance level in another).

3.1.2 Semantic Differences in Data Interpretation

Semantic differences were of interest only when semantic problems became more evident in the semantic interoperability of multi-database and information systems. In other words, semantic differences were of interest when semantic problems regarding the different interpretations of data and their UoD become an obstacle during the meaningful exchange of data. As a result, there have been numerous classifications of semantic conflicts that emphasise conflicts at the attribute and data-value levels of data granularity [10, 44, 77 and 151]. However, there has been little consensus on what each type of semantic conflict encompasses and as result have overlapped with conflicts belonging to structural differences. We outline the classifications of semantic conflicts proposed by [44, 50 and 240] because they provide the most

detailed list of semantic conflicts and have been the most cited and referenced over the years of research in semantic interoperability.

The most exhaustive list of semantic conflicts is available in [44]. They organize semantic conflicts between structured databases into: domain, entity, data value, abstraction level, and schematic levels of data granularities, where:

- domain conflicts arise between two objects when they have different definitions of attributes. Domain conflicts include *naming*, *data representation*, *data scaling*, *data precision*, *default value* and *attribute integrity constraint* sub types of conflicts;
- entity conflicts arise between two objects when the entity descriptors used by the objects are only partially incompatible. Entity conflicts include *database identifier*, *naming*, *union incompatibility*, *schema isomorphism* and *missing data item* sub types of conflicts;
- data-value conflicts arise between databases where data values already existing are different. Data-value conflicts include *known inconsistency*, *temporal inconsistency* and *acceptable inconsistency* sub types of conflicts.;
- abstraction-level conflicts arise between entities that are represented at differing levels of abstraction, i.e. different level of generality at which an entity is represented. Abstraction-level conflicts include *generalisation* and *aggregation* sub types of conflicts;
- schematic-level conflicts arise between databases when data in one database corresponds to the metadata of another. Schematic-level conflicts include the data-value attribute, attribute entity and data-value entity sub types of conflicts.

They also make clear distinction between the different levels of data granularity but many of the sub types of semantic conflicts listed, for instance, naming, data representation and database identifier conflicts are semantically the same type of conflict, but are considered separately if they occur at the entity level or the attribute level.

However, the classification of semantic conflicts as in [240] is catergorised along the three dimensions of ‘naming’, ‘abstraction’ and ‘level of heterogeneity’. The ‘naming’ dimension refers to the relationship between the object, attribute, or instance names, and thus contains naming conflicts. Naming conflicts include the *synonyms*, *homonyms* and *unrelated* sub types of semantic conflicts. The abstraction conflicts refer to the relationship between two schematic elements. Abstraction conflicts include the *class*, *generalization*, *aggregation* and *computed function* sub types of semantic conflicts. The ‘level of heterogeneities’ refers to the object, attribute, and instance levels of data granularity. Therefore, the naming and abstraction conflicts can exist at the object, attribute, and instance levels of the schema. Their classification is not as exhaustive as the classification available in [44] because they describe semantic conflicts with a fewer dimensions, hence reducing the redundant treatment of conflicts that are overlapping or those which are essentially the same.

Furthermore, Goh [50] characterises semantic conflicts between databases into four categories:

- schematic conflicts are concerned with the differences in the structure of data as a result of data heterogeneities and include *data type*, *labeling*, *aggregation* and *generalisation* sub types of schematic conflicts;

- semantic conflicts are concerned with the different interpretations of data, even when the corresponding database schemas are identical. Semantic conflicts include *naming*, *scaling* and *confounding* sub types of semantic conflicts;
- intension conflicts are concerned with the differences in the informational content present in data sources or information content expected by receivers. Intension conflicts include the *domain* and *integrity* sub types of conflicts.

It is obvious that Goh loosely categorises semantic conflicts as compared with classifications available in [44] and [240], i.e. *data representation*, *synonyms* and *homonyms* based semantic conflicts which are closely related to semantic interpretations of data have been excluded in their classification. However, out of all the three classifications of semantic conflicts, Goh is the only one that takes into account the expectations of the information content required by end users (i.e. receivers of data).

3.1.3 Ontological Mismatches

In modern computational environments today, outside structured databases, ontologies too, can be heterogeneous on many levels. Ontologies can have different models, different representations/meanings of the same reality, different naming conventions, and different ways to organize the taxonomy of information. Examples are available in [54, 161, 247, 250]. Subsequently, the problem of resolving semantic conflicts within heterogeneous ontologies has been documented in many works, such as in [15, 142, 241, 242 and 243].

However, classifications of semantic conflicts in the ontological community are commonly referred to ontological mismatches between different levels of modelling constructs, i.e. concept, property or instance, and are usually based on either: existing classifications of semantic conflicts from the database and information communities (as described above) [147, 153,173, 174, 245 and 246] or associated to ontology mapping, i.e. the alignment, merge or integration of multiple ontologies [247, 248 and 249]. The works of [54, 69, 250 and 251] give the best examples of the classifications of ontological mismatches that do not rely on classifications of semantic conflicts from the database community.

The classification of ontology mismatches available in [69] is based on the differences in the conceptualisation and explication (i.e. the way in which the conceptualisation is specified) of ontological concepts and their relations. Their classification includes *conceptualisation*, *class*, *categorisation* and *aggregation* ontological mismatches.

Klien [250] gives the most comprehensive list of ontological mismatches. Klien distinguishes between two levels of mismatches: the *language*/*meta-model* level that deals explicitly with non-semantic differences and the *ontology*/*model* level that deals explicitly with semantic differences. The mismatches in the first level are concerned with the specification of ontology i.e. the different mechanisms used to define classes and relations. They have the *language*, *syntax*, *logical representation*, *semantics of primitives*, and *language expressivity* sub types of ontological mismatches. The mismatches in the second level are concerned with the differences in the way the domain is modeled and have the *conceptualisation*, *explication*, *terminological* and *encoding values* sub types of ontological mismatches. Klien further classifies:

- *conceptualisation* ontological mismatches into differences in either the scope, model coverage or granularity of ontological concepts,
- *explication* ontological mismatches into differences in either the paradigm or concept description of ontologies, and
- *terminological* ontological mismatches into synonyms or homonyms based ontology concepts.

However, the simplest classification of ontology mismatches that occur when using multiple ontologies in the same application domain is available in [54]. Their ontological mismatches include *naming* (i.e. different names for labels), *restriction*, *property* and *concept mismatches*. Therefore, their classification is geared more towards the structural differences rather than the semantic differences in ontologies, which are described in [69]. However, the classification available in [54] distinguishes between concept-level and property-level of data granularity.

A classification of ontological mismatches based on the different meaning of terms used in ontologies and their overlapping content within the same domain is available in [251]. Ontological mismatches include *ambiguous reference*, *synonymical reference*, *one-to-many matching*, *uncertain matching* and *structural differences* between different ontologies. Their classification differs from classifications available in [54] and [69] because it is directly related to the problem of ontology alignment.

To summarise, there are many similarities and discrepancies in the semantic problems of the different types of semantic conflicts and ontological mismatches listed above. The semantic conflicts do not easily fall into the discrete categories and they do not neatly fit into the structural or semantic differences. The distinction between structural or semantic differences in each above mentioned classifications are not always clear as the logical organisation of data (or taxonomical structure of concepts in the case of ontologies) often conveys or implies semantic interpretation of data.

However, the above classifications do enumerate numerous types of semantic conflicts in advance. These semantic conflicts subsequently can be used as a basis of describing new types of semantic heterogeneities within different domains of interests, where they can lead to a clear picture of what to expect from semantic conflicts of the future. Hence, the classifications of semantic conflicts give us the space to extend and tailor their sub types according to future data repositories in modern computational environments. Future data repositories may not necessarily be structured as in the traditional relational database philosophy, i.e. we may have semantic conflicts in semi-structured data repositories which may have ontological concepts rather than database schemas. We may still be faced with some/limited structure in the ontological world, regardless of the existence of a database or not.

3.2 Resolving Semantic Conflicts through Different Methods and Approaches

Within the last two decades, there has been an array of different methods and approaches aimed at achieving interoperability among autonomous and heterogeneous databases and information systems. Some of the notable ones include federation, global conceptual schema, mediation and ontology based approaches. Mostly, these approaches have differed from one another along three key aspects:

- the choice of the underlying data model for achieving schematic and semantic transformations needed for resolving semantic conflicts (e.g. relational models, object oriented models or ontological models);
- the choice of preserving or destroying autonomy of databases, i.e. retaining the maximum number of 'original' semantics through non/partial-integration (e.g. translators, brokers and partial global schemas) or losing 'original' semantics through full-integration (e.g. federations and global schemas);
- the level of data sharing, i.e. a shared schema or interactions with a limited subset of the databases at any one time.

3.2.1 Federation, Global Conceptual Schema and Mediation in Databases

As we mentioned in chapter 2 (section 2.1), traditionally, federations, global schema and mediations have been the three most popular approaches to heterogeneous database integration. They namely resolve schematic conflicts and are more concerned with the differences in schema structures as opposed to semantic differences.

Federations rely on the construction of mappings between heterogeneous databases, and are usually accomplished by constructing a federated (or global) schema [25, 26 and 27]. Thus, federations assume the collection of cooperating but autonomous database systems, where each database is expected to export a portion of its schema that it was willing to share with other databases, i.e. federations assume the pre-integration of participating databases. Sheth and Larson [13] proposed two methodologies for managing the pre-integration of schemas resulting in either a tightly coupled system or loosely coupled systems.

In the case of a tightly-coupled system, the identification of semantic conflicts and the means of resolving them is created and maintained by a system administrator. The actual resolution of conflicts is done through one or more views which define the shared federated (or global) schema for the system, providing users with a canonical representation of the data originating from heterogeneous databases. An example of a tightly coupled federation system can be seen in [252]. However, the global or federated schema must be developed before issuing any queries in a federated system, thus requiring the pre-integration of schemas, hence compromising the level of independence required to preserve the autonomy of each participating database. Furthermore, any changes in the local schemas affect the global or federated schema implying rigidity in the flexibility of accommodating more semantics for data sharing.

In a loosely coupled system, a federated schema is created and managed by the user, i.e. it allows users to query local database systems directly by placing the integration responsibility on users. Hence, the users can directly interact with local databases instead of being restricted to querying federated schemas. Consequently, instead of resolving semantic conflicts through one or more views which define the shared federated schema for the system, conflict identification and resolution are undertaken by users themselves. Examples of loosely coupled federation systems can be seen in [20, 253 and 254]. However, loosely coupled federation systems require users to have semantic understanding and to be able to resolve conflicts in creating their schemas, thus placing too heavy a burden on users by requiring them to understand the underlying local databases [50]. Furthermore, users may only interact with a limited

subset of the sources at any one time, restricting the number of participating databases, thus reducing the level of data sharing.

To add, the choice of underlying model for a federated dictates the level of semantic richness and flexibility [255]. Federated schemas can be specified in the form of relational global schemas, object orientated global schemas or single domain model/global ontology. Examples of relational global schema can be seen in the Mermaid [252] and Multibase [256] that uses a data model that originates from the era before object orientation. Examples of object orientated schemas can be seen in Pegasus [257] that attempts to create super classes to subsume related data from several databases. Examples of global ontologies can be seen in [177 and 232], which both attempt to construct a single global ontology which expresses the shared semantics of participating databases, i.e. the use of a single domain ontology against which all data is integrated. However, all three forms of global schemas lack semantic richness and flexibility from the users' perspective. For example, Ziegler and Dittrich [24] discuss that the available information in global schemas may be too general or too fine-grained and hence conclude that inappropriately collected and selected content of global schemas contribute towards their lack of semantic richness.

Mediations on the other hand, rely on intermediary mechanisms such as mediators, wrappers, agents and ontologies. They usually require the use of domain specific knowledge, mapping knowledge, or rules for specifically coordinating various autonomous databases [28, 29, 31 and 34]. Examples of mediations can be seen in the Mediated Integration Framework [258], Information Manifold [30], Garlic [32] and TSIMMIS [33]. However, the use of intermediary mechanisms needs highly specialised translations for each pair of local database systems. Therefore, when the number of local database systems increases, the number of intermediary mechanisms grows resulting in numerous ad hoc programs. The development of these ad hoc programs is expensive in terms of both time and money.

3.2.2 Using Single Ontologies as Vocabularies in Heterogeneous Databases and Information Systems

As we mentioned in chapter 2 (section 2.4.1), ontology based approaches in resolving semantic conflicts (which may include schematic conflicts) have by far been the most successful attempt to achieving a certain level of semantic interoperability of databases and information systems through data sharing. Traditional ontology based approaches use a single global ontology to either [15, 142 and 251]:

- provide a shared vocabulary for the specification of a particular domain, eliminating the occurrence of semantic conflicts (i.e. a canonical representation of inter-domain knowledge), or
- provide a rich vocabulary for the explicit description and resolution of semantic conflicts,

thus allowing the sharing of data and knowledge to take place across different computing environments.

3.2.2.1 Ontology and Shared Vocabularies

From the data sharing perspective, the BioMediator ontology available in [181 and 259] provides a common interface to Web-accessible databases of biologic information and eliminates the need to deal with semantic conflicts through a shared vocabulary of biologic information. Similarly, a global ontology from [170] is used as common knowledge domain amongst a variety of databases systems to allow data

integration between them. Each pertaining local relational schema of the contributing database system is merged into a global schema in the form of an ontology. The classes established within the global ontology are a result of mapping and reasoning upon relationships in the local relational schemas, subsequently eliminating both syntactic and semantic conflicts of local relational schemas. However, all works in [170, 181 and 259], integrate databases against a single ontology, which subsequently requires domain expertise that can carry out manual mappings between database schemas and the single global ontology.

3.2.2.2 Ontology and Rich Vocabularies

Ontologies that are used as a rich vocabulary for the explicit description and resolution of semantic conflicts attempt to enumerate and describe all the possible semantic conflicts that can occur between heterogeneous DBs and information systems. Furthermore, they tend to use either semantic weightings or mappings as a means of comparing semantic conflicts in their reconciliation. For example, the Semantic Conflict representation Model (SCM) from [47] detects and resolves semantic conflicts in database integration. Their SCM is based on a Semantic Conflict Ontology (SCO) and an Extending relational database Schema Model (ESM). The SCO is used for the vocabulary of semantic conflicts based on data type, data format, data unit, data precision, default values and attribute integrity constraints. The ESM used to express data semantics of the databases explicitly. Semantic mediators between the SCO and ESM are used to detect semantic conflicts through 'conversion' knowledge carried forward into the ESM, according to the presence of semantic conflicts modelled in the SCO. The semantic conflicts are resolved through transforming semantics from the ESM and loading them into databases that need to be integrated. However, the underlying database schemas are still integrated losing the autonomy and evolution of their semantics.

Similarly, the Relational DataBase Ontology (RDBO) from [48] resloves semantic conflicts between databases automatically while allowing the individual DBs to evolve. RDBO is based on ontological classes that make up the semantic descriptions of the individual databases. Each ontological class conforms to a set of vocabularies, structures, and restrictions that are commonly agreed upon by participating DBs. A reasoning engine is used to validate and infer additional semantic relationships from the existing relationships. To resolve semantic conflicts, terms defined in different database ontologies are compared to each other semantically using semantic weights and the reasoning engine. However, the semantic weights are based on probabilistic methods that do not guarantee the resolution of semantic conflicts as per the users expectation, i.e. overlapping semantics are purely based on semantic descriptions that describe the structural characteristics of relationships, e.g. meta-data, class name or primary/foreign key constraints; they do not take into consideration overlapping information or degrees of similarities.

However, the Semantic Conflict Resolution Ontology (SCROL) from [49] can be used to identify and resolve semantic conflicts among heterogeneous databases. A common categorisation of semantic conflicts has been derived through the analysis of geographical data, which in turn constitutes towards their own semantic classification framework. The semantic classification framework helps users to identify the types of semantic conflicts taking place. Depending on the type of semantic conflict, SCROL

is used to deal with the resolution of the conflict. Ontological class 'conflict resolver' is used to hold such conflicts and through reasoning mechanisms is mapped to the 'semantic resolver' ontological class to provide a common consensus of the semantic conflict. However, although the definition of SCROL is a one-time effort and the resulting ontology can essentially be used repeatedly in many application domains, whilst keeping the autonomy of the local schemas, the burden still lays on each user of a domain to correctly establishing ontological mappings between SCROL and schemas of databases.

Furthermore, a shared ontology from [50] as part of the context interchange framework is used for the reconciliation of semantic conflicts between heterogeneous databases. In their context interchange framework it is assumed that the interpretations of data contained within databases are explicitly represented in the form of a shared ontology. Therefore, the shared ontology constitutes a shared vocabulary for describing the context in which data supplied by a database ('import' context) and the data expected by the data receiver ('export' context). A mediator is used for detecting the presence of semantic conflicts based on naming, measurement, representation and computational disparities between data supplied by a database and data expected by the data receiver. Upon detection of semantic conflicts the context mediator compares both the 'import' and 'export' contexts in the shared ontology and calls upon conversion functions to reconcile disparities. However, because the requirements of data receivers are diverse and can change rapidly, it is impractical for the shared ontology to capture all data expectations of the users, i.e. the shared ontology resolves only semantic conflicts in a subset of pre-defined user expectations of available data.

The choice of using a single global ontology to either provide a shared vocabulary for the specification of a particular domain, or provide a rich vocabulary for the explicit description and resolution of semantic conflicts no doubt provides more flexibility in terms of increasing the level of data sharing through accommodating a wider range of databases and making their semantics explicit. However, the resolution of semantic conflicts is very much similar to a semantic approach to database integration chosen by integrating databases against one domain model. Thus, full preservation of the autonomy of the databases cannot be made, hence leading to a compromise between retaining original semantics in database versus the level of their ontological commitment to the global ontology. For example, single global ontology approaches are susceptible to changes in the databases that can affect the conceptualisation of the domain represented in the single global ontology, i.e. depending on the nature of the changes in one database it can imply changes to the single global ontology and in the mappings to the underlying database.

Furthermore, current semantic conflict detection using a single global ontology mostly depends on human intervention which is an obstacle towards the automatic semantic interoperability between heterogeneous databases or any other data source for that matter. Automated solutions where the burden does not lie on the user to identify and resolve semantic conflicts implies the need for run time solution to resolving semantic conflicts, i.e. automatic identification and resolution of semantic conflicts. The inappropriate collected and selected content of the global ontology suggests the continuous problem of the available information being too general or too fine-grained from the users' perspective. These disadvantages lead to the development of multiple ontology approaches.

3.2.3 Using Multiple Ontologies and Reasoning in Modern Software Systems

Today modern ontology based approaches in resolving semantic conflicts rely on the use of multiple ontologies. Each heterogeneous data source (i.e. web data, web repositories, XML documents, databases, information systems etc.) is described by its own ontology, commonly referred to as source/local ontology. Semantic conflicts are resolved through either the process of ontology based *semantic matching* between source ontologies, or the process of ontology based *semantic mapping* of source ontologies into a domain (or upper) ontology.

3.2.3.1 Ontologies and Semantic Matching

In the case of *semantic matching*, semantic conflicts are usually resolved through a set of logical inferences that create a 'match' between related concepts of source ontologies. Each match is dictated by a correspondence between underlying data sources, where a particular correspondence implies a particular type of semantic conflict. Semantic access to reconciled matched ontological concepts of source ontologies harmonises underlying semantic conflicts, thus providing a homogenous view of data in heterogeneous data sources. The works in [153] and [173] use ontology based semantic matching to find correspondences based on structural and lexical characteristics between databases after their semantics have been expressed into ontologies.

For example, an ontology based OntoGrate system from [153] consists of Web-PDDL, and OntoEngine that uses multiple ontologies as a means of resolving semantic conflicts in heterogeneous databases. The Web-PDDL is an ontology language centred upon expressive first order predicate logic and the OntoEngine consists of a powerful knowledge inferring engine. Ontology usage is illustrated through source ontologies that are created as per the different schemas of databases, which are then matched through correspondences based on structural and lexical characteristics using rules defined in Web-PDDL. Furthermore, the rules defined in Web-PDDL act as bridging axioms to merge all databases schemas together, thus providing a means of data integration. By providing the bridging axioms in Web-PDDL, the OntoEngine is able to resolve semantic conflicts through powerful inferences upon source ontologies.

Similarly, a schema matching framework from [173] is used for the identification and resolution of semantic conflicts between relational schema attributes. Multiple ontologies are used to replicate the participating relational schemas so that the semantics behind each attribute can be extracted in the form of ontological classes and instances describing their structural and lexical characteristics. A semantic matching technique is used to identify correspondences, reconciling any semantic conflicts through their common quantifying ontological parent classes.

In both works multiple ontologies and schema matching prove to avoid the complexity and overheads of integrating underlying heterogeneous databases. The advantage seems to be that no ontology commitment to a single shared global ontology is needed, and that each source ontology, can be developed without the common agreement of all sources. However, in reality the lack of a common vocabulary makes it extremely difficult to compare different source ontologies, thus affecting the true comparison of semantic overlapping and similarities in heterogeneous databases.

3.2.3.2 Ontologies and Semantic Mapping

In the case of *semantic mapping*, semantic conflicts are usually identified through semantic mappings between related concepts of source ontologies, where semantic mappings make the use of a mapping criterion based on similarity measurements to dictate the type of semantic conflict among source ontologies. In order to resolve semantic conflicts, semantic mappings from source ontologies are mapped into ontological concepts of a domain (or upper) ontology, which acts very similar to that of a rich vocabulary of shared semantics describing basic terms of a domain. Semantic access to the domain ontology provides a shared representation of inter-domain knowledge, where underlying semantic conflicts of data sources are reconciled. The works in [51] and [52] use semantic mappings to reconcile differences in semantic definitions and terminological differences in heterogeneous web data.

A multiple ontology based approach from [51] is used as a solution for the reconciliation of data value, schema and data model conflicts between diverse sources of web data at both the syntactic and semantic levels. A domain ontology is created for a particular heterogeneous computing environment and covers all the semantic definitions of all the possible set of terms required for user querying (also known as a query path). Local ontologies are created to represent the web data which may be in the form of repositories or existing ontologies. An algorithm is used to create semantic mappings to relate similar terms from local ontologies according to the terms specified in a query path. The algorithm compares semantically related concepts, attribute and relationships in local ontologies using the query path as a mapping criterion. However, users of the system are expected to choose their query-terms from the domain ontology.

The GeoNis Semantic Mediator from [52] is used to resolve semantic conflicts between the terminologies of geospatial information sources (i.e. terminologies used within city services, local offices, local telecom, public utilities, water and power supply services, etc). The GeoNis Semantic Mediator formally (i) specifies the terminologies of geospatial information sources by translating them into local ontologies, and (ii) uses the relationship between concepts of different information sources (between local ontologies), in order to judge whether semantic conflicts exist (non-semantic equivalence) or not (semantic equivalence). In order to resolve semantic conflicts (if any), each relationship is mapped onto a top-level ontology based on a semantic mapping using fuzzy logic rules for computing the probability of similarity results. However, they do not use concept attributes to check similarity.

3.2.3.3 Ontologies and Ontology Mapping

Resolving semantic conflicts in heterogeneous computational environments is also motivated by the Semantic Web [139], and the maturity of Semantic Web technologies. As we mentioned in chapter 2 (section 2.4.2), the Semantic Web is an continous development of the WWW in order to make sense of the data stored on the WWW, whether it is in web documents, pages, or any other form of web data. Central to making sense of this data is Tim Berner-Lee's belief that in the near future ontologies will give meaning to the data on the Web [192]. Therefore, it is likely that there will be many different ontological models on the Semantic Web, resulting in a number of dissimilar ontologies for the same or overlapping domains, which subsequently generate the problem of resolving ontological mismatches between them

[142, 260 and 261]. Ontological mismatches are resolved through the process of inter-ontology mapping, either based on [57, 59, 249, 262 and 263]:

- *schema mapping* between heterogeneous ontologies,
- *instance mapping* between heterogeneous ontologies, or
- *hybrid mapping* that uses a combination of both schema and instance based ontology mappings.

Inter-ontology mapping based on *schema mapping* tries to infer semantic mappings by creating matches between information related to structure of heterogeneous ontologies, i.e. a semantic mapping can create a match between related topological properties, labels or description of nodes, and structural constraints defined on the schema of the ontologies. On the other hand, inter-ontology mapping based *instance mapping*, tries to infer semantic mappings by creating matches between information related to the information contained in the instances of each element in the ontological schemas of heterogeneous ontologies.

In both cases of inter-ontology mappings (schema or instance based) the semantic mappings are usually inferred through finding maximum similarity measures in the two heterogeneous ontologies being mapped. For example, an inter-ontology mapping method is used in [53], which employs machine learning to integrate ontology instances from heterogeneous ontologies that have same names but refer to different entities. and overlap with each other in terms of their intended meaning. The authors in [53] assume that for a particular domain, there could be one or more ontologies that encode the knowledge of this domain in the form of different concepts, properties and their semantic relations, resulting in ontological mismatches. Therefore, in order to resolve ontological mismatches between heterogeneous ontologies, the authors in [53] use a string based similarity measurement to check similarities based on:

(i) the computation of the subsumption relations between instances in the ontology schema, and
(ii) the examination of object properties of instances, i.e. the distance between semantically related ontological concepts.

Furthermore, the use of a support vector machine classifier as in [264], is trained with these similarity measures, and hence used to identify instances referring to same real entities, which enables creates semantic relations between different ontologies. However, the subsumption relations based on ontological constructs cannot be sufficient in identifying overlapping meaning or similarities in heterogeneous ontologies.

Another example can be found in [54] where an ontology alignment strategy is used for resolving ontological mismatches between ontologies of different applications running on the Semantic Web. Their ontology alignment is divided into two steps: inconsistency detection and action taking. Inconsistency detection involves the use of the TreeDiff algorithm proposed in [265] that identifies the largest common substructure of the two ontologies being compared. This includes detecting differences in ontological concepts or properties using:

- different names (i.e. labels) for the same meaning, and
- same names with different meaning.

Upon the identification of inconsistencies, ontological concepts, properties, axioms, restrictions are mapped into an intermediate representation using the degree of similarity between ontological hierarchies that point to disjointness, superposition, specialisation and equality. The result of ontology alignment is an intermediate representation of semantically related concepts that can be shared by both applications.

However, the similarities thresholds, i.e., how much is necessary to qualify as enough to ignore, tolerate or resolve ontological mismatches are not yet clear and will have to be determined.

The choice of using multiple ontologies avoid the commitment to a single global ontology, and hence give the flexibility of accommodating change such as adding or removing of additional data sources. New data sources can easily be added without the need of modification in existing ontology mappings or in the shared global ontology. Furthermore, each source ontology may be combined with additional semantics making them easier to compare for the resolution of semantic conflicts without losing the integrity of underlying original semantics. Thus, using source ontology to describe each heterogeneous data source naturally preserves the autonomy of underlying data sources and increases the level of data sharing across heterogeneous data sources. However, if one database has a different view on a domain (e.g. by providing another level of granularity) finding the minimal ontology commitment becomes a difficult task. Therefore, the above mentioned schema and instance based ontology mappings for resolving semantic conflicts are sufficient when mapping ontologies in the same level, i.e. where schema and instance based ontology mapping can provide a common layer from which several ontologies can be accessed. The need to consider multiple ontologies in environments in which different views and interpretations of ontological data (e.g. different aggregation and granularity of the ontology concepts) raise the question of comparing ontological concepts, in order to identify overlapping similarities between them.

3.3 Ontological Layering and Semantic Web Technology

As our proposal relies on the use of ontological layering, we aim to resolve semantic conflicts using multiple ontologies built on top of each other. We anticipate that multiple ontologies will be the best way to contribute towards semantic interoperability in PCEs. The use of multiple ontologies will be the most suitable approach to managing the growing abundance of semantics generated in heterogeneous data repositories in PCEs, and specifically make their various interpretations of data explicit. Additional representation formalism through the specification of semantic relationships between ontologies will allow a means of resolving semantic conflicts through ontology mappings between different layers of ontologies. Therefore, we believe it is appropriate to briefly review the research surrounding ontology mapping techniques and ontological reasoning mechanisms that do not necessarily overlay with the issue of resolving semantic conflicts in heterogeneous computing environments.

3.3.1 Ontology Mapping

Ontology mapping in essence is the specification of how concepts in different ontologies are related in a logical sense, and subsequently constitute towards a form of knowledge about the inter-relationship between two ontologies and the domain of discourse they model. Ontology mapping is closely related to the problem of ontology interoperability; in terms of necessitating the combination of distributed and heterogeneous ontologies in order to access multiple ontologies from different systems and understand similar ontologies (i.e. semantically related ontological data). According to Noy [260], the research into ontology mapping can be divided into three broader categories: ontology mapping discovery, declarative formal representation of mappings and reasoning with mappings.

The research in the area of **ontology mapping discovery** specifically deals with ontology mapping discovery, where the focus is on trying to find similarities between two ontologies in order to determine how and which concepts and properties represent similar notions. More commonly referred to as ontology alignment, integration and ontology merging depending upon the application and intended outcome, ontology mapping discovery concerns how we find similarities between two ontologies and how we determine which concepts and properties represent similar notions.

The research in the area of **declarative formal representation of mappings** deals with the representation of mappings and investigates the way in which we can represent ontology mappings between two ontologies in order to enable reasoning with mappings. Depending upon the intended use of the mapping, the representation of ontology mappings may be integral according to their outcome. For example, ontology mappings that may produce a translation of a source ontology into a target ontology, or merge two ontologies into a third ontology will no doubt depend on the way inter-relationships have been defined in ontology mappings. However, in other cases, ontology mappings may be represented as queries or bridging axioms that simply describe how one entity can be mapped or transformed into another, and stored separately from the ontologies they map [266].

In the ontology mapping process, a reasoning system finds the similar concepts between two ontologies and maps the corresponding concepts to each other. Hence, the research in the area **of reasoning with mappings deals** with performing the actual reasoning upon mappings between ontologies, i.e. once the mappings are defined, what do we do with them or what types of reasoning is involved? However, because this research is inter-linked with area of declarative formal representations of ontology mappings, given some formal representation of a mapping between ontologies one may be able to reason with the mapping itself. For example, reasoning may determine such things as the adequacy of a mapping to a given task or application, whether two mappings are equivalent or to query the ontologies that have been mapped [266].

We list ASCO [55], SAT [56], GLUE [57], Cupid [58], QOM [59], Anchor-PROMPT [60] and Chimaera [61] as some of the recent research and approaches in the ontology mapping domain. However, these tools are usually based on either *heuristics* that identify structural and naming similarities between models or using *machine learning* to learn mappings that require feedback from a user to further refine a proposed mapping as opposed to automatic creation of ontology mappings. We guide the reader to the ontology mapping surveys by [15, 247, 248, 249 and 267] in order to gain further knowledge on ontology mapping research and approaches.

3.3.2 Ontology Reasoning

In general, "reasoning" means to derive inferences in the form of logical conclusions from a corpus of explicitly stored information, to solve a range of problems [268]. Hence, in ontology based reasoning, "ontology reasoning" means that we must derive inferences that are sanctioned by the knowledge contained within an ontological model (i.e. an ontology's concepts, instances, axioms and constraints). However, ontology languages are good for describing knowledge adhering to the Open World Assumption (OWA), in which conclusions cannot be derived from an ontological model because its modeled knowledge is considered incomplete. Therefore, several rule languages which adhere to the

Closed World Assumption (CWA) are considered as partners for ontologies, in which rules add expressivity in the form of a closed view of the world i.e. everything which is not derivable from the ontological model is assumed to be false, hence, reducing the level of OWA in ontology reasoning.

Due to the complementary nature of existing ontology and rule languages, a plethora of rule based systems have been developed over the last years driven by the need for rule-based integration of constantly growing Semantic Web data. The issue of building rules either inside or outside OWL ontologies is in practice often underestimated and is an important milestone on the W3Cs agenda for completing the Semantic Web architecture. Subsequently, we explicitly list the works in [269, 270, 271 and 272] as an in-depth description of ontology reasoning using rule-based approaches ranging from deductive rule languages to probabilistic and fuzzy rule approaches. Specifically, the work of Eiter and Ianni [273] overview a number of rule-based formalisms, which can either work with rules built either inside or outside ontologies.

Furthermore, in [274 and 275] examples of ontology reasoning which do not rely on rule based formalisms and ontology knowledge bases are given. Howewer, their practicability in modern computational environments is questionable, because they do not embrace the vision of the Semantic Web.

The authors of [274] discuss the logical foundations of OWL in context-aware applications. They illustrate the various levels of descriptions for general properties of ontological concepts through terminological concepts defined in DL theory such as T-box (intentional knowledge in terms of terminology) and A-box (knowledge base - actual ontology) declarations, Using such DL-based reasoning through ontology query languages such as Resource Description Framework Data Query Language (RDQL)[31] and rule based extensions such as the Jess reasoning engine, the authors also emphasise the benefits of having reasoning support build into the logical basis of the ontology. However, the authors of [275] discuss OWL ontologies as a vehicle in decision making within pervasive spaces of self-care smart homes. They illustrate the limitations of OWL ontologies through ontology design models in which the distribution of computational power and reasoning capabilities are balanced through different OWL modeling constructs. They use ontological constructs and restrictions, the inference mechanism that uses either DL or SWRL rules, and create Java code through OWL Application Programming Interface (OWL API)[32].

Although the authors in both sources [274] and [275], raise different points on the use and practicality of ontology reasoning mechanisms. They both highlight the role of efficient reasoning in terms of the reasonable balance between ontological facts (i.e. the formalism of ontology's concepts, instances, axioms and constraints) and rules as additional expressivity. However, the design decision of ontological models and the number of constituting rules give rise to the trade-off between the expressiveness of OWL and the efficient OWL reasoning support.

[31] *www.w3.org/Submission/RDQL/*
[32] *owlapi.sourceforge.net/*

3.3.3 Managing Multiple Ontologies

The number of multiple ontologies available for modern heterogeneous computing environments gives rise to the problem of managing them. We mention a few works on ontology versioning, ontology storage and distributed ontology architecture as some issues relevant in the context of ontological layering.

Ontology versioning is just one task in managing multiple ontologies. According to Noy and Musen [60] ontology developers now face the same problem as software engineers encountered long ago: versioning and evolution. For example, ontology developers and users must be able to find and compare existing ontologies, reuse complete ontologies or their parts, maintain different versions, and translate between different formalisms. Noy and Musen categorise current ontology-versioning research issues into:

- identifying ontology versions in distributed environments (an example can be seen in [250]),
- explicitly specifying change logs between the different versions of ontologies (examples can be seen in [276 and 277]), and
- determining a set of additional ontology changes that each user-specified change incurs (examples can be seen in [164 and 278]).

Ontologies may contain millions of concepts in complex relationships, thus the need for ontology storage triggers the need for appropriate ontology servers. The majority of research on ontology storage concentrates on ontology servers described as either:

- an integrated tool for building ontologies such OntoEdit [219], WebODE [279] and Protégé-2000 [280], or
- stand alone servers that explicitly deal with storing ontologies such as tools OntoRama [281], Ontosaurus [282] and Ontolingua Server [283].

However, the research on ontology server technology is limited and very much immature compared to the rapid growth in the application of ontologies. We guide the reader to the evaluations of Ahmad and Colomb [284] for a more detailed understanding of the ontology server technology available today.

Lastly, it is worth mentioning that the evaluations in [285] have explored some important design questions in order to consider distributed architectures for heterogeneous ontologies. For instance, they question how one would define the various components of an ontology, and they would be related? Or what the consequences would be for the exchange of expressions between systems using different ontologies?

3.4 Conclusions

In this chapter we have reviewed the most important related works and demonstrated their benefits and drawbacks. To set this thesis into perspective, we have reviewed the various classifications of semantic conflicts as a means to understand where semantic conflicts can occur, i.e. in which type of data repositories and which level of data granularity they are concerned with, e.g. meta-data or data value. We have discovered that semantic conflicts do not easily fall into discrete categories of structural or semantic differences. However, semantic differences usually signal the need for the semantic interpretation of data.

We have discovered that methods and approaches based on the use of federations and global schemas have notably been successful in dealing with structural differences of databases. However, they tend to be rigid and inflexible, in terms of accommodating more semantics for data sharing, which

suggests that they lack semantic richness. Very often information they generate can be too general or too "fine grained". We have also briefly reviewed the use of mediations as an approach to resolving conflicts between heterogeneous databases, and have concluded that the development of them can be expensive in both time and money.

We have also given an insight into the multiple ways ontology based approaches are used to resolve semantic conflicts in traditional databases and information systems to modern computational environments today. It has been discussed that ontologies as a solution to resolving semantic conflicts between heterogeneous databases have no doubt been more successful in resolving semantic differences then previous approaches based on the use of federations, global schemas and mediations. However using ontologies as vocabularies in heterogeneous databases and information systems prove to be a similar semantic approach to data base integration as that of using global schemas in federations. Hence, compromising the level of original semantic contained with databases versus the level of ontological commitment to the global ontology. Furthermore, we have seen the heavy burden on manual intervention for the correct identification and resolution of semantic conflicts, hence hindering the posissibility to automating solutions for resolving semantic conflicts.

We have also extensively discussed the use of multiple ontologies that prove to be more suitable to the dynamic nature of modern computational environments today. We have shown how having heterogeneous data sources described by their own ontologies can significantly eliminate problems of flexibility in terms of adding new data sources, and avoid commitment to a single ontology. However, semantic matching and mappings are limited to a common layer from which multiple ontologies can be accessed, and subsequently suggest the need to consider multiple ontologies.

Finally, as our proposal relies on the use of ontological layering and reasoning mechanisms using the Semantic Web technology, we have reviewed research on ontology mapping and reasoning mechanisms that has identified the issue of having an reasonable balance between ontological facts (i.e. the formalism of an ontology's concepts, instances, axioms and constraints) and rules as additional expressivity. The question of having rules on built either inside or outside ontologies advocates the trade-off between the expressiveness of OWL versus an efficient OWL reasoning support.

Chapter 4

Software Architecture for Resolving Semantic Conflicts through Ontological Layering

In this chapter we describe our proposal, which supports retrievals from various data repositories and resolves semantic conflicts which arise from heterogeneities inherent in them. Our proposal is based on ontological layering, which is in the core of the proposed layered SA. Ontological layering generates Go-CID that ensures correct results of retrievals across heterogeneous data repositories, and secures inference mechanisms for resolving semantic conflicts. Our contribution and the novelty in our proposal are threefold:

1) the SA which accommodates ontological layering is generic and applicable across any domain where we need to manipulate the understanding of the environments where heterogeneous data repositories reside PLUS use the power of user's involvement in retrievals across these repositories;
2) the SA helps to resolve semantic conflicts and achieve data sharing and interoperability in any heterogeneous environment through unique ontological layering, which is based on a set of specific ontological mappings and reasoning performed upon ontological concepts;
3) our proposal leaves heterogeneous data repositories intact in terms of not changing their format and semantics stored in them. Ontological layering enables us to deal with semantic conflicts on an ad-hoc basis through ontological layers, by exploiting the meaning of user's requests for retrievals and the knowledge of the environment where retrievals take place through inference mechanisms.

The layered SA which accommodates ontological layering is presented in section 4.1. We introduce SA software architectural components, which are based on layered software architectural styles' principles, taken from [286] and [287]. Ontological layering is in the core of the SA and different layers have different purpose, i.e. they are generated through different reasoning mechanisms. The lowest layer is a Local Ontological layer, which accommodates Local Ontologies $\{LO_j \mid j = 1, ... n\}$, instantiated through translations of the content and structure of heterogeneous Data Repositories $\{Rep_i \mid i = 1, ... m\}$. All other ontological layers in our SA are dynamically generated from LO_j through a set of specific ontological mappings and reasoning performed upon ontological concepts. Consequently sections 4.1.1 - 4.1.3 detail the way our ontological layers are created from LO_j. Our core layering exists as a consequence of (a) the

existence of semantically related concepts in Rep_i and conflicts triggered by them and (b) our reasoning mechanisms for resolving semantic conflicts through ontological mappings *alignment*, *integration* and *merge*. Section 4.2 lays out our theoretical foundations for classifying similarities between semantically related concepts in heterogeneous Rep_i. We briefly review the term *semantic proximity model*, which characterises *similarities* between related concepts as defined by Sheth and Kashyap [44] and use it in our classification. Section 4.3 introduces the process for creating and deploying SA components which consists of 8 steps. Steps 1-5 "prepare" the semantics essential for creating core ontological layers. Steps 6-8 illustrate the exact way of resolving semantic conflicts through core ontological layering. In section 4.3.2 we give a specific scenario which illustrates heterogeneities of Rep_i, user's involvements in the retrievals across Rep_i and the way we generate core ontological layers for resolving semantic conflicts, generated from similarities of semantically related concepts involved in particular retrievals. Subsections 4.3.3 and 4.3.4 strictly distinguish between the preparation of the semantics essential for creating ontological layering (steps 1-5 of our process) and layering itself (steps 6-8 of our process).

4.1 The Proposal: Software Architecture for Ontological Layering

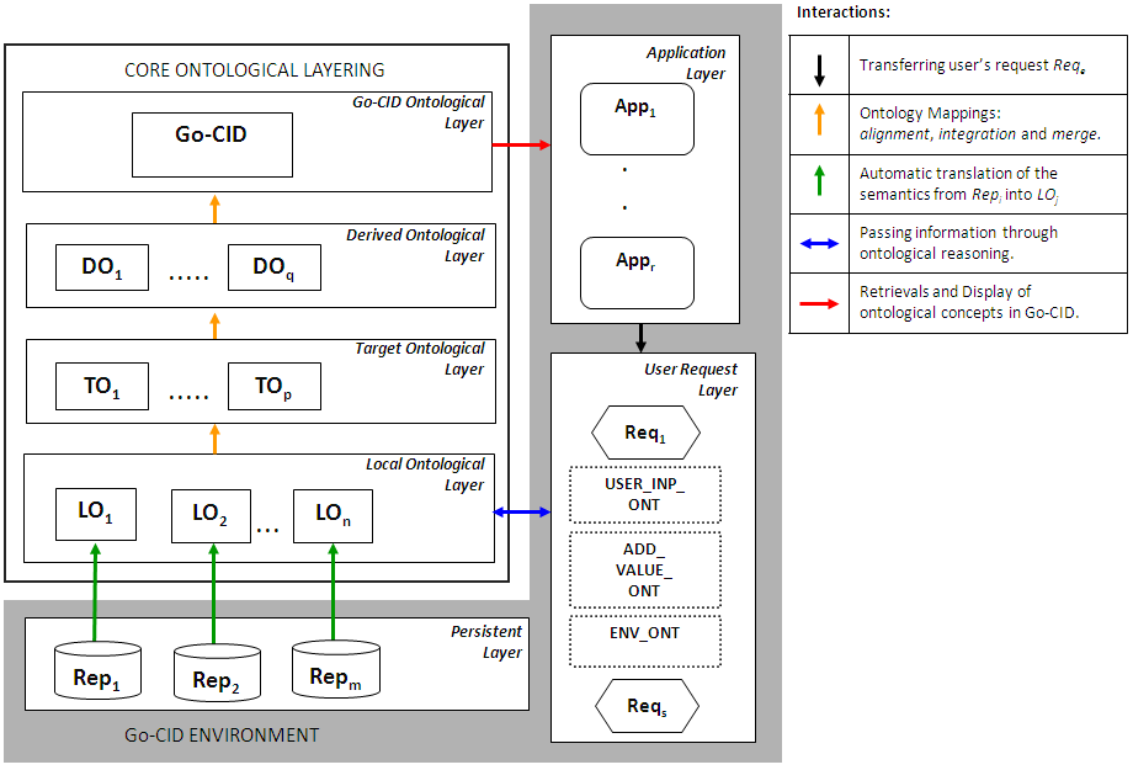

Figure 4.1 *SA for resolving semantic conflicts through ontological layering*

Figure 4.1 illustrates our layered SA which accommodates ontological layering and the Go-CID as its final result. We distinguish between the:

- environment (shaded in grey) in which heterogeneous data repositories reside: Application Layer which accommodates Software Applications $\{App_f \mid f = 1, ... r\}$; User Request Layer with User Requests $\{Req_e \mid e = 1, ... s\}$; Req_i and heterogeneous Rep_i from the Persistence Layer and

- core ontological layers: Local Ontologies LO_j, Target Ontologies $\{TO_k \mid k = 1, ... p\}$, Derived Ontologies $\{DO_g \mid g = 1, ... q\}$ and Go-CID that are responsible for resolving semantic conflicts which appear as a consequence of retrievals across Rep_i.

4.1.1 The Go-CID Environment

The Persistence Layer contains data repositories Rep_i that can be derived from a broad range of data repositories, such as databases in database systems, web services available on the web, file systems, web sites etc. Rep_i store data which are available for various retrievals across any particular domain; they may be of any format/technological specification, plus they may be distributed across any number of locations or computer network nodes. Rep_i have an impact on core ontological layering, because they instantiate the LO_j in the Local Ontological Layer through translations of Rep_i into LO_j. As soon as repositories/data providers subscribe to the environment in which heterogeneous data repositories reside the semantics from Rep_i are transferred into LO_j for the purpose of identifying and resolving semantic conflicts (if they exist) during particular retrievals upon Rep_i .

Software applications App_f, which are either built upon Rep_i or need data which is stored in them, are shown in the Application Layer in Figure 4.1. Their functionality may include retrievals across Rep_i and their interfaces can capture user's involvements, i.e. user requests articulated through application interfaces.

User requests Req_e, as a part of user involvements in these environments, are shown in the User Request Layer. Components from this layer are responsible for storing and interpreting the exact user's involvements in the ENV_ONT, USER_INP_ONT and ADDED_VAL_ONT ontologies. These ontologies correctly interpret the semantics behind user involvements and consequently identify semantically related concepts within Rep_i which (i) might be relevant to a particular user request Req_e and (ii) have impact to the way core ontological layering is created. User's involvements are modeled through their selection of Rep_i and possible Information Types $\{InfType_d \mid d = 1, \ldots, t\}$ stored within Rep_i. $InfType_d$ are actually groups of semantically related concepts which can be found across Rep_i and which are being chosen by the user as a part of his/her request Req_e for retrievals.

4.1.2 Core Ontological Layers

Core ontological layers store ontologies Local Ontologies LO_j, Target Ontologies TO_k, Derived Ontologies DO_g and finally Go-CID. It is obvious that we are moving from the Local Ontological Layer to the Go-CID Ontological Layer in order to:

- address user's request for retrievals across Rep_i , and
- resolve the different types of semantic conflicts as a consequence of user's request for retrievals across Rep_i.

Core ontological layers contain software architectural components which house ontologies responsible for resolving the different types of semantic conflicts. Each layer groups these components (i.e. ontologies) according to the roles they play in resolving semantic conflicts. Therefore, different types of semantic conflicts will dictate a specific set of ontological layers. In other words, the content of Rep_i and user requests Req_e upon Rep_i, reveal where and which types of semantic conflicts we may have, and which

ontologies within our core layers we have to create in order to resolve them. We have already mentioned at the beginning of the chapter that we use layered software architectural styles' principles, taken from [286] and [287]. Thus, our core "layering" means that components from a particular layer are "allowed to use" any component from their adjacent layers [288]. Our creation of core ontological layers starts at LO_j and proceeds upwards to the Go-CID Ontological Layer. In exceptional situations we allow to skip a layer, which depends on the exact Req_e placed upon a set of Rep_i. In other words, some types of semantic conflicts can be resolved without using all core layers.

However, the process of resolving semantic conflicts is based on ontology mapping. In other words, our ontological layering is dependent on ontology mappings. We use the definition which says that:

> *"mapping one ontology onto another means that for each entity (concept C, relation R, or instance I) in ontology O_1, we try to find a corresponding match, which has the same meaning, in ontology O_2"* [289].

The Semantic Web [139] terminology on ontology mappings [15, 247, 248, 249, 260, 266 and 267] allows us to re-use, adapt and itemise ontology mappings [289] into *alignment*, *integration* and *merge*.

Therefore, in our ontological mappings, LO_js are *aligned* into TO_ks, TO_ks are *integrated* into DO_gs and DO_gs are *merged* into the final Go-CID. The results of *alignment(s)* upon LO_j are stored in TO_k. The alignment is triggered by the existence of 'semantically related' concepts in LO_j, which have been carried forward from Rep_i. In other words, the alignment identifies overlapping semantics in 'semantically related' concepts in LO_j and stores them in TO_k. The results of *integration(s)* upon TO_k are stored in DO_g. The *integration(s)* is (are) triggered by the existence of 'semantically similar' concepts in TO_k. In other words, the integration identifies concepts with similar semantics in TO_k and stores them in DO_g. The results of *merge* upon DO_g are stored in Go-CID. The *merge* is triggered by the existence of 'semantically equivalent' concepts in DO_g. In other words, the merge identifies concepts with identical semantics and place them in the Go-CID Ontological Layer. Note: Go-CID concepts must become *'real world'* concepts as they have been initially stored in Rep_i.

4.1.3 Interactions between the Go-CID Environment and Core Ontological Layers

Interactions between layers vary because they have a specific purpose and may be bidirectional. They are explained in bullets below, following their graphical presentation in Figure 4.1.

- The Application and the User Request Layer interact by transferring user's request; which has been captured by application Graphical User Interfaces (GUIs) into ontological concepts within the User Request Layer (i.e. they are stored in the USER_INP_ONT). This interaction is denoted as the directional black arrow between layers.
- Interaction between the Persistence and the Local Ontological layer is done through a one way automatic translation of the semantics from Rep_i into LO_j. There are a number of available tools which can be deployed to perform this translation [147 and 227]. This interaction is denoted as the directional green arrow between layers.
- The results of reasoning upon ontological concepts in the USER_INP_ONT and the ENV_ONT, stored within the User Request Layer, trigger core ontological layering by passing information on

semantic conflicts from the User Request Layer to the Local Ontological Layer. Note: Interactions between these layers are bidirectional and they are based on ontological reasoning which moves ontological individuals between them. This interaction is denoted as the bidirectional blue arrow between layers.

- Interactions between core ontological layers are characterised by ontology mappings *alignment*, *integration* and *merge* which are all based on ontological reasoning. In other words, we match ontological individuals through a variety of reasoning rules in order to reach the Go-CID Ontological Layer and ultimately resolve all semantic conflicts. These interactions are denoted as the directional orange arrows between layers.
- The final interaction between the Go-CID Ontological Layer and the Application Layer appears in the form of "performing retrievals" of ontological concepts, including their individuals from Go-CID, and displaying them in applications GUIs. This interaction is denoted as the directional red arrow between layers.

4.2 Semantic Similarities versus Semantic Conflicts

Information retrievals in our research are concerned with identifying concepts in Rep_i which have to be retrieved, but are semantically related. If these concepts are semantically related, then semantic conflicts may exist between them, which in turn will have to be resolved through our ontological layering. Semantically related concepts of Rep_i may model real world concepts, therefore it is expected that they may have identical or overlapping meaning, which in turn may lead to a number of similarities between them that generate semantic conflicts [10, 74 and 83]. Thus, similarities between Rep_i concepts are often a good indication of whether the semantically related concepts refer to the same real world concepts they model.

Accordingly, in Sheth and Kashyap [44] the term *semantic proximity* characterises the similarity between a pair of objects in a database[33]. Given two objects O_1 and O_2, the *semantic proximity* between them is based on three key elements: (i) the 'context' in which two objects O_1 and O_2 are being compared (i.e. the situation in which a particular semantic similarity holds true between two objects), (ii) the 'abstraction' in which two objects O_1 and O_2 are being related (i.e. the mechanism used to map the domain of the objects to each other) and (iii) the 'domain' in which objects O_1 and O_2 are being defined (i.e. the sets of values from which the objects can take their definitions). Sheth and Kashyap emphasise (i)-(iii) in order to highlight their contribution in identifying, representing and understanding the similarities between related objects. Hence, we choose to use their 'context', 'abstraction' and 'domain' elements of the *semantic proximity* in order to identify the existence of similarities between semantically related data in Rep_i. The concept behind the 'context', 'abstraction' and 'domain' elements are applied to the User Request layer of our SA. Specifically:

- the choices of Rep_i and $InfType_d$ stored within Rep_i, create a 'context' in which semantically related concepts from Rep_i are compared;

[33] where database objects refer to objects in the model world (i.e. a representation or definition in a model world) as opposed to an entity or a concept in the real world [18].

- the ontological concepts in the ADDED_VAL_ONT create the 'abstraction' which helps to show that $InfType_d$ concepts are related to each other. In other words, after instantiating LO_j by converting Rep_i and $InfType_d$ into LO_j, ontological reasoning **groups** semantically related concepts from LO_j into the ADDED_VAL_ONT. Our grouping is equal to creating 'abstractions' because we map the related concepts of $InfType_d$ into the ADDED_VAL_ONT;
- ontological concepts in LO_j store ontological individuals from which related concepts of $InfType_d$ (stored in the ADDED_VAL_ONT) take their values. Our ontological individuals in LO_j is equal to the 'domain' because term "values" in [44] are equal to "ontological individuals" and their instance values, which we group into concepts of the ADDED_VAL_ONT.

Consequently, the ontological individuals from LO_j that have been grouped into ontological concepts in the ADDED_VAL_ONT define (and guarantee) the existence of similarities between semantically related concepts of Rep_i which in turn creates a situation in which a particular semantic similarity holds true across $InfType_d$.

4.2.1 Categorising Semantic Conflicts

Similarities of semantically related concepts in Rep_i may generate a variety of semantic conflicts due to the differences in either the:

- interpretation of semantically related data in respect to their meaning in a given context [20], or
- intended use of semantically related data within a given context [19], or
- way that semantically related data has been modeled in a universe of disclosure [21].

As various studies have demonstrated, it is difficult to resolve semantic conflicts across heterogeneous data repositories without the categorisation of the different types of semantic conflicts which may exist across them [44 and 49]. It was suggested in both works that the categorisation of semantic conflicts involves making their types and similarities explicit.

Table 4.1 in Appendix A.1 summarises the types of semantic conflicts that our SA resolves through core ontological layering. We place it in Appendix A.1 because of its length. Our types of semantic conflicts in Table 4.1 are categorised according to *Naming* and *Structural* conflicts. *Naming* conflicts occur when different names are used to represent semantically related concepts and often happen when names are arbitrarily assigned to concepts. *Structural* conflicts occur when different systems use different data organisation to represent semantically related concepts and apply to the structure of concepts which may be determined by different perspectives and incompatible design specifications.

Naming conflicts are divided into the *Mispelt*/*Case-sensative*, *Homonym*, and *Synonym* types of semantic conflicts. *Structural* conflicts are divided into the *Generalisation*, *Specialisation*, *Isomorphism*, *Union Incompatibility*, and *Aggregation* types of semantic conflicts.

The *Mispelt* semantic conflict refers to the difference in semantically related concepts because of incorrect spellings to describe named concepts[34].

The *Case sensitive* semantic conflict refers to the difference in semantically related concepts because of upper and lower cased letters used to describe named concepts.

[34] See the Glossary for the definition of 'named modeling concepts'.

The *Homonym* semantic conflict refers to the difference in semantically related concepts because of a named concept sounding alike another named concept but having a different meaning to each other.

The *Synonym* semantic conflict refers to the difference in semantically related concepts because of a named concept having the same or nearly the same meaning as another named concept.

The *Generalisation* semantic conflict refers to the difference in semantically related concepts because of a named concept having a 'super types' of their characteristics in terms of describing the same meaning as another named concept.

The *Specialisation* semantic conflict refers to the difference in semantically related concepts because of a named concept having a 'super types' of their characteristics in terms of describing the same meaning as another named concept. The *Isomorphism* semantic conflict refers to the difference in semantically related concepts because of a named concept having a 'different numbers' of their characteristics in terms of describing the same meaning as another named concept.

The *Aggregation* semantic conflict refers to the difference in semantically related concepts because of a named concept having a 'different aspects' of their characteristics in terms of describing the same meaning as another named concept.

The *Union Incompatible* semantic conflict refers to the difference in semantically related concepts because of a named concept having a 'different structures' of their characteristics in terms of describing the same meaning as another named concept.

In Table 4.1 a number of examples are given to help better understand the concepts behind all these kinds of conflicts.

4.2.2 Degrees of Semantic Similarities

In this section we describe our classification of semantically related concepts and the degree of similarity between them. The degree of similarity between semantically related concepts is defined by a taxonomy that judges the level of *semantic proximity* (section 4.2) between semantically related concepts that generate different types of semantics conflicts. Subsequently, the various degrees of similarities are used by our SA in order to compare the types of semantic conflicts during their resolution. Therefore, the classification of semantically related concepts and the degree of semantic similarities between them is necessary in judging the level of overlapping between semantically related concepts.

Our classification of semantically related concepts and the degree of similarity between them is illustrated in Figure 4.2. It contains seven degrees of similarities (or levels of "overlapping") that range from 'Semantic Disjoint (1)' to 'Semantic Equivalence (7)'. Semantically related concepts may belong to the same degree of similarities depending on their level of "overlapping". The 'Semantic Disjoint (1)' has the lowest level of similarity between semantically related concepts and the 'Semantic Equivalence (7)' has the highest. However, we do not measure the degree of similarity between semantically related concepts. We simply determine how far they are from 'Semantic Disjoint (1)' and 'Semantic Equivalence (7)'. In other words, to check the degree of similarity between semantically related concepts we judge how close they are to their semantic equivalence. For example, concepts belonging to the 'Semantic Subset - *contained within* (5)' have a higher degree of semantic similarities than the concepts

which belong to the 'Semantic Likeness (3)'. The definitions for each degree of similarity (1) – (7) between semantically related concepts are given in the next paragraph.

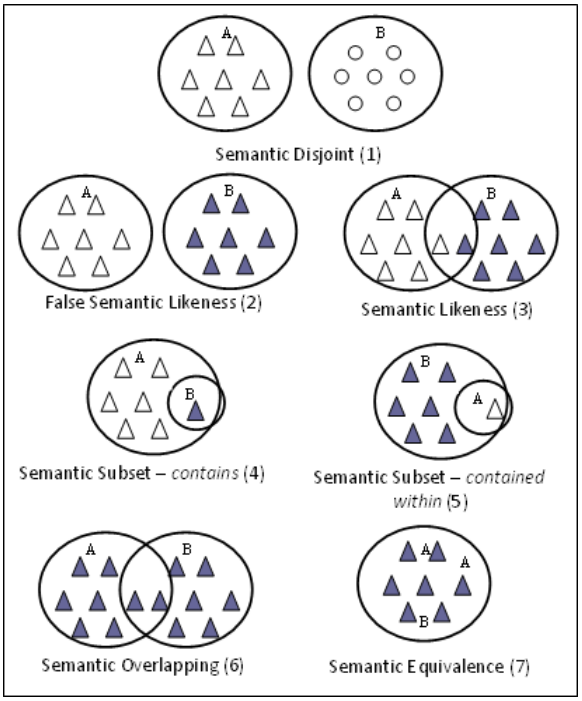

Figure 4.2 *Classification of semantically related concepts and the degree of similarity between them*

Semantic Disjoint (1) is a degree of similarity between two concepts that do not resemble[35] each other and do not have any similarities in a given 'context'. Concepts that belong to Semantic Disjoint (1) are not equivalent to each other and subsequently do not generate any semantic conflicts.

Semantic Equivalence (7) is a degree of similarity between two concepts that resemble each other and are identical to each other in a given context. Concepts that belong to Semantic Equivalence (7) share the exact same meaning. However, they may occasionally generate semantic conflicts because of the Mispelt/Case-Sensitivenaming conflicts. For example, we may have two modeling concepts **'**MEDICAL_SUMMARY' and 'MMMedical_Summary' which model a patient's summary of previous treatments in both cases. Both concepts MEDICAL_SUMMARY' and 'MMMedical_Summary' "resemble each other" (they are obviously Medical Summaries) PLUS have the same meaning: "patient's summary of previous treatments".

False Semantic Likeness (2) is a degree of similarity between two concepts that resemble each other but have no similarities in a given context. Concepts that belong to Semantic False Likeness (2) appear identical to each other but share different meaning and generate the Homonym based naming conflict. For example, we may have a modeling concept called **'**REPORT' which models the combinations of treatments and diagnosis per patient for a general practitioner, whereas 'REPORT' might also represent the list of patients who have positively reacted well to a particular treatment within a hospital. Thus, 'REPORT' has the same name in both environments: 'general practitioner' and 'hospital',

[35] 'resemble' refers to one or more 'characteristics' of the named modeling concept that may look alike.

but these two modeling concepts 'REPORT' do NOT have any similarities in respect to their meaning in a given context (general practitioners' and 'hospital').

Semantic Likeness (3) is a degree of similarity between two concepts that resemble each other and have some similarities in a given context. Concepts that belong to Semantic Likeness (3) share some similar meaning and generate the Synonym based naming conflict. For example, we may have a modeling concept **'**ELECTRONIC_HEALTH_RECORD' that models the patient records for a general practitioner and 'COMPUTATIONAL_RECORD' which models the patient records for a hospital. Thus, 'ELECTRONIC_HEALTH_RECORD' and 'COMPUTATIONAL_RECORD' have semantic similarities between them, they both model patients' records. However they are NOT equivalent to each other because they might model different aspects of patient records. A hospital patient record might have different structures ('characteristics') compared to a general practitioner patient record, for the same patient.

Concepts that belong to Semantic Likeness (3) may also generate the Aggregation based structural conflict. For example, we may have modeling concepts 'PATIENT NAME', 'PATIENT ADDRESS' and 'PATIENT CONTACT NO' that models patient personal details in a patient record for hospital, whereas 'PATIENT FIRST NAME', 'PATIENT LAST NAME', 'PATIENT ADDRESS' and 'PATIENT CONTACT NO' might also model patient personal details in a patient record for a clinic. Thus, the seven modeling concepts have semantic similarities between them because they all model patient personal details. However, it is obvious that there is some kind of "aggregation" between them in terms of their meaning in respect to the context of their meaning. Therefore, they are NOT equivalent to each other because they might model different aspects of patient demographics details (i.e. there is a difference between the concepts 'PATIENT_NAME' and 'PATIENT FIRST NAME' and 'PATIENT LAST NAME' in both environments: 'hospital' and 'clinic'). A hospital patient record might have different structures ('characteristics') compared to a clinic patient record, for the same patient.

Semantic Subset – *contains* (4) is a degree of similarity between two concepts that resemble each other and have some similarities that are 'super-types' between characteristics of the real world concepts they model in a given context. Concepts that belong to Semantic subset - *contains* (4) share some similar meaning in the generalisation of their characteristics and generate the Generalisation based structural conflict. For example, we may have a modeling concept **'**SUMMARY_OF_TREATMENTS' that models a patient's treatments over the last year in a general practitioner's environment, whereas 'PREVIOUS_TREATMENT_SUMMARY' and 'CURRENT_TREATMENT_SUMMARY' models a patient's treatments over the last six months in a clinic environment. Thus, the three modeling concepts have semantic similarities between them because they all model patient treatments and may be available for each patient. However, it is obvious that there is some kind of "generalisation" between them in terms of their meaning in respect to the context of their meaning. 'SUMMARY_OF_TREATMENTS' is a super-type of 'PREVIOUS_TREATMENT_SUMMARY' and 'CURRENT_TREATMENT_SUMMARY', i.e. SUMMARY_OF_TREATMENTS' is a concepts which 'contains' its subsets: 'PREVIOUS_TREATMENT_SUMMARY' and 'CURRENT_TREATMENT_SUMMARY'.

Semantic Subset – *contained within* (5) is a degree of similarity between two concepts that resemble each other and have some similarities that are 'sub-types' between characteristics of the real

world concepts they model in a given context. Concepts that belong to Semantic subset – *contained within* (5) share some similar meaning in the specialisation of their characteristics and generate the Specialisation based structural conflict. For example, we may have a modeling concepts 'PREVIOUS_PRESCRIPTION_SUMMARY' and 'CURRENT_PRESCRIPTION_SUMMARY' that models a patient's prescriptions over the last six months in a clinic environment, whereas **'**SUMMARY_OF_PRESCRIPTIONS' might also models a patient's prescriptions over the last year in a general practitioner's environment. Thus, the three modeling concepts have semantic similarities between them because they all model patient prescriptions and may be available for each patient. However, it is obvious that there is some kind of "specialisation" between them in terms of their meaning in respect to the context of their meaning. 'PREVIOUS_PRESCRIPTION_SUMMARY' and 'CURRENT_PRESCRIPTION_SUMMARY' are sub-types of **'**SUMMARY_OF_PRESCRIPTIONS' i.e. 'PREVIOUS_PRESCRIPTION_SUMMARY' and 'CURRENT_PRESCRIPTION_SUMMARY' are concepts which are 'contained within' its super-type: **'**SUMMARY_OF_PRESCRIPTIONS'.

Semantic Overlapping (6) is a degree of similarity between two concepts that resemble each other and have similarities in a given context. Concepts that belong to Semantic Overlapping (6) share overlapping meaning and generate the Isomorphism based structural conflict. For example, we may have modeling concepts 'LABTEST NAME', 'LABTEST TYPE' and 'LABTEST DATE' that models a patient's lab test in a lab test records for a clinic, whereas 'LABTEST NAME', 'LABTEST TYPE', 'LABTEST DATA' and 'LABTEST DATE' might also model a patient's lab test in a lab test records for a hospital. Thus, the seven modeling concepts have semantic similarities between them because they all model a patient's lab test and may be available for each patient. However, it is obvious that there is some kind of "isomorphism" between them in terms of their meaning in respect to the context of their meaning (i.e. there is a difference between the number of concepts that model a particular patient's lab test in both environments: 'clinic' and 'hospital'). A clinic lab test record might have different structures ('characteristics') compared to a hospital lab test record, for the same patient.

Concepts that belong to Semantic Overlapping (6) may also generate the Union Incompatibility based structural conflict. For example, we may have modeling concepts 'MEDICATION NAME', 'MEDICATION DESCRIPTION' and 'MANUFACTERING ADDRESS' that models a patient's prescribed medication in a medical record for a hospital, whereas 'MEDICATION NAME', 'MEDICATION DESCRIPTION' and 'MANUFACTERING DESCRIPTION' might also model a patient's prescribed medication in a medical record for a clinic. Thus, the six modeling concepts have semantic similarities between them because they all model a patient's prescribed medicine. However, it is obvious that there is some kind of "union incompatibility" between them in terms of their meaning in respect to the context of their meaning (i.e. there is a difference between the concepts 'MANUFACTERING ADDRESS' and 'MANUFACTERING DESCRIPTION' that model a particular patient's prescribed medicine in both environments: 'hospital' and 'clinic').). A hospital medical record might have different structures ('characteristics') compared to a clinic medical record, for the same patient.

4.3 Resolving Semantic Conflicts through Ontological Layering

4.3.1 The Process for Resolving Semantic Conflicts

Our process for resolving semantic conflicts during retrievals across heterogeneous data repositories is given in Figure 4.3. The process is tailored for our SA which accomadates ontological layering and influenced by technologies' imperatives (component technologies and semantic web tools).

In Figure 4.3, steps 1-5 of our process prepare the semantics needed for creating ontological layers. The preparation of semantics includes:

- transferring the semantics from heterogeneous Rep_i into LO_j, and
- preparing the semantics taken from user involvements (request Req_e for retrievals of $InfType_d$ across Rep_i in terms capturing, storing and interpreting the meaning of user's involvements).

Steps 6-8 of our process perform ontology mappings *alignment*, *integration* or *merge* of semantically related concepts. The mappings generate ontologies in the Target, Derived and Go-CID layers and resolve semantic conflicts. The detailed description of each step is given in the paragraphs below.

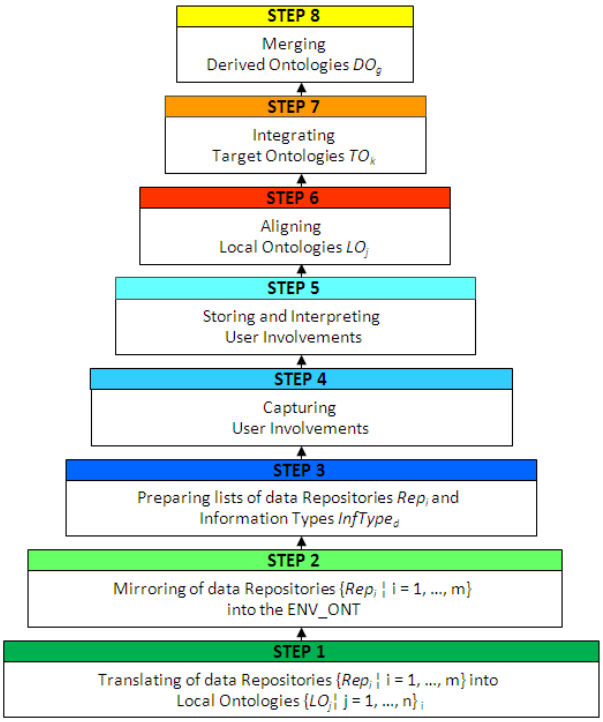

Figure 4.3 *The process for resolving semantic conflicts*

Step 1: Translating Rep_i into LO_j

The content of Rep_i (e.g. schemas and data values in cases of relational databases) are translated into LO_j. The Protégé 3.4 ontological editing toolkit environment [290] is used to generate LO_j with OWL DL [143]. Data translation techniques [227] are responsible for exporting data values from structured database elements into LO_j. Other types of data, such as images, audio and video data, or binary data files (Word, Excel, PDF etc.) are considered to be atomic files with no internal structure, thus they are described through pre-defined ontological concepts stored in the ENV_ONT.

Step 2: Mirroring Meta-Data from Rep_i in the ENV_ONT

Metadata from Rep_i are mirrored within the ENV_ONT concepts. They are "schemas" of Rep_i available for a particular retrieval. Thus, LO_j (which contain translated Rep_i) and the ENV_ONT concepts have the following characteristics: (i) the schemas in ENV_ONT give a list of $InfType_d$, and (ii) LO_j store ontological individuals which make up $InfType_d$. The Protégé 3.4 ontological editing toolkit environment is used to create the ENV_ONT with OWL DL.

Step 3: Preparing lists of Rep_i and $InfType_d$

We prepare a list of all possible repositories Rep_i and $InfType_d$ (metadata stored in the ENV_ONT) through GUIs of software application App_f to enable users' involvements, i.e. securing the retrieval of a particular $InfType_d$ from a particular Rep_i. NetBeans 6.4 Interactive Development Environment (NetBeans IDE)[36] and SWING[37] are used to provide a Java Application Programming Interface (Java API) for creating GUIs.

Step 4: Capturing User Involvements

We deal with user's involvement by capturing user inputs through the application's GUI. App_f use specific 'computations' to populate ontological concepts in the USER_INP_ONT, according to a user's inputs (i.e. captured user "clicks") which is a result of user's choices made upon the list of Rep_i and $InfType_d$. The Protégé 3.4 ontological editing toolkit environment is used to create the USER_INP_ONT with OWL DL. The Protégé-OWL API library [38] is used to provide a Java API for populating OWL DL ontologies.

Step 5: Storing and Interpreting User Involvements

We use the content of the USER_INP_ONT, which stored user's inputs (i.e. captured user "clicks") from the previous step and interpret them by creating concepts in the ADDED_VAL_ONT. Interpreting user's clicks involves reasoning upon ontological concepts in the USER_INP_ONT and the ENV_ONT at the same time, in order to group semantically related ontological concepts from local ontologies LO_j into the ADDED_VAL_ONT. Semantically related ontological concepts are grouped according to the $InfType_d$ relevant for a particular user request Req_e from the USER_INP_ONT. These semantically related concepts may have a number of similarities, which consequently generate semantic conflicts while retrieving from Rep_i. Therefore, grouped concepts in the ADDED_VALUE_ONT may tell us when ("context") and where (Rep_i and $InfType_d$) semantic conflicts exist. They may be at both the metadata and data level, i.e. they can be *naming* and *structural* conflicts (see Table 4.1, Appendix A.1). The SWRL is used to provide additional expressivity to the ontological concepts stored in the ENV_ONT and USER_INP_ONT. The execution of SWRL rules produces deductive inference capabilities for performing the grouping [39] semantically related concepts from LO_j into the ADDED_VAL_ONT.

[36] www.netbeans.org/
[37] http://www.netbeans.org/kb/trails/platform.html
[38] http://protege.stanford.edu/plugins/owl/api/
[39] See the Glossary for the definition of 'grouping' of ontological individuals.

Step 6: Aligning Local Ontologies LO_j

We align 'semantically related' ontological individuals[40] from LO_j, which have been grouped into ontological concepts of the ADDED_VAL_ONT in the previous step, into TO_k. Aligning of ontological individuals means establishing a 'match'[41] between ontological individuals which generate semantic conflicts. Establishing the match indicates that we have 'semantically similar' ontological indviduals[42], thus we create a 'semantic relation'[43] between them. The SWRL is used to provide additional expressivity to the ontological indviduals of LO_j. The execution of SWRL rules produces deductive inference capabilities for performing the '*Low-Level*' reasoning mechanism[44]. Alignment[45](s) may infer either new ontological individuals[46] from LO_j into TO_k or new axioms[47] LO_j into TO_k. Alignments may also transfer existing ontological individuals from LO_j into TO_k.

Step 7: Integrating Target Ontologies TO_k

'Semantically similar' ontological individuals from TO_k that have a 'semantic relation' as a consequence of their alignments are integrated into DO_g. Integrating of ontological individuals from TO_k into DO_g means establishing a 'link'[48] between ontological individuals of TO_k. Establishing the link indicates that we have 'semantically equivalent' ontological individuals[49], thus we create a 'semantic correspondence'[50] between them. As in the previous step, the SWRL is used to provide additional expressivity to the ontological indviduals stored in TO_k. The execution of SWRL rules produces deductive inference capabilities for performing the '*High-Level*' reasoning mechanism[51]. Integrations[52] may assert either existing ontological individuals from TO_k into DO_g or infer new axioms between ontological individuals from TO_k and DO_g.

Step 8: Merging Derived Ontologies DO_g

'Semantically equivalent' ontological individuals from DO_g that have a 'semantic correspondence' as a consequence of their integrations; are merged into the final ontological classes of Go-CID. Merging of ontological individuals from DO_g into classes of Go-CID means establishing a 'correlation' between ontological individuals of DO_g,. The 'correlation'[53] in turn indicates that ontological individuals in Go-CID classes have the same meaning in 'real world' concepts from data repositories Rep_i. The SWRL is used to provide additional expressivity to the ontological individuals stored in DO_g. The execution of SWRL rules produces deductive inference capabilities for performing the '*Post-High-Level*' reasoning

[40] See the Glossary for the definition of 'semantically related' ontological individuals.
[41] See the Glossary for the definition of a 'match' between aligned ontological individuals.
[42] See the Glossary for the definition of 'semantically similar' ontological individuals.
[43] See the Glossary for the definition of a 'semantic relation' between ontological individuals.
[44] See the Glossary for the definition of the 'Low-Level' reasoning mechanism.
[45] See the Glossary for the definition of ontological 'alignment'.
[46] See the Glossary for the definition of 'NEW' ontological individual.
[47] See Glossary for the definition of an ontological 'axiom'.
[48] See the Glossary for the definition of a 'link' between integrated individuals.
[49] See the Glossary for the definition of 'semantically equivalent' ontological individuals.
[50] See the Glossary for the definition of a 'semantic correspondence' between ontological individuals.
[51] See the Glossary for the definition of the 'High-Level' reasoning mechanism.
[52] See the Glossary for the definition of ontological 'integration'.
[53] See the Glossary for the definition of a 'correlation' between merged ontological individuals.

mechanism[54]. Merge[55] relocates ontological individuals from DO_g into the final Go-CID. App_f retrieves ontological concepts from the Go-CID ontology and display it within their GUIs as the output of the requested retrievals. The Protégé-OWL API library is used to provide a Java API for retrieving and displaying all ontological classes from OWL ontologies.

Figure 4.4 gives a complete illustration of:

a) where and when we resolve semantic conflicts in steps 1-8 of our process, and
b) how the classification of semantically related concepts and the degree of similarity between them from Figure 4.2 are used in steps 1-8 of our process.

It must be noted that although the steps in our process are carried out in different places in our SA, each step still follows a sequential order as in Figure 4.3.

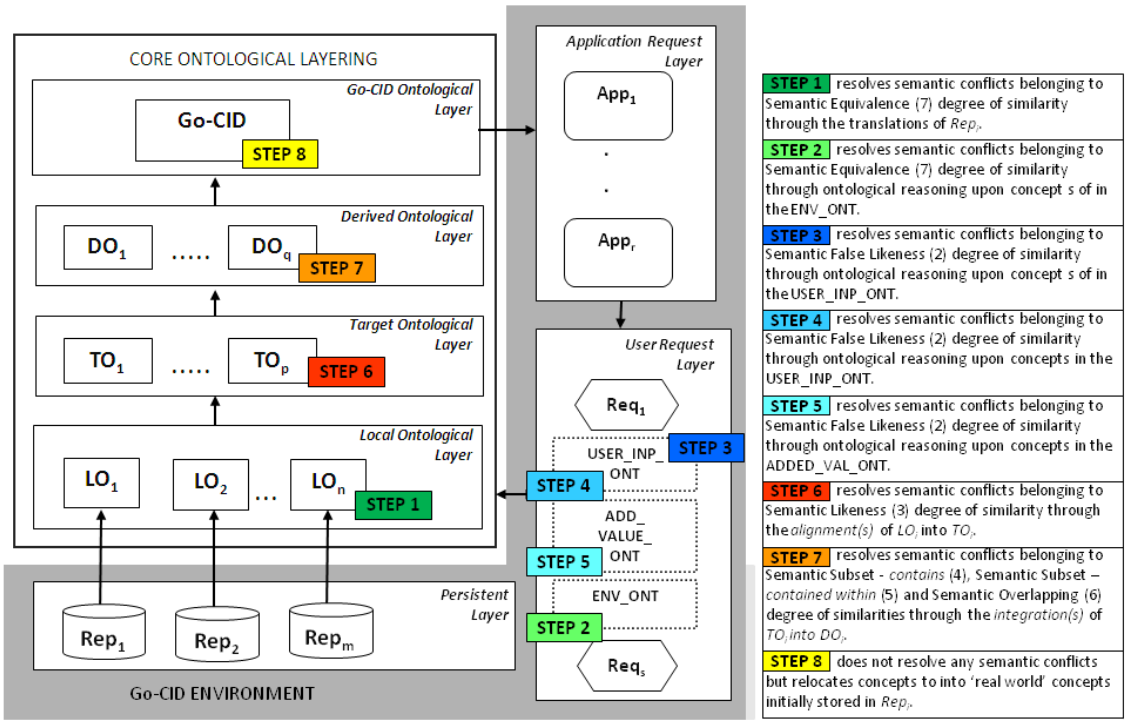

Figure 4.4 *Resolving different types of semantic conflicts generated by degrees of similarities between semantically related concepts through steps 1-8*

To summarise, in Figure 4.4, steps 1 and 2 resolve the Mispelt/Case-Sensitivesemantic conflicts that have been generated by the existence of 'Semantic Equivalence (7)'. Translations of Rep_i into LO_j and mirroring LO_j in the ontological concepts of the ENV_ONT resolve the Mispelt/Case-Sensitivesemantic conflicts in the following way:

i) Mispelt semantic conflicts are resolved through the possibility of changing the names of ontological concepts after their translation from Rep_i.
ii) Case-Sensitivesemantic conflicts are resolved through the option of choosing either lower or upper case characters for the names of ontological concepts which are being translated from Rep_i.

Steps 3, 4 and 5 resolve the Hymonym semantic conflict that has been generated by the existence of 'Semantic False Likeness (2)'. Capturing, storing and interpreting user involvements through reasoning

[54] See the Glossary for the definition of the 'Post-High-Level' reasoning mechanism.
[55] See the Glossary for the definition of ontological 'merge'.

upon the ENV_ONT, USER_INP_ONT and ADDED_VAL_ONT ontological concepts, resolve Hymonyms by creating a 'context' within which a choice of $Rep_i/InfType_d$ infers the semantics relevant to the user request.

Step 6 resolves the Synonym and Aggregation semantic conflicts that have been generated by the existence of 'Semantic Likeness (3)'. This is done though the *alignment (s)* of LO_j because it establishes a 'match' between ontological individuals in order to bring them to a state of being semantically similar to each other.

Step 7 resolves the Generalisation, Specialisation, Isomorphism and Union Incompatibility semantic conflicts that have been generated by the existence of 'Semantic Subset - *contains* (4)', 'Semantic Subset - *contained within* (5)' and 'Semantic Overlapping (6)'. The *integration(s)* of TO_k resolves all four conflicts through the establishment of a 'correspondence' between ontological individuals in order to bring them to a state of being semantically equivalent to each other.

Step 8 relocates ontological individuals from DO_g into the Go-CID, where ontological concepts Go-CID reflect the 'real world' concepts initially stored in Rep_i. Consequently, ontological concepts in Go-CID do not contain semantic conflicts because they have been resolved through previous alignments and integrations.

4.3.2 Example Scenario

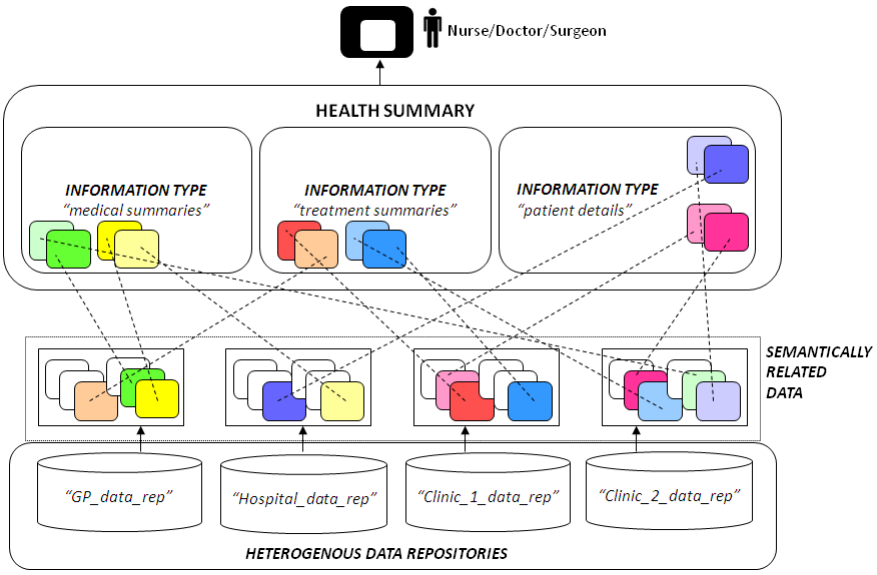

Figure 4.5 *Example scenario of a heterogeneous healthcare environment*

We give a scenario from a healthcare domain which illustrates our proposal. Figure 4.5 shows a number of heterogeneous data repositories (Rep_i) that are distributed across a number of locations such as hospitals, clinics, primary health care surgeries or any other healthcare institution that provides healthcare services.

The heterogeneous data repositories are defined in different formats, technological specifications, they may contain data from structured databases, unstructured data from web, sensors/mobile devices generated data and data created as a result of using semantic web technology. Retrievals across these data

repositories are required by a number of health care professionals such as surgeons, doctors and nurses. They issue requests (Req_e) upon these data repositories (Rep_i) and very often will require patient's medical data/information scattered across them. Furthermore, health care professionals expect that all relevant data repositories (Rep_i) are available for retrievals, and that all relevant medical data/information is stored within them. Therefore, retrievals across these data repositories (Rep_i) start with identifying semantically related concepts stored in them, and can not be correct if we do not resolve semantic conflicts which may exist across them (heterogeneous data repositories).

For example, if a doctor wants to make a decision about a medical diagnosis for a particular patient, then he/she may need to create the patient's health summary. He/she will have to retrieve relevant data, from any of the available data repositories *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep*, which generates the patient's health summary. However, apart from choosing a data repository, the doctor will also have to choose which exact information ($InfType_d$), such as patient's *medical summaries*, *treatment summaries*, and *patient details*, will make up a correct picture of the patient's health summary.

Furthermore, data which is relevant to *medical summaries*, *treatment summaries*, and *patient_details* are scattered across data repositories *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep*, and is very likely to contain identical or overlapping meaning. Thus, the data which:

- make up information ($InfType_d$) such as *medical summaries*, *treatment summaries*, and *patient details*, and consequently
- are essential in creating a correct picture of a particular patient's health summary

are semantically related and may have a number of similarities, which may create semantic conflicts when we try to create patient's health summary. Therefore we must resolve semantic conflicts which may appear within information ($InfType_d$), such as *medical summaries*, *treatment summaries*, and *patient details* if we wish to create the correct results (i.e. the correct health summary) of data retrievals.

The following subsections illustrate steps 1-8 of our process for resolving semantic conflicts by using the scenario above.

4.3.3 Preparing Semantics for Ontological Layering

4.3.3.1 Translating Data Repositories into Local Ontologies

The purpose of step 1 is to transfer the content of data repositories *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep*, into local ontologies *LO_gp*, *LO_hospital*, *LO_clinic_1* and *LO_clinic_2*. If any of these repositories are in the "format" of a relational/post-relational databases, then their corresponding local ontologies will describe their data values in terms of the 'column names', 'attribute names' and 'attribute values'. If any of these data repositories are in the format of XML documents, then their corresponding local ontologies will describe their data values in terms of their 'xml tag names', 'child elements' and 'root elements'. Finally, if any of these data repositories are in the format of RDF/OWL documents' their corresponding local ontologies will describe their data values in terms of their 'URI names', 'RDF object, subject and predicate triples', 'OWL classes', 'OWL instances', 'OWL datatype and object properties' etc. At the end of step 1, local ontologies *LO_gp*,

LO_hospital, *LO_clinic_1* and *LO_clinic_2* will store a number of ontological concepts that are generated from relational databases, XML documents and RDF triples/OWL classes.

4.3.3.2 Mirroring of data Repositories into ENV_ONT

The purpose of step 2 is to mirror the metadata (schemas) from *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep* within the ENV_ONT (ontological) concepts.

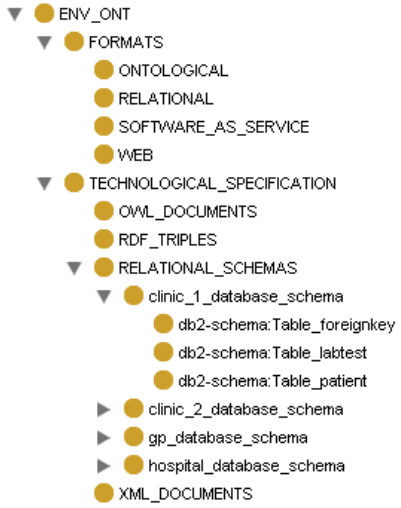

Figure 4.6 *An example of the ENV_ONT ontology from the User Request layer that stores ontological concepts that represent the availability of heterogeneous data repositories {Rep_i | i = 1, …, m} from the Persistent layer*

Figure 4.6 gives us the exact ENV_ONT hierarchies, which has two pre-defined parent ontological classes `TECHNOLOGY_SPECIFICATION` and `FORMATS`. Both of them store ontological concepts that are specific to the meta-data (schemas) of *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep*. The `TECHNOLOGY_SPECIFICATION` class makes provisions for storing a comprehensive list of all the possible technologies needed for environments where *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep* reside. We may have `OWL_DOCUMENTS`, `RDF_TRIPLES`, `RELATIONAL_SCHEMAS` and `XML_DOCUMENTS` as examples of a variety of technological specifications in our heterogeneous environments.

Note: the sub hierarchy of the `TECHNOLOGY_SPECIFICATION` ontological class can be extended by adding new concepts in order to accommodate other characteristics of heterogeneous data repositories. For example, if data repository:

- *Clinic_1_data_rep* is a relational/post-relational database then ontological concepts stored in the `TECHNOLOGY_SPECIFICATION` of ENV_ONT describe their meta-data in terms of the 'database name' and 'table names' as shown in class `clinic_1_database_schema` and its subclass `db2-schema:Table_patient`;
- *Hospital_data_rep* is in a set of XML documents then ontological concepts stored in the `TECHNOLOGY_SPECIFICATION` of ENV_ONT describe their meta-data in terms of the 'xml file names' and 'xml declarations' (due to space restrictions we do not show this in Figure 4.2);

- *Clinic_2_data_rep* is a set of RDF/OWL documents then ontological concepts stored in the TECHNOLOGY_SPECIFICATION of ENV_ONT describe their meta-data in terms of the 'URI names' and 'OWL classes' (due to space restrictions we do not show this in Figure 4.2);

The FORMATS parent ontological class in ENV_ONT makes provisions for storing a comprehensive list of all possible formats of data, from data types exploited in traditional databases to user defined or technology triggered types (images, audio, video, binary data files etc.). Consequently, the ONTOLOGICAL, RELATIONAL, SOFTWARE_AS_SERVICE and WEB subclasses of ENV_ONT are examples of possible formats we may deal with in our heterogeneous environments. Note: the sub hierarchy of the FORMATS ontological class in the ENV_ONT can be extended by adding new concepts in order to accommodate other characteristics of heterogeneous data repositories that are not covered by technological specifications [291 and 292].

4.3.3.3 Preparing lists of Data Repositories and Information Types

The purpose of step 3 is to prepare a list of all possible repositories and information types which support a particular retrieval, across heterogeneous repositories, through GUIs of software application App_f .The GUIs of App_f may provide a number of radio buttons, by giving the list of $\{Rep_i \mid i = 1, \ldots, m\}$ data repositories (in our scenario *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep*) and information types $\{InfType_d \mid d = 1, \ldots, t\}$ (in our scenario *medical summaries*, *treatment summaries*, and *prescription summaries*). Therefore, the doctor's choice of $Rep_i/InfType_d$ will define the nature of doctor's request Req_e.

Note: By giving a list of $Rep_i/InfType_d$ we make provisions for the doctor's involvement in terms of 'clicking' appropriate radio buttons from the App_f GUIs. This will be doctor's input, as an important part of the semantics behind user's involvements in retrievals across repositories and in understanding what it expected from them [46]. We further detail the doctor's involvements in the next section.

4.3.3.4 Capturing User's Involvements

The purpose of step 4 is to capture the doctor's inputs, through the App_f GUI, by populating ontological concepts in the USER_INP_ONT ontology, according to the doctor's "clicks" on radio buttons in the GUI. These "clicks" are actually doctor's choices of Rep_i (in our scenario *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep)* and $InfType_d$ (in our scenario *medical summaries*, *treatment summaries*, and *patient details).*

Figure 4.7 gives us the exact USER_INP_ONT hierarchies, which has two pre-defined parent ontological classes:

- the LIST_OF_DATA_REPOSITORIES class, which corresponds to a set of repositories $\{Rep_i \mid i = 1, \ldots, m\}$ made available for user's selection, and
- the LIST_OF_INFORMATION_TYPES class, which corresponds to information $\{InfType_d \mid i = 1, \ldots, t\}$, available in each of and across these data repositories, which are made available for user's selection.

It is obvious that the data repositories *GP_data_rep, Hospital_data_rep, Clinic_1_data_rep and Clinic_2_data_rep* (Rep_i) from our scenario will be modelled as subclasses of the

LIST_OF_DATA_REPOSITORIES in USER_INP_ONT. The information types *medical summaries, treatment summaries and patient details (*$InfType_d$) from our scenario will be modelled as subclasses of the LIST_OF_INFORMATION_TYPES. However, the sub hierarchy of the LIST_OF_DATA_REPOSITORIES and LIST_OF_INFORMATION_TYPES classes in the USER_INP_ONT can be extended by adding new concepts in order to accommodate any number of available data repositories.

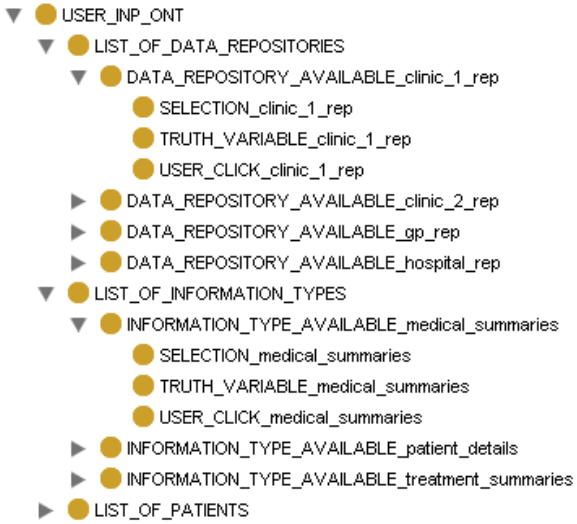

Figure 4.7 *An example of the USER_INP ontology from the User Request Layer that stores ontological concepts that correspond to the available heterogeneous data repositories* $\{Rep_i \mid i = 1, \ldots, m\}$ *from the Persistent Layer*

Figure 4.7 also shows that each subclass of LIST_OF_DATA_REPOSITORIES and LIST_OF_INFORMATION_TYPES contains three further subclasses named SELECTION_xxx/yyy, TRUTH_VARIABLE_xxx/yyy and USER_CLICK_xxx/yyy. "xxx/yyy" denotes the $Rep_i/InfType_j$ to which these three subclasses belong to, i.e. "xxx" may denote *GP_data_rep, Hospital_data_rep, Clinic_1_data_rep and Clinic_2_data_rep* and "yyy" may denote *medical summaries, treatment summaries and patient details.* These three subclasses SELECTION_xxx/yyy, TRUTH_VARIABLE_xxx/yyy and USER_CLICK_xxx/yyy are essential for capturing and storing the results of doctor's "clicks".

The ontological class USER_CLICK_xxx/yyy stores ontological individuals which have been inserted by the application App_f, as we noted at the beginning of this section. App_f populates the USER_CLICK_xxx/yyy class with an individual, which is equal to the doctor's "click" on a particular radio button. For example, if the doctor clicks on the radio button which is placed next to "Clinic_1_data_rep", then ontological class USER_CLICK_xxx/yyy will be populated with an ontological individual named "'USER_CLICK_clinic_1".

The role of ontological classes SELECTION_xxx/yyy and TRUTH_VARIABLE_xxx/yyy is describe in the next subsection, because we use them for storing and interpreting doctor's inputs (in this step we are ONLY capture doctor's inputs).

4.3.3.5 Storing and Interpreting User Involvements

The purpose of step 5 is to use the content of the USER_INP_ONT in order to:

5a) *store* doctor's clicks in the `SELECTION_xxx` subclass of `LIST_OF_DATA_REPOSITORIES` class and `SELECTION_yyy` subclass of `LIST_OF_INFORMATION_TYPES` class, and

5b) *interpret* doctor's clicks through reasoning upon the `TRUTH_VARIABLE_xxx/yyy` and `USER_CLICK_xxx/yyy` classes in the USER_INP_ONT.

In other words we store and interpret user involvements (i.e. user's clicks) through ontological reasoning. The inference created is a result of running SWRL rules[56] in order to:

- determine the doctor's selection of Rep_i and $InfType_d$, based on his/her captured clicks, and
- group semantically related concepts in a chosen $InfType_d$ from the selection of Rep_i.

The inference, as a result of 5a) and 5b) above, guarantees **correct semantics behind user involvements**, because it deduces which Rep_i and which $InfType_d$ from these Rep_i contribute to a correct picture of a particular patient's health summary.

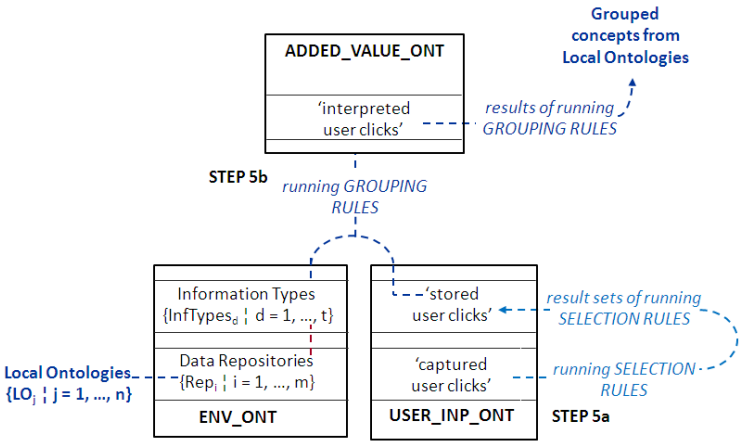

Figure 4.8 *An example of steps 5a and 5b in the process for resolving semantic conflicts*

Figure 4.8 demonstrates our sub-steps 5a and 5b. We distinguish between them by naming step 5a as "*storing*" and step 5b as "*interpreting*" the meaning of user's involvement. Note that the steps of storing and interpreting user involvements in Figure 4.8 show where ontological concepts relevant for Rep_i/$InfType_d$ reside, and where the SWRL rules responsible for inference are run. Therefore, for step 5a we run *Selection* rules and for step 5b we run *Grouping* rules. Furthermore, *Selection* and *Grouping* rules are pre-defined and generated as part of our process for preparing semantics essential for creating core ontological layering. They are both explained in the next two subsections.

4.3.3.5.1 Ontological Reasoning through Selection Rules

In step 5a of our process, 'captured user clicks' are stored in the `USER_CLICK_xxx` subclasses of `LIST_OF_DATA_REPOSITORIES` and the `USER_CLICK_yyy` subclasses of `LIST_OF_INFORMATION_TYPES` in the USER_INP_ONT, where "xxx" denotes the chosen Rep_i, and "yyy" denotes the chosen $InfType_d$. We run a set of *Selection* rules upon the user clicks captured in

[56] See Glossary for the definition of 'SWRL rule'.

the USER_CLICK_xxx/yyy subclasses in order to confirm which Rep_i and $InfType_d$ have been selected. As the result of the running *Selection* rules we can store user clicks in the SELECTION_xxx/yyy subclasses. For example, if the doctor clicks on the radio button which is placed next to *Clinic_1_data_rep* then:

1. the USER_CLICK_clinic_1_rep subclass of the DATA_REPOSITORY_AVAILABLE_clinic_1_rep class (see Figure 4.7) will be populated with an ontological individual "USER_CLICK_clinic_1".
2. the TRUTH_VARIABLE_clinic_1_rep subclass of the DATA_REPOSITORY_AVAILABLE_clinic_1_rep (see Figure 4.7) class will contain an ontological individual with a Boolean value set to 'true',
3. the SELECTION_clinic_1_rep subclass of the DATA_REPOSITORY_AVAILABLE_clinic_1_rep (see Figure 4.7) will store the *result sets* of running *Selection* rules which checks:
   a. if an ontological individual named "USER_CLICK_clinic_1" exists within the USER_CLICKS_clinic_1_rep subclass, and
   b. if an ontological individual named "TRUTH_VARIABLE_clinic_1" in the TRUTH_VARIABLE_clinic_1_rep subclass has a range value set to 'true'.

If both atoms[57] a. and b. equate to being TRUE, then the consequent (head of the SWRL rule) implies that a particular 'user selection' has been made, i.e. that the doctor has selected the data repository *Clinic_1_data_rep*. Table 4.2 gives the $SELECTION\_RULE_i$, that applies to the selection of *Clinic_1_data_rep*. However, a very similar rule can be run to store user clicks when other Rep_i and $InfType_d$ have been selected.

Table 4.2 $SELECTION\_RULE_i$

USER_CLICK_clinic_1_rep(?A) ∧ *TRUTH_VARIABLE_clinic_1_rep(?B)* *→ SELECTION_clinic_1_rep(?B)*

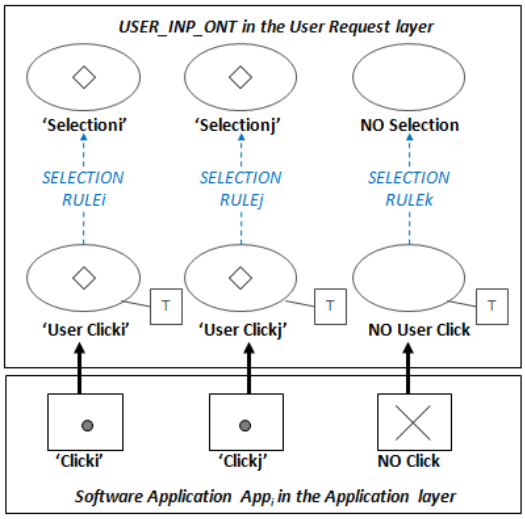

Figure 4.9 *The process of storing user inputs and determining user selections*

[57] See the Glossary for the definition of 'atoms' of ontological individuals.

Figure 4.9 illustrates step 5a as outlined in 1.- 3. above. '$Click_i$' and '$Click_j$' are examples of the doctor's selection of Rep_i/$InfType_d$. Consequently, if no Rep_i/$InfType_d$ have been selected, (i.e. `NO User Click`), the running of $SELECTION\_RULE_k$ would result in an empty result set of running a *selection* rule (SWRL).

4.3.3.5.2 Ontological Reasoning through Grouping Rules

In step 5b of our process, we run a set of *Grouping* rules upon the `SELECTION_xxx/yyy` subclasses of the USER_INP_ONT and ENV_ONT. *Grouping* rules move a selection of ontological individuals which make up $InfType_d$ from Rep_i (LO_j), into ontological concepts of the ADDED_VAL_ONT. In other words, our grouping is a mechanism of interpreting the doctor's clicks in terms of understanding which $InfType_d$ appears in which combinations of Rep_i selected by the doctor. We remind the reader that, when we run grouping rules, we use a metadata from ENV_ONT, but we group actual data from LO_j, which mirror Rep_i.

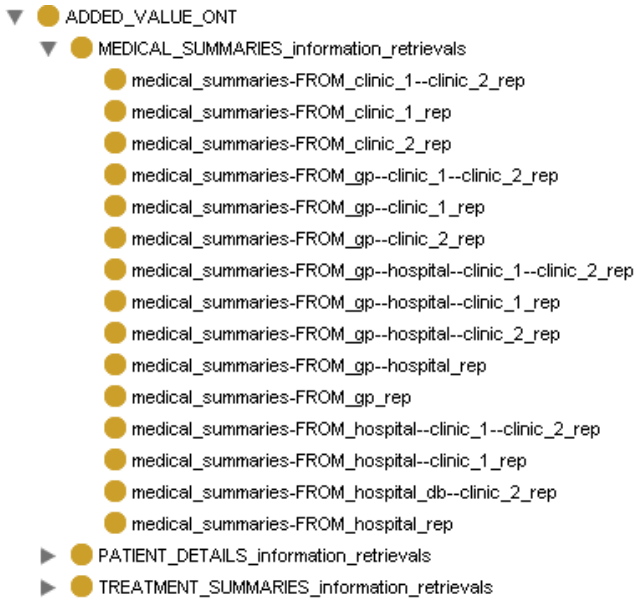

Figure 4.10 *An example of the USER_INP ontology from the User Request Layer that stores ontological concepts that correspond to the available heterogeneous data repositories {Rep_i | i = 1, …, m} from the Persistent Layer*

Figure 4.10 gives us the ADDED_VAL_ONT hierarchies, which stores the result of running *Grouping* rules. The subclasses:

- `MEDICAL_SUMMARIES_information_retrievals`,
- `PATIENT_DETAILS_information_retrievals`, and
- `TREATMENT_SUMMARIES_information_retrievals`

will accommodate ontological individuals from *LO_gp*, *LO_hospital*, *LO_clinic_1* and *LO_clinic_2* that make up information types *medical summaries*, *patient_details* and *treatment summaries.*

However, Figure 4.10 shows an excerpt from much longer list of subclasses in ADDED_VAL_ONT hierarchy. It shows what the subclasses of `MEDICAL_SUMMARIES_information_retrievals` would be. It is obvious that they mirror all possible combinations of repositories Rep_i where the chosen $InfType_d$ of *medical_summaries* can be found. If the doctor clicks the radio buttons next to

Clinic_1_data_rep, *GP_data_rep* as the only sources of *medical summaries*, the doctor might be interested in, then the corresponding subclass `medical_summaries-FROM_gp--clinic_1_rep` of the `MEDICAL_SUMMARIES_information_retrievals` will be populated with ontological individuals from both *LO_gp* and *LO_clinic_1*.

The symbol "--" in the naming convention for the subclasses of the `MEDICAL_SUMMARIES_information_retrievals` class means that a combination of more than one repository has been chosen by the doctor for a particular retrieval. Therefore in the `medical_summaries-FROM_gp--clinic_1_rep` subclass we denote that *medical_summaries* has been chosen as an information type from *Clinic_1_data_rep* and *GP_data_rep.*

Finally it is obvious that only ONE subclass of `MEDICAL_SUMMARIES_information_retrievals` class can be populated at any instance of running step 5b.

4.3.3.5.3 The Reasoning Mechanism behind Grouping Rules

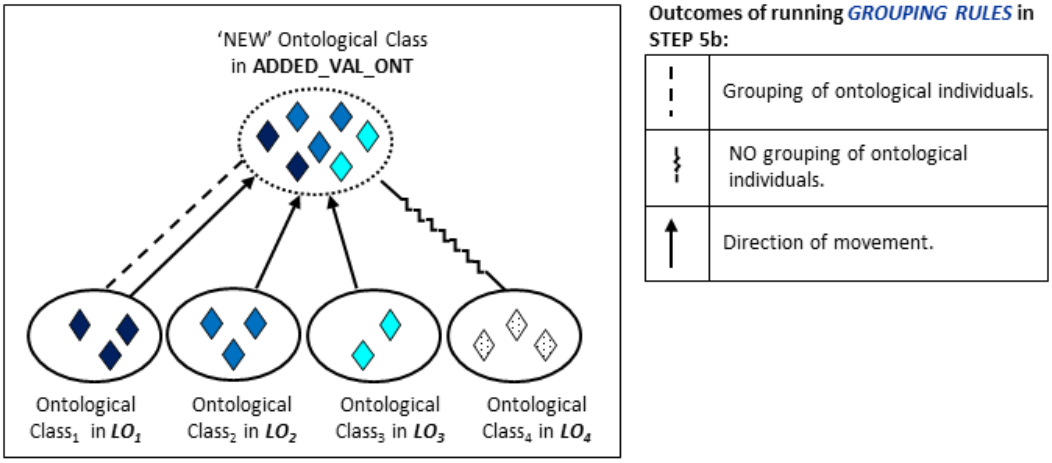

Figure 4.11 *The inference as a result of running Grouping rules in step 5b) of our process for resolving semantic conflicts*

It is obvious that *Grouping* rules secure inference in step 5b, which is graphically presented in Figure 4.11. Ontological individuals from Ontological Class$_1$, Class$_2$, and Class$_3$, in LO_1, LO_2, LO_3 are moved into a 'NEW' ontological class[58] in the ADDED_VAL_ONT. The choice of ontological individuals moved into the `NEW` ontological class depends on the doctor's clicks, i.e. his/her choice of Rep_i/$InfType_d$. Consequently, Class$_4$ from LO_4 in Figure 4.11 has not been chosen by the doctor, therefore its individuals have not been moved.

Grouping is shown in our diagram as a black broken line between ontological classes and NOT ontological individuals stored within them. This is because ontological individuals cannot exist without their classes; therefore they are moved according to a set criteria[59] which must be met by ontological individuals in order to be moved into (or to belong to) a `NEW` ontological class in the ADDED_VAL_ONT. Ontological individuals which cannot be moved between classes, remain in their

[58] See the Glossary for the definition of 'NEW' ontological class.
[59] See Glossary for the definition of 'set criteria' for class membership.

original place, which is illustrated by a black zigzag shaped line, with NO direction (the case of 'Ontological $Class_4$'). Thus, ontological individuals of $Class_4$ do not contribute to the grouping.

The rationale for our reasoning described in Figure 4.11, which secures grouping through SWRL *Grouping* rules is in bullets below:

- the direction of the movement is important (we use a black arrow). We must know where the ontological individual is moved from and where it has to go to;
- ontological individuals can only be moved into a `NEW` ontological class if they meet the *set criteria* for being a member of that `NEW` ontological class. In other words, a `NEW` ontological class may act like a 'secure space' in order to accommodate only ontological individuals that meet their *set criteria* for a `NEW` class membership;
- if at least one ontological individual within a particular class ($Class_1$, $Class_2$, $Class_3$ and $Class_4$) does not meet the *set criteria* for being grouped into a particular class, then no ontological individuals are moved from that class. This is because all ontological indvdiuals must meet the *set criteria* in order for grouping to be perfomed, i.e. all ontological indvdiuals must meet the *set criteria* in order to be moved into a `NEW` ontological class;
- ontological individuals can be moved between ontological classes if there is an *ontological property*[60] between these classes which define their relationship in terms of specifying '*domain*' and '*range*' values[61] for ontological individuals of these classes. Note: we have to become technology specific because our reasoning mechanism is dependent on the expressivity of OWL DL and SWRL.

Table 4.3 illustrates one example of a SWRL *Grouping* rule

Table 4.3 *GROUPING_RULE*$_i$

SELECTION_gp_rep(?A) ∧
has_variable_gp_rep(?A, true) ∧
SELECTION_Clinic_1_rep(?B) ∧
has_variable_Clinic_1_rep(?B, true) ∧
SELECTION_medical_summaries_details(?C) ∧
has_variable_medical_summaries_details(?C, true) ∧
LO_GP-MEDICATION (?D, ?E) ∧
LO_GP-TREATMENT (?F, ?G) ∧
LO_Clinic_1-CURRENT_SUMMARY (?H, ?I) ∧
LO_Clinic_1-PRESCRIPTIONS (?J, ?K) ∧
LO_Clinic_1-TREATMENTS (?L, ?M)
→ medical_summaries-FROM-gp--clinic_1 (?E) ∧
medical_summaries-FROM-gp--clinic_1 (?G) ∧
medical_summaries-FROM-gp--clinic_1 (?I) ∧
medical_summaries-FROM-gp--clinic_1 (?K) ∧
medical_summaries-FROM-gp--clinic_1 (?M)

. If it is confirmed that:

- `SELECTION_gp_db` subclass of the `DATA_REPOSITORY_AVAILABLE_gp_db` parent class,
- `SELECTION_clinic_1_db subclass of the DATA_REPOSITORY _clinic_1_db` parent class, and
- `SELECTION_medical_summaries` subclass of the `INFORMATION_TYPE_AVAILABLE_medical_summaries` parent class from the

[60] See Glossary for the definition of an 'ontological property'.
[61] See the Glossary for the definition of 'domain and range values' of ontological properties.

USER_INP_ONT (Figure 4.7) contains 'stored user clicks' as a consequence of running *Selection* rules (section 4..4.2.5.1),

than the running of *GROUPING RULE*$_i$ in Table 4.3 will group and move:

- the ontological individuals MEDICATION, DIAGNOSIS, and TREATMENT from *LO_gp,*
- the ontological individuals CURRENT_SUMMARY, PRESCRIPTIONS and TREATMENT from *LO_clinic_1*

into the medical_summaries-FROM_gp--clinic_1 subclass class of the MEDICAL_SUMMARIES_information_retrieval class in the ADDED_VALUE_ONT.

Note: we hard code in *GROUPING RULE*$_i$ all the ontological individuals which are supposed to be grouped as *"medical summary"* and moved into the ADDED_VALUE_ONT. In our full scale implementation, we fully utilise the content of the results of running *Selection* rules in step 5a by using ALL repositories {Rep_i | i = 1, …, m} and ALL information types {$InfType_d$ | d = 1, …, t} which may have been chosen by the doctor.

4.3.3.5.4 Technology-specific decisions for Grouping Rules

Creating and running *Grouping* rules is performed according to principles of Semantic Web technology. Our ontologies are created as OWL files[62] and are at the same time SWRL enabled.

We use OWL restrictions, object properties and SWRL rules in order to perform groupings. OWL restrictions are set up upon object properties[63] of ontological classes, which in turn are specified as relationships between ontological individuals. Thus, the technology-specific decisions in grouping of ontological individuals are:

- OWL restrictions[64] determine the set criteria for which classes are involved in a particular 'relationship' through an object properties 'domain' and 'range' values.
- the object property's 'range' value is set to ontological Class$_1$, Class$_2$, and Class$_3$ from Figure 4.11, in order to specify where to move ontological individuals from.
- the object property's 'domain' value is set to the NEW ontological classes from Figure 4.11, in order to specify where to move ontological individuals into.
- *Grouping* rules use object properties and names of ontological individuals to move ontological individuals from Class$_1$, Class$_2$, and Class$_3$.

Running *Grouping* rules infers that all the ontological individuals from Class$_1$, Class$_2$, and Class$_3$ have been 'classified'[65] as members of the NEW ontological class ('class membership') and are grouped into it. In other words, all the ontological individuals of a particular Ontological Class$_1$, Class$_2$, and Class$_3$ has meet the set criteria imposed through OWL restrictions. Note: we do not show object properties in Figure 4.11 because the purpose of the diagram is not to show a mechanism of running *Grouping* rules, but to show the inference it offers through the movement of ontological individuals[66].

[62] See Glossary for the definition of an 'OWL file'.
[63] See Glossary for the definition of an ontological 'object property'.
[64] See Glossary for the definition of an 'OWL restrictions'.
[65] See Glossary for the definition of 'classification' of ontological concepts.
[66] See the Glossary for the definition of 'movement' of ontological individuals.

4.3.3.6 Adding Value to User's Inputs

Running *Selection* and *Grouping* rules in steps 5a and 5b has resulted in adding more value to the existing semantics of Rep_i. This is achieved through reasoning upon user's inputs and grouping concepts of *LO_gp*, *LO_hospital*, *LO_clinic_1*, and *LO_clinic_2*, according to the doctor's selection of Rep_i/$InfType_d$. We can now understand the doctor's involvements and intent in terms of:

(i) knowing exactly which $InfType_j$ the doctor wishes to retrieve across Rep_i and

(ii) identifying which semantically related concepts in $InfType_d$ are present in Rep_i and relevant to this retrieval.

We would also like to emphasise that ADDING VALUE to user's inputs creates a 'context' in which a particular request for a particular retrieval across Rep_i happens. We have to remind the reader that in section 4.2 we clearly stated that:

- the choices of Rep_i and $InfType_d$ stored within Rep_i, create a 'context' in which semantically related concepts from Rep_i are compared.
- the ontological concepts in the ADDED_VAL_ONT create the 'abstraction' which helps to show that $InfType_d$ concepts are related to each other. In other words, after instantiating LO_j by converting Rep_i and $InfType_d$ into LO_j, ontological reasoning **groups** semantically related concepts from LO_j into the ADDED_VAL_ONT. Our grouping is equal to creating 'abstractions' because we map the related concepts of $InfType_d$ into the ADDED_VAL_ONT;
- ontological concepts in LO_j store ontological individuals from which related concepts of $InfType_d$ (stored in the ADDED_VAL_ONT) take their values. Our ontological individuals in LO_j is equal to the 'domain' because term "values" in [44] are equal to "ontological individuals" and their instance values, which we group into concepts of the ADDED_VAL_ONT.

This means that the impact of user's involvement is significant in terms of identifying semantically related concepts in Rep_i and semantic conflicts which may exist between them. However, the most important outcome from user's inputs (doctor's clicks) is that the content of ADDED_VAL_ONT (ontological concepts and their individuals) trigger the core ontological layering which will in turn resolve semantic conflicts between semantically related concepts in a particular "context". Thus, the ontological individuals moved into the concepts of the ADDED_VAL_ONT will trigger our core ontological layering by passing information on semantic conflicts which exists in this particular context. In the next section we deal with details of core ontological layering.

4.3.4 Core Ontological Layering

This section illustrates steps 6-8 of the process for resolving semantic conflicts. We perform core ontological layering through different types of reasoning mechanisms in order to execute *ontology mappings: alignment*, *integration* and *merge*.

Note: in this section we do NOT use examples from the scenario, given in section 4.3.2 for illustrating our ontological layering. We describe *alignment*, *integration* and *merge* though abstract concepts and their role in each reasoning mechanism that supports them. Their full illustration is given in chapter 5, which uses the same example as in section 4.3.2. Ontology mapping in our core ontological layering is always based on schemas and data from Rep_i and explanations of ontological schemas and

individuals from LO_j. They are both needed in the implementation of Go-CID software applications. Therefore, it is not appropriate to overload this section with implementation details. Furthermore, for a full understanding of our ontology mappings, we have also introduced a specific glossary in order to ease the explanations of certain terms related to semantic conflicts. Therefore, we advise the reader, through the footnotes 1-4 and 18-23 in this chapter, to consult the glossary for definitions of terms, which might have had an ambiguous meaning otherwise. The glossary is also needed because of (i) overloads of certain words (such as "semantic") and (ii) we still do not have a clear consensus on what exactly is "semantic relation", "semantic similarity", "match", "correspondence", etc. in the wider research community. This problem has been exuberated with the arrival of Semantic Web technologies and their numerous applications.

4.3.4.1 Reasoning Mechanisms in Core Ontological Layering

Figure 4.12 illustrates the *Low-Level*, *High-Level* and *Post-High-Level* reasoning mechanisms used in core ontological layers of our SA [293]. The reasoning mechanisms use SWRL reasoning rules as an extension to ontological expressivity in order to manipulate the semantics in ontologies LO_j, TO_k and DO_g [294]. Each reasoning mechanism differs according to the purpose of each ontological layer in terms of which type of ontology mappings are being performed: *alignment*, *integration* or *merge.*

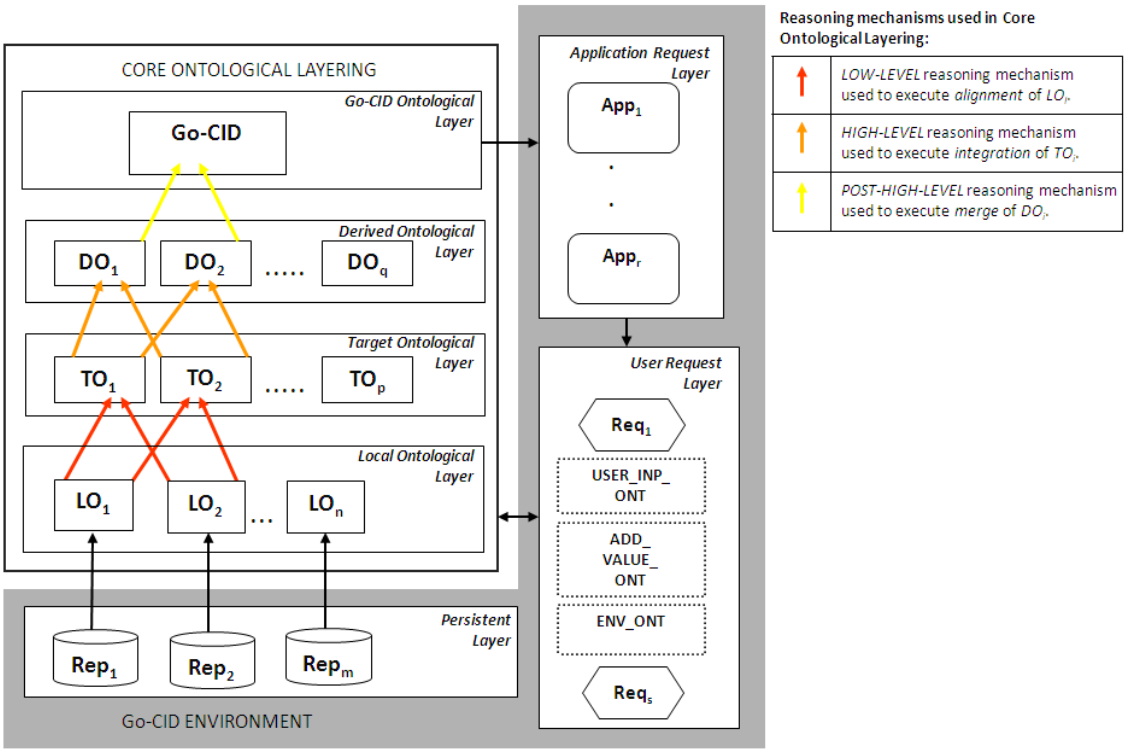

Figure 4.12 *Levels of reasoning mechanisms in core ontological layering*

Each reasoning mechanism in Figure 4.12 supports a specific ontological mapping, which results in the creation of a particular ontological layer. In general, we:

- use the power of ontological modeling in order to capture the 'meaning' behind semantically related data and different types of semantic conflicts between them,

- exploit ontological concepts that mirror semantically related data in Rep_i, through SWRL rules, in order to judge their degree of similarity. Through SWRL rules we are able to see how equivalent semantically related concepts are to each other, and
- achieve automation in terms running any number of SWRL rules in "a chain", and inferring/transferring ontological individuals as we create ontological layers.

The reasoning mechanisms used for ontology mappings, resolve different types of semantic conflicts. The *Low-Level* reasoning executes the *alignment* of LO_js into TO_ks in step 6. The *High-Level* reasoning executes the *integration* of TO_ks into DO_gs in step 7. The *Post-High-Level* reasoning executes the *merge* of DO_gs into ontological concepts of Go-CID in step 8. The use of rule chaining, similar to [295] allows the automatic 'incremental inference' which enables us to"move" from Local Ontological layer towards the final Go-CID layer.

4.3.4.2 Alignment of Local Ontologies

In step 6 of our process, we perform the *Low-Level* reasoning in order execute the ontological alignment of 'semantically related' ontological individuals from LO_j into TO_k. These semantically related ontological individuals generate semantic conflicts, because they have been grouped into subclasses in the ADDED_VAL_ONT in Step 5b of our process.

However, the choice of SWRL rules in the *Low-Level* reasoning, which establishes a 'match' between grouped ontological individuals, is dictated by our classification of semantically related concepts and the degree of similarity between them, which was given in Figure 4.2 from section 4.3.1.

The 'match' indicates that certain ontological individuals are 'semantically similar', thus we can create a 'semantic relation' between them. A 'match' between 'semantically related' ontological individuals should happen/exist, because ontological individuals would not have been grouped into the ADDED_VAL_ONT in the step 5b of our process for resolving semantic conflicts if they have not been semantically related.

The *Low-Level* reasoning creates inference, which is graphically presented in Figure 4.13. It infers or transfers ontological individuals from two or more classes in LO_1 and LO_2 (Ontological $\texttt{Class}_1$, $\texttt{Class}_2$ $\texttt{Class}_3$, $\texttt{Class}_4$, $\texttt{Class}_5$, $\texttt{Class}_6$, $\texttt{Class}_7$ and $\texttt{Class}_8$), which are matched into a CRADLE ontological class[67]. This process of:

- creating a 'semantic relation' as a consequence of establishing a 'match' between two semantically related ontological individuals, and
- transferring or inferring ontological individuals to a CRADLE ontological class

is called the *alignment* between ontological individuals.

Ontological individuals that are matched are named 'originals'[68]. When 'originals' are being transferred into a CRADLE ontological class (where the CRADLE class belongs to TO_1), they are actually transferred as 'duplicates'[69]. Therefore, transferring ontological individuals[70], as a consequence of a 'match', means that we do not move any of our ontological individuals. They remain as 'originals'

[67] See the Glossary for the definition of 'CRADLE' ontological class.
[68] See the Glossary for the definition of 'original' ontological individuals.
[69] See the Glossary for the definition of 'duplicate' ontological individuals.
[70] See the Glossary for the definition of 'transferring' of ontological individuals.

within their original classes and are copied into the CRADLE class as 'duplicates'. However, when a 'match' does not result in transferring of ontological individuals, we say that we rather **infer**[71] a new individual into the CRADLE class. In this case individuals in the CRADLE class initially did not belong to any of 'originals' found in Ontological $Class_1$ and $Class_2$ from LO_1 and LO_2. Transferring of ontological individuals is shown in our diagram as a black broken line between ontological individuals and NOT ontological classes. This is because ontological individuals cannot be transferred without being copied from their 'originals' in Ontological $Class_3$, $Class_4$, $Class_5$, $Class_6$, $Class_7$ and $Class_8$. Inferring of ontological individuals is shown in our diagram as a blue broken line between ontological individuals and NOT Ontological $Class_1$ and $Class_2$ from LO_1 and LO_2.

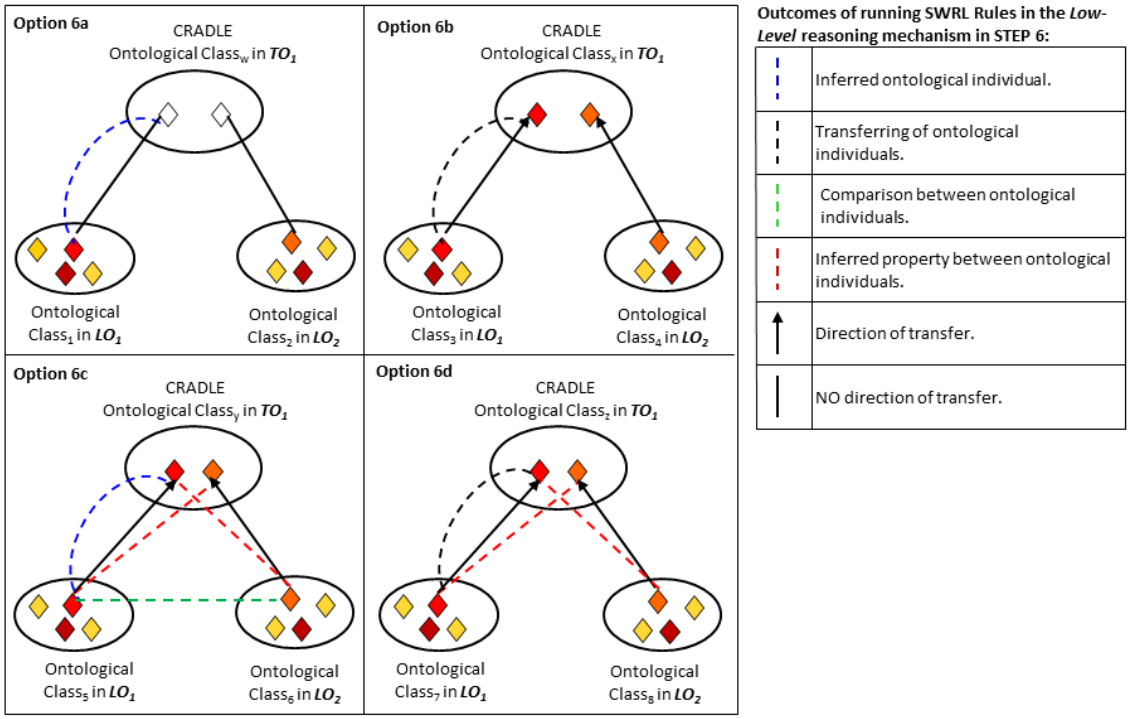

Figure 4.13*The inference as a result of running SWRL rules as part of the Low-Level reasoning which secures ontology alignment*

Note: before ontological individuals are being inferred into a CRADLE ontological class; we may run a comparison[72] between them. The comparison of ontological individuals is shown in our diagram as a green broken line between ontological individuals and NOT ontological classes. This is because the comparison will check if two particular ontological individuals satisfy OWL conditions[73] which determine the strength of the match that exists as a result of grouping of ontological individuals in step 5b of our process. In other words, the result of comparison will give the 'strength' of the match[74] between compared semantically related ontological individuals, which further generates the inferred ontological individual/s (e.g. $Class_w$ of option 6a). Thus, in step 6 of our process for resolving semantic conflicts, different *alignment* options can either:

[71] See the Glossary for the definition of the 'inference' of ontological concepts.
[72] See the Glossary for the definition of 'comparison' between ontological individuals.
[73] See Glossary for the definition of an 'OWL conditions'.
[74] See Glossary for the definition of 'strength' of the match.

6a) infer ontological individuals as a consequence of running a comparison between ontological individuals,

6b) transfer ontological individuals without running a comparison between ontological individuals,

6c) infer axioms in the form of ontological properties between ontological individuals as a consequence of running a comparison between ontological individuals or

6d) infer axioms in the form of ontological properties between ontological individuals without running a comparison between ontological individuals.

6a)-6d) are **our own** set of options that we allow to be used, if we want to claim that we perform *alignment*. We can not predict in advance which one of the options will be used in alignments of real life examples. It will depend on the exact request for the retrieval and semantics (including semantic conflicts) stored in Rep_i.

We expect that one of the options would be sufficient to perform alignment. However, there is a possibility of choosing more than one option in Figure 4.13 and still claim that we perform alignment. This depends on the exact level of similarities between semantically related concepts, which have been classified in Figure 4.2 from section 4.2. In all our examples, option 6b) has always been sufficient for performing alignment.

The rationale for our reasoning described in Figure 4.13, which secures alignment through SWRL *Low-Level* rules is in bullets below:

- option 6a is an example where we do NOT transfer (the solid black lines without any direction) ontological individuals into the Ontological $\texttt{Class}_w$. Instead, we infer ontological individuals (white diamonds), by placing them in $\texttt{Class}_w$, as a consequence of comparing (green broken line) ontological individuals from Ontological $\texttt{Class}_1$ and Ontological $\texttt{Class}_2$. SWRL rules, as a part of the *Low-Level* reasoning mechanisms 'decide' which ontological individuals have to be 'inferred' (white diamonds) into the Ontological $\texttt{Class}_w$ i.e. these inferred individuals do NOT play the role of 'originals' or 'duplicates';
- option 6b shows an example where we DO transfer (directional black arrows) ontological individuals (red and dark orange diamonds) into the Ontological $\texttt{Class}_x$, without running a comparison between ontological individuals from Ontological $\texttt{Class}_1$ and Ontological $\texttt{Class}_2$. It is obvious that there was a 'match' between individuals, marked as red and dark orange diamonds in option 6b, therefore they are transferred into $\texttt{Class}_x$. Note: Ontological individuals in $\texttt{Class}_x$ are actually 'duplicates' copied from their 'originals' in Ontological $\texttt{Class}_1$ and Ontological $\texttt{Class}_2$;
- option 6c shows an example where we DO transfer (one directional black arrows) ontological individuals (red and dark orange diamonds) from Ontological $\texttt{Class}_1$ and Ontological $\texttt{Class}_2$ into the Ontological $\texttt{Class}_y$. However, at the same time, we also infer (broken blue line) an ontological property (red broken line) between ontological individuals from Ontological $\texttt{Class}_1$ and Ontological $\texttt{Class}_2$ as a consequence of running a comparison (broken green line) between them. It is obvious that there was a 'match' between individuals, marked as red and dark orange diamonds in option 6c, therefore they are transferred into $\texttt{Class}_y$. The inferred ontological property strengthens the 'match' and SWRL rules, as a part of *Low-Level* reasoning mechanism

'decide' which ontological properties have to be 'inferred' (broken blue line) into the Ontological $\texttt{Class}_y$;

- option 6d shows an example where we DO transfer (directional black arrows) ontological individuals (red and dark orange diamonds) from Ontological $\texttt{Class}_1$ and Ontological $\texttt{Class}_2$ into the Ontological $\texttt{Class}_z$. However, at the same time, we also infer (broken blue line) an ontological property (red broken line) between ontological individuals from Ontological $\texttt{Class}_1$ and Ontological $\texttt{Class}_2$ WITHOUT running a comparison between them. It is obvious that there was a 'match' between individuals, marked as red and dark orange diamonds in option 6d, therefore they are transferred into $\texttt{Class}_z$.

In options **6a and 6c,** running the comparison between ontological individuals may result in inferring (i.e. shall we say that 6a and 6c guarantee the existence of) either:

- new ontological individuals (white diamond shapes) in Ontological $\texttt{Class}_w$, or
- ontological property between ontological individuals (red and dark orange diamonds) in Ontological $\texttt{Class}_y$.

In options **6c and 6d** we transfer individuals which are matched ('originals') by creating a 'duplicate' of them in the Ontological $\texttt{Class}_z$ and $\texttt{Class}_x$. This type of transferring individuals and creating an ontological property between them strengthens the match between 'duplicates'.

In options **6b, 6c, 6d** we must transfer (i.e. copy) 'originals' that make 'duplicates' of ontological individuals into the CRADLE Ontological $\texttt{Class}_x$, $\texttt{Class}_y$ and $\texttt{Class}_z$. These are the cases where we do NOT infer new ontological individuals. Consequently, not all ontological individuals are becoming 'originals', i.e. some ontological individuals may never have their 'duplicates' created, which means that they are never transferred into the CRADLE ontological class (as in option 6a).

4.3.4.3 Integration of Target Ontologies

In step 7 of our process, we perform the *High-Level* reasoning in order to execute the ontological integration of 'semantically similar' ontological individuals from TO_k into DO_g. These semantically similar ontological individuals generate semantic conflicts, because they have been aligned into LO_js in step 6 of our process. However, the choice of SWRL rules in the *High-Level* reasoning, which establishes a 'link' between similar individuals is dictated by our classification of semantically related concepts and the degree of similarity between them, which was given in Figure 4.2 from section 4.3.1. The 'link' indicates that some ontological individuals are 'semantically equivalent', thus we create a 'semantic correspondence' between them. A 'link' between 'semantically similar' ontological individuals MUST happen/exist, because these ontological individuals would not have been aligned into TO_ks if they were not semantically related.

The *High-Level* reasoning creates inference, which is graphically presented in Figure 4.14. It asserts and transfers ontological individuals that are linked into a COMMON ontological class[75] from two or more ontological classes. This process of:

[75] See the Glossary for the definition of 'COMMON' ontological class.

- creating a 'semantic correspondence' as a consequence of establishing a 'link' between two semantically similar ontological individuals, and
- asserting and transferring ontological individuals into a COMMON ontological class

is called the *integration* between ontological individuals.

When originals are being transferred into a COMMON ontological class (where the COMMON class belongs to DO_1), they are actually transferred as 'duplicates'. Therefore transferring ontological individuals, as a consequence of a 'link', means that we do not move any of our ontological individuals. They remain as 'originals' within their original classes and are copied into the COMMON class as 'duplicates'. Asserting of ontological individuals[76] is shown in our diagram as a black broken line between ontological individuals and ontological classes. This is because ontological individuals have to asserted by transferring their 'duplicates' into the Ontological $\texttt{Class}_a$, $\texttt{Class}_b$, $\texttt{Class}_c$, and $\texttt{Class}_d$, in $\texttt{TO}_1$ and $\texttt{TO}_2$.

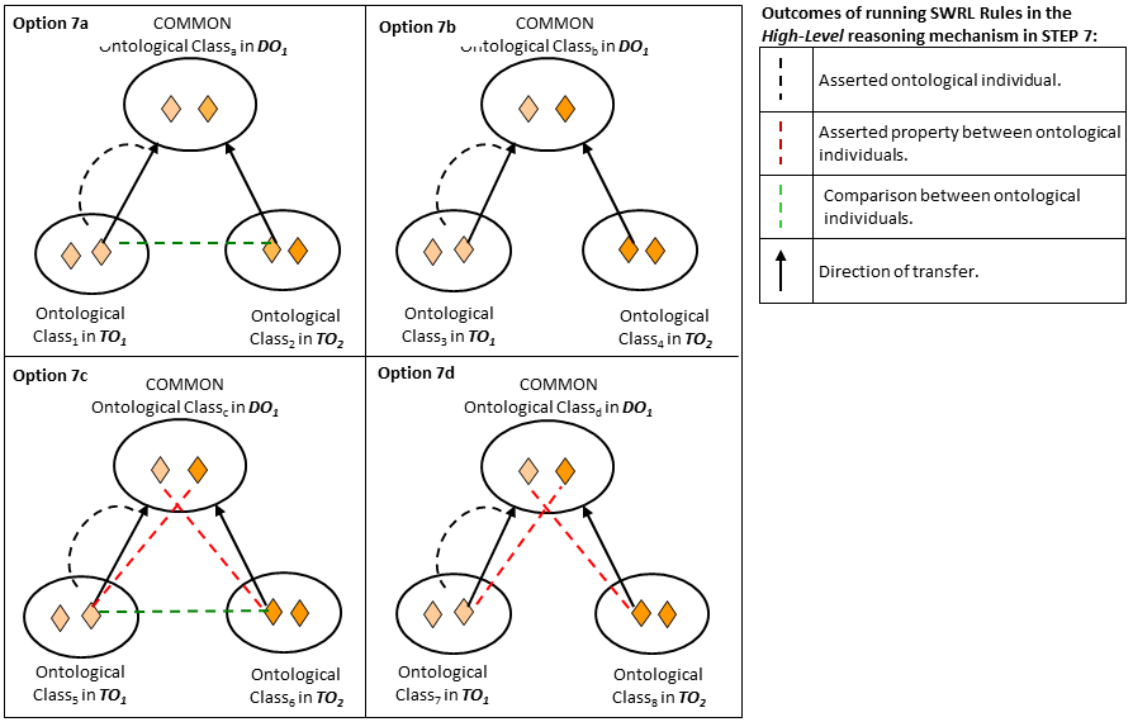

Figure 4.14 *The inference as a result of running SWRL rules as part of the High-Level reasoning which secures ontology integration*

Note: Before ontological individuals are being asserted, we may or may not perform a comparison between them (the green broken line in option 7a and 7c) in terms of checking if two particular ontological individuals satisfy conditions which guarantee their assertion into a COMMON ontological class. Furthermore, we can say that the comparison is needed to determine the "equality" between two ontological individuals that exists as a result of their 'semantic correspondence'. The result of the comparison in options 7a and 7c is asserted ontological individuals in $\texttt{Class}_a$ of option 7a and in $\texttt{Class}_c$ of option 7c. Thus, in step 7 different *integration* options can either:

[76] See the Glossary for the definition of 'assertion' of ontological individuals.

7a) assert and transfer ontological individuals as a consequence of running a comparison between ontological individuals,

7b) assert and transfer ontological individuals without running a comparison between ontological individuals,

7c) assert axioms in the form of ontological properties between ontological individuals as a consequence of running a comparison between ontological individuals or

7d) assert axioms in the form of ontological properties between ontological individuals without running a comparison between ontological individuals.

7a)-7d) are **our own** set of options that we allow to be used if we want to claim that we perform *integration*. We cannot predict in advance which one of the options will be used in the integrations of real life examples. It will depend on the exact request for the retrieval and semantics (including semantic conflicts) stored in Rep_i. We expect that one of the options would be sufficient to perform integrations. However, there is a possibility of choosing more than one option in Figure 4.14 and still claim that we perform integrations. This depends on the exact level of similarities between semantically related concepts, which have been classified in Figure 4.2 from section 4.2. In all our examples, option 7b) has always been sufficient for performing integration.

The rationale for our reasoning described in Figure 4.14, which secures integration through SWRL *High-Level* rules is in bullets below:

- option 7a is an example where we DO transfer (directional black arrows) and assert (black broken line) ontological individuals (light orange and orange diamonds) into the Ontological $\texttt{Class}_a$ as a consequence of running a comparison (green broken line) between ontological individuals from Ontological $\texttt{Class}_1$ and Ontological $\texttt{Class}_2$;
- option 7b shows an example where we DO transfer (one directional black arrows) ontological individuals (light orange and orange diamonds) into the Ontological $\texttt{Class}_b$, without running a comparison between ontological individuals from Ontological $\texttt{Class}_1$ and Ontological $\texttt{Class}_2$. It is obvious that there was a 'link' between individuals, marked as light orange and orange diamonds in option 7b, therefore they are transferred into $\texttt{Class}_b$. Note: Ontological individuals in $\texttt{Class}_b$ are actually 'duplicates' copied from their 'originals' in Ontological $\texttt{Class}_1$ and Ontological $\texttt{Class}_2$;
- option 7c shows an example where we DO transfer (one directional black arrows) ontological individuals (light orange and orange diamonds) from Ontological $\texttt{Class}_1$ and Ontological $\texttt{Class}_2$ into the Ontological $\texttt{Class}_c$. However, at the same time, we also assert (broken black line) an ontological property (red broken line) between ontological individuals from Ontological $\texttt{Class}_1$ and Ontological $\texttt{Class}_2$ as a consequence of running a comparison (broken green line) between them. It is obvious that there was a 'linkh' between individuals, marked as light orange and orange diamonds in option 7c, therefore they are transferred into Ontological $\texttt{Class}_c$. The asserted ontological property strengthens the 'match' and SWRL rules, as a part of *High-Level* reasoning mechanism 'decide' which ontological properties have to be 'inferred' (broken blue line) into the Ontological $\texttt{Class}_c$;

- option 7d shows an example where we DO transfer (one directional black arrows) ontological individuals (light orange and orange diamonds) from Ontological $\texttt{Class}_1$ and Ontological $\texttt{Class}_2$ into the Ontological $\texttt{Class}_d$. However, at the same time, we also assert (broken black line) an ontological property (red broken line) between ontological individuals from Ontological $\texttt{Class}_1$ and Ontological $\texttt{Class}_2$ WITHOUT running a comparison between them. It is obvious that there was a 'link' between individuals, marked as light orange and orange diamonds in option 7d, therefore they are transferred into $\texttt{Class}_d$.

In options **7a and 7c,** running the comparison between ontological individuals may result in either:

- transferred ontological individuals in Ontological $\texttt{Class}_a$, or
- ontological property between ontological individuals (orange and light orange diamond shapes) in Ontological $\texttt{Class}_c$.

In options **7c and 7d** we transfer individuals which are linked ('originals') by creating a 'duplicate' of them in the Ontological $\texttt{Class}_c$ and $\texttt{Class}_d$. This type of transferring individuals and creating an ontological property between them strengthens the link between 'duplicates'.

In options **7a, 7b, 7c, 7d** we must transfer copy 'originals' that make 'duplicates' of ontological individuals into the COMMON Ontological $\texttt{Class}_a$, $\texttt{Class}_b$, $\texttt{Class}_c$ and $\texttt{Class}_d$.

4.3.4.4 Merge of Derived Ontologies

In step 8 of our process, we perform the *Post-High-Level* reasoning in order to execute the ontological merge of 'semantically equivalent' ontological individuals from DO_g into Go-CID. These semantically equivalent ontological individuals **do not** generate semantic conflicts, because they have been previously aligned into LO_js and integrated into TO_ks in steps 6 and 7 of our process. However, the choice of SWRL rules in the *Post-High-Level* reasoning, which establishes a 'correlation' between equivalent individuals indicates that ontological individuals have the same meaning as in 'real world' concepts from Rep_i. A 'correlation' between 'semantically equivalent' ontological individuals should happen/exist, because these ontological individuals would not have been integrated into DO_g's in the previous step.

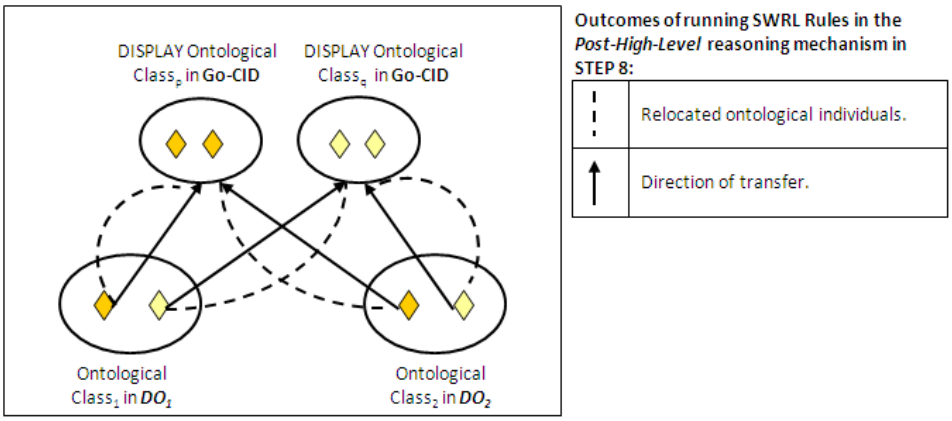

Figure 4.15 *The inference as a result of running SWRL rules as part of the Post-High-Level reasoning which secures ontology merge*

The *Post-High-Level* reasoning mechanism creates inference, which is graphically presented in Figure 4.15. It relocates and transfers ontological individuals that are correlated into a DISPLAY ontological class[77] from two or more ontological classes. This process of:

- establishing a 'correlation' between equivalent individuals, and
- relocating and transferring ontological individuals into a DISPLAY ontological class

is called the *merge* between ontological individuals.

When originals are being transferred into a DISPLAY ontological class (where the DISPLAY class belongs to Go-CID), they are actually transferred as 'duplicates'. Therefore by transferring ontological individuals, as a consequence of a 'correlation', means that we do not move any of our ontological individuals. They remain as 'originals' within their original classes and are copied into the DISPLAY class as 'duplicates'. Relocating of ontological individuals is shown in our diagram as a black broken line between ontological individuals and ontological classes. This is because ontological individuals are relocated into DISPLAY ontological class that that reflect real-world concepts in terms of accommodating ontological individuals that have achieved a semantic-equivalence between them, i.e. ontological individuals that are ready to be retrieved by App_f in order to satisfy a particular retrieval according to the user's selection of Rep_i/$InfType_d$.

The inference, as a result of running SWRL rules in the *Post-High-Level* reasoning mechanism in Figure 4.15. transfers (directional black arrow) individuals (we call them 'originals' marked as yellow and light yellow diamonds), which are relocated, by creating a 'duplicates' of them in Ontological $\texttt{Class}_p$ and $\texttt{Class}_q$.

4.3.4.5 Technology-specific decisions in Ontology Mappings

Creating and running *Low-Level*, *High-Level* and *Post-High-Level* rules are performed according to principles of Semantic Web technology. Our ontologies are created as OWL files and are at the same time SWRL enabled. It is important to draw reader's attention to the power of exploiting OWL modeling constructs and the deployments of a range of SWRL rules in our reasoning. Please note that the text below is very specific to OWL terminology and will require familiarity with OWL modeling constructs and constraints, for its full understanding.

In the *Low-Level* reasoning mechanism used to execute ontological alignment of LO_j into TO_k, OWL conditions and SWRL rules are set up and run upon either object properties or/and datatype properties defined upon ontological classes in LO_j. The only way of transferring individuals into the CRADLE ontological class in TO_k is to use object properties or/and datatype properties as a mechanism for describing the "conditions" under which a 'semantic relation' can be created. Therefore, the technology-specific decisions for ontology *alignment* in step 6 (options 6a, 6b, 6c and 6d) of our process are:

- OWL conditions upon ontological concepts are related to 'allowed literal values' that exactly expresses the criteria for the establishment of a semantic relation. The OWL conditions can be set between any 'range' value for an ontological property. An object property's 'range' value is set to an ontological individual and a datatype property's 'range' value is set to literal value;

[77] See the Glossary for the definition of 'DISPLAY' ontological class.

- the transferring of ontological individuals into a CRADLE ontological class in TO_k is performed through OWL Conditions upon object properties in options 6b, 6c and 6d in Figure 4.13. These conditions are applied to the names of ontological individuals from LO_j in the SWRL rules used for *Low-Level* reasoning.
- an axiom (in the form of a ontological property) is always defined between ontological 'duplicates' and 'originals', where the 'domain' is set as the CRADLE ontological class in TO_k and the 'range' is set as either the class which contains "original" individuals (when using object properties) or a particular literal value for the "originals" (when using data-type properties), i.e. LO_j;
- the comparison of ontological individuals is performed through SWRL rules using an object property or datatype property[78] to compare their 'range' values in options 6a and 6c;
- the inference of an ontological individual/property is performed through SWRL rules, using an object property or datatype property and their 'domain' and 'range' values, as in options 6a and 6c.

In the *High-Level* reasoning mechanism used for ontological integration of TO_k into DO_g, OWL conditions and SWRL rules are set up and run upon either object properties or/and datatype properties defined to ontological classes in TO_k. The only way of asserting individuals into the COMMON ontological class in DO_g is to set up SWRL rules to run upon the result-sets, which is an output from previously run SWRL rule/s, i.e. rule/s that have already been run during the ontological alignment process.

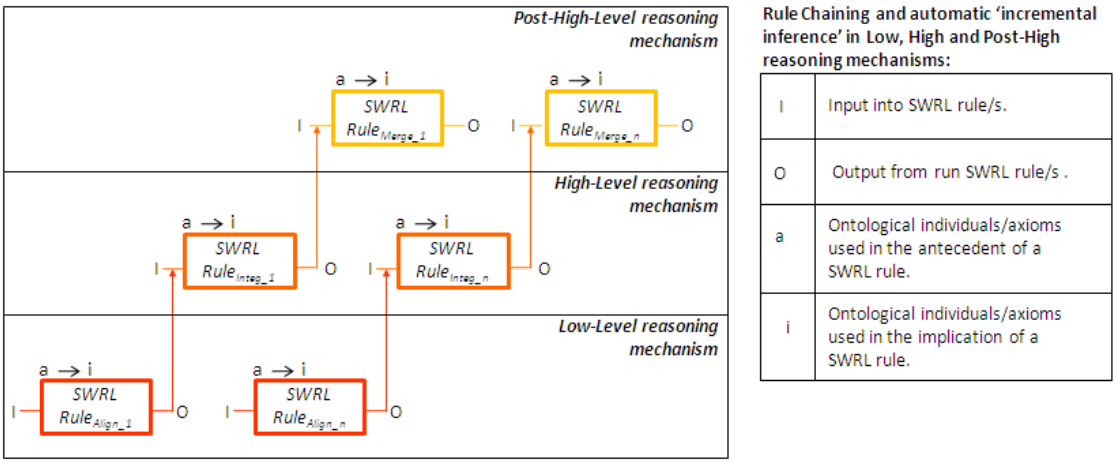

Figure 14.16 *The incremental inference as a result of running SWRL rules as part of the Low-Level, High-Level and Post-High-Level reasoning in ontology alignment, integration and merge*

Figure 4.16 depicts where the inferred/transferred ontological individuals/axioms as a consequence of ontological alignment are used in the antecedent of a SWRL rule in the *High-Level* reasoning mechanism for ontology integration. It allows 'rule chaining'[79], i.e. incremental inference through the implication of the consequent in SWRL rule. Rule chaining exploits existing ontological individuals which are part of a 'semantic relation' that has been created through the *alignment* process. Usage of OWL conditions in the *integration* process is the same as that described in the alignment process above. Therefore, the technology-specific decisions for ontology *integration* in step 7 of our process are:

[78] See Glossary for the definition of an ontological 'datatype property'.
[79] See Glossary for the definition of 'rule chaining'.

- the transferring of ontological individuals is performed through OWL Conditions upon object properties in options 7a, 7b, 7c and 7d that are applied to the name of ontological individuals in SWRL rules;
- an axiom (in the form of a ontological property) is always defined between ontological 'duplicates' and 'originals', where the 'domain' is set as the `COMMON` ontological class in DO_g and the 'range' is set as either the class which contains "original" individuals (when using object properties) or a particular literal value for the "originals" (when using data-type properties), i.e. TO_k;
- the comparison of ontological individuals is performed through SWRL Rules using datatype property to compare their 'range' values, as in options 7a and 7c;
- the assertion of an ontological individual/property is performed through SWRL Rules using object property or datatype property and their 'domain' and 'range' values as in option 7a and 7c.

In the *Post-High-Level* reasoning mechanism, used to execute ontological merge of DO_g into Go-CID, SWRL rules are set up and run upon either object properties or/and datatype properties belonging to ontological classes in DO_g. Note: that NO OWL conditions are used in *Post-High-Level* reasoning mechanism. Thus, the only way of relocating individuals into the `DISPLAY` ontological class in Go-CID is to set up SWRL rules to run upon the result-sets, which are outputs from previously run SWRL rule/s, i.e. rule/s that have already been run during the ontological integration process. Figure 4.16 depicts where the asserted ontological individuals/axioms as a consequence of ontological integration are used in the antecedent of a SWRL rule in the *Post-High-Level* reasoning for ontology merge. Rule chaining exploits existing ontological individuals which are part of a 'semantic correspondence' that has been created through the *integration* process. Therefore, the technology-specific decisions for ontology *merge* in Step 8 of our process are:

- rule chaining in *Post-High-Level* reasoning is the only way of relocating individuals in the ontology merge, where existing ontological individuals, which are part of a 'semantic equivalence' (that has been created through the integration process), are used;
- if for any reason we have skipped layers, i.e. if we skipped ontological alignment and/or integration, then we can re-locate individuals in ontology merge, where existing ontological individuals are part of 'semantic relation' (see section 4.4.3.4) or 'semantic 'correspondence' (see section 4.4.3.5).

4.4 Summary

In this chapter we have defined our SA which accommodates ontological layering and Go-CID software applications. It consists of core ontological layering and its environment, which accommodates a family of ontologies, generated dynamically, in order to support retrievals from various data repositories and to resolve semantic conflicts which arise from heterogeneities inherent in them. Each core ontological layer contains a specific set of ontologies that are created for the purpose of resolving different types of semantic conflicts which appear in our retrievals. We have adopted the Sheth and Kashyap's [44] *semantic proximity* in our classification of semantically related concepts and the way we judge their degree of similarities. Our classification of semantic similarities is used in the process for resolving semantic conflicts, which was given in section 4.3.1. We summarise the main technical characteristics of our SA as:

- supporting heterogeneous environments and allowing any number of data repositories to be included into any of it instances, but does not have to know in advance which repositories are needed in these instances;
- including information from all original sources without the need for changing underlying data repositories;
- core ontological layering is dynamic, i.e. a set of ontological layers are created as soon as a request imposed on the heterogeneous environment has been issued. This means that the core of Go-CID is changeable and corresponds to the semantics stored in the issued requests by applications;
- automation of core ontological layering is based on reasoning upon ontological concepts that directly relate to user's requests for retrievals, thus, making provisions for addressing many issues in heterogeneous environments: from resolving semantic conflicts to deriving more semantics to answer requests.

Chapter 5

Illustration of Ontological Layering

In this chapter we illustrate the SA by demonstrating the example of creating ontological layers through mappings and associated reasoning. We use the same scenario from chapter 4, where a specific example of retrievals of semantically related data, across heterogeneous repositories in healthcare domain, generates semantic conflicts.

In section 5.1 we enrich the scenario from chapter 4, section 4.3.2 in order to accommodate specific details of (i) schemas for heterogeneous data repositories *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep* and (ii) semantically related data in information types *medical summaries*, *treatment summaries*, and *patient details*. Section 5.2 highlights steps 1-5 of our process for resolving semantic conflicts. They include (a) showing the results from the translations of the content and structure of schemas from heterogeneous data repositories into Local Ontologies $\{LO_j \mid j = 1, \ldots n\}$ and the ENV_ONT (sections 5.2.1 and 5.2.2), and (b) storing user involvements in these retrievals within the USER_INP_ONT and interpreting user's inputs in the ADDED_VAL_ONT (sections 5.2.4 and 5.2.5). Section 5.3 highlights steps 6-8 of our process, which generates core ontological layering. They include aligning LO_j into Target Ontologies $\{TO_k \mid k = 1, \ldots p\}$ in order to resolve synonym based naming conflicts (section 5.3.1), integrating TO_k into Derived Ontologies $\{DO_g \mid g = 1, \ldots q\}$ in order to resolve generalisation, specialisation, isomorphism and union incompatibility based structural conflicts (section 5.3.2), and finally merging DO_g into the final Go-CID (section 5.3.3). In section 5.4 we describe our full scale implementation of a Go-CID software application in terms of connecting GUIs through Java Interactive Development Environments (NetBeans 6.4 IDE) with OWL ontologies and automatic execution of SWRL rules, in order to perform ontology mappings and associated reasoning. We discuss some technology-specific decisions in (section 5.5) and finally end the chapter with the summary of our implementation (section 5.6).

5.1 Retrievals across Heterogeneous Healthcare Environments

Dr Smith has to create an ad-hoc health summary for his patient Mrs Jane Flee. He is aware that all relevant data which may be used for created such a summary is scattered across various data repositories, which he has permission to access. Therefore, he is in a position to choose which of these data

repositories would be suitable for his retrieval, on this particular occasion. His choices are therefore limited to four data repositories: Mrs Flee's GP database (*GP_data_rep*), a database from the hospital where Dr Smith is employed and where Mrs Flee was treated (*Hospital_data_rep*), and two databases from a clinic (*Clinic_1_data_rep*) and healthcare center (*Clinic_2_data_rep*), where Mrs Flee chose to have all her regular lab test/s done. Dr Smith is aware of the existence of various medical records, which belong to Mrs Flee, across all these four data repositories. At the same time Dr Smith becomes aware that he will have to retrieve Mrs Flee's medical records from ALL four of them. Therefore, his job is to confirm his own selection of data repositories for this particular retrieval.

Furthermore, Dr Smith does not need Mrs Flee's complete medical records from each of these repositories. He is now in a position to choose which exact information type $\{InfType \mid d = 1, \ldots t\}$: Mrs Flee's *medical summaries*, *treatment summaries*, and Mrs Flee's *demographic and clinical data* will make up a correct picture of the Mrs Flee's health summary for this retrieval.

The heterogeneous data repositories *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep*, where we can find Mrs Flee's medical records are defined in the format of relational databases, where the structures of their computational models are based on:

- relational tables (i.e. entities that denote the subject of interest in the real world which exists physically or conceptually and can be distinctly identified), and
- a number of attributes describing the semantics contained within them (i.e. columns belong to the tables).

We can expect that semantics of data relevant to Mrs Flee's medical records, available across all four of these repositories, and consequently any type of health summaries we may have in them, can be in a variety of database elements that share the same meaning, may be stored under different table/attribute names and belong to different database structures.

5.1.1 Heterogeneous Relational Schemas

Relational schemas for data repositories *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep* are in Figures 5.1-5.4. The SQL source code for each relational schema, including the 'insert' SQL statements for patient Mrs Jane Flee can be found in Appendix A.2. Note: In our relational schemas, we have deliberately created database elements which show identical and overlapping database elements as well as the complex nature of semantic conflicts that may be triggered by them (as introduced in chapter 4, section 4.2.1).

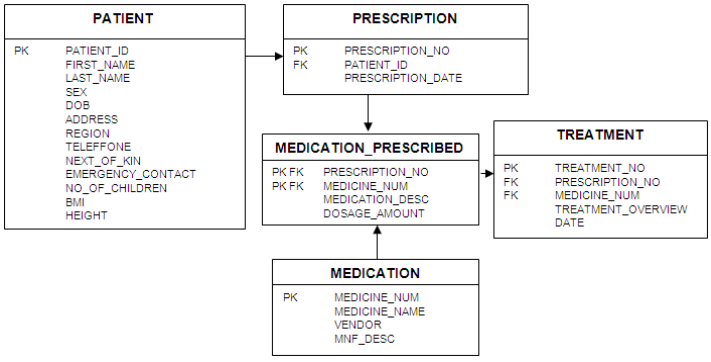

Figure 5.1 *Relational schema for the GP_data_rep*

Figure 5.1 shows the relational schema for the *GP_data_rep*. The PATIENT table contains semantics of demographic and clinical data about patients (i.e. attributes that describe the entity PATIENT). The PATIENT table contains semantics of demographic and clinical data about patients (i.e. attributes that describe the entity PATIENT). In Figure 5.1 the PATIENT table is uniquely identified by the PATIENT_ID attribute. The PRESCRIPTION table contains semantics of prescriptions given to patients, and is uniquely identified by the PRESCRIPTION_ID attribute. It also contains the PATIENT_ID attribute as a foreign key linking to the PATIENT table (i.e. the association between the tables PATIENT and PRESCRIPTION). The MEDICATION table contains semantics of medications given through prescriptions and is uniquely identified by the MEDICINE_NUM attribute. The MEDICATION_PRESCRIBED is a look up table which connects PRESCRIPTION, and MEDICATION tables. Therefore it has a compound identifier which consists of two foreign keys: PRESCRIPTION_ID and MEDICINE_NUM. The TREATMENT table contains semantics of treatments associated to medications prescribed for patients, and is uniquely identified by the TREATMENT_NO attribute. It has also has two foreign keys PRESCRIPTION_ID'and MEDICINE_NUM as a consequence of the 'one to many' relationship between MEDICATION PRESCRIBED and TREATMENT tables.

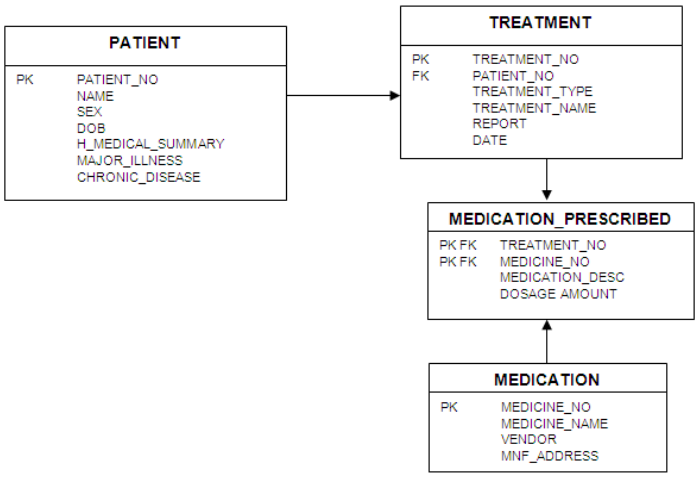

Figure 5.2 *Relational schema for the Hospital_data_rep*

Figure 5.2 shows the relational schema for the *Hospital_data_rep*. The patient table contains semantics based on personal and clinical data about patients, and is uniquely identified by the PATIENT_NO attribute. The TREATEMENT table contains semantics based on treatments for patients, and is uniquely identified by the TREATMENT_NO attribute. The TREATMENT table also contains the PATIENT_NO attribute as a foreign key linking to the patient table. The MEDICATION table contains semantics based on given through prescriptions and is uniquely identified by the MEDICINE_NO attribute. The MEDICATION_PRESCRIBED is a look up table which connects TREATMENT, and MEDICATION tables. Therefore, it has a compound identifier which consists of two foreign keys: TREATMENT_NO and MEDICINE_NO.

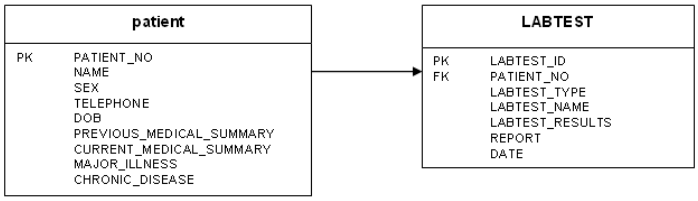

Figure 5.3 *The Relational schema for the Clinic_1_data_rep*

Figure 5.3 shows the relational schema for the *Clinic_1_data_rep*. The `PATIENT` table contains semantics based on personal and clinical data about patients, and is uniquely identified by the `PATIENT_NO` attribute. The `LABTEST` table contains semantics based on lab test(s) for patients, and is uniquely identified by the `LABTEST_ID` attribute. The `LABTEST` table also contains the `PATIENT_NO` attribute as a foreign key linking to the `PATIENT` table.

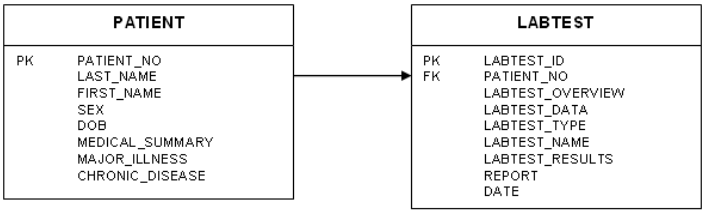

Figure 5.4 *Relational schema for the Clinic_2_data_rep*

Figure 5.4 illustrates the relational schema for the *Clinic_2_data_rep*. The `PATIENT` table contains semantics based on personal and clinical data about patients, and is uniquely identified by the `PATIENT_NO` attribute. The `LABTEST` table contains semantics based on lab test(s) for patients, and is uniquely identified by the `LABTEST_ID` attribute. The `LABTEST` table also contains the `PATIENT_NO` attribute as a foreign key linking to the `PATIENT` table.

5.1.2 Types of Semantic Conflicts in Relational Schemas

We outline below the extent of identical and overlapping data in database elements for the *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep*. We concentrate on the number of on similarities in semantically related data which belong to *medical summaries*, *treatment summaries*, and *patient details* across *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep*.

Semantically related data in *medical summaries* appear when Dr Smith requires retrieving data from:

- `H_MEDICAL_SUMMARY`, `MAJOR_ILLNESS` and `CHRONIC_ILLNESS` from the `PATIENT` table in the *Hospital_data_rep* database,
- `PREVIOUS_MEDICAL_SUMMARY`, `CURRENT_MEDICAL_SUMMARY`, `MAJOR_ILLNESS` and `CHRONIC_ILLNESS` from the `patient` table in the *Clinic_1_data_rep* database,
- `LABTEST_TYPE`, `LABTEST_NAME`, `LABTEST_RESULTS`, and `DATE` from the `LABTEST` table in *Clinic_1_data_rep* database,
- `MEDICAL_SUMMARY`, `MAJOR_ILLNESS` and `CHRONIC_ILLNESS` from the `PATIENT` table in the *Clinic_2_data_rep* database, and

- LABTEST_OVERVIEW, LABTEST_DATA, LABTEST_TYPE, LABTEST_NAME, LABTEST_RESULTS and DATE from the LABTEST table in the *Clinic_2_data_rep* database.

Note: In this example, we do not have data stored in *GP_data_rep* database which is related to *medical summaries.*

We label in bullets (a) and (b) below, overlapping database elements in the information type *medical summaries* across *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep* databases. We indicate the degree of similarity between them (as defined in chapter 4, section 4.2.2) and the types of semantic conflicts they may generate (as defined in chapter 4, section 4.2.1) when retrieving *medical summaries*.

(a) The data stored in MEDICAL_SUMMARY from the PATIENT table in the *Hospital_data_rep* database, models Mrs Flee's clinical data and medical summaries over the last year in the hospital environment where Dr Smith works. The data stored in CURRENT_MEDICAL_SUMMARIES and PREVIOUS_MEDICAL_SUMMARIES from the patient table in the *Clinic_1_data_rep* database, contains Mrs Flee's clinical data and medical summaries over the past 6 months, while she was taking various lab tests in a clinic. We could see through attribute naming, that the data stored in MEDICAL_SUMMARY, CURRENT_MEDICAL_SUMMARIES and PREVIOUS_MEDICAL_SUMMARIES are semantically related. They belong to the Semantic Subset – *contains* (4) degree of similarity and may generate the Generalisation based structural conflict: CURRENT_MEDICAL_SUMMARIES and PREVIOUS_MEDICAL_SUMMARIES from the *Clinic_1_data_rep* database may have been seen by Dr Smith as "parts" of MEDICAL_SUMMARY he would like to add to the Mrs Flee's *medical summaries* he holds within his *Hospital_data_rep* database.

(b) The data stored in LABTEST_ID, PATIENT_NO, LABTEST_TYPE, LABTEST_NAME, LABTEST_RESULTS, REPORT and DATE from the LABTEST table in the *Clinic_1_data_rep* database, models Mrs Flee's lab test(s) carried out over the last two years in a clinic (clinic 1). Their data is semantically related to the data stored in LABTEST_ID, PATIENT_NO, LABTEST_OVERVIEW, LABTEST_DATA, LABTEST_TYPE, LABTEST_NAME, LABTEST_RESULTS, REPORT and DATE from the LABTEST table in the *Clinic_2_data_rep* database, which models Mrs Flee's lab test(s) carried out over the last two years in a healthcare center (clinic 2). Data in these attributes have some semantic similarities, but they are not semantically equivalent to each other. Hence, they belong to the Semantic Overlapping (6) degree of similarity and may generate the Isomorphism based structural conflict.

Semantically related data in *treatment summaries* appear when Dr Smith requires retrieving data from:

- TREATMENT_OVERVIEW and DATE from the TREATMENT table in the *GP_data_rep* database,
- MEDICINE_NUM, MEDICINE_NAME, VENDOR and MNF_DESC from the MEDICATION table in the *GP_data_rep* database,
- DOSAGE_AMOUNT from the MEDICTAION_PRESCRIBED table in the *GP_data_rep* database,
- TREATMENT_TYPE, TREATMENT_NAME and DATE from the TREATMENT table in the *Hospital_data_rep* database,

- MEDICINE_NO, VENDOR and MNF_ADDRESS from the MEDICATION table in the *Hospital_data_rep* database, and
- DOSAGE_AMOUNT from the MEDICTAION_PRESCRIBED table in for the *Hospital_data_rep* database.

Note: In this example, we do not have data stored in *Clinic_1_data_rep* and *Clinic_2_data_rep* databases which is related to *treatment summaries.*

We label in bullets (c), (d) and (e) below, overlapping database elements in the information type *treatment summaries* across *GP_data_rep* and *Hospital_data_rep* databases. We indicate the degree of similarity between them and the types of semantic conflicts they may generate when retrieving *treatment summaries*.

(c) The data stored in MEDICINE_NUM from the MEDICATION table in the *GP_data_rep* database, is an identifier for a particular medicine in Mrs Flee's GP surgery. Its data is semantically related to the data stored in MEDICINE_NO from the MEDICATION table in the *Hospital_data_rep* database, which also identifies a particular medicine in this database. The data stored in MEDICINE_NUM and MEDICINE_NO resemble each other and have semantic similarities. Both of them belong to Semantic Likeness (3) as degree of similarity, and may generate the Synonym based naming conflict.

(d) The data stored in TREATMENT_TYPE, TREATMENT_NAME and TREATMENT_DATE from the TREATMENT table in the *Hospital_data_rep* database, models Mrs Flee's treatments over the last year in Dr Smith's hospital. Their data is semantically related to the data stored in TREATMENT_OVERVIEW and TREATMENT_DATE from the TREATMENT table in the *GP_data_rep* database, that models Mrs Flee's treatments over the last year in her GP surgery. We could see through attribute naming, that the data stored in TREATMENT_TYPE, TREATMENT_NAME and TREATMENT_OVERVIEW are semantically related. They belong to the Semantic Subset – *contained within* (5) degree of similarity and may generate the Specialisation based structural conflict: TREATMENT_OVERVIEW from the *GP_data_rep* database may have been seen by Dr Smith as "parts" of TREATMENT_TYPE and TREATMENT_NAME he would like to add to the Mrs Flee's *treatment summaries* he holds within his *Hospital_data_rep* database.

(e) The data stored in MEDICINE_NUM, MEDICINE_NAME, VENDOR, and MNF_DESC from the MEDICATION table in *GP_data_rep* database, models a Mrs Flee's prescribed medicine in her GP surgery. Its data is semantically related to the data stored in MEDICINE_NO, MEDICINE_NAME, VENDOR, and MNF_ADDRESS from the MEDICATION table in the *Hospital_data_rep* database, which also models Mrs Flee's medicine prescribed in Dr Smith's hospital. These two sets of attributes contain semantically related data which and have semantic similarities because they all model the same concepts: Mrs Flee's prescribed medicine. However, attributes MNF_DESC and MNF_ADDRESS which belong to these two sets are NOT modelling the same semantics: one of them stores data which describes the manufacturer of the prescribed medicine, and the other gives the manufacturer's address. Consequently, data stored in these two sets of attributes belong to Semantic Overlapping (6) degree of similarity and may generate the Union Incompatibility based structural conflict.

Semantically related data in *patient details* appear when Dr Smith requires retrieving data from:

- FIRST_NAME, LAST_NAME, SEX, DOB, ADDRESS, REGION, TELEFFONE, NEXT_OF_KIN, EMERGENCY_CONTACT, NO_OF_CHILDREN, BMI and HEIGHT from the PATIENT table in the *GP_data_rep* database,
- NAME, SEX, and DOB from the PATIENT table in the *Hospital_data_rep* database,
- NAME, SEX, TELEPHONE and DOB from the patient table in the *Clinic_1_data_rep* database, and
- FIRST_NAME, LAST_NAME, SEX, and DOB from the PATIENT table in the *Clinic_2_data_rep* database.

We label in bullets (f), (g), (h) and (i), overlapping database elements in the information type *patient details* across the *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep* databases.

(f) The data stored in TELEFONNE from the PATIENT table in the *GP_data_rep* database, models Mrs Flee's demographic data in her GP surgery. It is semantically related to the data stored in TELEPHONE from the patient table in the *Clinic_1_data_rep* database, that models the same thing, but in the clinic environment. The data stored in TELEPHONE and TELEFONNE are identical, because they store Mrs Flee's telephone number. They have the strongest similarity because they are identical (they resemble each other wholly). Data stored in both attributes belong to Semantic Equivalence (7) degree of similarity and may generate the Mispelt based naming conflict.

(g) The data stored in PATIENT_NO, NAME, SEX, DOB, H_MEDICAL_SUMMARY, MAJOR_ILLNESS and CHRONIC_DISEASE from the patient table in the *Hospital_data_rep* database, models Mrs Flee's demographic data over the last one year in Dr Smith's hospital. Their data is semantically related to the data stored in PATIENT_NO, FIRST_NAME, LAST_NAME, SEX, DOB, MEDICAL_SUMMARY, MAJOR_ILLNESS and CHRONIC_DISEASE of the PATIENT table in the in the *Clinic_2_data_rep* database, that also models Mrs Flee's demographic clinical data over the last one year in a healthcare center. The data in NAME, FIRST_NAME, and LAST_NAME attributes do resemble each other (they store Mrs Flee's name and surname) but their attributes have different structures. They belong to the Semantic Likeness (3) degree of similarity, and may generate the Aggregation based structural conflict.

(h) The data stored in the patient table in *Clinic_1_data_rep* database, models demographic and clinical details for Mrs Flee in a clinic environment. It is semantically related to the data stored in PATIENT table within the *GP_data_rep* database, which also models Mrs Flee's demographic and clinical details according to Mrs Flee's GP surgery. Data stored in the patient and PATIENT tables belong to the Semantic Equivalence (7) degree of similarity and may generate the Case-Sensitivebased naming conflict. (For the purpose of illustrating the case-sensitive based naming conflict we look at possible conflicts at the level of table names).

The illustration of the Homonym based naming conflict, through data stored in REPORT from the TREATMENT table in the *Hospital_data_rep* database, and data stored in REPORT from the LABTEST table in the *Clinic_1_data_rep* database is labelled in bullet (i) below. However, data stored in these two attributes might not appear in this particular creation of Mrs Flee's health summary by Dr Smith (i.e. they might not belong to the information types *medical summaries*, *treatment summaries* and *patient details)*.

(i) The data stored in REPORT from the TREATMENT table in the *Hospital_data_rep* database, models the combinations of prescriptions and treatments for Mrs Flee in Dr Smith's hospital. It may be semantically related to the data stored in REPORT from the LABTEST table in the *Clinic_1_data_rep* database. However, the *Clinic_1_data_rep* database decides to store in REPORT the number and information on their patients who have normal lab tests (and Mrs Flee's data might not be there!) Therefore, data stored in both 'REPORT' may initially resemble each other but there is **no** semantic similarity between them. They belong to the Semantic False Likeness (2) degree of similarity and may generate the Homonym based naming conflict.

5.2 Example of Preparing the Semantics for Core Ontological Layering

In the first five steps of our process of resolving semantic conflicts we have to prepare semantics, which will ensure a correct way of creating core ontological layers. The preparation for ontological layering is done by:

- translating relational database schemas and the content of their databases into local ontologies LO_j and ENV_ONT, and
- reasoning upon the content of user inputs stored in USER_INP_ONT, captured through application GUI, and interpreted by creating new concepts in the ADDED_VAL_ONT.

In these five steps we do not intend to resolve any semantic conflicts, because we need to identify first where semantically related data exist and which semantic conflicts they generate. However, by identifying semantically related data through this preparation for ontological layering, we resolve HOMONYMS because we are able to eliminate data or information, which is a consequence of False Semantic Likeness. To be more precise, in this preparation, HOMONYMS are eliminated because semantically related data are grouped together according to the exact meaning of user's input (i.e. choices of data repositories $\{Rep_i \mid i = 1, \ldots m\}$ or information types $\{InfType_d \mid t = 1, \ldots t\}$ captured through user's clicks).

Furthermore, at this stage, we are also able to identify if any of the semantically related data are equivalent to each other, and hence resolve Mispelt and Case-Sensitivesemantic conflicts. This may happen because of the existence of semantic equivalence and this identification is easy because (a) DataMaster (Nylus 2007) changes all characters into lower-cases, thus eliminates Case-Sensitivesemantic conflicts, and (b) we are able to inspect any possible Mispelt names of attributes and check the equivalence of their data values, before we declare them semantically equivalent. In the following seven subsections we describe each step in the preparation for ontological layering which ultimately shows the way we identify semantically related data. Step 1 of our process is described in section 5.2.1, step 2 is in section 5.2.2, steps 3 and 4 are in section 5.2.4 and step 5 is in section 5.2.5. .Section 5.2.3 deals with Mispelt and Case-Sensitiveconflicts and section 5.2.6 deals with Homonyms.

5.2.1 Step 1: Translating Relational Schemas into Local Ontologies

In Step 1, the content of *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep* databases are translated into local ontologies *LO_gp*, *LO_hospital*, *LO_clinic_1* and *LO_clinic_2*. The translation is performed through the Protégé 3.4 ontological editing toolkit environment (Knublauch et al. 2004) using the DataMaster plug-in that allows for the automatic translation of relational schema to OWL ontology, i.e. it allows the importation of relational schema and the content of its tables/attributes (data values) into Protégé via a Java Database/Open Database connector. After translating each database *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep* into local ontologies *LO_gp*, *LO_hospital*, *LO_clinic_1* and *LO_clinic_2* we have their ontological hierarchies, associated data type properties and their range values created by DataMaster.

It is important to note that we chose one of three methods of translations offered by DataMaster. Details of each method are available in [294 and 296]. The results of the translations are shown in Figures 5.5, 5.8, 5.9 and 5.10. In all four figures "db", "db1", "db2" and "db3" denotes the ontological version of the relational schemas for *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep* respectively. Furthermore, all these ontological hierarchies from Figures 5.5, 5.8, 5.9 and 5.10 have an identical base parent class `Local_ontological_layer` and its subclasses `LO_gp`, `LO_hospital`, `LO_clinic_1` and `LO_clinic_2`. However, each of these subclasses has their own hierarchies which are separately shown in each of these figures. Their names denote that each subclass and its hierarchies belong to the local ontological layer of our SA.

In this section we only give a detailed explanation of the way we created local ontology *LO_gp*. In order to avoid repetitions in explaining the creation of other three local ontologies *LO_hospital*, *LO_clinic_1* and *LO_clinic_2* we only give a short description of their hierarchies. The OWL source code for all local ontologies *LO_hospital*, *LO_clinic_1* and *LO_clinic_2* *LO_gp*, can be found in Appendix A.3 (Note: Appendix A.3 is stored on the CD-ROM due to its size - 153 pages long!)

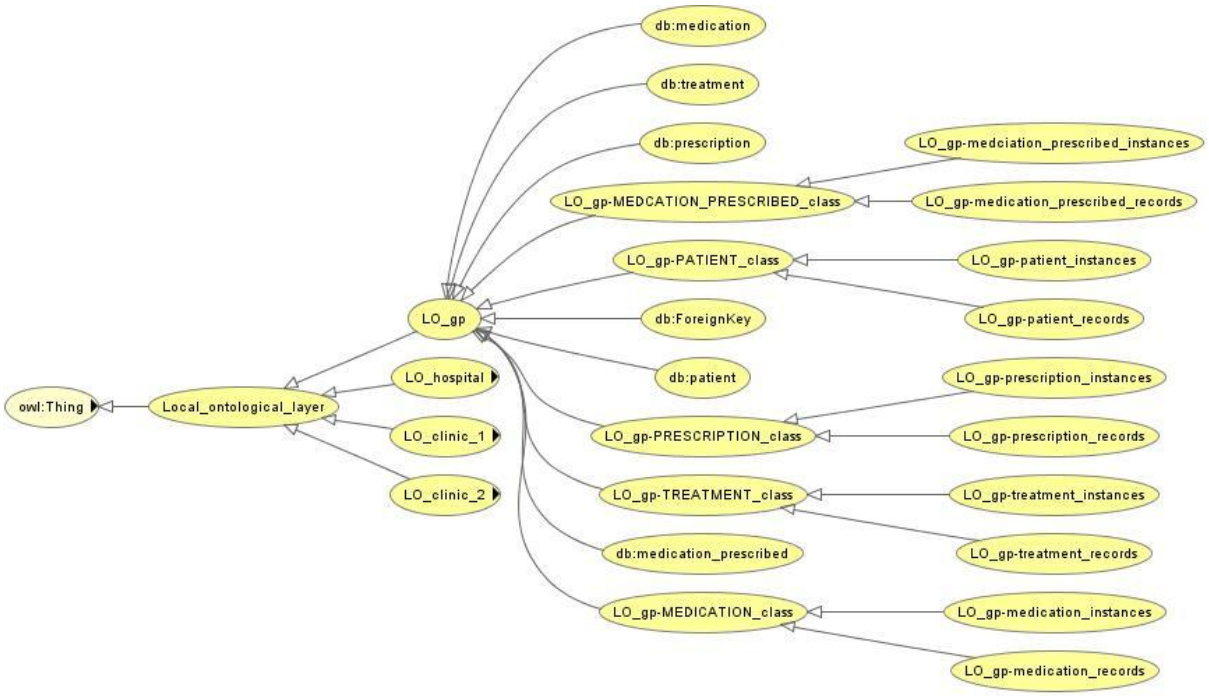

Figure 5.5 *Results from the translation of GP_data_rep database into the local ontology LO_gp*

In Figure 5.5 the five subclasses db:patient, db:prescription, db:medication_prescribed, db:medication and db:treatment of the LO_gp parent class are the direct output from the automatic translation of the PATIENT, PRESCRIPTION, MEDICATION_PRESCRIBED, MEDICATION and TREATMENT tables in the *GP_data_rep* database (see Figure 5.1) into the local ontology *LO_gp* through the DataMaster plug-in. The db:ForiegnKey subclass contains a list of all the foreign keys from the *GP_data_rep* database. The automatic translation of the PATIENT, PRESCRIPTION, MEDICATION_PRESCRIBED, MEDICATION and TREATMENT tables also generates datatype properties that mirror the concept of column names and data values. For example, if we have the PATIENT table in the *GP_data_rep* database, with the:

- column names (attributes): PATIENT_ID, FIRST_NAME, LAST_NAME, ADDRESS, REGION, TELEFFONE, NEXT_OF_KIN, EMERGENCY_CONTACT, NO_OF_CHILDREN, BMI and HEIGHT, and their
- data values relevant to Mrs Flee: P3344A, JANE, FLEE, 167_BOULEVARD_RD_W1W_5TU, LONDON, 02075698899, NEMANJA_FLEE, 07965896456, 0, NORMAL, and 5_feet_8_inches,

then the automatic translation of PATIENT table into *LO_gp* translates each row of the PATIENT table as an ontological individual db1:patient_Instance_1 belonging to the db:patient class.

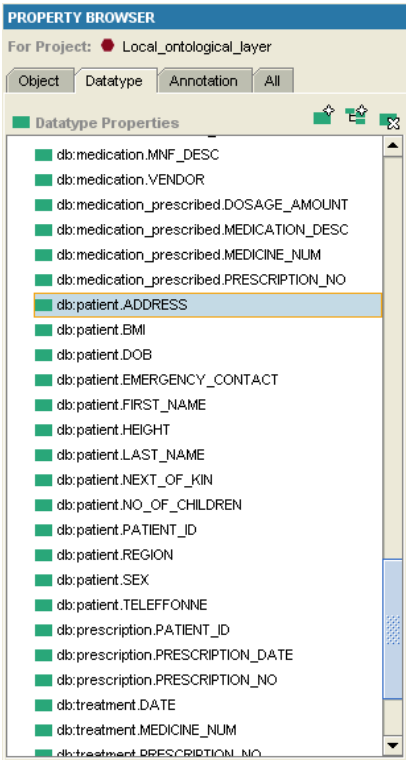

Figure 5.6 *Examples of datatype properties in the 'db:patient' class generated as a result of translating the GP_data_rep database into local ontology LO_gp*

Figure 5.6 shows that column names (attributes) are translated into datatype properties db1:patient.PATIENT_ID, db1:patient.FIRST_NAME, db1:patient.LAST_NAME db1:patient.ADDRESS, db1:patient.REGION, db1:patient.TELEFFONE,

db1:patient.NEXT_OF_KIN, db1:patient.EMERGENCY_CONTACT, db1:patient.NO_OF_CHILDREN, db1:patient.BMI and db1:patient.HEIGHT. The complete set of domain and range constraints for all the above datatype properties in the local ontology *LO_gp* can be found in Table 5.1, Appendix A.4.

Note: for all datatype properties listed in Table 5.1 from Appendix A.4, the range values are set as the "data types" of their corresponding column names (attributes) in the *GP_data_rep* database. In other words, the data type "varchar" for data values JANE, FLEE and 167_BOULEVARD_RD_W1W_5TU in the *GP_rep_data* database are translated as the ontological range values of type of "string literal" for a datatype properties db1:patient.FIRST_NAME, db1:patient.LAST_NAME and db1:patient.ADDRESS respectively.

Subsequently, the number of datatype properties generated as a consequence of translations, correspond directly to the number of column names (attributes) within each table of a particular relational schema. Thus, in our example scenario we gain a total of 81 datatype properties across the *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep* databases.

There are five subclasses in Figure 5.5 which are not a result of translations through DataMaster. They are named as LO_gp-PATIENT_class, LO_gp-TREATMENT_class, LO_gp-MEDICATION_class, LO_gp-PRESCRIPTION_class and LO_gp-MEDICATION_PRESCRIBED_class. We had to create them because of the following reason.

Our dependence on Protégé 3.4 and DataMaster during the translation, has dictated that all database elements are translated into ontological datatype properties, with their range values defined as string literals. However, in order to perform ontological mapping, which will secure the creation of all ontological layers, we need to manipulate **ontological individuals**, which are not available with datatype properties generated by DataMaster. What we need, in order to ultimately perform reasoning within our SA, are object properties with range values of ontological individuals. Consequently we have to convert all the datatype properties in the local ontologies *LO_gp*, *LO_hospital*, *LO_clinic_1*, and *LO_clinic_2* into object properties. Therefore each of these subclasses LO_gp-PATIENT_class, LO_gp-TREATMENT_class, LO_gp-MEDICATION_class, LO_gp-PRESCRIPTION_class and LO_gp-MEDICATION_PRESCRIBED_class (remember, we have 5 tables in the relational schema for *GP_ data_rep*, see Figure 5.1) contains further two subclasses:

- one subclass for storing all the instances (data values) of attributes within a particular table (e.g. the subclass named LO_gp-patient_instances), and
- one subclass for storing all the records, i.e. rows and columns (e.g. the subclass named LO_gp-patient_records) of a particular table.

All the stored instances within a particular table in a database are modelled as ontological individuals. For example, the data value 167_BOULEVARD_RD_W1W_5TU becomes the ontological individual named 167_BOULEVARD_RD_W1W_5TU stored in the subclass LO_gp-patient_instances of the LO_gp-PATIENT_class class. All the stored records (i.e. rows) within a particular table in a database are modelled as ontological individuals. For example, the ontological individual db1:patient_Instance_1 is modelled to represent a particular row stored in PATIENT table in the relational schema for the *GP_data_rep* and stored in the subclass LO_gp-patient_records of

the `LO_gp-PATIENT_class` class. Additionally, in order to define the relationships between each row of a table and its corresponding data values in the *GP_data_rep* database, object properties are modelled between the classes `LO_gp-patient_instances` and `LO_gp-patient_records` of local ontology *LO_gp*. For example, for `PATIENT` table in *GP_data_rep* database we had to create 11 object properties in local ontology *LO_gp*.

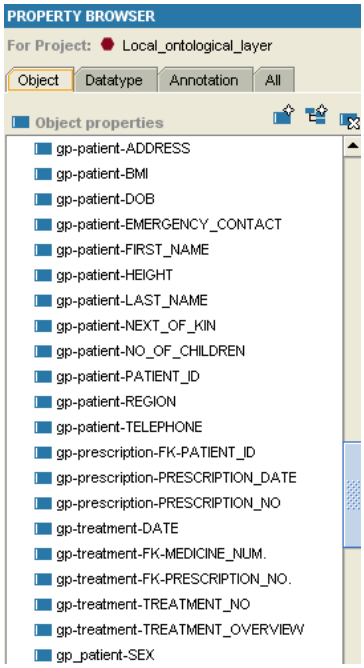

Figure 5.7 *Examples of object properties created as part of the relationships existing in between the classes 'LO_gp-patient_instances' and 'LO_gp-patient_records' in local ontology LO_gp*

Figure 5.7 shows a partial view of a few object properties which create the relationships between the classes `LO_gp-patient_instances` and `LO_gp-patient_records`. The complete set of domain and range constraints for the object properties in the local ontology *LO_gp* can be found in Table 5.2, Appendix A.5.

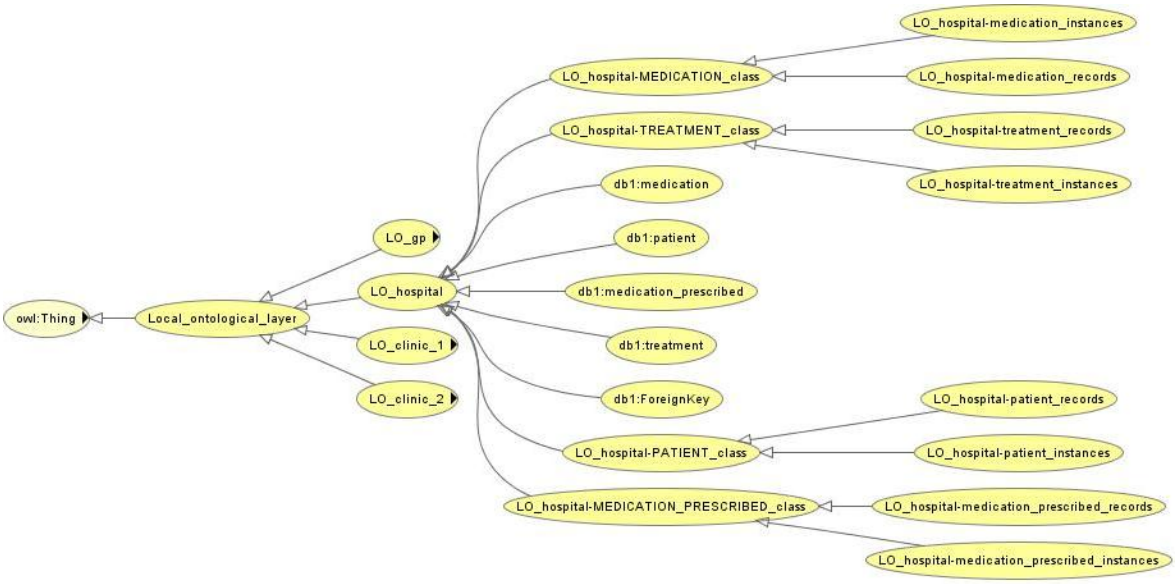

Figure 5.8 *Results from the translation of Hospital_data_rep database into local ontology LO_hospital*

In Figure 5.8 the four subclasses db1:patient, db1:prescription, db1:medication_prescribed, and db1:medication of the LO_hospital parent class are the direct output from the automatic translation of the PATIENT, PRESCRIPTION, MEDICATION, and MEDICATION_PRESCRIBED tables in the *Hospital_data_rep* database (see Figure 5.2) into the local ontology *LO_hospital* through the DataMaster plug-in. The four subclasses LO_hospital-PATIENT_class, LO_hospital-PRESCRIPTION_class, LO_gp-MEDICATION_PRESCRIBED_class and LO_hospital-MEDICATION_class are created to accommodate ontological individuals through object properties, as in the previous case (see page 89)

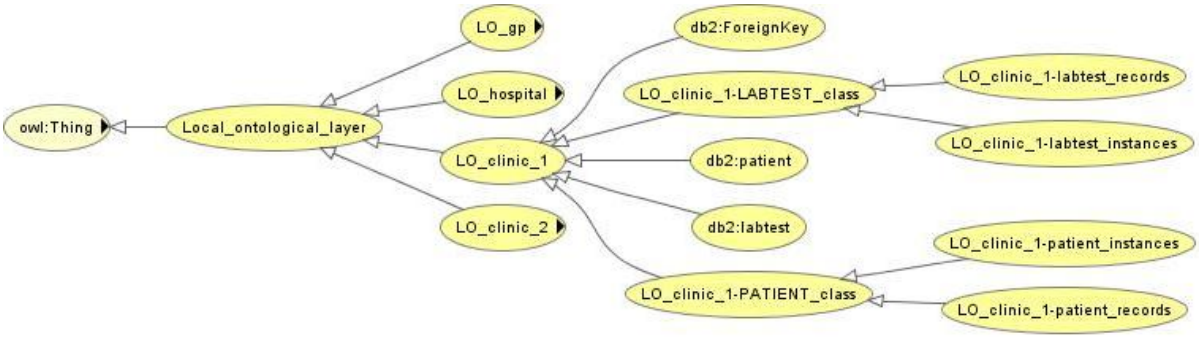

Figure 5.9 *Results from the translation of Clinic_1_data_rep database into local ontology LO_clinic_1*

In Figure 5.9 the two subclasses db2:patient and db2:labtest of the LO_clinic_1 parent class are the direct output from the automatic translation of the PATIENT and LABTEST tables in the *Clinic_1_data_rep* database (see Figure 5.3) into the local ontology *LO_clinic_1* through the DataMaster plug-in. The two subclasses LO_clinic_1-PATIENT_class and LO_clinic_1-LABTEST_class are the consequence of modelling additional semantics through human intervention.

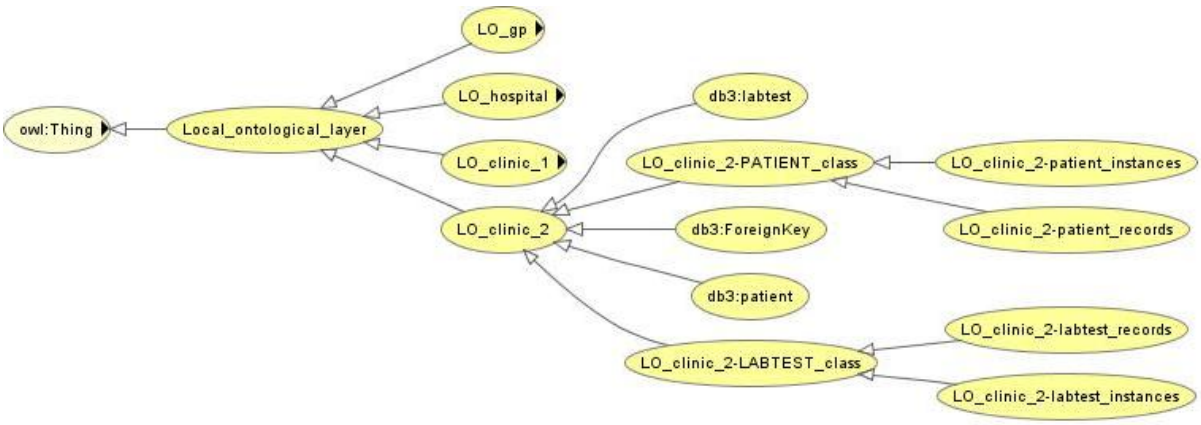

Figure 5.10 *Results from the translation of Clinic_2_data_rep database into local ontology LO_clinic_2*

In Figure 5.10 the two subclasses db3:patient and db3:labtest of the LO_clinic_2 parent class are the direct output from the automatic translation of the PATIENT and LABTEST tables in the *Clinic_2_data_rep* database (see Figure 5.4) into the local ontology *LO_clinic_2* through the DataMaster plug-in. The two subclasses LO_clinic_2-PATIENT_class and LO_clinic_2-LABTEST_class are the consequence of modelling additional semantics through human intervention.

After the translation of the *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep* databases into local ontologies, their semantic similarities (i.e. semantically related data) are naturally carried forward into the ontological concepts of *LO_gp*, *LO_hospital*, *LO_clinic_1*, and *LO_clinic_2*. Table 5.3 in Appendix A.7 lists the semantic similarities between local ontologies, and where semantic conflicts have been carried forward into ontological concepts. The number of similarities between semantically related concepts in local ontologies remains the same as in their underlying databases; hence, they keep the same classification of the degree of similarities as defined in section 5.1.2.

5.2.2 Step 2: Mirroring of Relational Schemas into ENV_ONT

In Step 2, the metadata from the *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep* databases are mirrored within the predefined parent class `TECHNOLOGICAL_SPECIFICATION` in the ENV_ONT, as explained in chapter 4, section 4.1

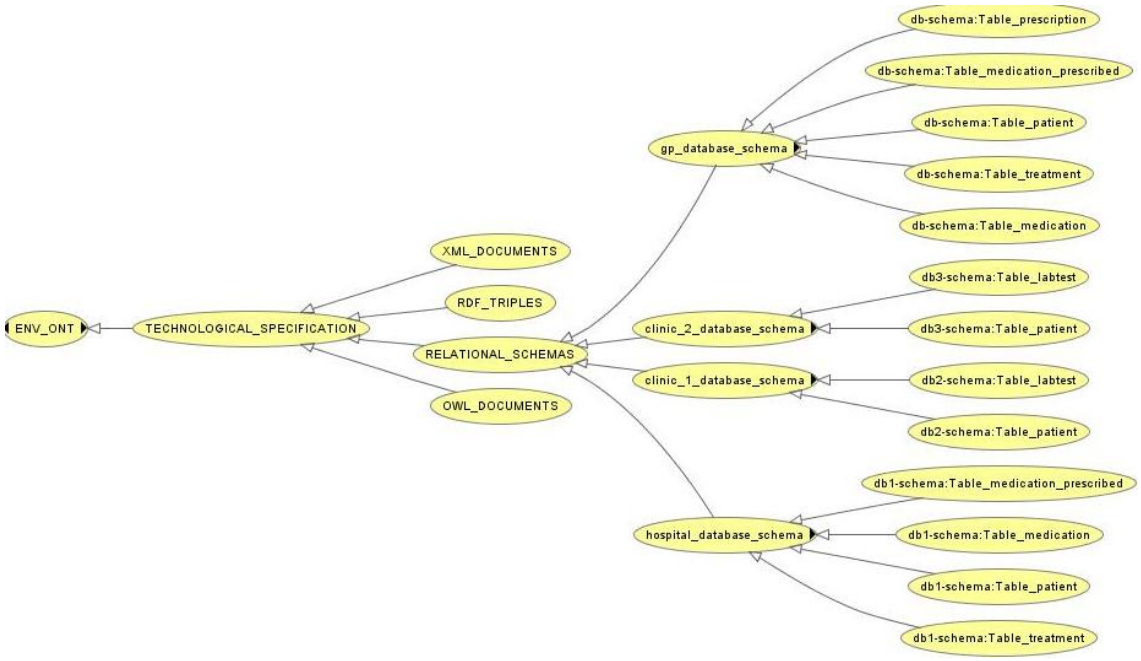

Figure 5.11 *Results from the modelling of metadata in GP_data_rep, Hospital_data_rep, Clinic_1_data_rep and Clinic_2_data_rep databases into the ENV_ONT*

The metadata from the *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep* databases are shown in Figure 5.11 as subclasses of `RELTIONAL_SCHEMAS`. The OWL source code for the ENV_ONT can be found in Appendix A.6. Note: Appendix A.6 is stored on the CD-ROM due to its size. The child-classes in ontological hierarchies from Figure 5.11 show that:

- `db:patient`, `db:prescription`, `db:medication_prescribed`, `db:medication` and `db:treatment` of the `RELATIONAL_SCHEMA` subclass, are the consequence of having five tables in the *GP_data_rep* database,
- `db1:patient`, `db1:prescription`, `db1:medication_prescribed` and `db1:medication` of the `RELATIONAL_SCHEMA` subclass, are the consequence of having four tables in the *Hospital_data_rep* database,

- `db2:patient` and `db2:labtest` of the `RELATIONAL_SCHEMA` subclass, are the consequence of having two tables in the *Clinic_1_data_rep* database, and
- `db3:patient` and `db3:labtest` of the `RELATIONAL_SCHEMA` subclass, are the consequence of having two tables in the *Clinic_2_data_rep*.

5.2.3 Resolving Mispelt and Case-Sensitive Semantic Conflicts

Having identified, while looking at the possible types of semantic conflicts in relational schemas in section 5.1.2, that there are Mispelt and Case-Sensitive attribute names, we discovered that:

- data stored in '`TELEFONNE`' from the `PATIENT` table in the *GP_data_rep* database, and
- data stored in '`TELEPHONE`' from the `patient` table in the *Clinic_1_data_rep* database,

exhibit Semantic Equivalence (7) as their degree of similarity. This is an example of Mispelt attribute name, which is resolved during the translation. However, we are again dependent on DataMaster and its mechanism of translating database elements into ontological concepts. As we mentioned earlier, in section 5.2.2, we need object properties for our ontological mapping, therefore Mispelt conflict is resolved by changing the name of the attribute `TELEFONNE` into the object property `gp-patient.TELEPHONE` of the `LO_gp-patient_class` parent class

Similarly, having identified Case-Sensitive naming of database tables we have discovered that:

- the `patient` table in the *Clinic_1_data_rep* database, and
- the `PATIENT` table in the *GP_data_rep* database,

exhibit Semantic Equivalence (7) as their degree of similarity. We resolve the conflict by choosing lower case characters to define the name of the `patient` table after its translation into `LO_clinic_1-patient_class` ontological class.

5.2.4 Steps 3 and 4: Preparing Lists of Data Repositories and Capturing Dr Smith's Involvements

In step 3, we prepare a list of all four databases: *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep*, and information types available within them: *medical summaries*, *treatment summaries* and *patient details*, as choices for Dr Smith's retrievals through the application's GUI.

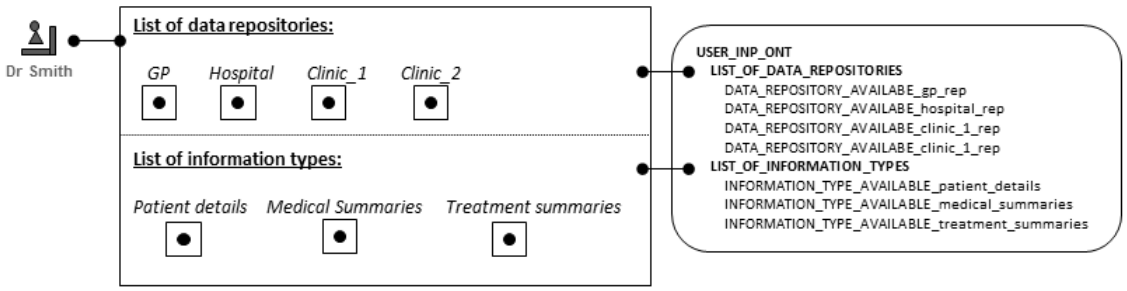

Figure 5.12 *Example of performing "clicks" on radio buttons offering data repositories Rep_i and information types $InfType_d$*

We assume that Dr Smith's involvements in step 3 are in the form of performing "clicks" on radio buttons shown in the GUI from Figure 5.12. The scenario from section 5.1 indicates that Dr Smith is

interested in retrieving data/information from all four databases and therefore he will have an opportunity to click on radio buttons which determine the selection of these databases. However, more importantly Dr Smith knows that he would like to create a heath summary for Mrs Flee, which can be only generated from various information types available in these databases. Therefore, the GUI above also lists three information types available for clicking: *medical summaries*, *treatment summaries* and *patient details*.

As soon as Dr Smith clicks on radio buttons, we have to capture his clicks by populating ontological individuals into the subclasses `LIST_OF_DATA_REPOSITORIES` and `LIST_OF_INFORMATION_TYPES` of the USER_INP_ONT as explained in chapter 4, section 4.1. The OWL source code for the USER_INP_ONT can be found in Appendix A.8. Note: Appendix A.8 is stored on the CD-ROM.

Hence, let us assume that Dr Smith has selected all seven radio buttons and has entered the patient name Jane Flee, in the text box of the application GUI. Therefore, in step 4, we populate subclasses of the USER_INP_ONT (they are illustrated in Figure 5.13):

- `DATA_REPOSITORY_AVAILABLE_gp_rep`,
- `DATA_REPOSITORY_AVAILABLE_hospital_rep`,
- `DATA_REPOSITORY_AVAILABLE_clinic_1_rep`,
- `INFORMATION_TYPE_AVAILABLE_medical_summaries`,
- `INFORMATION_TYPE_AVAILABLE_treatment_summaries`,
- `INFORMATION_TYPE_AVAILABLE_patient_details` and
- `PATIENT_AVAILABLE_jane_flee`,

with ontological individuals, therefore:

- `USER_CLICK_gp_rep` class is populated with ontological individual named "USER_CLICK_gp", thus implying that Dr Smith has clicked on radio button placed next to the *GP_data_rep* database;
- `USER_CLICK_hospital_rep` is populated with ontological individual named "USER_CLICK_hospital", thus implying that Dr Smith has clicked on radio button placed next to the *Hospital_data_rep* database;
- `USER_CLICK_clinic_1_rep` is populated with ontological individual named "USER_CLICK_clinic_1", thus implying that Dr Smith has clicked on radio button placed next to the *Clinic_1_data_rep* database;
- `USER_CLICK_clinic_2_rep` is populated with ontological individual named "USER_CLICK_clinic_2", thus implying that Dr Smith has clicked on radio button placed next to the *Clinic_2_rep* database;
- `USER_CLICK_medical_summaries` is populated with ontological individual named "USER_CLICK_medical_summaries", thus implying that Dr Smith has clicked on radio button placed next to the information type *medical summaries*;
- `USER_CLICK_treatment_summaries` is populated with ontological individual named "USER_CLICK_treatment_summaries", thus implying that Dr. Smith has clicked on radio button placed next to the information type *treatment summaries*;

- `USER_CLICK_patient_details` is populated with ontological individual named "USER_CLICK_patient_details", thus implying that Dr Smith has clicked on radio button placed next to the information type *patient details*;
- '`TEXT_ENTERED_jane_flee`' is populated with ontological individual named '`TEXT_ENTERED_jane_flee`', thus implying that Dr Smith has entered the patient name Jane Flee.

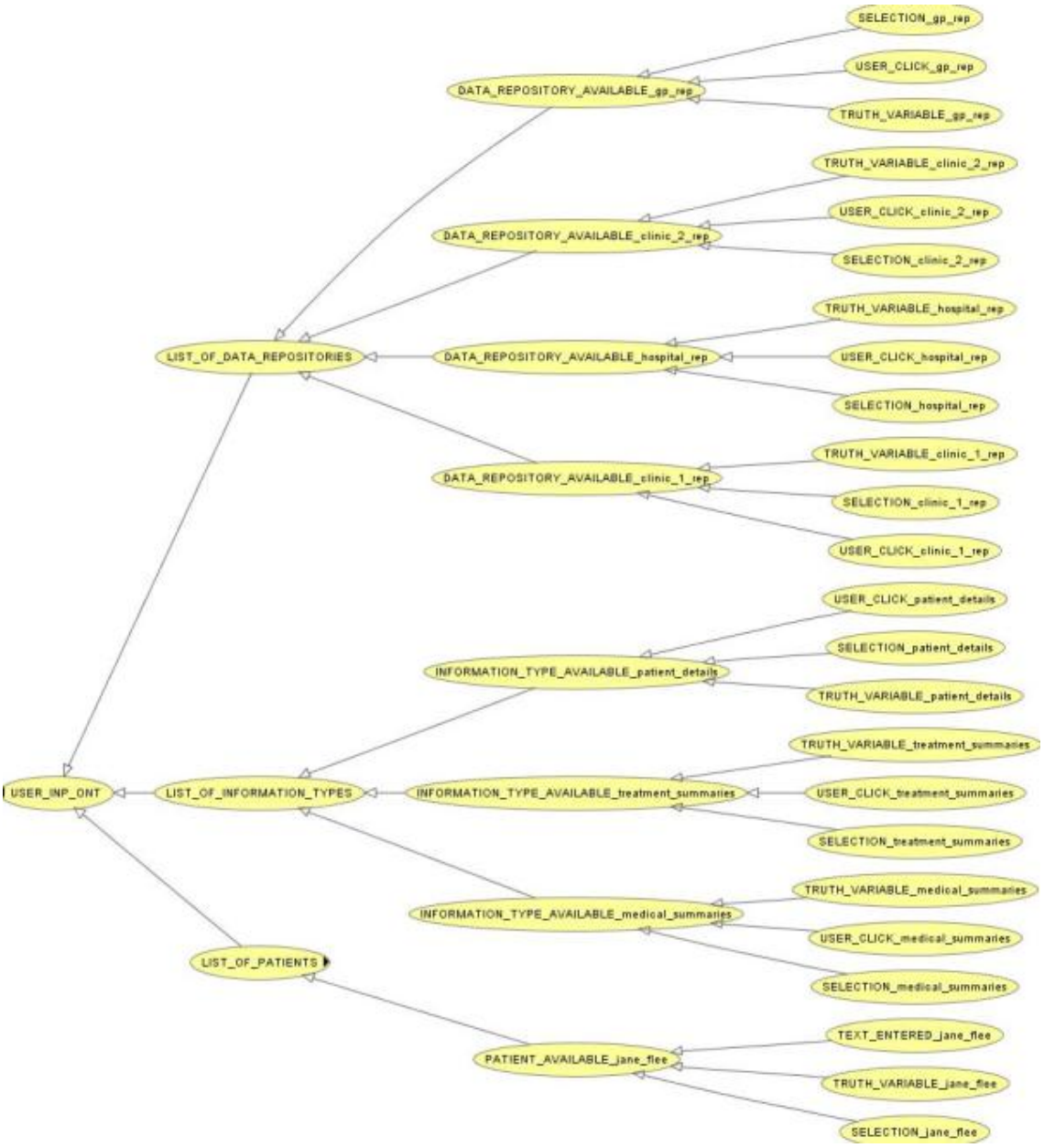

Figure 5.13 *Example of the classes we populate in the USER_INP_ONT*

Populating ontological individuals into the subclasses of the classes `LIST_OF_DATA_REPOSITORIES` and `LIST_OF_INFORMATION_TYPES` in the USER_INP_ONT (in Figure 5.13) is performed through the Protégé OWL API library [80] to provide a Java API for populating OWL ontologies.

[80] http://protege.stanford.edu/plugins/owl/api/

5.2.5 Step 5: Storing and Interpreting Dr. Smith's Involvements – Running Selection and Grouping SWRL rules

In step 5, we use the content of the USER_INP_ONT, which stores Dr Smith's inputs (i.e. captured "clicks") from the previous step and interpret them by creating concepts in the ADDED_VAL_ONT as explained in chapter 4, section 4.1. In other words, Dr. Smith's captured clicks based on his choice of data repositories and information types are:

5c) *stored* in the 'SELECTION_xxx/yyy/zzz' subclass of 'LIST_OF_DATA_REPOSITORIES' and 'LIST_OF_INFORMATION_TYPES' in the USER_INP_ONT by running *Selection* rules (where 'xxx' denotes Dr Smith's choice of databases *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep*, 'yyy' denotes information types *medical summaries*, *treatment summaries*, and *patient_details*, and 'zzz' denotes patient name Jane Flee), and

5d) *interpreted* through reasoning upon ontological concepts in the USER_INP_ONT and the ENV_ONT as explained in chapter 4, section 4.1 at the same time, in order to group semantically related ontological concepts from local ontologies *LO_gp*, *LO_hospital*, *LO_clinic_1* and *LO_clinic_2* into the ADDED_VAL_ONT by running *Grouping* rules .

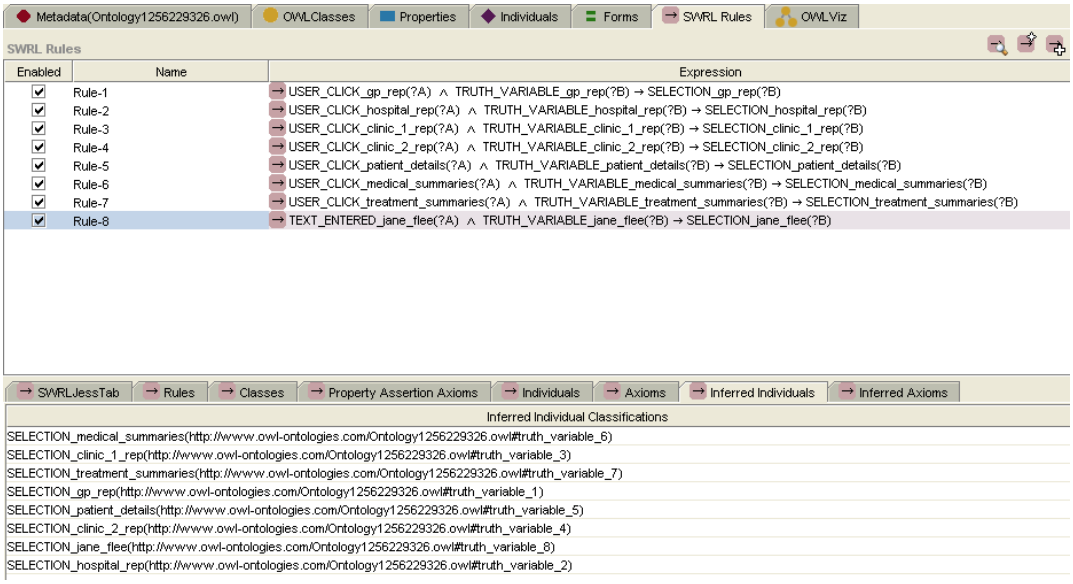

Figure 5.14 *Results of running Selection rules 1 – 8 against the Jess engine in the Protégé 3.4 ontological editing toolkit environment*

The *Selection* rules are created in the SWRL, and run through the SWRL-plug-in[81] in Protégé 3.4, using the Java Expert System Shell (Jess) reasoning engine[82], which performs the:

(i) conversion of SWRL rule to Jess rules,

(ii) running of Jess rules against the Jess engine, and

(iii) inference of ontological individuals into specified ontological classes implied in the SWRL rules.

[81] protege.cim3.net/cgi-bin/wiki.pl?SWRLTab

[82] http://www.jessrules.com/

Running Selection Rules (8 rules)

The results of running the *Selection* rules are shown in Figure 5.14, and are based on the assumption that:

- Dr Smith has made a choice of selecting all 7 radio buttons and entering the patient name Jane Flee through the GUI in Figure 5.12, and
- relevant subclasses of the pre-defined parent ontological classes named `LIST_OF_DATA_REPOSITORIES` and `LIST_OF_INFORMATION_TYPES` in the USER_INP_ONT have been populated with ontological individuals (see section 5.2.4).

The SWRL source code for the *Selection* rules 1 – 8 in Figure 5.14 can be found in Appendix A.9. Screen shots of the inference as a result of running Selection rules 1 – 8 in Appendix A.9 can be found Appendix A.10. (Note Appendix A.10 is stored on the CD-ROM due to its size).

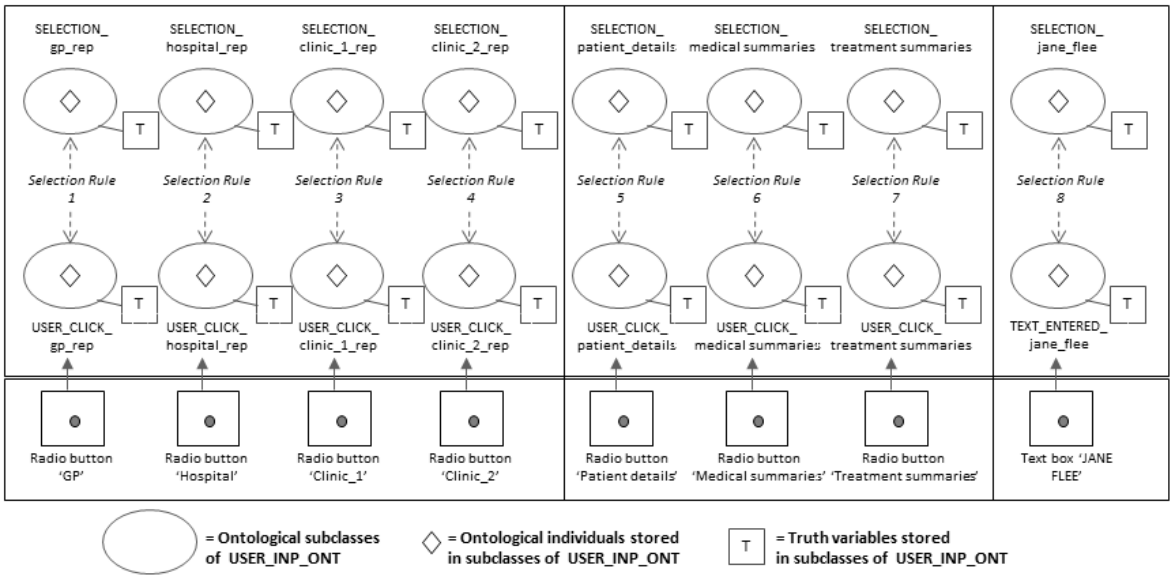

Figure 5.15 *Example of inferring ontological individuals as a consequence of running SWRL Selection rules*

Figure 5.15 shows the way of inferring individuals if Dr Smith has selected a certain set of radio buttons on the application GUI. We give one example. The `SELECTION_gp_rep` subclass in the USER_INP_ONT stores the result sets of running *Selection* rule 1, given in Appendix A.9, i.e. it stores the ontological individual "truth_variable_1" (T in the Figure 5.15) that has been transferred into `SELECTION_gp_rep` subclass, as a consequence of checking if both bullets below are correct:

- ontological individual "USER_CLICK_gp" exists within the `USER_CLICKS_gp_rep` subclass;
- ontological individual "truth_variable_1" in the `TRUTH_VARIABLE_gp_rep` subclass is set to 'true'.

The same applies to any other subclass of the USER_INP_ONT as illustrated in Figure 5.14.

Running Grouping Rules (14, 15, 16, 17, 22, 23, 24, 29 and 30)

The *Grouping* rules are created in the SWRL and run through the SWRL-plug-in in Protégé 3.4, using the Jess engine. The results of running the *Grouping* rules are stored in the subclasses of the:

- `MEDICAL_SUMMARIES_information_retrievals`,
- `TREATMENT_SUMMARIES_information_retrievals` and

- `PATIENT_DETAILS_information_retrievals` classes

in the ADDED_VAL_ONT shown in Figures 5.16, 5.19, and 5.22. The ontological hierarchy of the ADDED_VAL_ONT and its subclasses are created through Protégé 3.4, and are specific to the combination of data repositories *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep* and information types *medical summaries*, *treatment summaries* and *patient details*.

However, we wish to remind the reader that when we run *Grouping* rules we use metadata from ENV_ONT and actual data from local ontologies *LO_gp*, *LO_hospital*, *LO_clinic_1* and *LO_clinic_2*, i.e. ontological individuals stored in the classes `LO_gp`, `LO_hospital`, `LO_clinic_1` and `LO_clinic_2`. The OWL source code for the ADDED_VAL_ONT can be found in Appendix A.11. The SWRL source code for the *Grouping* rules can be found in Appendix A.12. Screen shots of the inference as a result of running *Grouping* rules Appendix A.12 can be found Appendix A.13. (Note Appendix A.11 and A.13 is stored on the CD-ROM).

5.2.5.1 Grouping Semantically Related Data in Patient Details

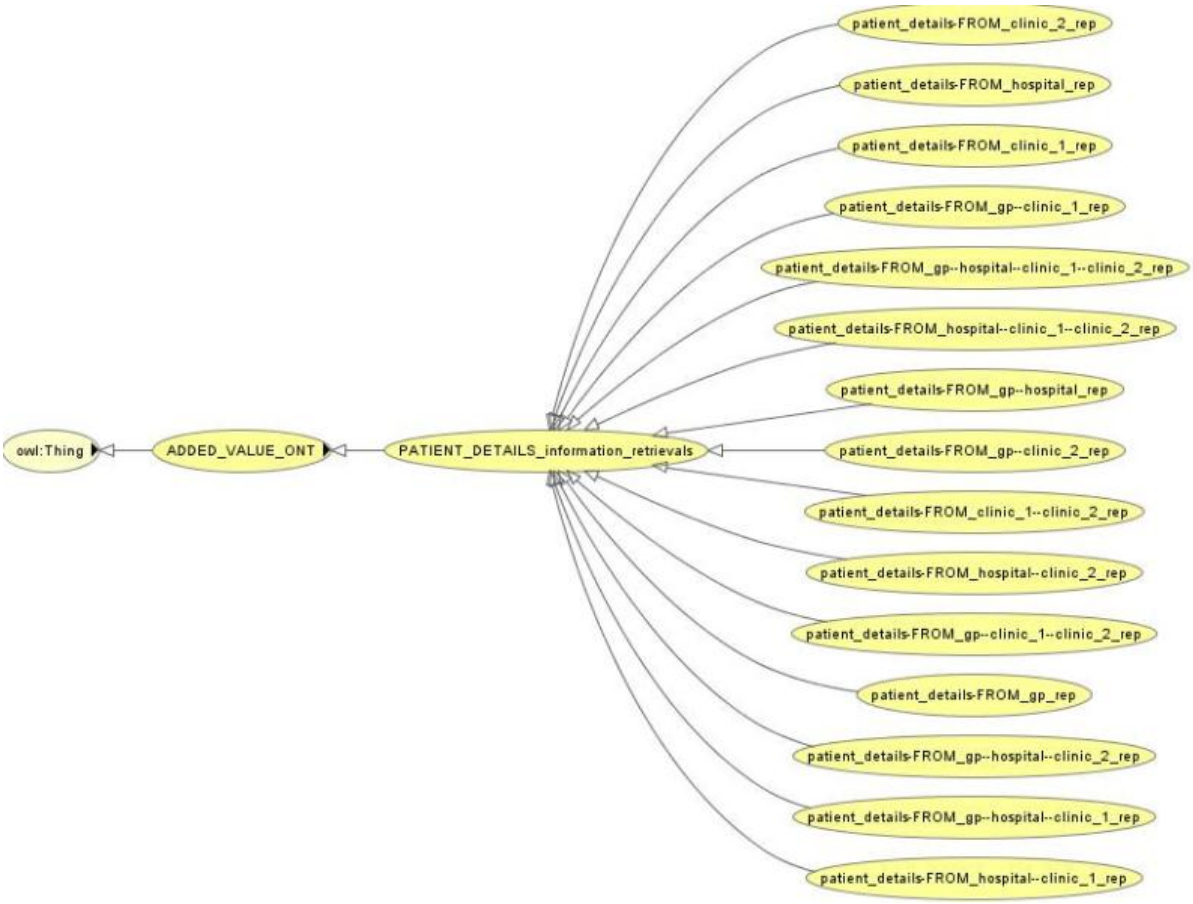

Figure 5.16 *Example of the 'PATIENT_DETAILS_information_retreivals' class in the ADDED_VAL_ONT*

In Figure 5.16 fifteen subclasses of the `PATIENT_DETAILS_information_retrievals` parent class are created to store the results of running *Grouping* rules 14, 15, 16 and 17, specific to the combination of *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep* database, and the information type *patient details*. We only give a detailed explanation for the subclass

patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep in order to avoid repetition and to illustrate semantically related data across all four databases.

Grouping rules 14, 15, 16 and 17 in Appendix A.12 are run against the Jess engine. They use the following set of object properties to group and move ontological individuals from classes in the local ontologies *LO_gp*, *LO_hospital*, *LO_clinic_1* and *LO_clinic_2*, defined in their range values:

- object properties gp-patient-FIRST_NAME, gp-patient-LAST_NAME, gp-patient-SEX, gb-patient-DOB, gp-patient-ADDRESS, gp-patient-REGION, gp-patient-TELEFFONE, gp-patient-NEXT_OF_KIN, gp-patient-EMERGENCY_CONTACT, gp-patient-NO_OF_CHILDREN, gp-patient-BMI and gp-patient-HEIGHT, which are defined upon the LO_gp-patient_instances class in local ontology *LO_gp*;
- object properties hospital-patient-NAME, hospital-patient-SEX, and hospital-patient-DOB, which are defined upon the LO_hospital-patient_instances class in local ontology *LO_hospital*;
- object properties clinic_1-patient-NAME, clinic_1-patient-SEX, clinci_1-patient-TELEPHONE and clinic_1-patient-DOB, which are defined upon the LO_clinic_1-patient_instances class in local ontology *LO_clinic_1*;
- object properties clinic_2-patient-FIRST_NAME, clinic_2-patient-LAST_NAME, clinic_2-patient-SEX, and clinic_2-patient-DOB, which are defined upon the LO_clinic_2-patient_instances class in local ontology *LO_clinic_2*.

Table 5.4 *The results of running the Grouping rules 14, 15, 16 and 17*

1. JANE	24. H_5_feet_4_inchesJANE FLEE
2. FLEE	25. F
3. F	26. JULY_04_1970
4. JULY_04_1970	27. JULIA FOX
5. 167_BOULEVARD_RD_W1W_5TU	28. F
6. LONDON	29. JUNE_17_1971
7. TEL_02075698899	30. JANE FLEE
8. NEMANJA_FLEE	31. F
9. TEL_07965896456	32. JULY_04_1970
10. CHILDREN_0	33. JULIA FOX
11. NORMAL	34. F
12. H_5_feet_8_inches	35. JUNE_17_1971
13. JULIA	36. TEL_02075698899
14. FOX	37. TEL_02078691369
15. F	38. JANE
16. JUNE_17_1971	39. FLEE
17. 27_HANGER_LANE_RD_N1E_6SU	40. F
18. LONDON	41. JULY_04_1970
19. TEL_02078691369	42. JULIA
20. PETER_FOX	43. FOX
21. TEL_07949538498	44. F
22. CHILDREN_1	45. JUNE_17_1971
23. NORMAL_PER_BMI	

After running *Grouping* rules 14, 15, 16 and 17, we move 45 ontological individuals, listed in Table 5.4, into patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep class. The inference, as a result of running *Grouping* rules 14, 15, 16 and 17 is graphically shown in Figure 5.17.

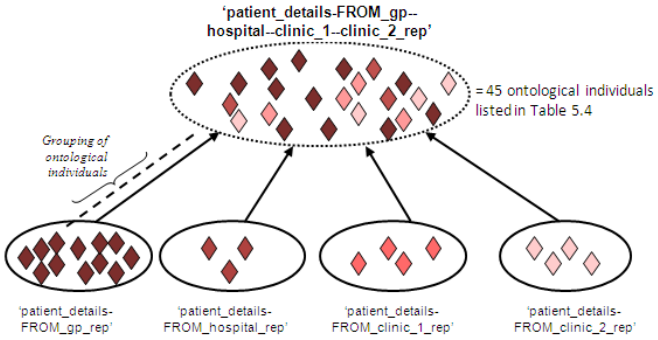

Figure 5.17 *Grouping ontological individuals from local ontologies LO_gp, LO_hospital, LO_clinic_1 and LO_clinic_2 into the concepts of ADDED_VAL_ONT that make up information type Patient details*

We model the object property `patient_details-treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep` in order to create a relationship between the `treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep` class and the classes of local ontologies *LO_gp*, *LO_Hospital*, *LO_clinic_1* and *LO_clinic_2*. The domain for the object property is set to the `patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep` ontological class, in order to specify where to move ontological individuals into. The range for the object property is set to the names of the ontological classes in local ontologies *LO_gp*, *LO_Hospital*, *LO_clinic_1* and *LO_clinic_2*, in order to specify where to move ontological individuals from (i.e. semantically related data that make up information type *patient details*).

Figure 5.18 shows a set of OWL restrictions applied to the object property `criteria_for-patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep` that determine the set criteria for `patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep` class membership.

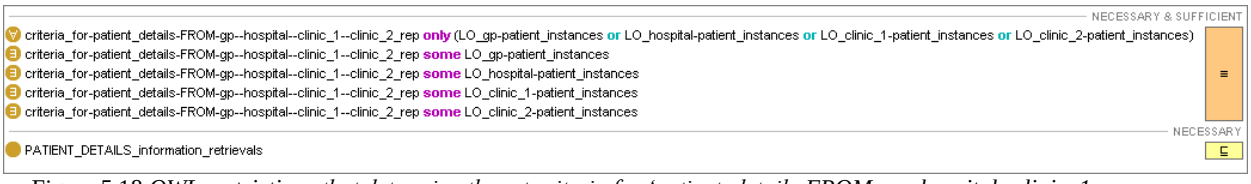

Figure 5.18 *OWL restrictions that determine the set criteria for 'patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep' class membership in the ADDED_VAL_ONT*

In Figure 5.18 the existential restriction (∃) is used to describe that the `patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep` ontological class has **some** ontological individuals from local ontologies *LO_gp*, *LO_Hospital*, *LO_clinic_1* and *LO_clinic_2*: `LO_gp-patient_instances`, `LO_gp-hospital_instances`, `LO_gp-clinic_1_instances`, and `LO_gp-clinic_2_instances`.

The universal restriction (∀) is used to describe that the `patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep` ontological class has **only** ontological individuals from local ontologies *LO_gp*, *LO_Hospital*, *LO_clinic_1* and *LO_clinic_2*: `LO_gp-patient_instances`, `LO_gp-hospital_instances`, `LO_gp-clinic_1_instances`, and `LO_gp-clinic_2_instances`.

Both restrictions are made 'necessary and sufficient' conditions to imply the concreteness of the `patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep` class.

5.2.5.2 Grouping Semantically Related Data in Medical Summaries

In Figure 5.19 fifteen subclasses of the `MEDICAL_SUMMARIES_information_retrievals` parent class are created to store the results of running the *Grouping* rules specific to the combination of *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep* databases, and the information type *medical summaries*. We only give a detailed explanation for the subclass `medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep` in order to avoid repetition and to illustrate semantically related data across all four databases.

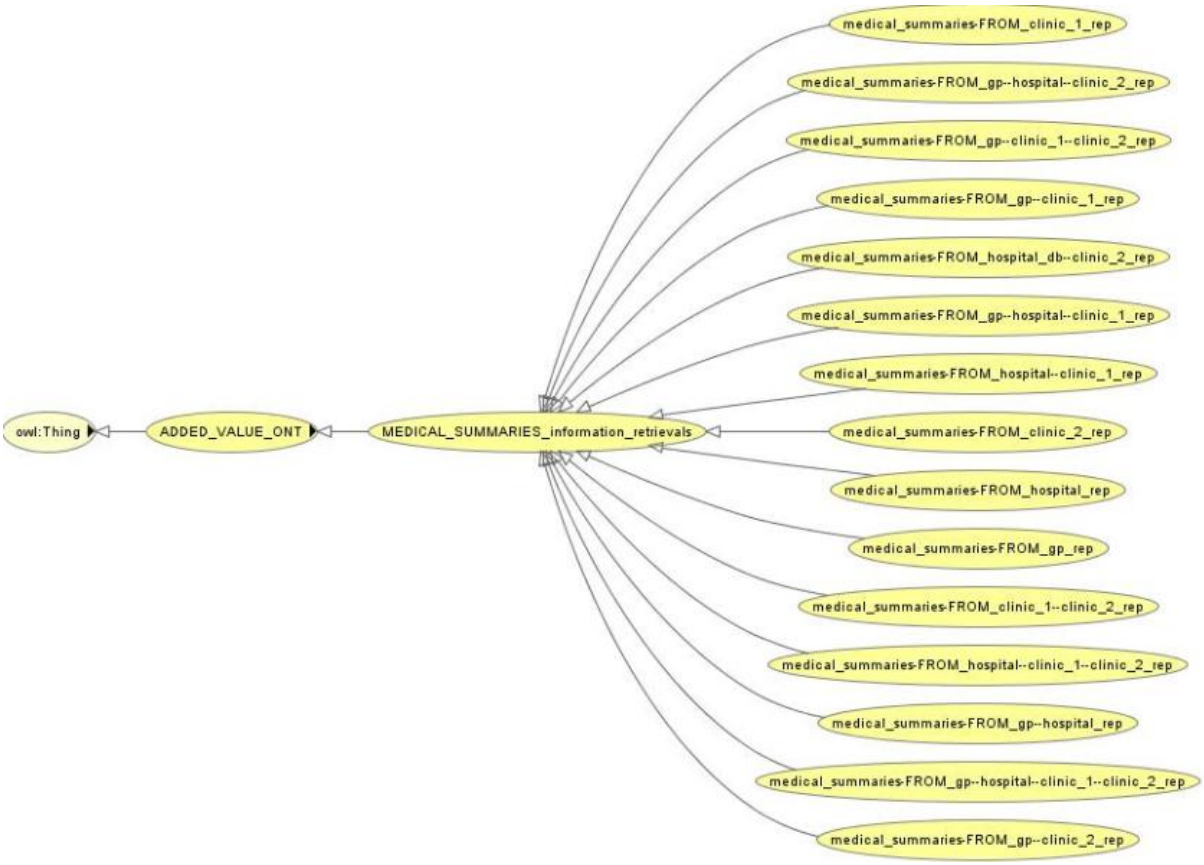

Figure 5.19 *Example of the 'MEDICAL_SUMMARIES_information_retreivals' class in the ADDED_VAL_ONT*

Grouping rules 22, 23 and 24 in Appendix A.12 are run against the Jess engine. They use the following set of object properties to group and move ontological individuals defined in their range values from the classes in local ontologies *LO_hospital*, *LO_clinic_1* and *LO_clinic_2*:

- object properties `hospital-patient-MEDICAL_SUMMARY`, `hospital-patient-MAJOR_ILLNESS` and `hospital-patient-CHRONIC_ILLNESS`, which are defined upon the `LO_hospital-patient_instances` class in local ontology *LO_hospital*;
- object properties `clinic_1-patient-PREVIOUS_MEDICAL_SUMMARY`, `clinic_1-patient-CURRENT_MEDICAL_SUMMARY`, `clinic_1-patient-MAJOR_ILLNESS` and

clinic_1-patient-CHRONIC_ILLNESS, which are defined upon the LO_clinic_1-patient_instances class in local ontology *LO_clinic_1*;

- object properties clinic_1-labtest-LABTEST_TYPE, clinic_1-labtest-LABTEST_NAME, clinic_1-labtest-LABTEST_RESULTS, and clinic_1-labtest-DATE, which are defined upon the LO_clinic_1-labtest_instances class in local ontology *LO_clinic_1*;
- object properties clinic_2-patient-MEDICAL_SUMMARY, clinic_2-patient-MAJOR_ILLNESS and clinic_2-CHRONIC_ILLNESS, which are defined upon the LO_clinic_2-patient_instances class in local ontology *LO_clinic_2* ;
- object properties clinic_2-labtest-LABTEST_OVERVIEW, clinic_2-labtest-LABTEST_DATA, clinic_2-labtest-LABTEST_TYPE, clinic_2-labtest-LABTEST_NAME, clinic_2-labtest-LABTEST_RESULTS and clinic_2-labtest-DATE, which are defined upon the LO_clinic_2-labtest_instances class in local ontology *LO_clinic_2*.

Table 5.5 *The results of running the Grouping rules 22, 23 and 24*

1. Mrs_Flee_complains_of_severe_pain_in_left_ankle_Minor_swelling_evident_and_xrays_taken_admitted_as_overnight_stay_and_found_to_have_acute_COPD_exacerbation	20. LT123_Blood_test_Type_4123
2. no_major_illness_evident	21. LT123_anaemia_level_46
3. no_chronic_disease_evident	22. LT123_16-02-08
4. Mrs_Fox_complains_of_severe_pain_in_left_ankle_and_xrays_taken_admitted_as_overnight_stay_and_found_to_have_acute_COPD_exacerbation	23. LT659_Normal_ABt
5. no_major_illness	24. LT659_16-01-08
6. no_chronic_disease	25. LT659_Smear_test_1
7. Mrs_Flee_has_a_regular_cervical_smear_test_results_appear_normal	26. LT659_Cervical_T2
8. Mrs_Flee_has_come_into_the_clinic_for_a_blood_test_complains_of_fatigue	27. Mrs_Flee_complains_of_shortness_of_breath_and_feels_intervals_of_pain_in_chest_area
9. none_found	28. No_MJ.
10. none	29. no_cd_found
11. Mrs_Fox_has_a_regular_cervical_smear_test_results_appear_normal	30. Mrs_Fox_complains_of_shortness_of_breath_and_feels_intervals_of_pain_in_chest_area
12. Mrs_Fox_has_come_into_the_clinic_for_a_blood_test_complains_of_fatigue	31. MJ_not_found.
13. no_trace	32. CD_not_found
14. no	33. LL456_Used_to_identify_lung_diseases
15. LT256_Smear_test	34. LL456_Radiation
16. LT256_Cervical_Type_3	35. LL456_Xray
17. LT256_Normal	36. LL456_fileID_wavelength908
18. LT256_16-01-08	37. LL456_28-04-09
19. LT123_Pathology	38. LL456_data_aa2
	39. Ll14569_Used_to_identify_Pneumonia_lung_cancer_fluid_collection_the_lungs
	40. LLl4569_Radiation_143
	41. LLl4569_X_ray
	42. LLl4569_fileID_wavelength203
	43. LLl4569_29-03-09
	44. LL14569_data_aa1

The results of running *Grouping* rules 22, 23 and 24 move the 44 ontological individuals listed in Table 5.5, into medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep class. The inference created as a result of running *Grouping* rules 22, 23 and 24 is graphically shown in Figure 5.20.

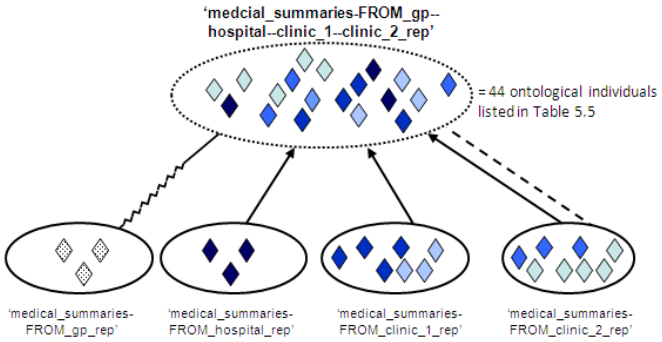

Figure 5.20 *Grouping ontological individuals from local ontologies LO_gp, LO_hospital, LO_clinic_1 and LO_clinic_2 into the concepts of ADDED_VAL_ONT that make up information type Medical summaries*

We model the object property `criteria_for-medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep` in order to create a relationship between the `medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep` class and the classes of local ontologies *LO_Hospital*, *LO_clinic_1* and *LO_clinic_2*. The domain for the object property is set to the `medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep` ontological class, in order to specify where to move ontological individuals into. The range for the object property is set to the names of the ontological classes in local ontologies *LO_hospital*, *LO_clinic_1* and *LO_clinic_2*, in order to specify where to move ontological individuals from (i.e. semantically related data that make up information type *medical summaries*).

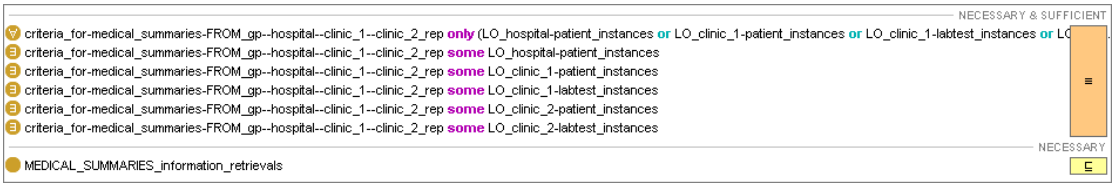

Figure 5.21 *OWL restrictions that determine the set criteria for 'medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep' class membership in the ADDED_VAL_ONT*

Figure 5.21 shows a set of OWL restrictions applied to the object property '`criteria_for-mecical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep`' that determine the set criteria for `medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep` class membership. The existential restriction (∃) is used to describe that the `medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep` ontological class has **some** ontological individuals from local ontologies *LO_hospital*, *LO_clinic_1* and *LO_clinic_2*: `LO_hospital-patient_instances`, `LO_clinic_1-patient_instances`, `LO_clinic_1-labtest_instances`, `LO_clinic_1-patient_instances`, and `LO_clinic_1-labtest_instances`.

The universal restriction (∀) is used to describe that the `medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep` ontological class has **only** ontological individuals from local ontologies *LO_hospital*, *LO_clinic_1* and *LO_clinic_2*: `LO_hospital-patient_instances`, `LO_clinic_1-patient_instances`, `LO_clinic_1-labtest_instances`, `LO_clinic_1-patient_instances`, and `LO_clinic_1-labtest_instances`.

Both restrictions are made 'necessary and sufficient' conditions to imply the concreteness of the `medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep` class.

5.2.5.3 Grouping Semantically Related Data in Treatment Summaries

In Figure 5.22 fifteen subclasses of the `TREATMENT_SUMMARIES_information_retrievals` parent class are created to store the results of running the *Grouping* rules specific to the combination of *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep* databses, and the information type *treatment summaries*. We only give a detailed explanation for the subclass `treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep` in order to avoid repetition and to illustrate semantically related data across all four databases.

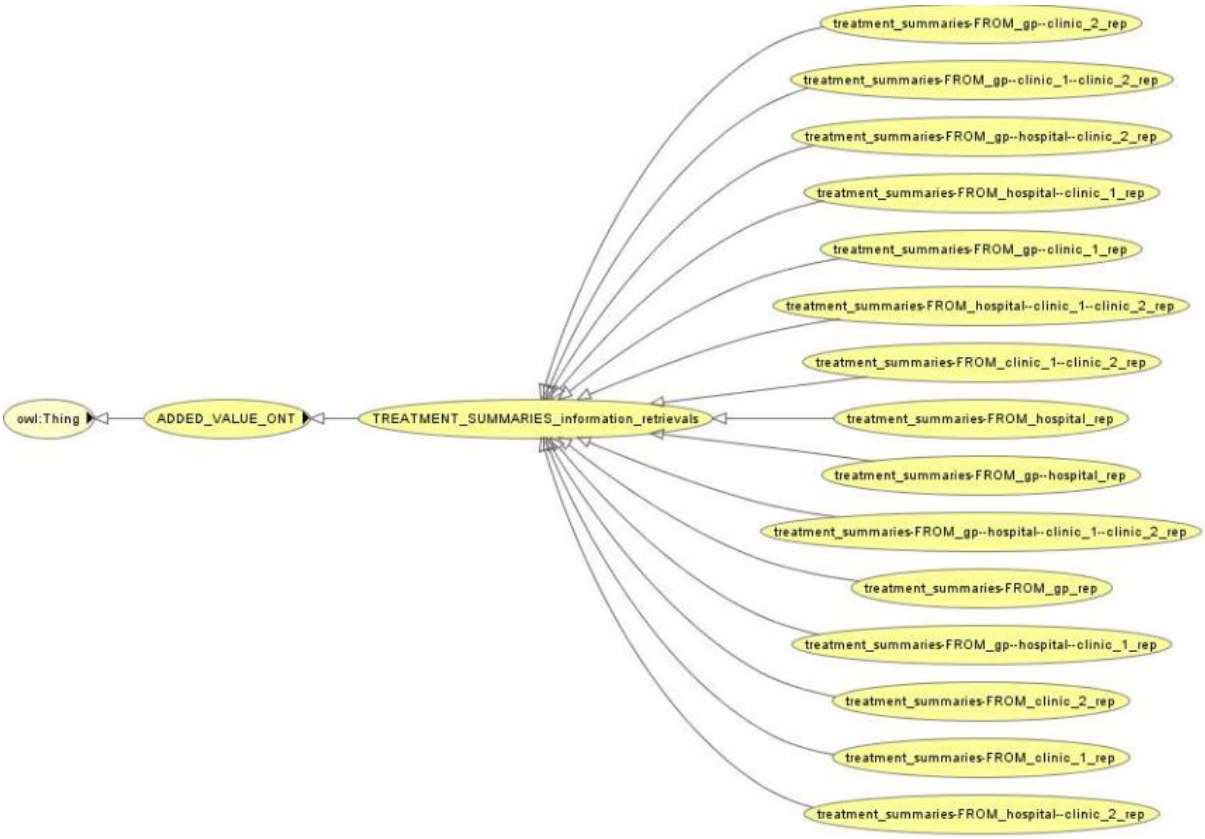

Figure 5.22 *Example of the 'TREATMENT_SUMMARIES_information_retreivals' class in the ADDED_VAL_ONT*

Grouping rules 29 and 30 in Appendix A.12 are run against the Jess engine. They use the following set of object properties to group and move ontological individuals defined in their range values from the classes in local ontologies *LO_gp* and *LO_hospital*:

- object properties `gp-treatment-TREATMENT_OVERVIEW` and `gp-treatment-DATE`, which are defined upon the `LO_gp-treatment_instances` class in local ontology *LO_gp*;
- object properties `gp-medication-MEDICINE_NUM`, `gp-medication-MEDICINE_NAME`, `gp-medication-VENDOR`, and `gp-medication-MNF_DESC`, which are defined upon the `LO_gp-medication_instances` class in local ontology *LO_gp*;

- object property `gp-medication_prescribed-DOSAGE_AMOUNT`, which are defined upon the `LO_gp-medication_prescribed_instances` class in local ontology *LO_gp*;
- object properties `hospital-treatment-TREATMENT_TYPE`, `hospital-treatment-TREATMENT_NAME` and `hospital-treatment-DATE`, which are defined upon the `LO_hospital-treatment_instances` class in local ontology *LO_hospital*;
- object properties `hospital-medication-MEDICINE_NUM`, `hospital-medication-MEDICINE_NAME`, `hospital-medication-VENDOR`, and `hospital-medication-MNF_ADDRESS`, which are defined upon the `LO_hospital-medication_instances` class in local ontology *LO_hospital;*
- object property `hospital-medication_prescribed-DOSAGE_AMOUNT`, which are defined upon the `LO_hospital-MEDICATION_PRESCRIBED_class` class in local ontology *LO_hospital*.

Table 5.6 *The results of running the Grouping rules 29 and 30*

1. M0031	14. T01245_COPDT_Chronic_pain_recovery
2. M0031_Capzasin	15. T01245_COPDT_exacerbation
3. M0031_Xhing_Ltd	16. T01245_18-04-09
4. M0031_China_pharmaceuticals	17. M222p
5. M0031_2_tablets_per_day	18. M225i
6. M0031_3_tablets_per_day	19. M222p_NAPROXEN
7. TT1989_Patient_is_suffering_from_aches_in_lower_limbs_and_has_minor_swelling_to_ankle_pain_support_through_chronic_pain_recovery_is_suggested	20. M222p_Risedronate
	21. M222p_Andheri_east_India
	22. M225i_EHOSUXIMIDE
	23. M225i_Emeside
	24. M225i_South_coast_Canada
	25. M222p_1_or_2_tablets_to_be_taken_4_times_a_day
8. TT1989_12-03-09	26. M225i_1_tablets_to_be_taken_4_times_a_day
9. TT4563_Patient_is_suffering_from_pains_in_lower_limbs_and_has_minor_swelling_to_ankle_see_prescription_given	27. M225i_1_tablets_to_be_taken_2_times_a_day
10. TT4563_14-03-09	
11. T09851_COPD_Chronic_pain_recovery	
12. T09851_COPD_exacerbation	
13. T09851_17-04-09	

The results of running *Grouping* rules 29 and 30 move the 27 ontological individuals listed in Table 5.6, into `treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep` class. The inference created as a result of running *Grouping* rules 29 and 30 is graphically shown in Figure 5.23.

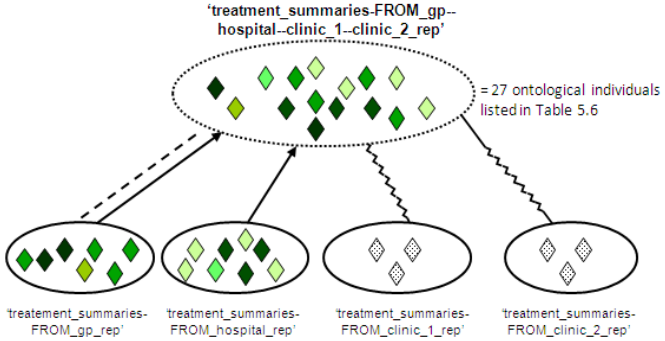

Figure 5.23 *Grouping ontological individuals from local ontologies LO_gp, LO_hospital, LO_clinic_1 and LO_clinic_2 into the concepts of ADDED_VAL_ONT that make up information type Treatment summaries*

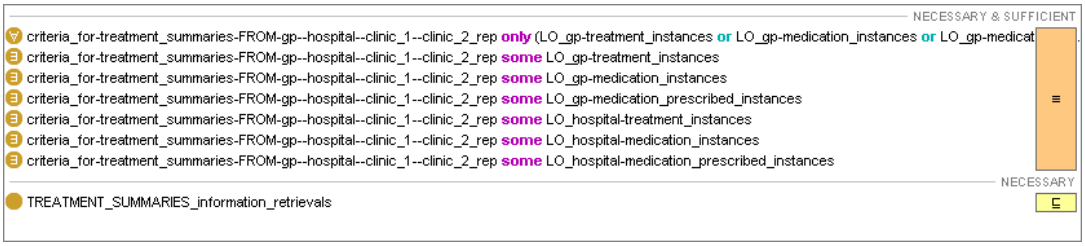

Figure 5.24 *OWL restrictions that determine the set criteria for 'treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep' class membership in the ADDED_VAL_ONT*

We model the object property `criteria_for-treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep` in order to create a relationship between the `treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep` class and the classes of local ontologies *LO_gp* and *LO_hospital*. The domain for the object property is set to the `treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep` ontological class, in order to specify where to move ontological individuals into. The range for the object property is set to the names of the ontological classes in local ontologies *LO_gp* and *LO_hospital*, in order to specify where to move ontological individuals from (i.e. semantically related data that make up information type *treatment summaries*).

Figure 5.24 shows a set of OWL restrictions applied to the object property `criteria_for-treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep` that determine the set criteria for `treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep` class membership. The existential restriction (∃) is used to describe that the `treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep` ontological class has **some** ontological individuals from local ontologies *LO_gp* and *LO_hospital*: `LO_gp-treatment_instances`, `LO_gp-medication_instances`, `LO_gp-medication_prescribed_instances`, `LO_hospital-treatment_instances`, `LO_hospital-medication_instances`, and `LO_hospital-medication_prescribed_instances`.

The universal restriction (∀) is used to describe that the 'treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep' ontological class has **only** ontological individuals local ontologies *LO_gp* and *LO_hospital*: `LO_gp-treatment_instances`, `LO_gp-medication_instances`, `LO_gp-medication_prescribed_instances`, `LO_hospital-treatment_instances`, `LO_hospital-medication_instances`, and `LO_hospital-medication_prescribed_instances`.

Both restrictions are made 'necessary and sufficient' conditions to imply the concreteness of the `treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep` class.

To avoid repetition, we do not further describe the other subclasses shown in Figures 5.16, 5.19, and 5.22 in the ADDED_VAL_ONT. However, it is obvious that similar object properties, OWL restrictions and *Grouping* rules are modelled and inferred as described above for the `medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep`, `treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep` and `patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep` classes.

5.2.6 Resolving Homonym Semantic Conflict

Having identified, while looking at the possible types of semantic conflicts in relational schemas in section 5.1.2, that there are Homonym semantic conflicts concerning attribute names, we discovered that:

- data stored in `REPORT` attribute from the `TREATMENT` table in the *Hospital_data_rep* database, and
- data stored in `REPORT` attribute from the `LABTEST` table in the *Clinic_1_data_rep* database,

exhibit False Semantic Likeness (2) as their degree of similarity. This false likeness means that that we may have originally thought that these two attribute names `REPORT` might model the same semantics but it is obvious that there is have **no** semantic similarity between them (i.e. the *Hospital_data_rep* database, models the `REPORT` as the combinations of prescriptions and treatments for Mrs Flee in Dr Smith's hospital and the *Clinic_1_data_rep* database decides to store in `REPORT` the number and information on their patients who have normal lab tests (and Mrs Flee's data might not be there!). We resolve the Homonym conflict by defining the exact object properties that guarantees that the correct `REPORT` is included in information types *medical summaries*, *treatment summaries* and *patient details*. In other words we define the exact object properties in *Grouping* rules, i.e. the exact object properties would associated to range values of the class that stores the correct `REPORT`.

5.3 Example of Generating Core Ontological Layering

The first five steps of our process have resolved the Mispelt and Case-Sensitiveand Homonym semantic conflicts. In the next three steps we have to resolve Aggregation, Synonym, Generalisation, Isomorphism, Specialisation and Union Incompatibility through ontology mappings: *alignment*, *integration* and *merge*.

It is important to note, that ADDED_VAL_ONT contains information on which exact concepts from LO_j are 'semantically related' and must be aligned into TO_k. In other words, concepts from LO_j are aligned into TO_k according to information stored in ADDED_VALUE_ONT. Alignments of LO_js into TO_k happen in step 6: we resolve Aggregation and Synonym (section 5.31).

The result set of alignments (TO_i) indicates that there are 'semantically similar concepts' in TO_k which must be integrated into DO_g. Integration of TO_k into DO_g happens in step 7: we resolve Generalisation, Isomorphism, Specialisation and Union Incompatibility (section 5.32).

The result set of integration (DO_g) indicates that there are 'semantically equivalent' concepts in DO_g that contain no semantic conflicts. Therefore DO_g and any other concepts from DO_g, which never exhibit any semantic conflicts, are merged into concepts of Go-CID (section 5.3.3). They are in the format suitable for the retrievals and will contain correct answers to Dr Smith's requests to obtain healthcare summary for Mrs Flee by retrieving a particular $InfType_d$ (*medical summaries*, *treatment summaries* and *patient details*) across Rep_i (*GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data _rep* and *Clinic_2_data_rep*).

5.3.1 Step 6: Aligning Local Ontologies

In step 6, we perform the *Low-Level* reasoning mechanism, which allows us to choose options 6a and 6b (see chapter 4, section 4.4.3.2) to execute the ontological alignment of semantically related ontological individuals stored in local ontologies *LO_gp*, *LO_hospital*, *LO_clinic_1* and *LO_clinic_2*. Figure 5.25 shows exactly what happens in step 6.

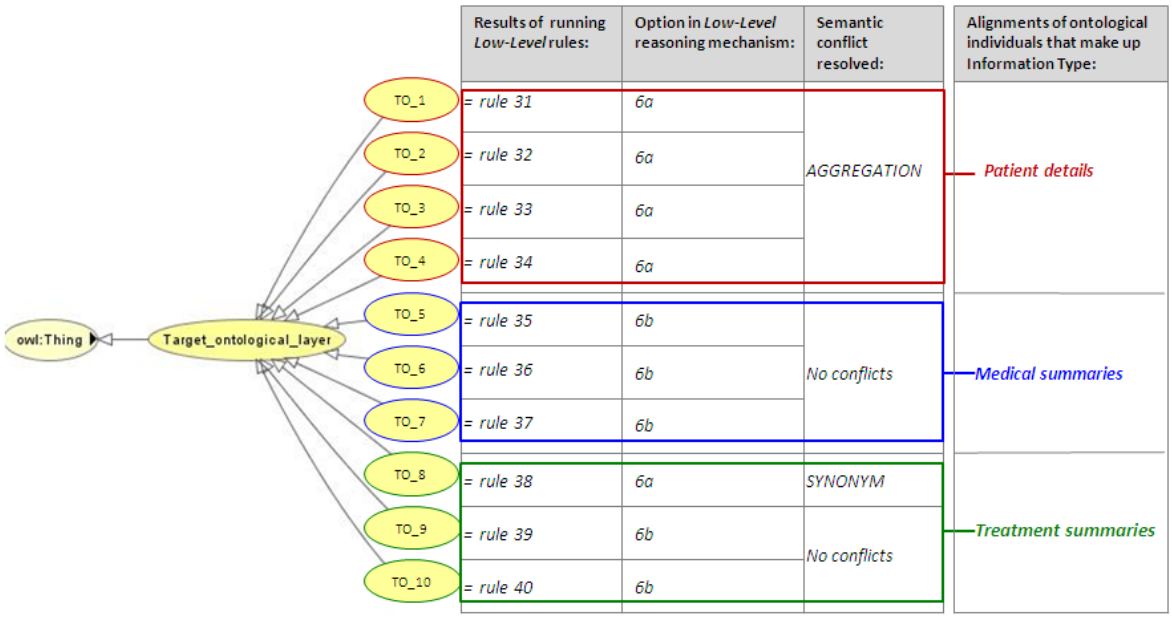

Figure 5.25 *Example of target ontologies {TO_k | k =1 ...10 }*

We align semantically related ontological individuals from local ontologies *LO_gp*, *LO_hospital*, *LO_clinic_1* and *LO_clinic_2*, listed in Table 5.4 from section 5.2.5.1, for the purpose of making *Patient details* information type. We run four SWRL *Low-Level* rules 31 – 34 (available in Appendix A.14), which in turn create target ontologies `TO_1`, `TO_2`, `TO_3` and `TO_4`. These four *Low-Level* rules enable us to create a **match** between semantically related ontological individuals and resolve the Aggregation semantic conflict which has existed across *LO_gp*, *LO_hospital*, *LO_clinic_1* and *LO_clinic_2*, when retrieving *Patient details*. For the exact list of ontological individuals which store *Patient details* and cause Aggregation conflict see Table 5.3 in Appendix A.7. The number of alignments, i.e. the number of SWRL rules, (four in this particular case) depends on the number of concepts involved in the Aggregation semantic conflict and the number of local ontologies in which these concepts reside. Details on the exact alignments for resolving Aggregation and creating *Patient details* are given in section 5.3.1.1.

The same applies to the alignment of semantically related ontological individuals, listed in Table 5.5 from section 5.2.5.2, from local ontologies *LO_gp*, *LO_hospital*, *LO_clinic_1* and *LO_clinic_2*, for the purpose of making *Medical summaries*. We run three SWRL *Low-Level* rules 35 - 37, which in turn create target ontologies `TO_5`, `TO_6` and `TO_7`. With these alignments we create matches in order to make up *Medical summaries* information type and we do not resolve any conflicts at this stage because there are no ontological individuals, listed in Table 5.5 which exhibit Aggregation and Synonym semantic

conflicts. *Low-Level* rules 35 - 37 are available in Appendix A.14. Details on the exact alignments of target ontologies TO_5 - TO_7 individuals for creating *Medical summaries* are given in section 5.3.1.2.

The creation of information type *Treatment summaries* is again done through the alignment of semantically related ontological individuals, listed in Table 5.6 from section 5.2.5.3, from local ontologies *LO_gp*, *LO_hospital*, *LO_clinic_1* and *LO_clinic_2*. We run three SWRL *Low-Level* rules 38 - 40, which in turn create target ontologies TO_8, TO_9 and TO_10. However, SWRL rule 38, which results in the creation of TO_8 resolves the Synonym conflict. All other rules help to create *Treatment summaries* at this level. *Low-Level* rules 38 - 40 are available in Appendix A.14. Details on the exact alignments for resolving Synonym semantic conflict and creating *Treatment summaries* are given in section 5.3.1.3.

It is important to note that all *Low-Level* rules for ontology alignments are run through the SWRL-plug-in in Protégé 3.4, using the JESS engine. All classes in target ontologies TO_1 – TO_10 are created through Protégé 3.4. Screen shots of the inference (e.g. ontological individuals stored in TO_1 – TO_10), as a result of running *Low-Level* rules 31 - 40 can be found Appendix A.15. (Note: Appendix A.15 is stored on the CD-ROM).

5.3.1.1 Aligning Semantically Related Data in Patient Details

Low-Level rule 31 transfers and infers the following set of semantically related ontological individuals from the classes in local ontologies *LO_gp* and *LO_hospital* into TO_1:

- ontological individuals JANE, FLEE, FEMALE and JULY_04_1970 defined in the range values of object properties:
  - gp-patient-FIRST_NAME, gp-patient-LAST_NAME and gp-patient-DOB, which are defined upon the LO_gp-patient_instances class in local ontology *LO_gp*;
- ontological individuals JANE_FLEE, Female and JULY_04_1970 defined in the range values of the object properties:
  - hospital-patient-NAME and hospital-patient-DOB, which are defined upon the LO_hospital-patient_instances class in local ontology *LO_hospital*.

Low-Level rule 31 use option 6a in the *Low-Level* reasoning mechanism, to infer the new ontological individuals JANE and FLEE through running a comparison between ontological individuals JANE, FLEE-JANE_FLEE. The comparison is run using OWL conditions set upon datatype properties:

- has_same_FIRST_NAME allowed values JANE (i.e. literal value = JANE, Figure 5.26), and
- has_same_LAST_NAME allowed values FLEE (i.e. literal value = FLEE, Figure 5.27).

We model the datatype properties has_same_FIRST_NAME and has_same_LAST_NAME in order to create a relationship between the TO_1 class and the allowed values of JANE and FLEE. The domain for the datatype property is set to the TO_1 ontological class and the range is set to a string literal.

A match is established between the pair of ontological individuals: FEMALE-Female and the pair of ontological individuals JULY_04_1970-JULY_04_1970 without running a comparison between the ontological individuals.

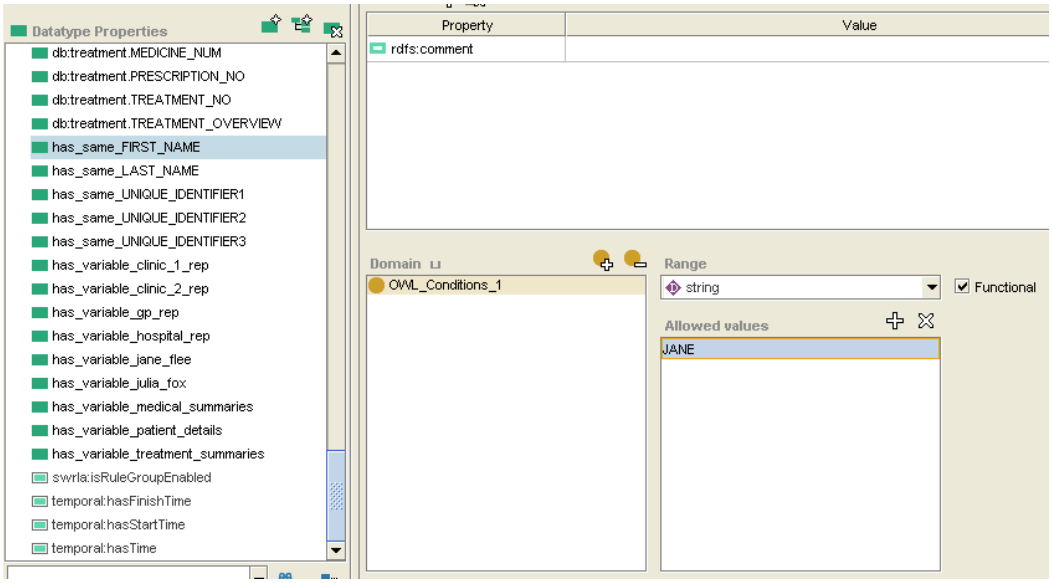

Figure 5.26 *Example of OWL conditions set upon the datatype property 'has_same_FIRST_NAME'*

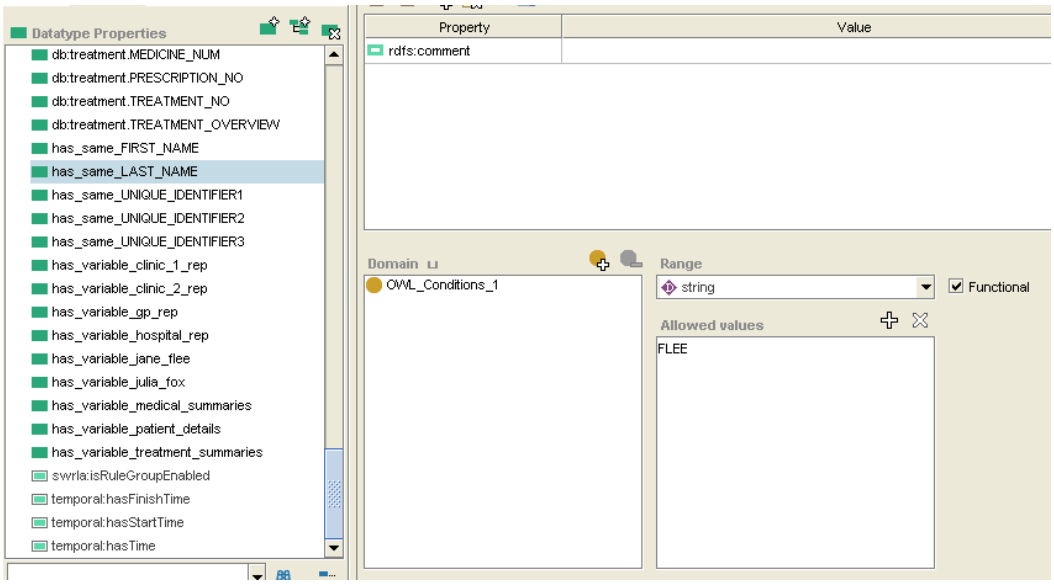

Figure 5.27 *Example of OWL conditions set upon the datatype property 'has_same_LAST_NAME'*

Table 5.7 *The results of running the Low-Level rule 31*

1.	JANE
2.	FLEE
3.	FEMALE
4.	Female
5.	JULY_04_1970
6.	JULY_04_1970

After running *Low-Level* rule 31, 2 ontological individuals numbered 1. – 2. listed in Table 5.7 are inferred into `TO_1`, and 4 ontological individuals numbered 3. – 6. listed in Table 5.7 are transferred into `TO_1` class, thus creating a 'semantic relation' between them. The inference created as a result of running *Low-Level* rule 31 is graphically shown in Figure 5.28. We use the same rational as mentioned in Figure 4.13 in chapter 4, i.e. inference of ontological individuals is as a blue broken line between ontological individuals and NOT ontological classes. Comparison of ontological individuals is shown as a green broken line between ontological individuals. Transference of ontological individuals is shown as a black broken line between ontological individuals and NOT ontological classes.

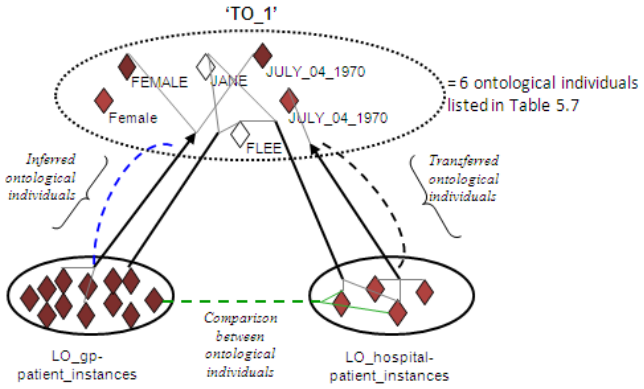

Figure 5.28 *Transferring and inferring ontological individuals from local ontologies LO_gp and LO_hospital into the TO_1 target ontology*

Low-Level rule 32 in Appendix A.14 transfers and infers the following set of semantically related ontological individuals from the classes in local ontologies *LO_gp* and *LO_clinic_1* into `TO_2`:

- ontological individuals `JANE`, `FLEE`, `FEMALE`, `JULY_04_1970` and `TEL_02075698899` defined in the range values of object properties:
  - `gp-patient-FIRST_NAME`, `gp-patient-LAST_NAME`, `gp-patient-DOB` and `gp-patient-TELEPHONE`, which are defined upon the `LO_gp-patient_instances` class in local ontology *LO_gp*;
- ontological individuals `JANE_FLEE`, `Female`, `JULY_4_1970` and `Tel_02075698899` defined in the range values of the object properties:
  - `clinic_1-patient-NAME`, `clinic_1-patient-SEX`, `clinic_1-patient-DOB` and `clinic_1-patient-TELEPHONE`, which are defined upon the `LO_clinic_1-patient_instances` class in local ontology *LO_clinic_1*.

Low-Level rule 32 use option 6a in the *Low-Level* reasoning mechanism, to infer the new ontological individuals `JANE` and `FLEE` through running a comparison between ontological individuals `JANE`, `FLEE-JANE_FLEE`. The comparison is run using OWL conditions set upon datatype properties:

- `has_same_FIRST_NAME` allowed values `JANE` (i.e. literal value = JANE, Figure 5.26), and
- `has_same_LAST_NAME` allowed values `FLEE` (i.e. literal value = FLEE, Figure 5.27).

A match is established between the pair of ontological individuals: `FEMALE-Female`, the pair of ontological individuals: `JULY_04_1970-JULY_04_1970` and the pair of ontological individuals: `TEL_02075698899-TEL_02075698899` without running a comparison between the ontological individuals.

Table 5.8 *The results of running the Low-Level rule 32*

1.	JANE	5.	JULY_04_1970
2.	FLEE	6.	JULY_04_1970
3.	FEMALE	7.	TEL_02075698899
4.	Female	8.	TEL_02075698899

After running *Low-Level* rule 32, 2 ontological individuals numbered 1. – 2. listed in Table 5.8 are inferred into `TO_2`, and 6 ontological individuals numbered 3. – 8. listed in Table 5.8 are transferred into `TO_2` class, thus creating a 'semantic relation' between them. The inference created as a result of running *Low-Level* rule 32 is graphically shown in Figure 5.29.

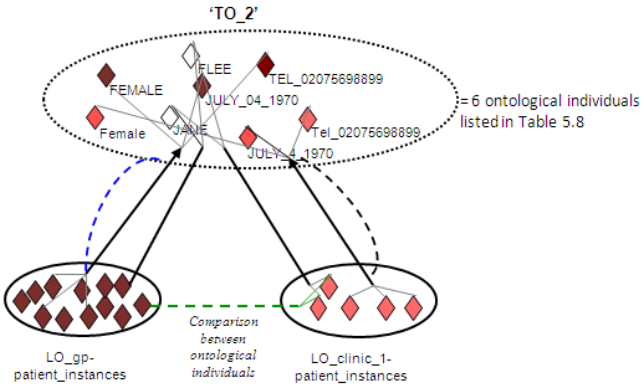

Figure 5.29 *Transferring and inferring ontological individuals from local ontologies LO_gp and LO_clinic_1 into the TO_2 target ontology*

Low-Level rule 33 in Appendix A.14 transfers and infers the following set of semantically related ontological individuals from the classes in local ontologies *LO_hospital* and *LO_clinic_2* into `TO_3`:

- ontological individuals `JANE_FLEE`, `Female` and `JULY_04_1970` defined in the range values of the object properties:
  - `hospital-patient-NAME` and `hospital-patient-DOB`, which are defined upon the `LO_hospital-patient_instances` class in local ontology *LO_hospital*;
- ontological individuals `JANE`, `FLEE`, `Female` and `JULY_04_1970` defined in the range values of the object properties:
  - `clinic_2-patient-FIRST_NAME`, `clinic_2-patient-LAST_NAME` and `clinic_2-patient-DOB`, which are defined upon the `LO_clinic_2-patient_instances` class in local ontology *LO_clinic_2*.

Low-Level rule 33, use option 6a in the *Low-Level* reasoning mechanism, to infer the new ontological individuals `JANE` and `FLEE` through running a comparison between ontological individuals `JANE`, `FLEE-JANE_FLEE`. The comparison is run using OWL conditions set upon datatype properties:

- `has_same_FIRST_NAME` allowed values `JANE` (i.e. literal value = JANE, Figure 5.26), and
- `has_same_LAST_NAME` allowed values `FLEE` (i.e. literal value = FLEE, Figure 5.27).

A match is established between the pair of ontological individuals: `Female` - `Female` and the pair of ontological individuals: `JULY_04_1970-JULY_04_1970` without running a comparison between the ontological individuals.

Table 5.9 *The results of running the Low-Level rule 33*

1.	JANE
2.	FLEE
3.	Female
4.	Female
5.	JULY_04_1970
6.	JULY_4_1970

After running *Low-Level* rule 33, 2 ontological individuals numbered 1. – 2. listed in Table 5.9 are inferred into `TO_3`, and 4 ontological individuals numbered 3. – 6. listed in Table 5.9 are transferred into

TO_3 class, thus creating a 'semantic relation' between them. The inference created as a result of running *Low-Level* rule 33 is graphically shown in Figure 5.30.

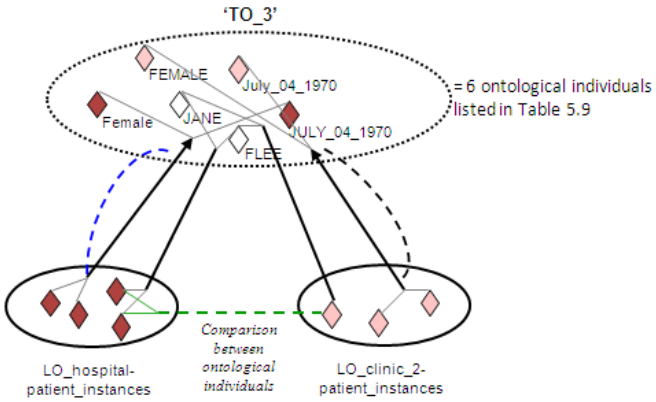

Figure 5.30*Transferring and inferring ontological individuals from local ontologies LO_hospital and LO_clinic_2 into the TO_3 target ontology*

Low-Level rule 34 in Appendix A.14 transfers or infers the following set of semantically related ontological individuals from the classes in local ontologies *LO_clinic_1* and *LO_clinic_2* into TO_4:

- ontological individuals JANE_FLEE, Female and JULY_4_1970 defined in the range values of the object properties:
  - clinic_1-patient-NAME, clinic_1-patient-SEX and clinic_1-patient-DOB, which are defined upon the LO_clinic_1-patient_instances class in local ontology *LO_clinic_1*;
- ontological individuals JANE, FLEE, Female and JULY_04_1970 defined in the range values of the object properties:
  - clinic_2-patient-FIRST_NAME, clinic_2-patient-LAST_NAME and clinic_2-patient-DOB, which are defined upon the LO_clinic_2-patient_instances class in local ontology *LO_clinic_2*.

Low-Level rule 34, use option 6a in the *Low-Level* reasoning mechanism, to infer the new ontological individuals JANE and FLEE through running a comparison between ontological individuals JANE, FLEE-JANE_FLEE. The comparison is run using OWL conditions set upon datatype properties:

- has_same_FIRST_NAME allowed values JANE (i.e. literal value = JANE, Figure 5.26.), and
- has_same_LAST_NAME allowed values FLEE (i.e. literal value = FLEE, Figure 5.27.).

A match is established between the pair of ontological individuals: Female-Female and the pair of ontological individuals JULY_4_1970-JULY_04_1970 without running a comparison between the ontological individuals.

Table 5.10 *The results of running the Low-Level rule 34*

1.	JANE
2.	FLEE
3.	Female
4.	Female
5.	JULY_04_1970
6.	JULY_4_1970

After running *Low-Level* rule 34, 2 ontological individuals numbered 1. – 2. listed in Table 5.10 are inferred into `TO_4`, and 4 ontological individuals numbered 3. – 6. listed in Table 5.10 are transferred into `TO_4` class, thus creating a 'semantic relation' between them. The inference created as a result of running *Low-Level* rule 34 is graphically shown in Figure 5.31.

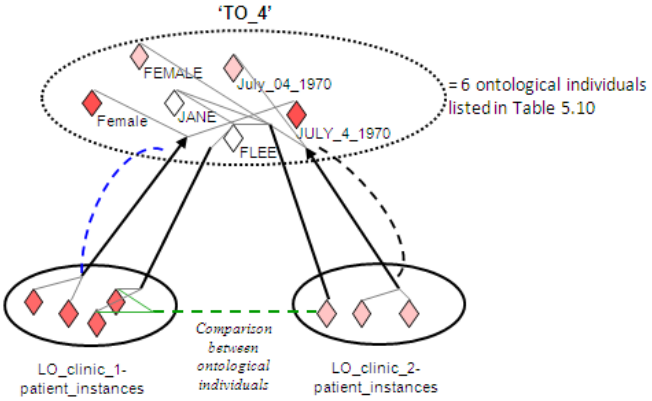

Figure 5.31 *Transferring and inferring ontological individuals from local ontologies LO_clinic_1 and LO_clinic_2 into the TO_4 target ontology*

5.3.1.1.1 Resolving Aggregation Semantic Conflict

Having identified, while looking at the possible types of semantic conflicts in relational schemas in section 5.1.2, that there are Aggregation semantic conflicts concerning attribute names, we discovered that:

- data stored in the `LAST_NAME` and `FIRST_NAME` from the `PATIENT` table in the *Hospital_data_rep* database, and
- data stored in the `NAME` attribute from the `PATIENT` table in the *Clinic_2_data_rep* database,

exhibit Semantic Likeness (3) as their degree of similarity. This is an example of Aggregation in attribute names, which is resolved after the alignment of ontological individuals from local ontologies *LO_hospital* and *LO_clinic_2* into the `TO_3` target ontology. Specifically, the conflict is resolved by inferring the new ontological individuals `JANE` and `FLEE` into target ontology `TO_3` as a consequence of running a comparison against the OWL conditions set upon datatype properties `has_same_FIRST_NAME = JANE` and `has_same_LAST_NAME = FLEE`. Ontological individuals `JANE` and `FLEE` will ultimately be merged into the ontological concepts of Go-CID, i.e. the new ontological individuals `JANE` and `FLEE` will be relocated into the `FIRST_NAME` and `LAST_NAME` classes of the Go-CID respectively.

5.3.1.2 Aligning Semantically Related Data in Medical Summaries

Low-Level rule 35 in Appendix A.14 transfers the following set of semantically related ontological individuals from the classes in local ontologies *LO_hospital* and *LO_clinic_1* into `TO_5`:

- ontological individuals:
  `Mrs_Flee_complains_of_severe_pain_in_left_ankle_Minor_swelling_evident_and_xrays_taken_admitted_as_overnight_stay_and_found_to_have_acute_COPD_exacerbation`, `no_major_illness_evident` and
  'no_chronic_disease_evident` defined in the range values of object properties:

- hospital-patient-MEDICAL_SUMMARY, hospital-patient-MAJOR_ILLNESS and hospital-patient-CHRONIC_ILLNESS, which are defined upon the 'LO_hospital-patient_instances' class in local ontology *LO_hospital*;

- ontological individuals:

  Mrs_Flee_has_a_regular_cervical_smear_test_results_appear_normal, Mrs_Flee_has_come_into_the_clinic_for_a_blood_test_complains_of_fatigue, none and none_found defined in the range values of the object properties:

  - clinic_1-patient-PREVIOUS_MEDICAL_SUMMARY, clinic_1-patient-CURRENT_MEDICAL_SUMMARY, clinic_1-patient-MAJOR_ILLNESS and clinic_1-patient-CHRONIC_ILLNESS, which are defined upon the LO_clinic_1-patient_instances class in local ontology *LO_clinic_1*.

A match is established between the set of ontological individuals: Mrs_Flee_complains_xxx-Mrs_Flee_has_xxx-Mrs_Flee_has_xxx, pair of ontological individuals: no_major_xxx-none, and the pair of ontological individuals: no_chro_xxx-none_xxx (where 'xxx' denotes the full name of the ontological individual mentioned above).

Table 5.11 *The results of running the Low-Level rule 35*

1.	Mrs_Flee_complains_of_severe_pain_in_left_ankle_Minor_swelling_evident_and_xrays_taken_admitted_as_overnight_stay_and_found_to_have_acute_COPD_exacerbation
2.	Mrs_Flee_has_a_regular_cervical_smear_test_results_appear_normal
3.	Mrs_Flee_has_come_into_the_clinic_for_a_blood_test_complains_of_fatigue
4.	no_major_illness_evident
5.	no_chronic_disease
6.	none
7.	none_found

After running *Low-Level* rule 35, 7 ontological individuals listed in Table 5.11 are transferred into TO_5 class, thus creating a 'semantic relation' between them. The inference created as a result of running *Low-Level* rule 35 is graphically shown in Figure 5.32. We use the same rational is used as mentioned in Figure 4.13 in chapter 4. The transference of ontological individuals is as a black broken line between ontological individuals and NOT ontological classes.

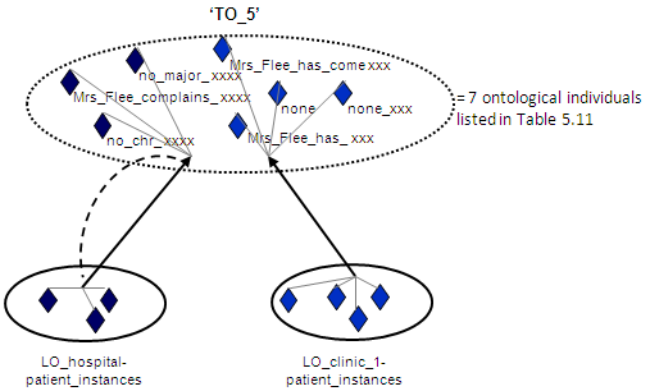

Figure 5.32 *Transferring ontological individuals from local ontologies LO_hospital and LO_clinic_1 into the TO_5 target ontology*

Low-Level rule 36 in Appendix A.14 transfers the following set of semantically related ontological individuals from the classes in local ontologies *LO_hospital* and *LO_clinic_1* into `TO_6`:

- ontological individuals:

  `Mrs_Flee_has_a_regular_cervical_smear_test_results_appear_normal`, `Mrs_Flee_has_come_into_the_clinic_for_a_blood_test_complains_of_fatigue`, `none` and `none_found` defined in the range values of the object properties:

  - `clinic_1-patient-PREVIOUS_MEDICAL_SUMMARY`, `clinic_1-patient-CURRENT_MEDICAL_SUMMARY`, `clinic_1-patient-MAJOR_ILLNESS` and `clinic_1-patient-CHRONIC_ILLNESS`, which are defined upon the `LO_clinic_1-patient_instances` class in local ontology *LO_clinic_1*;
- ontological individuals `Mrs_Flee_complains_of_shortness_of_breath`, `no_MJ` and `none_cd_found` defined in the range values of the object properties:
  - `clinic_2-patient-MEDICAL_SUMMARY`, `clinic_2-patient-MAJOR_ILLNESS` and `clinic_2-patient-CHRONIC_ILLNESS`, which are defined upon the `LO_clinic_2-patient_instances` class in local ontology *LO_clinic_2*.

A match is established between the set of ontological individuals: `Mrs_xxx-Mrs_xxx-Mrs_xxx`, the pair of ontological individuals: `none-no_MJ`, and `none_xxx-no_xxx`, (where 'xxx' denotes the full name of the ontological individual mentioned above).

Table 5.12 *The results of running the Low-Level rule 36*

1.	Mrs_Flee_complains_of_shortness_of_breath
2.	Mrs_Flee_has_a_regular_cervical_smear_test_results_appear_normal
3.	Mrs_Flee_has_come_into_the_clinic_for_a_blood_test_complains_of_fatigue
4.	none
5.	none_found
6.	No_MJ
7.	no_cd_found

After running the *Low-Level* rule 36, 7 ontological individuals listed in Table 5.12 are transferred into `TO_6` class, thus creating a 'semantic relation' them. The inference created as a result of running *Low-Level* rule 36 is graphically shown in Figure 5.33.

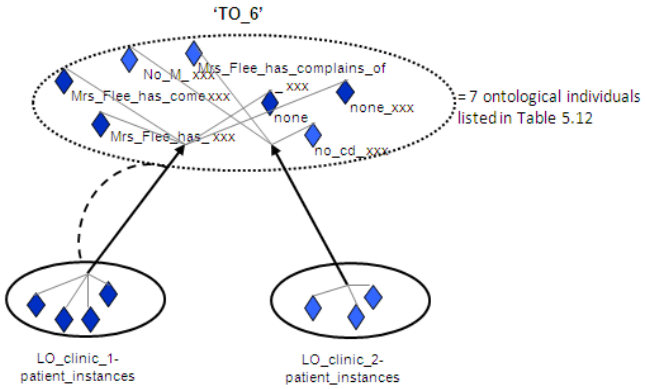

Figure 5.33 *Transferring ontological individuals from local ontologies LO_clinic_1 and LO_clinic_2 into the TO_6 target ontology*

Low-Level rule 37 in Appendix A.14 transfers the following set of semantically related ontological individuals from the classes in local ontologies *LO_hospital* and *LO_clinic_1* into `TO_7`:

- ontological individuals `LT256_Smear_test`, `LT256_Cervical_Type_3`, `LT256_Normal`, `LT256_16-01-08`, `LT123_Pathology`, `LT123_Blood_test_Type_4123`, `LT123_anaemia_level_46` and `LT123_16-02-08` defined in the range values of the object properties:
  - `clinic_1-labtest-LABTEST_TYPE`, `clinic_1-labtest-LABTEST_NAME`, `clinic_1-labtest-LABTEST_RESULTS`, and `clinic_1-labtest-DATE`, which are defined upon the `LO_clinic_1-labtest_instances` class in local ontology *LO_clinic_1*;
- ontological individuals `LL456_Used_to_identify_lung_diseases`, `LL456_Radiation`, `LL456_Xray`, `LL456_fileID_wavelength908` and `LL456_28-04-09` defined in the range values of the object properties:
  - `clinic_2-labtest-LABTEST_OVERVIEW`, `clinic_2-labtest-LABTEST_DATA`, `clinic_2-labtest-LABTEST_TYPE`, `clinic_2-labtest-LABTEST_NAME`, `clinic_2-labtest-LABTEST_RESULTS` and `clinic_2-labtest-DATE`, which are defined upon the `LO_clinic_2-labtest_instances` class in the local ontology *LO_clinic_2*.

A match is established between ALL the set of ontological individuals mentioned above.

Table 5.13 *The results of running the Low-Level rule 37*

1. LT256_Smear_test	8. LT123_16-02-08
2. LT256_Cervical_Type_3	9. LL456_Radiation
3. LT256_Normal	10. LL456_Xray
4. LT256_16-01-08	11. LL456_fileID_wavelength908
5. LT123_Pathology	12. LL456_28-04-09
6. LT123_Blood_test_Type_4123	13. LL456_data_aa2
7. LT123_anaemia_level_46	

After running the *Low-Level* rule 37, 13 ontological individuals in listed in Table .13 are transferred into `TO_7` class, thus creating a 'semantic relation' between them. The inference created as a result of running *Low-Level* rule 37 is graphically shown in Figure 5.34.

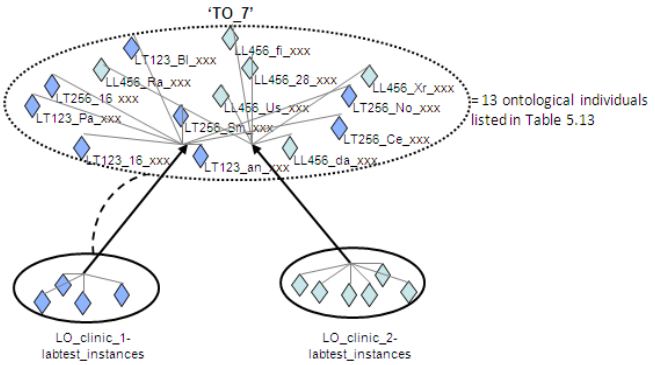

Figure 5.34 *Transferring ontological individuals from local ontologies LO_clinic_1 and LO_clinic_2 into the TO_7 target ontology*

5.3.1.3 Aligning Semantically Related Data in Treatment Summaries

Low-Level rule 38 in Appendix A.14 infers the following set of semantically related ontological individuals from the classes in local ontologies *LO_gp* and *LO_hospital* into `TO_8`:

- ontological individual `M0031` defined in the range value of object property:
  - `gp-medication-MEDICINE_NUM`, which is defined upon the `LO_gp-patient_instances` class in local ontology *LO_gp*;
- ontological individuals `M222p` and `M225i` defined in the range values of the object property:
  - `hospital-medication-MEDICINE_NO`, which is defined upon the `LO_hospital-patient_instances` class in local ontology *LO_hospital*.

Low-Level rule 38 uses option 6a to infer the new ontological individuals `M0031`, `M222p` and `M225i`, through running a comparison between existing ontological individuals `M0031`, `M222p` and `M225i`. The comparison is run using OWL conditions set upon datatype properties:

- `has_same_UNIQUE_IDENTIFIER1` with allowed values `M0031` (literal value = M0031, Figure 5.35)
- `has_same_UNIQUE_IDENTIFIER2` with allowed values `M222p` (literal value = M222p, Figure 5.36.), and
- `has_same_UNIQUE_IDENTIFIER3` with allowed values `M225i` (literal value = M225i, Figure 5.37.)

We model the datatype properties `has_same_UNIQUE_IDENTIFIER1`, `has_same_UNIQUE_IDENTIFIER2` and ‘`has_same_UNIQUE_IDENTIFIER3` in order to create a relationship between the `TO_8` class and the allowed values of `M0031`, `M222p` and `M225i`. The domain for the datatype property is set to the `TO_8` ontological class and the range is set to string literal.

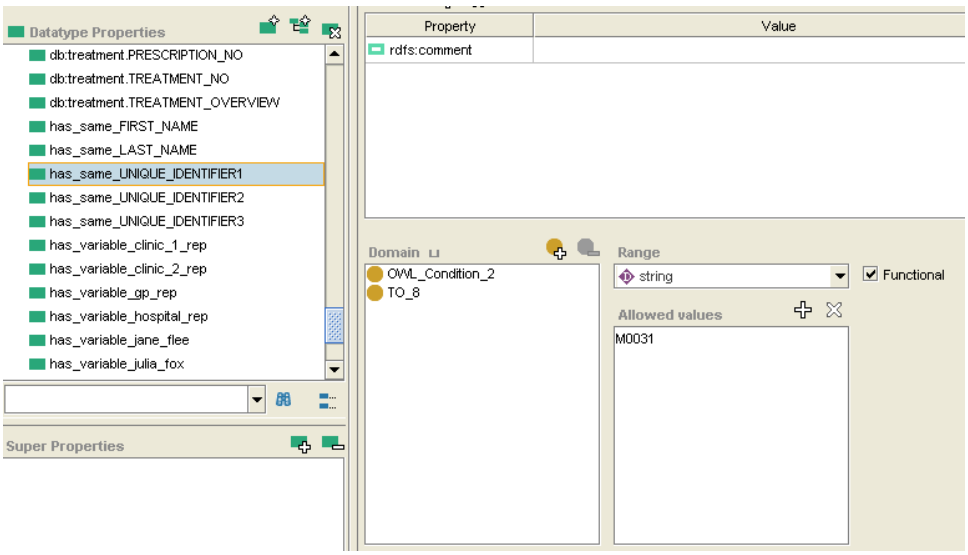

Figure 5.35. *Example of OWL conditions set upon the datatype property ‘has_same_UNIQUE_IDENTIFIER1’*

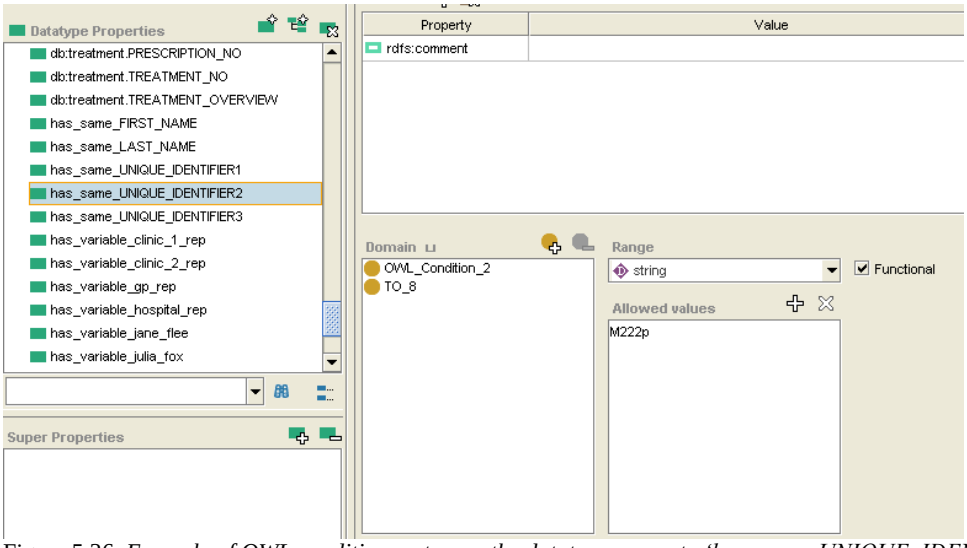

Figure 5.36. *Example of OWL conditions set upon the datatype property 'has_same_UNIQUE_IDENTIFIER2'*

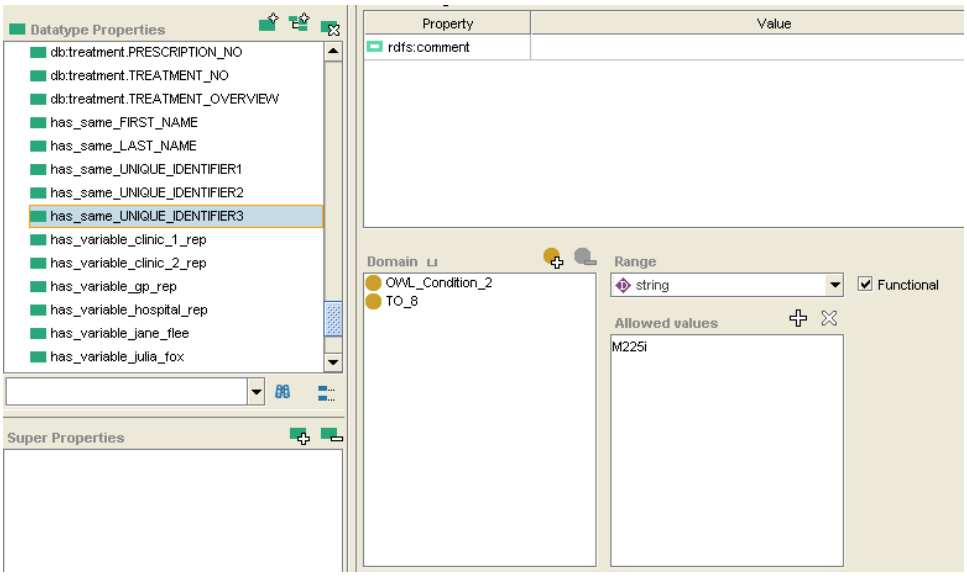

Figure 5.37. *Example of OWL conditions set upon the datatype property 'has_same_UNIQUE_IDENTIFIER3'*

Table 5.14 *The results of running the Low-Level rule 38*

1. M0031
2. M222p
3. M225i

After running the *Low-Level* rule 38, 3 ontological individuals listed in Table 5.14 are inferred into `TO_8` class, thus creating a 'semantic relation' between them. The inference created as a result of running *Low-Level* rule 38 is graphically shown in Figure 5.38. We use the same rational is used as that mentioned for Figure 4.13 in chapter 4, i.e. inference of ontological individuals is shown as a blue broken line between ontological individuals and NOT ontological classes. Comparison of ontological individuals is shown as a green broken line between ontological individuals.

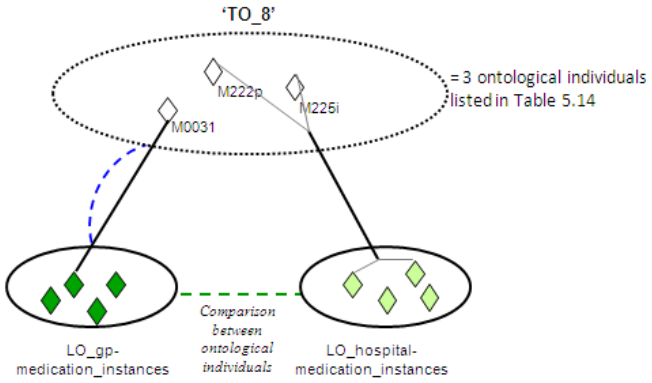

Figure 5.38 *Inferring ontological individuals from local ontologies LO_gp and LO_hospital into the TO_8 target ontology*

Low-Level rule 39 in Appendix A.14 uses option 6b to transfer the following set of semantically related ontological individuals from the classes in local ontologies *LO_gp* and *LO_hospital* into `TO_9`:

- ontological individuals: `T1989_Patient_is_suffering_from_aches_in_lower_limbs_and_has_minor_swelling_to_ankle_pain_support_through_chronic_pain_recovery_is_suggested` and `TT1989_12-03-09` defined in the range values of the object properties:
  - `gp-treatment-TREATMENT_OVERVIEW` and `gp-treatment-DATE`, which are defined upon the `LO_gp-treatment_instances` class in local ontology *LO_gp*;
- ontological individuals `T09851_COPD_Chronic_pain_recovery`, `T09851_COPD_exacerbation` and `T09851_17-04-09` defined in the range values of the object properties:
  - `hospital-treatment-TREATMENT_TYPE`, `hospital-treatment-TREATMENT_NAME` and `hospital-treatment-DATE`, which are defined upon the `LO_hospital-treatment_instances` class in local ontology *LO_hospital*.

A match is established between ALL the set of ontological individuals mentioned above.

Table 5.15 *The results of running the Low-Level rule 39*

1.	T1989_Patient_is_suffering_from_aches_in_lower_limbs_and_has_minor_swelling_to_ankle_pain_support_through_chronic_pain_recovery_is_suggested
2.	TT1989_12-03-09
3.	T09851_COPD_Chronic_pain_recovery
4.	T09851_COPD_exacerbation
5.	T09851_17-04-09

After running the *Low-Level* rule 39, 5 ontological individuals in listed in Table 5.15 are transferred into `TO_9` class, thus creating a 'semantic relation' as between them. The inference created as a result of running the *Low-Level* rule 39 is graphically shown in Figure 5.39.

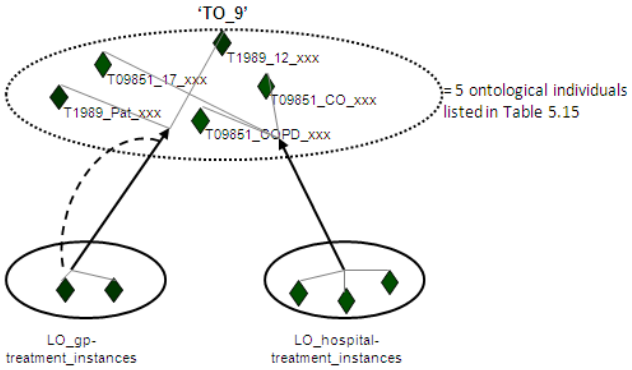

Figure 5.39 *Transferring ontological individuals from local ontology LO_gp and LO_hospital into the TO_9 target ontology*

Low-Level rule 40 in Appendix A.14 uses option 6b to transfer the following set of semantically related ontological individuals from the classes in local ontologies *LO_hospital* and *LO_clinic_1* into `TO_10`:

- ontological individuals `M0031_Capzasin`, `M0031_Xhing_Ltd` and `M0031_China_pharmaceuticals` defined in the range values of the object properties:
  - `gp-medication-MEDICINE_NAME`, `gp-medication-VENDOR` and `gp-medication-MNF_DESC`, which are defined upon the `LO_gp-treatment_instances` class in local ontology *LO_gp*;
- ontological individual `M0031_2_tablets_per_day` defined in the range values of object property:
  - `gp-medication_prescribed-DOSAGE_AMOUNT`, which are defined upon the `LO_gp-treatment_instances` class in local ontology *LO_gp*;
- ontological individuals `M222p_NAPROXEN`, `M222p_Risedronate`, `M222p_Andheri_east_India`, `M225i_EHOSUXIMIDE`, `M225i_Emeside`, `M225i_South_coast_Canada` defined in the range values of object properties:
  - `hospital-medication-MEDICINE_NAME`, `hospital-medication-VENDOR` and `hospital-medication-MNF_ADD`, which are defined upon the `LO_hospital-treatment_instances` class in local ontology *LO_hospital*;
- ontological individuals `M222p_1_or_2_tablets_to_be_taken_4_times_a_day`, and `M225i_1_tablets_to_be_taken_4_times_a_day` defined in the range values of the object property:
  - `hospital-medication_prescribed-DOSAGE_AMOUNT` , which are defined upon the `LO_hospital-treatment_instances` class in local ontology *LO_hospital*;

A match is established between ALL the set of ontological individuals mentioned above.

Table 5.16 *The results of running the Low-Level rule 40*

1. M0031_Capzasin	8. M225i_EHOSUXIMIDE
2. M0031_Xhing_Ltd	9. M225i_Emeside
3. M0031_China_pharmaceuticals	10. M225i_South_coast_Canada
4. M0031_2_tablets_per_day'	11. M222p_1_or_2_tablets_to_be_taken_4_times_a _day
5. M222p_NAPROXEN	
6. M222p_Risedronate	12. M225i_1_tablets_to_be_taken_4_times_a_day
7. M222p_Andheri_east_India	

After running the *Low-Level* rule 40, 12 ontological individuals in listed in Table 5.16 are transferred into `TO_10` class, thus creating a 'semantic relation' as between them. The inference created as a result of running the *Low-Level* rule 40 is graphically shown in Figure 5.40.

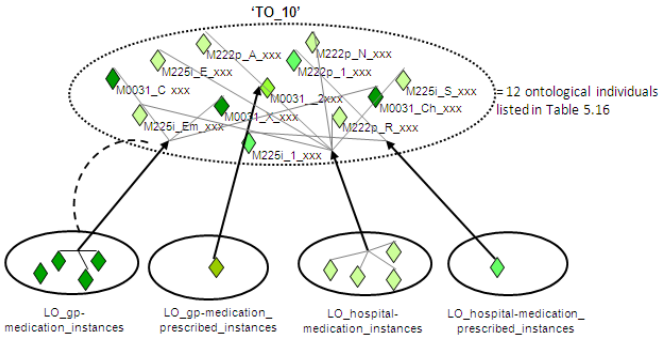

Figure 5.40 *Transferring ontological individuals from classes from local ontologies LO_gp and LO_hospital into the TO_10 target ontology*

5.3.1.3.1 Resolving Synonym Semantic Conflict

Having identified, while looking at the possible types of semantic conflicts in relational schemas in section 5.1.2, we have discovered that:

- data stored in the `MEDICINE_NUM` attribute from the `MEDICATION` table in the *GP_data_rep* database, and
- data stored in the `MEDICINE_NO` attribute from the `MEDICATION` table in the *Hospital_data_rep* database,

exhibit Semantic Likeness (3) as their degree of similarity. This is an example of Synonym semantic conflict concerning attribute names, which is resolved after the alignment of ontological individuals from local ontologies *LO_gp* and *LO_hospital* into the `TO_8` target ontology. Specifically, the conflict is resolved by inferring the new ontological individuals `M0031`, `M222p` and `M225` into the target ontology `TO_8` as a consequence of running a comparison against the OWL conditions set upon datatype properties `has_same_UNIQUES_IDENTIFIER1` = `M0031`, `has_same_UNIQUES_IDENTIFIER2` = `M222p`, and `has_same_UNIQUES_IDENTIFIER3` = `M225i`. These three individuals will ultimately be merged into the ontological concepts of Go-CID, i.e. the new ontological individuals `M0031`, `M222p` and `M225i` will be relocated into the `MEDICINE_NUMBER` class of Go-CID.

5.3.2 Step 7: Integrating Target Ontologies

In step 7, we perform the *High-Level* reasoning mechanism, which allows us to choose option 7b (see chapter 4, section 4.3.4.3) to execute the integration of semantically similar ontological individuals stored in target ontologies `TO_1` - `TO_10` (created though alignment in the previous step). Figure 5.45 shows exactly what happens in step 7.

It is obvious from ontological individuals listed in Tables 5.7 - 5.10 from section 5.3.1.1 that we have no semantic conflicts within target ontologies `TO_1`, `TO_2`, `TO_3` and `TO_4`, because we have already resolved Aggregation conflict in the previous step when creating information type *Patient details*. However, individuals contained in target ontologies `TO_1` - `TO_4` must be made semantically similar in

order to establish their semantic equivalence (see section 4.3.4.3 in chapter 4). Therefore, in order to establish semantic equivalence we integrate semantically similar individuals from target ontologies TO_1 - TO_4 by asserting them into derived ontologies DO_1, DO_2, DO_3, and DO_4. Assertion happens by running four SWRL *High-Level* rules 41-44 (available in Appendix A16). The outcome of assertion is a **link** between individuals in derived ontologies DO_1 - DO_4 which are semantically equivalent. We also say that asserted ontological individuals (i) are integrated into derived ontologies DO_1 - DO_4, (ii) exhibit 'semantic correspondence' as a consequence of their integrations. Details on the exact integrations target ontologies TO_1 - TO_4 individuals which create information type *Patient details* at this stage (i.e. step 7) are given in section 5.3.2.1.

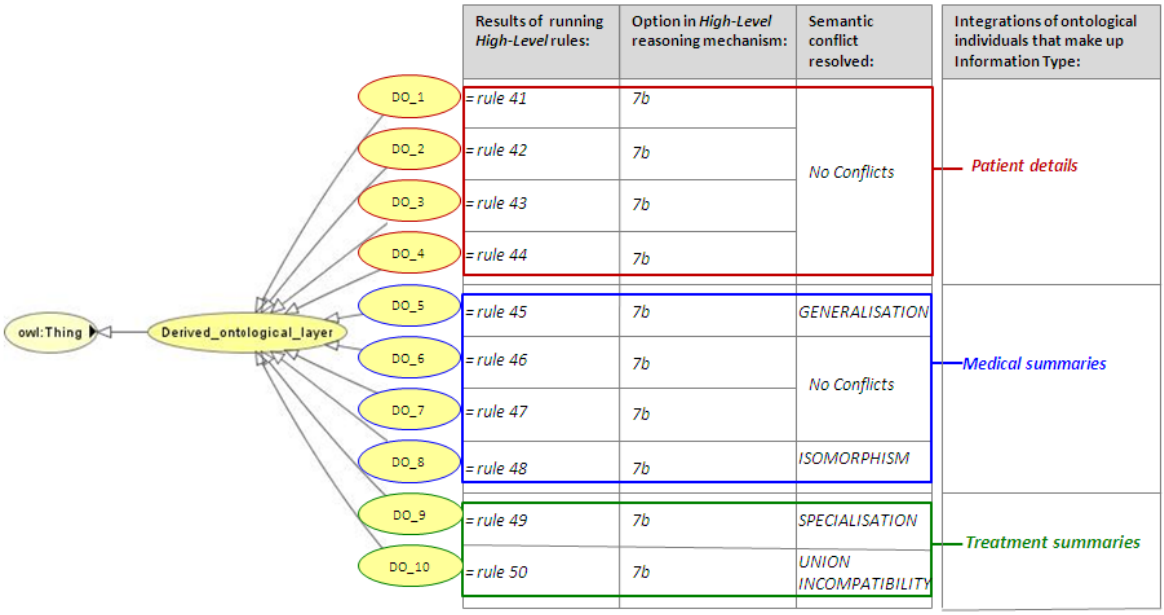

Figure 5.41 *Example of derived ontologies {DO_g | g =1 ...10}*

When creating information type *Medical summaries* at this stage, we observe from Figure 5.41 that Generalisation and Isomorphism semantic conflicts will be resolved. These two conflicts are found in ontological individuals from target ontologies TO_5 and TO_8 respectively, which is listed in Tables 5.11 and 5.13 from section 5.3.1.2. We run SWRL *High-level* rule 35 (available in Appendix A16), to resolve the Generalisation and SWRL *High-level* rules 38 (available in Appendix A16) to resolve the Isomorphism. This is how we resolve them at this stage:

- we must integrate, into derived ontologies DO_5 and DO_8, ontological individuals which have Generalisation (in target ontology TO_5) and Isomorphism (target ontology TO_8) semantic conflicts, and which have been classified as 'semantically similar' in step 6;
- we support integration by running *High-level* rules 35 and 38, in order to assert into derived ontologies DO_5 and DO_8 semantically similar ontological individuals from target ontologies TO_5 and TO_8. The outcome of assertion is a link between individuals in ontologies DO_5 and DO_8, which do not exhibit Generalisation and Isomorphism any more.

However, information type *Medical summaries* are not completed after the elimination of Generalisation and Isomorphism semantic conflicts. Figure 5.41 shows that we run extra two SWRL *High-level* rules,

numbered as 46 and 47 (available in Appendix A16), which have the same role/purpose as SWRL *High-level* rules 41-44, essential for creating information type *Medical summaries*. We need these two rules for the assertion of semantically similar individuals in target ontologies TO_5 and TO_6, as listed in Tables 5.11 and 5.13 from section 5.3.1.2, into derived ontologies DO_6 and DO_7. The assertion happens by running *High-level* rules 46 and 47. The outcome of assertion is a link between ontological individuals in derived ontologies DO_6 and DO_7, which are now semantically equivalent. Details on the exact integrations for resolving Generalisation and Isomorphism for creating *Medical summaries* are given in section 5.3.2.2.

When creating information type *Treatment summaries*, Specialisation and Union Incompatibility semantic conflicts will be resolved. These two conflicts are found in ontological individuals from target ontologies TO_9 and TO_10 respectively, which are listed in Tables 5.15 and 5.16 from section 5.3.1.2. We run SWRL *High-level* rule 49 (available in Appendix A16), to resolve the Specialisation and SWRL *High-level* rules 50 (available in Appendix A16) to resolve the Union Incompatibility. This is how we resolve them at this stage:

- we must integrate, into derived ontologies DO_9 and DO_10, ontological individuals which have Specialisation (in target ontology TO_9) and Union Incompatibility (target ontology TO_10) semantic conflicts, and which have been classified as 'semantically similar' in step 6;
- we support integration by running *High-level* rules 49 and 50, in order to assert into derived ontologies DO_9 and DO_10 semantically similar ontological individuals from target ontologies TO_9 and TO_10. The outcome of assertion is a link between individuals in ontologies DO_9 and DO_10, which do not exhibit Specialisation and Union Incompatibility any more.

Details on the exact integrations for resolving Specialisation and Union Incompatibility for creating *Treatment summaries* are given in section 5.3.2.3.

It is important to note that all *High-Level* rules are run through the SWRL-plug-in in Protégé 3.4, using the Jess engine. All classes are created through Protégé 3.4. Screen shots of the inference (e.g. ontological individuals stored in DO_1 - DO_10), as a result of running *High-Level* rules 41-50 can be found Appendix A.17. (Note: Appendix A.17 is stored on the CD-ROM).

5.3.2.1 Integrating Semantically Similar Data in Patient Details

High-Level rule 41 in Appendix A.16 asserts and transfers the semantically similar ontological individuals JANE, JANE, JANE and JANE from the target ontologies TO_1, TO_2, TO_3 and TO_4 into DO_1. A link is established between ALL the ontological individuals.

Table 5.17 *The results of running the High-Level rule 41*

1. JANE
2. JANE

After running *High-Level* rule 41, 2 ontological individuals listed in Table 5.17 are transferred into DO_1 class, thus creating a 'semantic correspondence' between them. (The assertion of ontological individuals into DO_1 eliminates the duplicate ontological individuals in TO_1, TO_2, TO_3 and TO_4). The

inference created as a result of running the *High-Level* rule 41 is graphically shown in Figure 5.42. We use the same rational is used as mentioned in Figure 4.14 in chapter 4, i.e. assertion of ontological individuals is shown as a black broken line between ontological individuals and ontological classes.

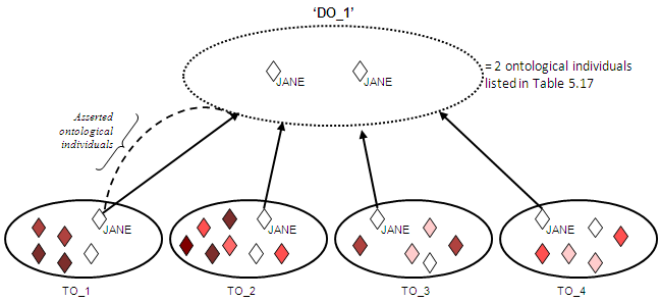

Figure 5.42 *Transferring ontological individuals from target ontologies 'TO_1', 'TO_2', 'TO_3' and 'TO_4' into the DO_1 derived ontology*

High-Level rule 42 in Appendix A.16 asserts and transfers the semantically similar ontological individuals `FLEE`, `FLEE`, `FLEE` and `FLEE` from the target ontologies `TO_1`, `TO_2`, `TO_3` and `TO_4` into `DO_2`. A link is established between all four of the ontological individuals.

Table 5.18 *The results of running the High-Level rule 42*

1. FLEE
2. FLEE

After running *High-Level* rule 42, 2 ontological individuals in listed in Table 5.18 are transferred into `DO_2` class, thus creating a 'semantic correspondence' between them. (The assertion of ontological individuals into `DO_2` eliminates duplicate ontological individuals in `TO_1`, `TO_2`, `TO_3` and `TO_4`). The inference created as a result of running the *High-Level* rule 42 is graphically shown in Figure 5.43.

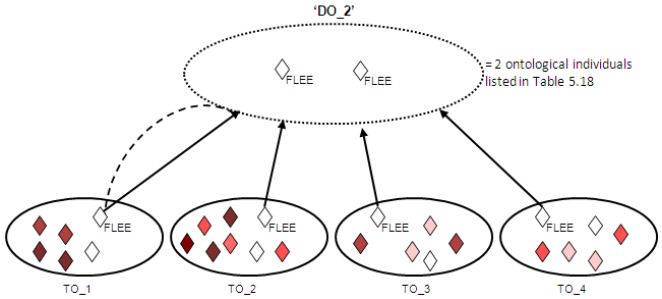

Figure 5.43 *Transferring ontological individuals from target ontologies 'TO_1', 'TO_2', 'TO_3' and 'TO_4' into the DO_2 derived ontology*

High-Level rule 43 in Appendix A.16 asserts and transfers the semantically similar ontological individuals `FEMALE` from the target ontologies `TO_1`, `TO_2`, `TO_3` and `TO_4` into `DO_3`. A link is established between both ontological individuals `FEMALE`.

Table 5.19 *The results of running the High-Level rule 43*

1.	FEMALE
2.	Female
3.	FEMALE
4.	Female

After running *High-Level* rule 43, 4 ontological individuals listed in Table 5.19 are transferred into `DO_3` class, thus creating a 'semantic correspondence' between them. The inference created as a result of running the *High-Level* rule 43 is graphically shown in Figure 5.44.

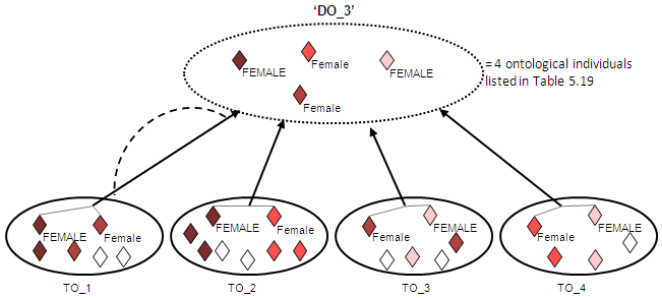

Figure 5.44 *Transferring ontological individuals from target ontologies 'TO_1', 'TO_2', 'TO_3' and 'TO_4' into the DO_3 derived ontology*

High-Level rule 44 in Appendix A.16 asserts and transfers the semantically similar ontological individuals `JULY_04_1970` from the target ontologies `TO_1`, `TO_2`, `TO_3` and `TO_4` into `DO_4`. A link is established between all four of the ontological individuals.

Table 5.20 *The results of running the High-Level rule 44*

1.	JULY_04_1970
2.	JULY_04_1970
3.	JULY_4_1970
4.	July_04_1970

After running *High-Level* rule 44, 4 ontological individuals listed in Table 5.20 are transferred into `DO_4` class, thus creating a 'semantic correspondence' between them. The inference created as a result of running the *High-Level* rule 44 is graphically shown in Figure 5.45.

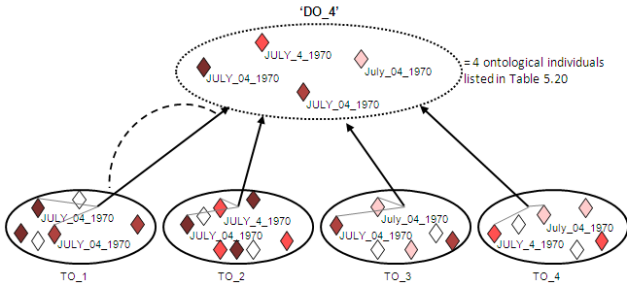

Figure 5.45 *Transferring ontological individuals from target ontologies 'TO_1', 'TO_2', 'TO_3' and 'TO_4' into the DO_4 derived ontology*

5.3.2.2 Integrating Semantically Similar Data in Medical Summaries

High-Level rule 45 in Appendix A.16 asserts and transfers the semantically similar ontological individuals: `Mrs_Flee_complains_xxx`, `Mrs_Flee_has_come_xxx` and `Mrs_Flee_complains_of_xxx` (where 'xxx' denotes the whole name of the ontological individuals) from the target ontologies `TO_5` and `TO_6` into `DO_5`. A link is established between all the ontological individuals.

Table 5.21 *The results of running the High-Level rule 45*

1.	Mrs_Flee_complains_of_severe_pain_in_left_ankle_Minor_swelling_evident_and_xrays_taken_admitted_as_overnight_stay_and_found_to_have_acute_COPD_exacerbation
2.	Mrs_Flee_has_come_into_the_clinic_for_a_blood_test_complains_of_fatigue
3.	Mrs_Flee_complains_of_shortness_of_breath

After running *High-Level* rule 45, 3 ontological individuals listed in Table 5.21 are transferred into `DO_5` class, thus creating a 'semantic correspondence' between them. The inference created as a result of running the *High-Level* rule 45 is graphically shown in Figure 5.46.

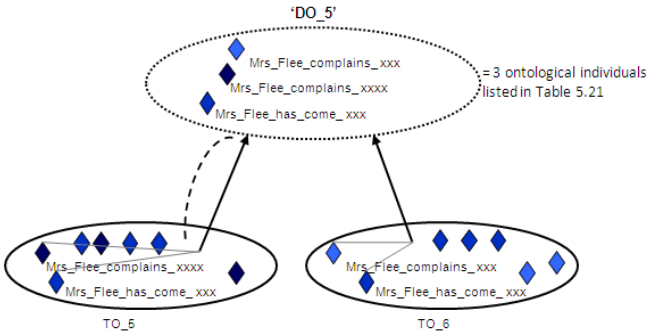

Figure 5.46 *Transferring ontological individuals from target ontologies 'TO_5' and 'TO_6' into the DO_5 derived ontology*

High-Level rule 46 in Appendix A.16 asserts and transfers the semantically similar ontological individuals `no_major_illness_evident`, `none` and `No_MJ` from the target ontologies `TO_5` and `TO_6` into `DO_6`. A link is established between ALL the ontological individuals mentioned above.

Table 5.22 *The results of running the High-Level rule 46*

1.	no_major_illness_evident
2.	none
3.	No_MJ

After running *High-Level* rule 46, 3 ontological individuals listed in Table 5.22 are transferred into `DO_6` class, thus creating a 'semantic correspondence' between them. The inference created as a result of running the *High-Level* rule 46 is graphically shown in Figure 5.47.

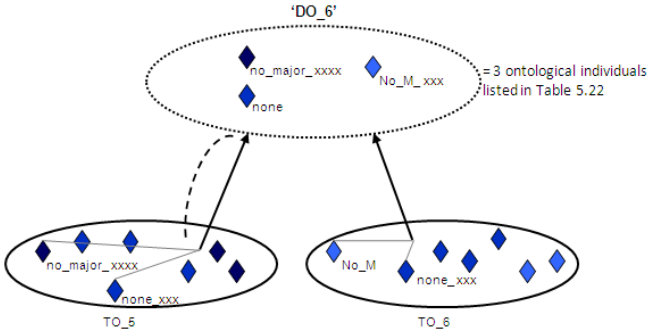

Figure 5.47 *Transferring ontological individuals from target ontologies 'TO_5' and 'TO_6' into the DO_6 derived ontology*

High-Level rule 47 in Appendix A.16 asserts and transfers the semantically similar ontological individuals `no_chronic_disease_evident`, `none_found` and `no_cd_found` from the target ontologies `TO_5` and `TO_6` into `DO_7`. A link is established between all the ontological individuals.

Table 5.23 *The results of running the High-Level rule 47*

1.	no_chronic_disease_evident
2.	none_found
3.	no_cd_found

After running *High-Level* rule 47, 3 ontological individuals listed in Table 5.23 are transferred into `DO_7` class, thus creating a 'semantic correspondence' between them. The inference created as a result of running the *High-Level* rule 47 is graphically shown in Figure 5.48.

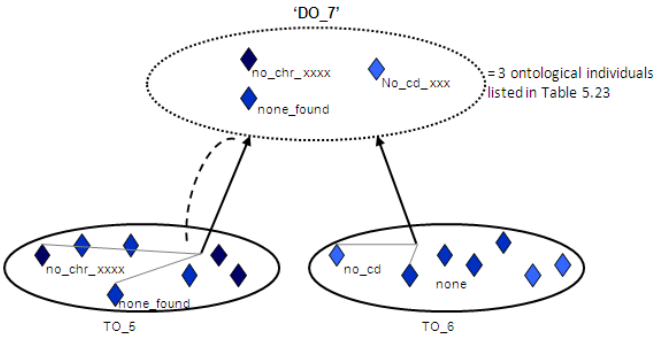

Figure 5.48 *Transferring ontological individuals from target ontologies 'TO_5' and 'TO_6' into the DO_7 derived ontology*

High-Level rule 48 in Appendix A.16 asserts and transfers the semantically similar ontological individuals `LT256_Sm_xxx`, `LT256_Ce_xxx`, `LT256_No_xxx`, `LT256_16_xxx`, `LT123_Pa_xxx`, `LT123_Bl_xxx`, `LT123_an_xxx`, `LT123_16_xxx`, `LL456_xxx`, `LL456_Xr_xxx`, `LL456_fi_xxx` and `LL456_28_xxx` (where 'xxx' denotes the whole name of the ontological individuals) from the target ontology `TO_7` into `DO_8`. A link is established between all the ontological individuals.

Table 5.24 *The results of running the High-Level rule 48*

1.	LT256_Smear_test	7.	LT123_anaemia_level_46
2.	LT256_Cervical_Type_3	8.	LT123_16-02-08
3.	LT256_Normal	9.	LL456_Radiation
4.	LT256_16-01-08	10.	LL456_Xray
5.	LT123_Pathology	11.	LL456_fileID_wavelength908
6.	LT123_Blood_test_Type_4123	12.	LL456_28-04-09

After running *High-Level* rule 48, 12 ontological individuals listed in Table 5.24 are transferred into DO_8 class, thus creating a 'semantic correspondence' between them. The inference created as a result of running the *High-Level* rule 48 is graphically shown in Figure 5.49.

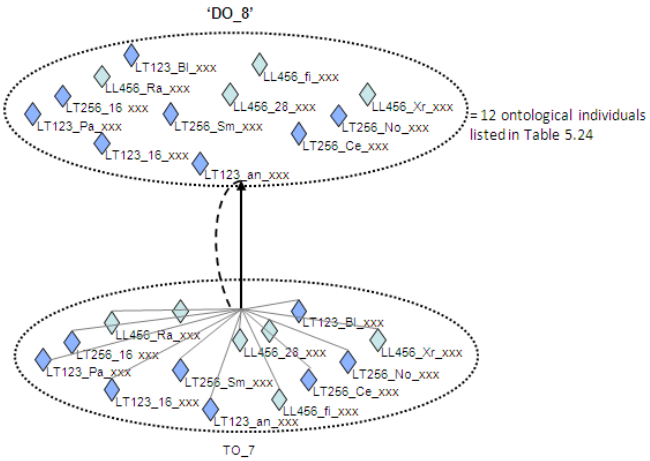

Figure 5.49 *Transferring ontological individuals from target ontologies 'TO_7' into the DO_8 target ontology*

5.3.2.2.1 Resolving Generalisation Semantic Conflict

Having identified, while looking at the possible types of semantic conflicts in relational schemas in section 5.1.2, that there are Generalisation semantic conflicts in attribute names, we discovered that:

- data stored in the MEDICAL_SUMMARY from the PATIENT table in the *Hospital_data_rep* database, and
- data stored in the CURRENT_MEDICAL_SUMMARIES and PREVIOUS_MEDICAL_SUMMARIES attributes from the patient table in the *Clinic_1_data_rep* database,

exhibit Semantic Subset - *contains* (4) as their degree of similarity. This is because the data stored in CURRENT_MEDICAL_SUMMARIES and PREVIOUS_MEDICAL_SUMMARIES attributes may have been seen by Dr Smith as "parts" of data needed for MEDICAL_SUMMARY, which he would like to add to Mrs Flee's *medical summaries* Dr Smith holds within his *Hospital_data_rep* database. This is an example of Generalisation, which is resolved after the integration of ontological individuals into from target ontologies TO_5 and TO_6 into the DO_5 derived ontology. Specifically, the conflict is resolved by asserting the ontological individuals: Mrs_Flee_complains_of_severe_xxx, Mrs_Flee_has_come_xxx and Mrs_Flee_complains_of_xxx (where 'xxx' denotes the whole name of the ontological individuals) from the target ontologies TO_5 and TO_6, into the derived ontology DO_5, and which will ultimately be merged into the concepts of Go-CID, i.e:

- ontological individuals Mrs_Flee_complains_of_severe_xxx and Mrs_Flee_complains_of_shortness_of_breath will be relocated into the SUMMARIES class of Go-CID, and
- ontological individual: Mrs_Flee_has_come_into_the_clinic_for_a_blood_test_complains_of_fatigue will be relocated into the PREVIOUS_MEDICAL_SUMMARIES class of Go-CID.

5.3.2.2.2 Resolving Isomorphism Semantic Conflict

Having identified, while looking at the possible types of semantic conflicts in relational schemas in section 5.1.2, that there are Isomorphism semantic conflicts concerning attribute names, we discovered that:

- data stored in the LABTEST_ID, PATIENT_NO, LABTEST_TYPE, LABTEST_NAME, LABTEST_RESULTS, REPORT and DATA from the LABTEST table in the *Clinic_1_data_rep* database, and
- data stored in the LABTEST_ID, PATIENT_NO, LABTEST_OVERVIEW, LABTEST_TYPE, LABTEST_NAME, LABTEST_RESULTS, REPORT and DATA from the LABTEST table in the *Clinic_2_data_rep* database,

exhibit Semantic Overlapping (6) as their degree of similarity. This is an example of Isomorphism, which is resolved after the integration of ontological individuals from the target ontology TO_7 into the DO_8 derived ontology. Specifically, the conflict is resolved by asserting ontological individuals LT256_Sm_xxx, LT256_Ce_xxx, LT256_No_xxx, LT256_16_xxx, LT123_Pa_xxx, LT123_Bl_xxx, 'LT123_an_xxx, LT123_16_xxx, LL456_xxx, LL456_Xr_xxx, LL456_fi_xxx, and LL456_28_xxx (where 'xxx' denotes the whole name of the ontological individuals) from the target ontology TO_7 into the derived ontology DO_8, which will ultimately be merged into the concepts of Go-CID, i.e.:

- ontological individuals LT256_Smear_test, LL456_Radiation and LT123_Blood_test_Type_4123 will be relocated into the class LABTEST_TYPE of Go-CID,
- ontological individuals LT256_Cervical_Type_3, LL456_Xray and LT123_Pathology will be relocated into the class LABTEST_NAME of Go-CID,
- ontological individuals LT256_Normal, LT123_anaemia_level_46 and LL456_fileID_wavelength908 will be relocated into the class 'LABTEST_RESULTS' of Go-CID,
- ontological individuals LT123_16-02-08, LT256_16-01-08 and LL456_28-04-09 will be relocated into the LABTEST_DATE class of Go-CID, and
- ontological individuals LT123_16-02-08, LT256_16-01-08 and LL456_28-04-09 will be relocated into the LABTEST_OVERVIEW class of Go-CID.

5.3.2.3 Integrating Semantically Similar Data in Treatment Summaries

High-Level rule 49 in Appendix A.16 asserts and transfers the semantically similar ontological individuals TT1989_12-03-09 and T09851_17-04-09 from the target ontology TO_9 into DO_9. A link is established between both the ontological individuals.

Table 5.25 *The results of running the High-Level rule 49*

1.	TT1989_12-03-09
2.	T09851_17-04-09

After running *High-Level* rule 49, 2 ontological individuals listed in Table 5.25 are transferred into `DO_9` class, thus creating a 'semantic correspondence' between them. The inference created as a result of running the *High-Level* rule 49 is graphically shown in Figure 5.50.

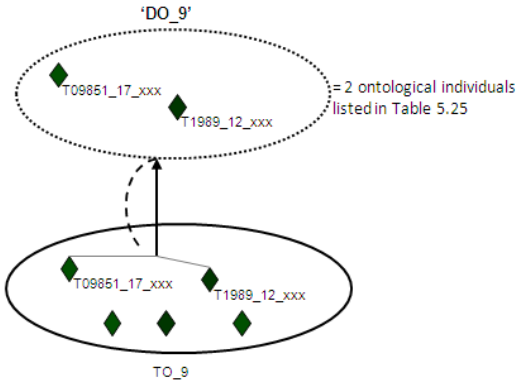

Figure 5.50 *Transferring ontological individuals from target ontologies 'TO_9' into the DO_9 derived ontology*

High-Level rule 50 in Appendix A.16 asserts and transfers the semantically similar ontological individuals `M0031_C_xxx`, `M0031_X_xxx`, `M0031_2_xxx`, `M222p_N_xxx`, `M222p_R_xxx`, `M225i_E_xxx`, `M225i_Em_xxx`, `M222p_1_xxx` and `M225i_1_xxx` (where 'xxx' denotes the whole name of the ontological individuals) from the target ontology `TO_10` into `DO_10`. A link is established between all the ontological individuals.

Table 5.26 *The results of running the High-Level rule 50*

1. M0031_Capzasin	6. M225i_EHOSUXIMIDE
2. M0031_Xhing_Ltd	7. M225i_Emeside
3. M0031_2_tablets_per_day	8. M222p_1_or_2_tablets_to_be_taken_4_times_a_day
4. M222p_NAPROXEN	
5. M222p_Risedronate	9. M225i_1_tablets_to_be_taken_4_times_a_day

After running the *High-Level* rule 50, 9 ontological individuals listed in Table 5.26 are transferred into `DO_10` class, creating a 'semantic correspondence' between them. The inference created as a result of running the *High-Level* rule 50 is graphically shown in Figure 5.51.

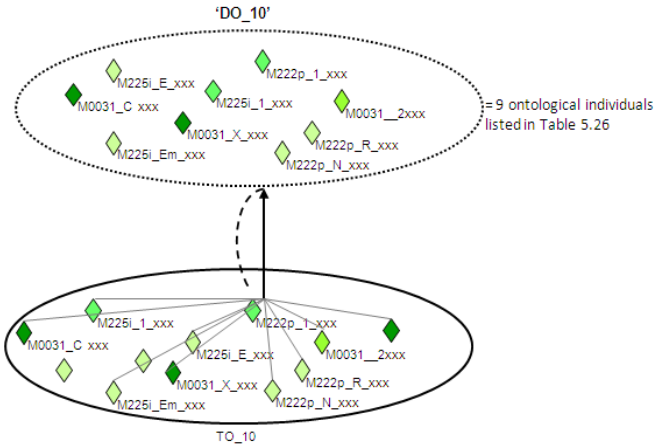

Figure 5.51 *Transferring ontological individuals from target ontologies 'TO_10' into the DO_10 derived ontology*

5.3.2.3.1 Resolving Specialisation Semantic Conflict

Having identified, while looking at the possible types of semantic conflicts in relational schemas in section 5.1.2, that there are Specialisation semantic conflicts concerning attribute names, we discovered that:

- data stored in the `TREATMENT_TYPE`, `TREATMENT_NAME` and `TREATMENT_DATE` from the `TREATMENT` table in the *Hospital_data_rep* database, and
- data stored in the `TREATMENT_OVERVIEW` and `TREATMENT_DATE` attributes from the `TREATMENT` table in the *GP_data_rep* database,

exhibit Semantic Subset – *contained within* (5) as their degree of similarity. This is because the data stored in the `TREATMENT_OVERVIEW` and `TREATMENT_DATE` attributes may have been seen by Dr Smith as "parts" of data which is stored in the `TREATMENT_TYPE` and `TREATMENT_NAME` attributes, which Dr Smith would like to add to Mrs Flee's *treatment summaries* he holds within his *Hospital_data_rep* database. This is an example of Specialisation semantic conflict, which is resolved after the integration of ontological individuals from the target ontology `TO_9` into the `DO_9` derived ontology. Specifically, the conflict is resolved by asserting ontological individuals `TT1989_12-03-09` and `T09851_17-04-09` from the target ontology `TO_9` into derived ontology `DO_9`, and which will ultimately be merged into the concepts of Go-CID, i.e.:

- ontological individual:

`T1989_Patient_is_suffering_from_aches_in_lower_limbs_and_has_minor_swelling_to_ankle_pain_support_through_chronic_pain_recovery_is_suggested` will be relocated into the `TREATMENT_OVERVIEW` class of Go-CID., and

- ontological individual `T09851_COPD_Chronic_pain_recovery` will be relocated into the `TREATMENT_NAME` class of Go-CID.

5.3.2.3.2 Resolving Union Incompatibility Semantic Conflict

Having identified, while looking at the possible types of semantic conflicts in relational schemas in section 5.1.2, that there are Union Incompatibility semantic conflicts concerning attribute names, we discovered that:

- data stored in the `MEDICINE_NUM`, `MEDICINE_NAME`, `VENDOR`, and `MNF_DESC` from the `MEDICATION` table in *GP_data_rep* database, and
- data stored in the `MEDICINE_NO`, `MEDICINE_NAME`, `VENDOR`, and `MNF_ADDRESS` from the `MEDICATION` table in the *Hospital_data_rep* database,

exhibit Semantic Overlapping (6) as their degree of similarity. This is an example of Union Incompatibility, which is resolved after the integration of ontological individuals from the target ontology `TO_10` into the `DO_10` derived ontology. Specifically, the Union Incompatibility conflict is resolved by asserting ontological individuals `M0031_C_xxx`, `M0031_X_xxx`, `M0031_2_xxx`, `M222p_N_xxx`, `M222p_R_xxx`, `M225i_E_xxx`, `M225i_Em_xxx`, `M222p_1_xxx` and `M225i_1_xxx` (where 'xxx' denotes the whole name of the ontological individuals) from the target ontology `TO_10` into the derived ontology `DO_10`, and which will ultimately be merged into the concepts of Go-CID, i.e.:

- ontological individuals M0031_Capzasin, M225i_EHOSUXIMIDE, and M222p_NAPROXEN will be relocated into the MEDICINE_NAME class of Go-CID,
- ontological individuals M0031_Xhing_Ltd, M222p_Risedronate, and M225i_Emeside will be relocated into the MEDICINE_DETAILS class of Go-CID, and
- ontological individuals:

M0031_2_tablets_per_day,
M222p_1_or_2_tablets_to_be_taken_4_times_a_day, and
M225i_1_tablets_to_be_taken_4_times_a_day will be relocated into the DOSAGE_AMOUNT class of Go-CID.

5.3.3 Step 8: Merging Derived Ontologies

In step 8, we finally create the final Go-CID ontological layer by merging ontological individuals from lower ontological layers. Ontological individuals stored in subclasses of Go-CID will (a) obviously exhibit no semantic conflicts and (b) have been merged because they must become ontological individuals, which correspond to 'real world' concepts' originally modelled in and placed within data repositories Rep_i (see chapter 4, section 4.3.4.4). In Figures 5.52 – 5.54 we illustrate exactly how we merge concepts in order to create a final version of information types *Patient details*, *Medical summaries* and *Treatment summaries*. To summarise:

- we have three figures because we allocate each figure to a separate information type *Patient details*, *Medical summaries* and *Treatment summaries*,
- we perform "merge" of ontological individuals from lower ontological layers into Go-CID when creating final formats of the information types requested by Dr Smith (i.e. in order to create a Health Summary for Mrs Flee),
- we do NOT resolve semantic conflicts because they have been resolved through previous steps, and
- we run 28 SWRL *Post-High-Level* rules, as a part of *Post-High-Level* reasoning mechanism (see chapter 4, section 4.3.4.4), which conduct this merge by relocating ontological individuals from lower ontological layers into subclasses of Go-CID.

Therefore, ontological individuals stored in subclasses of Go-CID are not always the result of merge of individuals from DO_g. In Figures 5.52 – 5.54 we shade (i) in grey, rules which perform the merge upon ontological individuals from LO_j (*Post-High-Level* rules 55 to 61 in Figure 5.52) and (ii) in dark grey, *Post-High-Level* rules which perform the merge upon ontological individuals from TO_k (*Post-High-Level* rules 69-71 in Figure 5.53 and *Post-High-Level* rules 76-79 in Figure 5.54).

In Figure 5.52 through *Post-High-Level* rules 51-54 we merge semantically equivalent ontological individuals from derived ontologies DO_1, DO_2, DO_3 and DO_4, listed in Tables 5.17 - 5.20 from section 5.3.2.1, and re-locate them into subclasses FIRST_NAME, LAST_NAME, SEX and DOB of Go-CID. *Post-High-Level* rules 55-61 merge semantically equivalent ontological individuals from *LO_gp* (they are listed in the Appendix A.3 stored on CD-ROM) and relocate them into subclasses ADDRESS, REGION, NEXT_OF_KIN, EMERGENCY_CONTACT, NO_OF_CHILDREN, BMI and HEIGHT of Go-CID. These eleven *Post-High-Level* rules are available in Appendix A.18. Their number (eleven in this particular case), depends on the numbers of concepts needed when creating final formats of information

type *Patient details* requested by Dr Smith in order to create a health summary for Mrs Flee. Details on the exact merges for creating *Patient details* in section 5.3.3.1.

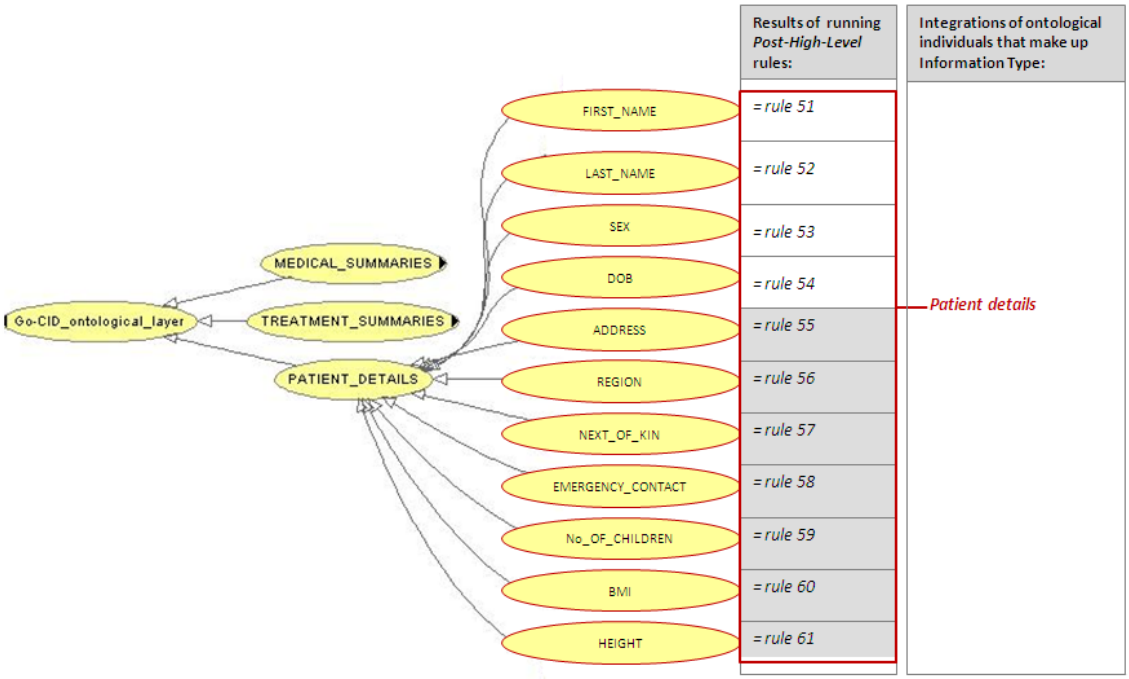

Figure 5.52 *Example of the 'PATIENT_DETAILS' subclasses in Go-CID*

In Figure 5.53 through *Post-High-Level* rules 62-68 we merge semantically equivalent ontological individuals from derived ontologies `DO_5`, `DO_6`, `DO_7` and `DO_8`, listed in Tables 5.21 - 5.24 from section 5.3.2.2, and re-locate them into subclasses `SUMMARIES`, `MAJOR_ILLNESS`, `CHRONIC_DISEASE`, `LABTEST_TYPE`, `LABTEST_NAME`, `LABTEST_RESULTS` and `LABTEST_DATE` of Go-CID. *Post-High-Level* rules 69-71 merge semantically equivalent ontological individuals from target ontologies `TO_5` and `TO_7` (they are listed Tables 5.11 and 5.13 from section 5.3.1.2) and relocate them into subclasses `PREVIOUS_MEDICAL_SUMMARIES`, `LABTEST_OVERVIEW`, and `LABTEST_DATA` of Go-CID. These three *Post-High-Level* rules are available in Appendix A.18. Their number (three in this particular case), depends on the numbers of concepts needed when creating final formats of information type *Medical summaries* requested by Dr Smith in order to create a health summary for Mrs Flee. Details on the exact merges for creating *Medical summaries* in section 5.3.3.2.

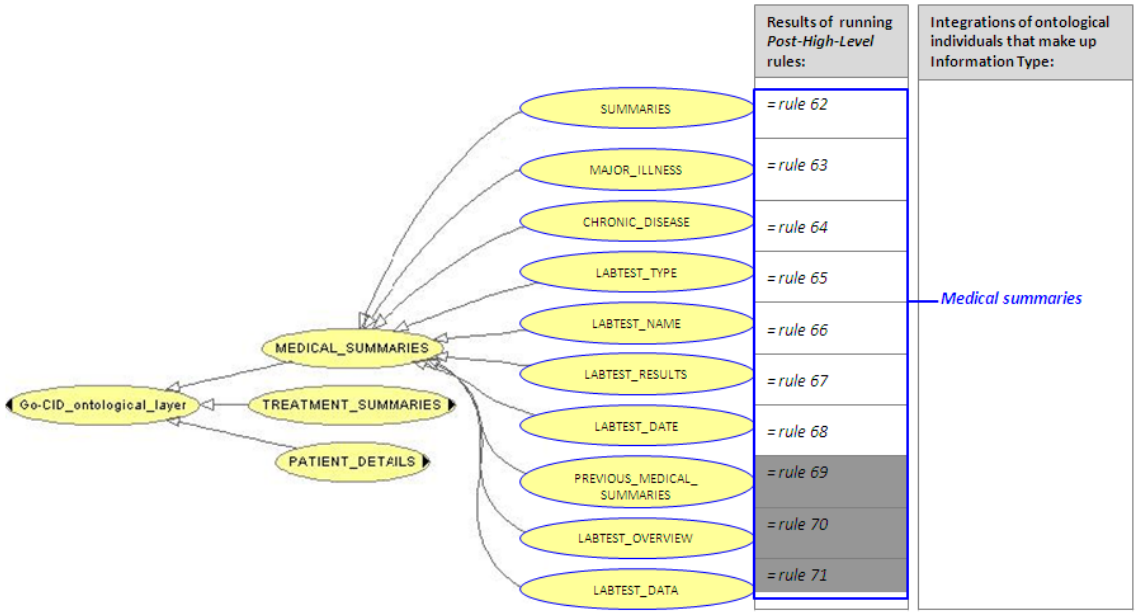

Figure 5.53 *Example of the 'MEDICAL_SUMMARIES' subclasses in Go-CID*

In Figure 5.54 through *Post-High-Level* rules 72-75 we merge semantically equivalent ontological individuals from derived ontologies `DO_9` and `DO_10`, listed in Tables 5.25 and 5.26 from section 5.3.2.3, and re-locate them into subclasses `TREATMENT_DATE`, `MEDICINE_NAME`, `VENDOR` and `DOSAGE_AMOUNT` of Go-CID. *Post-High-Level* rules 76-79 merge semantically equivalent ontological individuals from target ontologies `TO_8` and `TO_10` (they are listed Tables 5.14 and 5.16 from section 5.3.1.3) and relocate them into subclasses `MEDICINE_NUMBER`, `TREATMENT_OVERVIEW`, `TREATMENT_NAME` and `MEDICATION_DETAILS` of Go-CID. These four *Post-High-Level* rules are available in Appendix A.18. Their number (four in this particular case), depends on the numbers of concepts needed when creating final formats of information type *Treatment summaries* requested by Dr Smith in order to create a health summary for Mrs Flee. Details on the exact merges for creating *Treatment summaries* in section 5.3.3.3.

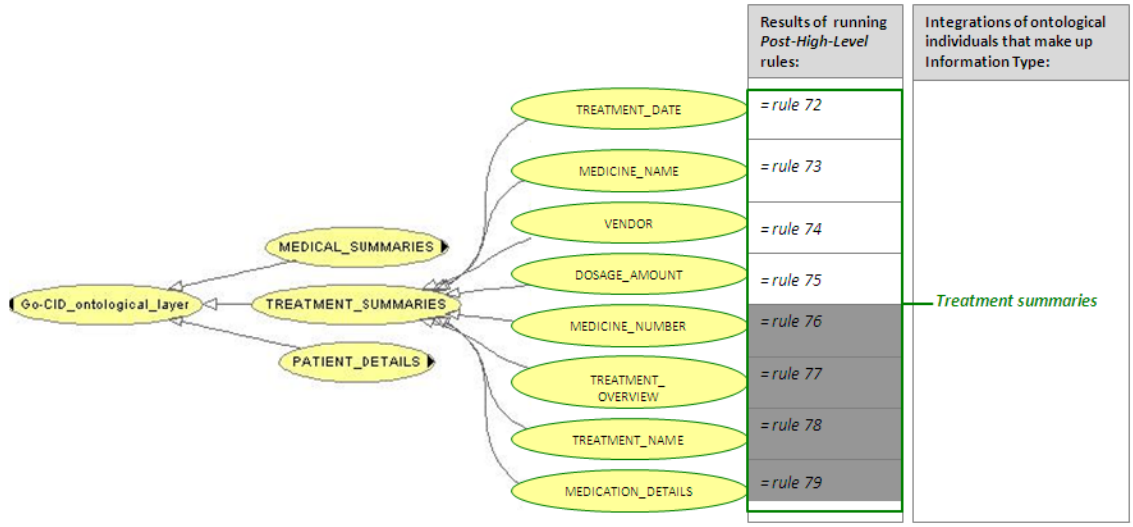

Figure 5.54 *Example of the 'TREATMENT_SUMMARIES' subclasses in Go-CID*

It is important to note that all *Post-High-Level* for ontology merges are run through the SWRL-plug-in in Protégé 3.4, using the Jess engine. All classes in Go-CID are created through Protégé 3.4. Screen shots of the inference (e.g. ontological individuals stored in subclasses FIRST_NAME, LAST_NAME, SEX, DOB, ADDRESS, REGION, NEXT_OF_KIN, EMERGENCY_CONTACT, NO_OF_CHILDREN, BMI, HEIGHT, SUMMARIES, MAJOR_ILLNESS, CHRONIC_DISEASE, LABTEST_TYPE, LABTEST_NAME, LABTEST_RESULTS, LABTEST_DATE, PREVIOUS_MEDICAL_SUMMARIES, LABTEST_OVERVIEW, and LABTEST_DATA TREATMENT_DATE, MEDICINE_NAME, VENDOR and DOSAGE_AMOUNT, MEDICINE_NUMBER, TREATMENT_OVER, TREATMENT_NAME and MEDICATION_DETAILS in Go-CID) as a result of running *Post-High-Level* rules can be found Appendix A.19. (Note Appendix A.19 is stored on the CD-ROM).

5.3.3.1 Merging Semantically Equivalent Data in Patient Details

Post-High-Level rules 51, 52, 53 and 54 in Appendix A.18 relocates the semantically equivalent ontological individuals JANE, FLEE, FEMALE and JULY_04_1970 from the derived ontologies DO_1, DO_2, DO_3 and DO_4 into FIRST_NAME, LAST_NAME, SEX and DOB respectively.

Table 5.27 *The results of running the Post-High-Level rules 51, 52, 53 and 54*

1.	JANE
2.	FLEE
3.	FEMALE
4.	JULY_04_1970

After running *Post-High-Level* rules 51, 52, 53 and 54, 4 ontological individuals listed in Table 5.27 are relocated into FIRST_NAME, LAST_NAME, SEX and DOB subclasses of Go-CID. The inference created as a result of running *Post-High-Level* rules 51, 52, 53 and 54 is graphically shown in Figure 5.55. We use the same rational as mentioned in Figure 4.15 in chapter 4, i.e. relocation of ontological individuals is shown as a black broken line between ontological individuals and ontological classes.

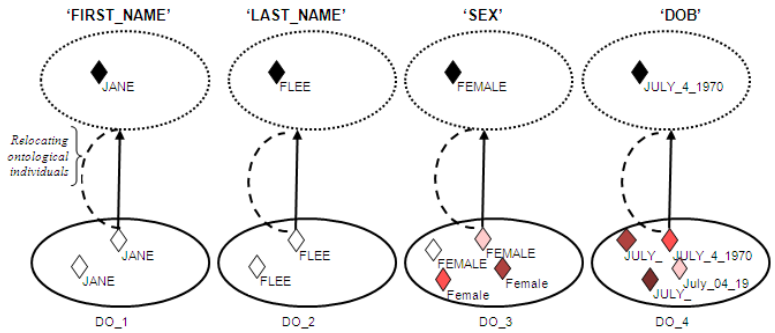

Figure 5.55 *Relocating ontological individuals from derived ontologies 'DO_1', 'DO_2', 'DO_3' and 'DO_4' into the Go-CID*

Post-High-Level rules 55, 56, 57, 58, 59, 60 and 61 in Appendix A.18 relocates the semantically equivalent ontological individuals ADD_167_BOULEVARD_RD_W1W_5TU, LONDON, NEMANJA_FLEE, TEL_07965896456, CHILDREN_0, NORMAL and H_5_feet_8_inches from

the LO_gp_patient_instances class in local ontology *LO_gp* into ADDRESS, REGION, NEXT_OF_KIN, EMERGENCY_CONTACT, NO_OF_CHILDREN, BMI and HEIGHT respectively.

Table 5.28*The results of running the Post-High-Level rules 55, 56, 57, 58, 59, 60 and 61*

1. ADD_167_BOULEVARD_RD_W1W_5TU	5. CHILDREN_0
2. LONDON	6. NORMAL
3. NEMANJA_FLEE	7. H_5_feet_8_inches
4. TEL_07965896456	

After running *Post-High-* rules 55, 56, 57, 58, 59, 60 and 61, 7 ontological individuals listed in Table 5.28 are relocated into ADDRESS, REGION, NEXT_OF_KIN, EMERGENCY_CONTACT, NO_OF_CHILDREN, BMI and HEIGHT subclasses of Go-CID. The inference created as a result of running *Post-High-Level* rules 55, 56, 57, 58, 59, 60 and 61 is graphically shown in Figure 5.56.

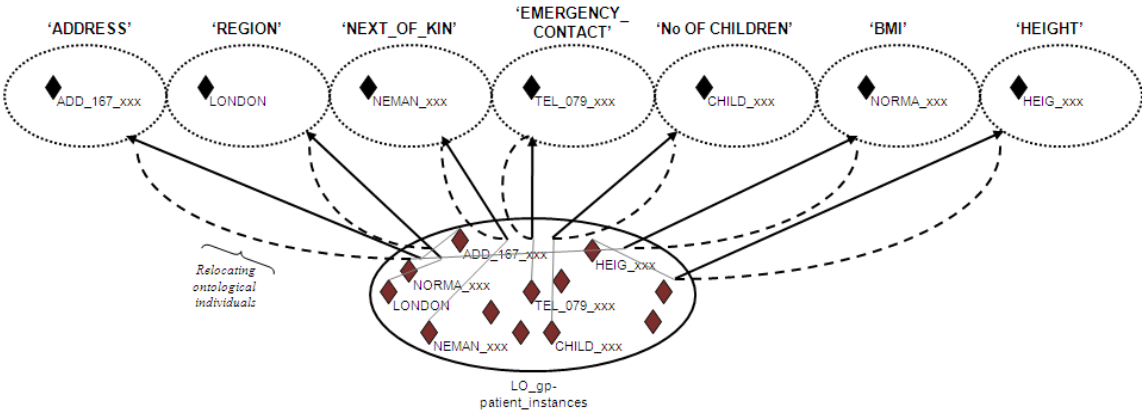

Figure 5.56 *Relocating ontological individuals from 'LO_gp-patient_instances' in the LO_gp into the Go-CID*

Note: in Figure 5.56 we skip the Target and Derived ontological layers because existing ontological individuals stored in LO_gp-patient_instances subclass of local ontology *LO_gp* already share a semantic correspondence (see section chapter 4, section 4.4.3.5).

5.3.3.2 Merging Semantically Equivalent Data in Medical Summaries

Post-High-Level rules 62, 63, 64, 65, 66, 67 and 68 in Appendix A.18 relocates the semantically equivalent ontological individuals LT256_Sm_xxx, LT256_Ce_xxx, LT256_No_xxx, LT256_16_xxx, LT123_Pa_xxx, LT123_Bl_xxx, LT123_an_xxx, LT123_16_xxx, LL456_xxx, LL456_Xr_xxx, LL456_fi_xxx and LL456_28_xxx (where 'xxx' denotes the whole name of the ontological individuals) from the derived ontologies DO_5, DO_6, DO_7 and DO_8 into SUMMARIES, MAJOR_ILLNESS, CHRONIC_DISEASE, LABTEST_TYPE, LABTEST_NAME, LABTEST_RESULTS and LABTEST_DATE respectively.

Table 5.29 *The results of running the Post-High-Level rules 62, 63, 64, 65, 66, 67 and 68*

1. Mrs_Flee_complains_of_severe_pain_in_left_ankle_Minor_swelling_evident_and_xrays_taken_admitted_as_overnight_stay_and_found_to_have_acute_COPD_exacerbation	8. LT256_Smear_test
2. Mrs_Flee_has_come_into_the_clinic_for_a_blood_test_complains_of_fatigue	9. LT123_Pathology
3. Mrs_Flee_complains_of_shortness_of_breath'	10. LT256_Cervical_Type_3
4. none_major_illness_evident	11. LL456_Xray
5. none_found	12. LT256_Normal
6. LT123_Blood_test_Type_4123	13. LT123_anaemia_level_46
7. LL456_Radiation	14. LL456_fileID_wavelength908
	15. LT256_16-01-08
	16. LT123_16-02-08
	17. LL456_28-04-09

After running *Post-High-Level* rules 62, 63, 64, 65, 66, 67 and 68, 17 ontological individuals listed in Table 5.29 are relocated into `SUMMARIES`, `MAJOR_ILLNESS`, `CHRONIC_DISEASE`, `LABTEST_TYPE`, `LABTEST_NAME`, `LABTEST_RESULTS` and `LABTEST_DATE` subclasses of Go-CID. The inference created as a result of running *Post-High-Level* rules 62, 63, 64, 65, 66, 67and 68 is graphically shown in Figure 5.57.

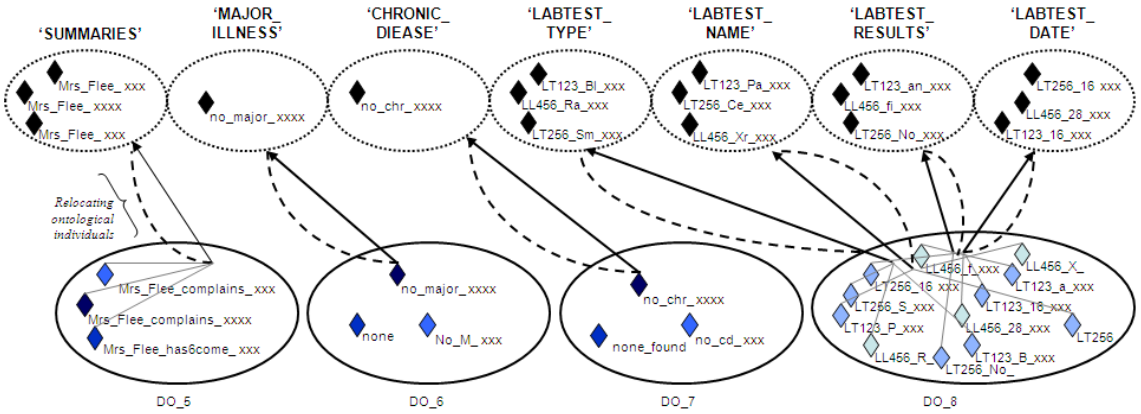

Figure 5.57 *Relocating ontological individuals from derived ontologies 'DO_5', 'DO_6', 'DO_7' and 'DO_8' into the Go-CID*

Post-High-Level rules 69, 70 and 71 in Appendix A.18 relocates the semantically equivalent ontological individuals `Mrs_Flee_has_a_xxx`, `LL456_Used_xxx` and `LL456_data_xxx` (where 'xxx' denotes the whole name of the ontological individuals) from the target ontologies `TO_5` and `TO_7` into `PREVIOUS_MEDICAL_SUMMARIES`, `LABTEST_OVERVIEW` and `LABTEST_DATA` respectively.

Table 5.30 *The results of running the Post-High-Level rules 69, 70 and 71*

1. Mrs_Flee_has_a_regular_cervical_smear_test_results_appear_nor
2. LL456_Used_to_identify_lung_diseases
3. LL456_data_aa2

The result of running *Post-High-Level* rules 69, 70 and 71, 3 ontological individuals listed in Table 5.30 are relocated into `PREVIOUS_MEDICAL_SUMMARIES`, `LABTEST_OVERVIEW` and `LABTEST_DATA` subclasses of Go-CID. The inference created as a result of running the *Post-High-Level* rules 69, 70 and 71 is graphically shown in Figure 5.58. Note: in Figure 5.58 we skip the Derived ontological layer

because existing ontological individuals stored in TO_5 and TO_7 already share a semantic correspondence (see section chapter 4, section 4.4.3.5).

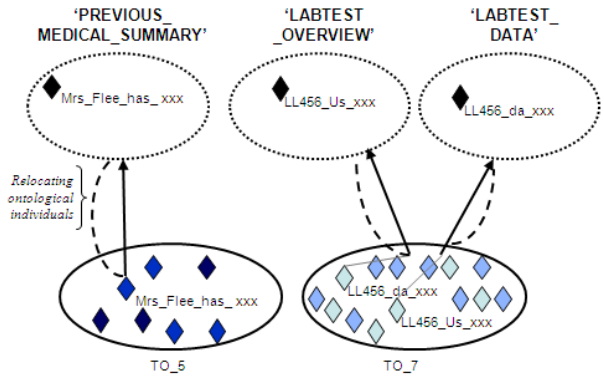

Figure 5.58 *Relocating ontological individuals from 'TO_5' and 'TO_7' target ontologies into the Go-CID*

5.3.3.3 Merging Semantically Equivalent Data in Treatment Summaries

Post-High-Level rules 72, 73, 74 and 75 in Appendix A.18 relocates the semantically equivalent ontological individuals M0031_C_xxx, M0031_X_xxx, M0031_2_xxx, M222p_N_xxx, M222p_R_xxx, M225i_E_xxx, M225i_Em_xxx, M222p_1_xxx and M225i_1_xxx (where 'xxx' denotes the whole name of the ontological individuals) from the derived ontologies DO_9 and DO_10 into TREATMENT_DATE, MEDICINE_NAME, VENDOR and DOSAGE_AMOUNT respectively.

Table 5.31 *The results of running the Post-High-Level rules 72, 73, 74 and 75*

1. TT1989_12-03-09	7. M222p_Risedronate
2. T09851_17-04-09	8. M225i_Emeside
3. M0031_Capzasin	9. M222p_1_or_2_tablets_to_be_taken_4_times_a_day
4. M222p_NAPROXEN	10. M225i_1_tablets_to_be_taken_4_times_a_da
5. M225i_EHOSUXIMIDE	11. M0031_2_tablets_per_day
6. M0031_Xhing_Ltd	

After running *Post-High-Level* rules 72, 73, 74 and 75, 11 ontological individuals listed in Table 5.31 are transferred into TREATMENT_DATE, MEDICINE_NAME, VENDOR and DOSAGE_AMOUNT subclasses of Go-CID. The inference created as a result of running *Post-High-Level Level* rules 69, 70 and 71 is graphically shown in Figure 5.59.

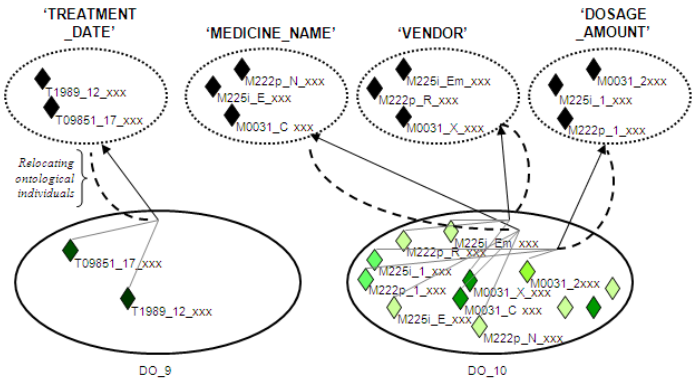

Figure 5.59 *Relocating ontological individuals from derived ontologies 'DO_9' and 'DO_10' into the Go-CID.*

Post-High-Level rules 76, 77, 78 and 79 in Appendix A.18 relocates the semantically equivalent ontological individuals `M0031`, `M222p`, `M225i`, `T1989_Patient_xxx`, `T09851_COPD_xxx`, `T09851_COPD_ex_xxx`, `M222p_An_xxx`, `M225i_So_xxx` and `M0031_Chin_xxx` (where 'xxx' denotes the whole name of the ontological individuals) from the target ontologies `TO_8`, `TO_9` and `TO_10` into `MEDICINE_NUMBER`, `TREATMENT_OVER`, `TREATMENT_NAME` and `MEDICATION_DETAILS` respectively.

Table 5.32 *The results of running the Post-High-Level rules 76, 77, 78 and 79*

1. M0031	3. M225i
2. M222p	6. T09851_COPD_exacerbation
4. T1989_Patient_is_suffering_from_aches_in_lower_limbs_and_has_minor_swelling_to_ankle_pain_support_through_chronic_pain_recovery_is_suggested	7. M222p_Andheri_east_India
5. T09851_COPD_Chronic_pain_recovery	8. M225i_South_coast_Canada
	9. M0031_China_pharmaceuticals

After running the *Post-High-Level* rules 76, 77, 78 and 79, 9 ontological individuals listed in Table 5.32 are relocated into `MEDICINE_NUMBER`, `TREATMENT_OVER`, `TREATMENT_NAME` and `MEDICATION_DETAILS` subclasses of Go-CID. The inference created as a result of running *Post-High-Level* rules 76, 77, 78 and 79 is graphically shown in Figure 5.60. Note: in Figure 5.60 we skip the Derived ontological layer because existing ontological individuals stored in `TO_8`, `TO_9` and `TO_10` already share a semantic correspondence (see section chapter 4, section 4.4.3.5).

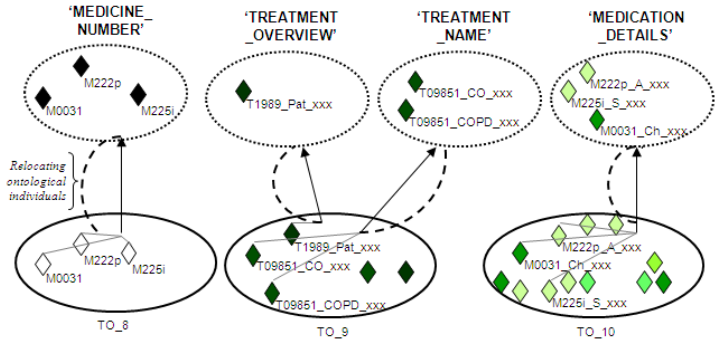

Figure 5.60 *Relocating ontological individuals from 'TO_8', TO_9' and 'TO_10' target ontologies into the Go-CID.*

5.4 Example of Software Application built upon Ontological Layering

Figure 5.61 illustrates the architectural model for software application built upon ontological layering, which takes Dr Smith's inputs and displays results of retrievals based on his request. The application model in Figure 5.61 distinguishes between a front-end which manages Dr Smith's involvements through the application GUI (i.e. *USER_INPUTS/OUTPUTS*), and a data centric back end (i.e. *ONTOLOGICAL LAYERS*), which manages local ontologies LO_j, target ontologies TO_k, derived ontologies DO_g and Go-CID. The connection between the front-end and back-end consists of a set of application elements named *PRIMARY APPLICATION LAYER*, *CORE COMPUTATIONS*, *REASONING ENGINE* and *OWL API*.

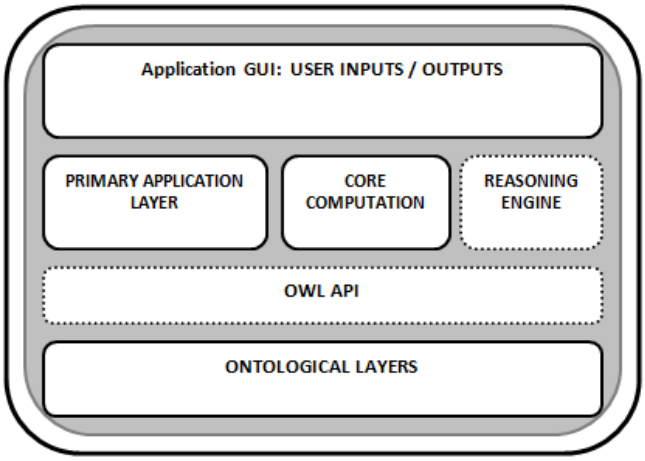

Figure 5.61 *Architectural elements of the software application built upon ontological layering*

The *PRIMARY APPLICATION LAYER* in Figure 5.61, controls the management of the whole application in terms of accepting Dr Smith's input from the application GUI and preparing the results of retrievals for the GUI. It also decides which particular core computations will be executed at which stage.

The purpose of the *CORE COMPUTATIONS* is to create, manipulate, and retrieve ontologies local ontologies LO_j, target ontologies TO_k, derived ontologies DO_g and Go-CID, which may also include either populating a particular ontology with individuals, modifying a particular ontology or deleting content from a particular ontology. As a consequence of running *CORE COMPUTATIONS*, a connection may be made to the *REASONING ENGINE* in order to run rules upon the ontologies in the *ONTOLOGICAL LAYER*. The *OWL API* will be used to (i) link the *CORE COMPUTATIONS* to the *ONTOLOGICAL LAYER* and (ii) connect to the *Reasoning engine*.

5.4.1 Illustrating Architectural Elements of the Software Application built upon Ontological Layering

Figure 5.62 illustrates an example of the elements within software application derived from Figure 5.61 and according to (i) the scenario of the case study from section 5.1 and (ii) ontology mappings and their reasoning described in sections 5.3.1, 5.3.3 and 5.3.5. We define a set of application GUIs that will allow Dr Smith's inputs though *USER_INPUTS.* In other words, Dr. Smith will be allowed to click on radio buttons available within the application GUI which will be his own selection of data repositories *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep* and information types *patient details*, *medical summaries* and *treatment summaries*, relevant for his request for creating a health summary for Mrs Flee.

OUTPUTS in Figure 5.62 are defined in terms of displaying the output of retrievals across data repositories *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep*. Subsequently, the *APPLICATION MANAGER* manages the GUIs in terms of deciding what *CORE COMPUTATIONS* should be performed according to Dr. Smith's inputs. Thus, the *APPLICATION MANAGER* may trigger from the *CORE COMPUTATIONS* elements which:

- *insert* Dr. Smith's 'clicks' into ontological concepts of the USER_INP_ONT, i.e. populate ontological concepts that correspond to radio buttons selected (see section 5.2.5), or
- *prepare the results of retrievals* to fit the OUTPUTS in GUI, i.e. prepares for displaying all ontological concepts available within the Go-CID, which are relevant to Dr Smith's request.

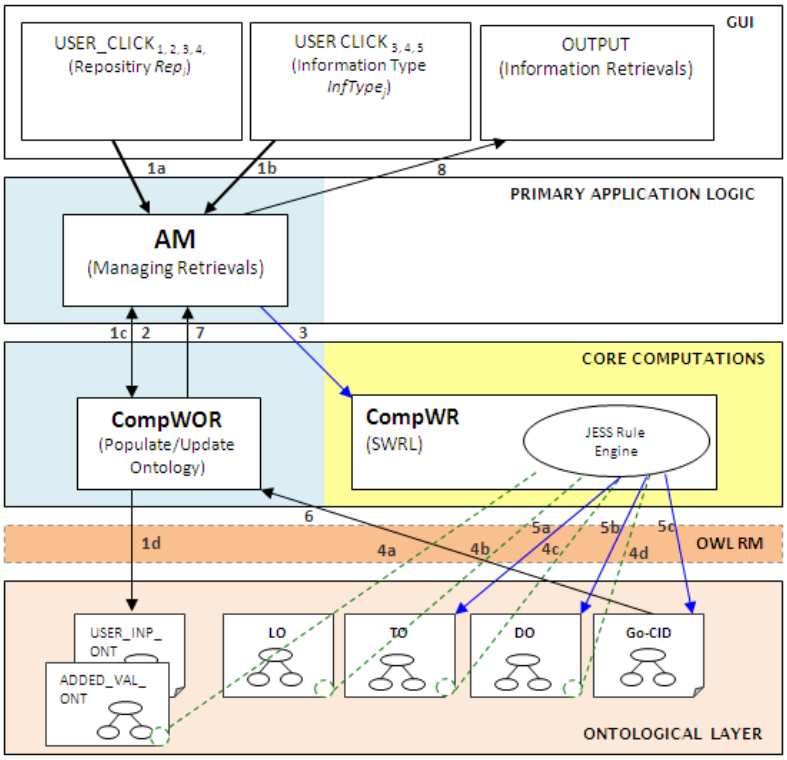

Figure 5.62 *Example of illustrating the elements contained within the architectural model of the software applications built upon ontological layering*

We divide the computations which deliver the functionalities of the software application built upon ontological layering in terms of creating:

- traditional or java based software code which may need NO SWRL rules for its execution (i.e. *Computations* ***without*** *Rules* and
- SWRL rules which underpin the semantic aspects of such applications and would need reasoning engines to perform them (i.e. *Computations* ***with*** *Rules*).

Thus, these two types of computations (*Computations* ***without*** *Rules* and *Computations* ***with*** *Rules*) may share ontologies, they might be run upon each other's results of computations, and may be able to accommodate reasoning rules, which in turn use the result set of reasoning created by another core computation. For example, *Computations* ***without*** *Rules* will contain java based code that will populate and update the USER_INP_ONT in the *ONTOLOGICAL LAYER*, and will not require the SWRL rules for their execution. On the other hand, *Computations* ***with*** *Rules* will trigger the running of SWRL rules upon the LO_j, TO_k, DO_g and Go-CID in the *ONTOLOGICAL LAYER*. This is how ontological layers are created in the first place.

5.4.2. Technology-specific Design Decisions for the Software Application

Figure 5.62 also shows how the tools and languages used for implementing ontological layering have influenced the way the software application has been designed. Therefore, the choice of tools and languages play a significant role in the way we connect the elements of our application model in Figure 5.61. The same factors have impact on the flow of data and the order of computations carried out within the software application, i.e. the numbering shown in Figure 5.62. Subsequently, we outline our choice of tools and languages used in the implementation of the application (section 5.4.2.1.1), followed by the explanations of the flow of data and the order of computations (section 5.4.2.1.2).

5.4.2.1 Tools and Languages

OWL DL is used to create the ontologies ENV_ONT, USER_INP_ONT, ADDED_VAL_ONT, LO_i, TO_i, DO_g and Go-CID, using the Protégé 3.4 ontological editing toolkit environment.

NetBeans 6.4 IDE is used to develop and implement the software application built upon ontological layering as it provides extensible application development environments and the functionality of uploading all the Protégé plug-ins (i.e. all the packages that make up the functionality of the Protégé 3.4 ontological toolkit environment). NetBeans 6.4 IDE allows its functionality to be extended by the selection of Protégé-OWL API libraries in order to create, manipulate and serialise OWL ontologies. NetBeans 6.4 IDE also includes the inbuilt swing package and provides a swing GUI builder.

Protégé-OWL API library is used to provide a Java API and reference implementation for creating and manipulating OWL ontologies. Thus, the software application takes the help of Protégé-OWL API in order to extend the functionality of abstract classes and methods for populating ontologies with individuals, creating a bridge to a reasoning engine and running the SWRL rules. It must be noted that the only way to manipulate ontologies created in Protégé 3.4 (i.e. the Protégé OWL model), is to convert them, into a 'Jena OWL reference model', in order to get a static snapshot of the ontological model at run time. Thus, the 'Jena OWL reference model' generated for each ontology in the *ONTOLOGICAL_LAYER* behaves like a run-time copy so that the software application can manipulate them.

Jess reasoning engine is used to infer logical consequences from a set of asserted facts or axioms in OWL ontolgies. The Jess reasoning engine is the preferred engine for running the SWRL rules as it comes as an inbuilt package with Protégé 3.4 itself. Many of the other reasoning engines Bossam[83], Pellet[84], RacerPro[85] and Jena[86] need to be imported externally while the others do not fully support the SWRL rules. Thus, the software application connects to the Jess reasoning engine whenever it needs to execute/run the SWRL rules.

SWRL Rule Engine Bridge[87] from the OWL API library is extended to create an instance of a bridge between the Jena OWL reference model (that include the SWRL rules) and the Jess reasoning engine. Thus, the Jess reasoning engine opens the Jena OWL reference model and accesses SWR rules within it, using the SWRL rule engine bridge. The SWRL rules are then imported onto the bridge from

[83] http://bossam.wordpress.com/about-bossam/
[84] http://clarkparsia.com/pellet/
[85] http://www.racer-systems.com/products/racerpro/index.phtml
[86] http://jena.sourceforge.net/
[87] http://protege.cim3.net/cgi-bin/wiki.pl?SWRLRuleEngineBridgeFAQ

the Jena OWL reference model and are executed/run to generate the results. Therefore, the SWRL rule engine bridge gives the software application the necessary infrastructure to incorporate the Jess reasoning engine into the software application in order to provide a mechanism to:

- import the SWRL rules and relevant OWL classes, individuals and object/datatype properties in the Jena OWL reference model and to write knowledge obtained from them, onto the Jess reasoning engine,
- allow the Jess reasoning engine to perform inference and to assert its new knowledge back to the bridge, and
- insert that asserted knowledge into the Jena OWL reference model.

Thus, the software application is capable of extracting information from the ontologies, running the SWRL rules and storing back the results to the original ontologies.

5.4.2.2 Flow of Data and Order of Computations

The numbering in Figure 5.62 shows the flow of data and order of computations that are required to deploy the elements contained within our architectural model of the software application built upon ontological layering (Figure 5.61) Hence, the numbers:

- 1a and 1b denote **the collection of the user inputs** through Dr. Smith's clicks, i.e. USER $CLICKS_{1,2,3,4}$ and USER $CLICKS_{4,5,6}$;
- 1c denotes **the opening of ontologies USER_INP_ONT and ADDED_VAL_ONT, plus the automatic creation of their Jena OWL Reference Models**, i.e. the *APPLICATION MANAGER* uses java technology to open the ontologies USER_INP_ONT and ADDED_VAL_ONT, which upon opening, automatically creates Jena OWL reference models for them;
- 1d denotes **the creation of ontological individuals for the subclasses in the ontology USER_INP_ONT** according to Dr. Smith's captured clicks;
- 2 denotes **the passing of information to the APPLICATION MANAGER** that confirms the USER_INP_ONT has been successfully populated with ontological individuals and *Computation with Rules* may start. Note: number 2 starts at the same time as number 1;
- 3 denotes **the establishment of a connection to the SWRL Rule Engine Bridge**, i.e. preparing activation of running the SWRL rules through the *Core computations with Rules* using the Jess reasoning engine;
- 4a denotes **the running of SWRL Selection and Grouping rules using the SWRLJessBridge**, i.e. opening the LO_j, USER_INP_ONT and ADDED_VAL_ONT Jena OWL reference models and importing the *Selection* and *Grouping* rules onto the 'SWRLJessBridge' in order to execute and generate results of inference;
- 4b denotes **the running of SWRL Low-Level rules using the SWRLJessBridge**, i.e. opening the LO_j Jena OWL reference model and importing *Low-Level* SWRL rules onto the 'SWRLJessBridge' in order to execute and generate results of inference;
- 4c denotes **the running of SWRL High-Level rules using the SWRLJessBridge**, i.e. opening the TO_k Jena OWL reference model and importing *High-Level* SWRL rules onto the 'SWRLJessBridge' in order to execute and generate results of inference;

- 4d denotes **the running of SWRL Post-High-Level rules using the SWRLJessBridge**, i.e. opening the DO_g Jena OWL reference model and importing *Post-High-Level* SWRL rules onto the 'SWRLJessBridge' in order to execute and generate results of inference;
- 5a denotes **the update of the TO Jena OWL reference model** by storing the results generated by running *Selection*, *Grouping* and *Low-Level* SWRL rules in numbers 4a and 4b. The TO_k Jena OWL reference model is saved back as the TO_k ontology in the *ONTOLOGICAL LAYER*, i.e. target ontologies has been created;
- 5b denotes **the update of the DO Jena OWL reference model** by storing the results generated by running *High-Level* SWRL rules in number 4c. The DO_g Jena OWL reference model is saved back as the DO_g ontology in the *ONTOLOGICAL LAYER*, i.e. derived ontologies has been created;
- 5c denotes **the update of the Go-CID Jena OWL reference model** by storing the results generated by running *Post-High-Level* SWRL rules in number 4d. The Go-CID Jena OWL reference model is saved back as the final Go-CID in the *ONTOLOGICAL LAYER*, i.e. the final Go-CID ontology has been created;
- 6 and 7 denote **retrieving classes from Go-CID** and the transference of ontological concepts to the *APPLICATION MANAGER*, i.e. uses java technology for retrieving ontological individuals from the final result set of reasoning saved in the Go-CID ontology through *Core computations without Rules*;
- 8 denotes **the displaying of ontological individuals retrieved from Go-CID**, i.e. the *APPLICATION MANAGER* uses java technology for displaying ontological individuals from the Go-CID in the OUTPUT GUI through *Core computations without Rules*.

The green dotted lines in Figure 5.62 (numbers 4a, 4b, 4c and 4d), mean that we conduct the *Computation with Rules* through the Jess reasoning engine and upon the subclasses of the ontologies in the *ONTOLOGICAL LAYER*. However, it is important to note these SWRL reasoning rules:

- run in a chain, i.e. they are executed immediately one after another (transparent to the user),
- may run upon a result set of a predecessor rule, and
- must conform in their purpose to the semantics of the results sets of the preceding rule.

Consequently, these green dotted lines in numbering 4a, 4b, 4c and 4d are 'teamed-up' with blue lines with arrow heads in numbering 5a, 5b and 5c which are directed towards subclasses of the ontologies USER_INP_ONT, ADDED_VAL_ONT, LO_j, TO_k, DO_g and Go-CID, which subsequently, represent 'classification', 'inference' and 'assertion' within the software application. Therefore, 'classification', 'inference' and 'assertion' (numbering 5a, 5b and 5c) are consequences of running the SWRL rules in numbering 4a, 4b, 4c and 4d, and are responsible for creating/transferring new ontological individuals and ontological properties within existing ontologies. The creation/transference of new ontological individuals or ontological properties is seen as a 'result set of a particular reasoning' which is made ready for further reasoning through the SWRL rules.

5.4.3 Examples of GUIs and Java Code of the Software Application

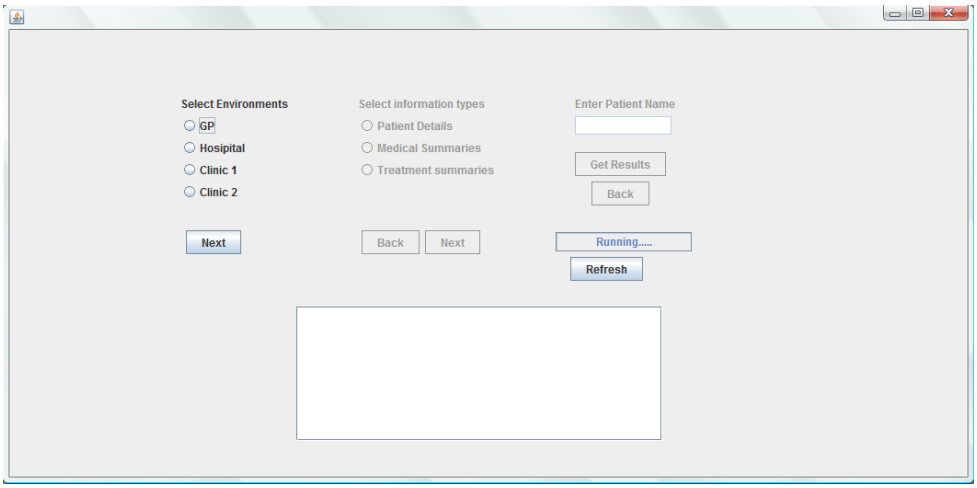

Figure 5.63 *Example of the interface design used for the software application*

Figure 5.63 illustrates the design of the GUI used in the software application for retrievals across heterogeneous pervasive healthcare environments in section 5.1. The interface is designed such that the number of available data repositories and information types can be selected from a set of radio buttons offered. Hence, each radio button corresponds to:

- the 'Selection of Environments' (which includes their available repositories: *GP_data_rep*, *Hospital_data_rep*, *Clinic 1_data_rep* and *Clinic 2_data_rep*), and
- the 'Selection of Information Types' (from available repositories in environments: *medical summaries*, *treatment summaries* and *patient details*).

Figure 5.63 also shows a text box for allowing Dr. Smith to enter a Patients name (i.e. Jane Flee) and a text box for displaying the results of his retrievals. Additional functionalities such as 'Next', 'Back' and 'Refresh' buttons are added for ease of use and navigation. The 'Running' progress bar is displayed to judge the running status of the Go-CID software application.

Note: for testing purposes one interface is created to accomadate radio buttons offering the choice of Rep_i and $InfType_d$ and display of retrievals. The 'GO_CID_Implementation.owl' file is created to store all the OWL classes that belong to ontologies USER_INP_ONT, ADDED_VAL_ONT, LO_j, TO_k, DO_g and Go-CID, plus all *Selection*, *Grouping*, *Low-Level*, *High-Level* and *Post-High-Level* SWRL rules. This has been done for experimentation and testing purposes in order to test the feasibility of triggering rule-chaining across ontological layering and subsequently, resolving semantic conflicts through ontology alignment, integration and merge.

All the .jar files from the Protégé plug-in directory are imported into the Java application project library. This is so that the Java program (the software application) can execute methods to work with the 'GO_CID_Implementation.owl' file. Thus, the first java code in our software application is responsible for importing the packages from which we wish to use functions, methods and objects. The full Java source code for the software application rules can be found in Appendix A.20. (Appendix A.20 is stored on the CD-ROM, however, we show an excerpt from the code below). Note: from now onwards all Java

code will be listed in the in blue text of font type '`courier font`' and comments will be made in '/**…. /*' using '*italic*' black text of font type 'times new roman'.

```
import com.hp.hpl.jena.util.FileUtils;
import edu.stanford.smi.protege.exception.OntologyLoadException;
import edu.stanford.smi.protegex.owl.ProtegeOWL;
import edu.stanford.smi.protegex.owl.jena.JenaOWLModel;
import edu.stanford.smi.protegex.owl.model.OWLIndividual;
import edu.stanford.smi.protegex.owl.model.OWLModel;
import edu.stanford.smi.protegex.owl.model.OWLNamedClass;
import edu.stanford.smi.protegex.owl.model.RDFObject;
import edu.stanford.smi.protegex.owl.swrl.bridge.BridgeFactory;
import edu.stanford.smi.protegex.owl.swrl.bridge.SWRLRuleEngineBridge;
import
edu.stanford.smi.protegex.owl.swrl.bridge.exceptions.SWRLRuleEngineBridgeExcep
tion;
import edu.stanford.smi.protegex.owl.swrl.exceptions.SWRLRuleEngineException;
import edu.stanford.smi.protegex.owl.swrl.model.SWRLFactory;
```

The following sections describe the Java source code for the implementation of the computations required to deploy architectural elements of the software application, i.e. description of java code according to the order of computation specified in Figure 5.61.

Collecting User Inputs

Dr. Smith's inputs through his USER CLICKS$_{1,2,3,4}$ (corresponding to the selection of data repositories) and USER CLICKS$_{4,5,6}$ (corresponding to the selection of information types) are collected through JFrames designed in the GUI, which is powered by swing components. Thus, a GUI consisting of 'jButtons', 'jLabels' and 'jTextField' is created by using the following Java code:

```
public class NewJFrame1 extends javax.swing.JFrame { /** Creates new form NewJFrame1 */
        public NewJFrame1() {
        initComponents();
    }
    private void initComponents() { /**This method is called from within the constructor to
       initialize the form/*
   buttonGroup1 = new javax.swing.ButtonGroup(); /**assigns a new JButton()/*
   buttonGroup2 = new javax.swing.ButtonGroup();/**assigns a new JButton()/*
    buttonGroup3 = new javax.swing.ButtonGroup();/**assigns a new JButton()/*
    jFrame1 = new javax.swing.JFrame(); /**assigns a new JFrame()/*
    jFrame2 = new javax.swing.JFrame();/**assigns a new JFrame()/*
    jFrame3 = new javax.swing.JFrame();/**assigns a new JFrame()/*
    jFrame4 = new javax.swing.JFrame();/**assigns a new JFrame()/*
    jFrame5 = new javax.swing.JFrame();/**assigns a new JFrame()/*
    jFrame6 = new javax.swing.JFrame();/**assigns a new JFrame()/*
    jDialog1 = new javax.swing.JDialog();/**assigns a new JFrame()/*
    jFrame7 = new javax.swing.JFrame();/**assigns a new JFrame()/*
    jDialog2 = new javax.swing.JDialog();/**assigns a new JFrame()/*
    jRadioButton1 = new javax.swing.JRadioButton(); /**assigns a new JButton()/*
    jRadioButton2 = new javax.swing.JRadioButton();/**assigns a new JButton()/*
    jRadioButton3 = new javax.swing.JRadioButton();/**assigns a new JButton()/*
    jRadioButton4 = new javax.swing.JRadioButton();/**assigns a new JButton()/*
    jLabel1 = new javax.swing.JLabel(); /**assigns a new JLabel()/*
    jButton1 = new javax.swing.JButton();/**assigns a new JButton()/*
    jLabel2 = new javax.swing.JLabel();/**assigns a new JLabel()/*
    jLabel3 = new javax.swing.JLabel(); /**assigns a new JLabel()/*
    jRadioButton5 = new javax.swing.JRadioButton(); /**assigns a new JRadioButton()/*
```

```
jRadioButton6 = new javax.swing.JRadioButton(); /**assigns a new JRadioButton()/*
jRadioButton7 = new javax.swing.JRadioButton(); /**assigns a new JRadioButton()/*
jTextField1 = new javax.swing.JTextField(); /**assigns a new JTextFeild()/*
jInternalFrame1 = new javax.swing.JInternalFrame(); /**assigns a new
    JInternalFrame()/*
jScrollPane1 = new javax.swing.JScrollPane(); /**assigns a new JScrollpane()/*
jTextArea1 = new javax.swing.JTextArea();/**assigns a new JTextArea()/*
jButton2 = new javax.swing.JButton(); /**assigns a new JButton()/*
jButton3 = new javax.swing.JButton(); /**assigns a new JButton()/*
jButton4 = new javax.swing.JButton(); /**assigns a new JButton()/*
jButton5 = new javax.swing.JButton(); /**assigns a new JButton()/*
jProgressBar1 = new javax.swing.JProgressBar(); /**assigns a new JProgressBar()/*
jButton6 = new javax.swing.JButton(); /**assigns a new JButton()/*
```

An 'Action Listener' is added to each of the 'jButtons' and 'jTextField' which execute a piece of code to perform an action. This is how the working of the Go-CID software application is invoked from the interface. The code used to perform an action upon clicking the 'jButtons' is:

```
jButton1.addActionListener(new java.awt.event.ActionListener() { /**this is the
    text that appears on the button and can be modified/*
public void actionPerformed(java.awt.event.ActionEvent evt) {
jButton1ActionPerformed(evt); /**this code will run the "Action Performed" method upon a click,
    which is generally written by a user to tell the program what to do/*
```

Similar action events for the rest of the '`jButtons`' and '`jTextFields`' are defined in the Java source code for the Go-CID software application in Appendix A.20.

Creating the Jena OWL Reference Model

The Jena OWL reference model from the 'GO_CID_Implementation.owl' file is created by giving the path of the ontology from the function:

'`owlModel = ProtegeOWL.createJenaOWLModelFromURI(uri)`'

using the string URI from the specified path of where the 'GO_CID_Implementation.owl' file is stored:

"`file:///Users/KAT/desktop/NetBeansProjects/Protegeowl/GO_CID_Implementation.owl`".

Creating individuals for OWL classes in the USER_INP_ONT

The 'GO_CID_Implementation.owl' file is populated with OWL Individuals by attaching 'action events' to each of the '`jButtons`' and '`jTextFields`'. Thus, the code used to create an ontological individual in the OWL class of the 'GO_CID_Implementation.owl' file, if a particular '`JButton`' is clicked is:

```
        if (jRadioButton1.isSelected() == true) /**if the 'jRadioButton1' is selected, then do
        the following/*
{
            FileOutputStream fos = null;
            try {
                String uri =
"file:///c:/Users/Kat/Desktop/Protegeowl/GO_CID_Implementation2.owl";
                File openAs = new File("c:/netbeansproject/system_out.txt");
```

```
                fos = new FileOutputStream(openAs);
                PrintStream ps = new PrintStream(fos);
                System.setOut(ps);
                OWLModel owlModel = null;
                owlModel = ProtegeOWL.createJenaOWLModelFromURI(uri); /**the
                GO_CID_Implementation2  Jena OWL reference model is created/*
        String lot = "userinput1_1"; /**the name of the individual is entered into a string/*
        OWLNamedClass clientclass =
        owlModel.getOWLNamedClass("USER_CLICK_gp_rep"); /**the GO_CID_Implementation2
        Jena OWL reference model  is queried to get the class that has the name USER_CLICK_gp_rep/*
        OWLIndividual ind = clientclass.createOWLIndividual(lot); /**an owl individual
        is created from the string lot into the class USER_CLICK_gp_rep/*
        String filename = "GO_CID_Implementation2.owl"; /**file name is assigned with the
        name of the owl file/*
       Collection errors = new ArrayList(); /**errors are collected in a new array list/*
        ((JenaOWLModel) owlModel).save(new File(filename).toURI(),/**the
          GO_CID_Implementation2 Jena OWL reference model  is saved replacing the existing the
          GO_CID_Implementation2.owl with a new 'populated' OWL model /*
 FileUtils.langXMLAbbrev, errors);
            } catch (OntologyLoadException ex) {
                Logger.getLogger(NewJFrame1.class.getName()).log(Level.SEVERE,
null, ex);
            } catch (FileNotFoundException ex) {
                Logger.getLogger(NewJFrame1.class.getName()).log(Level.SEVERE,
null, ex);
            } finally {
                try {
                    fos.close();
                } catch (IOException ex) {

Logger.getLogger(NewJFrame1.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
}
```

Similar 'if' statements including the 'OWLIndividual ind = clientclass.createOWLIndividual(lot)' for the rest of the 'jButtons' are specified in the Java source code in Appendix A.20. The code used to create an individual in the OWL class of the 'GO_CID_Implementation.owl' file, if a particular 'JTextfeild' is entered with text is below:

```
String x = jTextField1.getText();
if (x.equals("JANE FLEE")) /**if the text entered is equivalent to 'JANE FLEE', then do the following/*
{
            {
                FileOutputStream fos = null;
                try {
                    String uri =
"file:///c:/Users/Kat/Desktop/Protegeowl/GO_CID_Implementation2.owl";
                    File openAs = new
File("c:/netbeansproject/system_out.txt");
                    fos = new FileOutputStream(openAs);
                    PrintStream ps = new PrintStream(fos);
                    System.setOut(ps);
                    OWLModel owlModel = null;
         owlModel = ProtegeOWL.createJenaOWLModelFromURI(uri); /**creates the
         GO_CID_Implementation2 Jena OWL reference model/*
         String lot = "user_input8_1"; /**the name of the individual is entered into a string/*
                    OWLNamedClass clientclass =
owlModel.getOWLNamedClass("TEXT_ENTERED_jane_flee"); /**the GO_CID_Implementation  is
queried to get the class that has the name TEXT_ENTERED_jane_flee/*
         OWLIndividual ind = clientclass.createOWLIndividual(lot); /**an owl
         individual is created from the string lot into the class TEXT_ENTERED_jane_flee/*
```

```
                    String filename = "GO_CID_Implementation2.owl"; /**file name is
                    assigned with the name of the owl file/*
          Collection errors = new ArrayList();/**errors are collected in a new array list/*
          ((JenaOWLModel) owlModel).save(new File(filename).toURI(), /**the
           GO_CID_Implementation2 Jena OWL reference model  is saved replacing the existing the
           GO_CID_Implementation2.owl with a new 'populated' OWL model /*
FileUtils.langXMLAbbrev, errors);
                } catch (FileNotFoundException ex) {

Logger.getLogger(NewJFrame1.class.getName()).log(Level.SEVERE, null, ex);
                } catch (OntologyLoadException ex) {

Logger.getLogger(NewJFrame1.class.getName()).log(Level.SEVERE, null, ex);
                } finally {
                    try {
                        fos.close();
                    } catch (IOException ex) {

Logger.getLogger(NewJFrame1.class.getName()).log(Level.SEVERE, null, ex);
                    }
                }
            }
}
```

Similar '`if (x.equals)`' statements including the '`OWLIndividual ind = clientclass.createOWLIndividual(lot)`' for the 'jTextFeilds' are specified in the Java source code in Appendix A.20.

Establishing a SWRL Rule Engine Bridge

The SWRL rules are triggered through establishing a connection to the SWRL Rule engine bridge. The code below is used to connect to the Jess reasoning engine through the extension of the Protégé-OWL API library: 'OWLModel' object and 'BridgeFactory' abstract class (this piece of code assumes that the owl model of a given ontology is already created):

```
SWRLFactory factory = new SWRLFactory(owlModel); //A SWRL factory is created from the
owlmodel//
SWRLRuleEngineBridge bridge = BridgeFactory.createBridge("SWRLJessBridge",
owlModel); // SWRL rule engine bridge called SWRLJessBridge is created, this returns a successful registration
of a bridge else throws an exception//.
```

Running SWRL rules using the SWRLJessBridge

The 'SWRLJessBridge' is used to import all the SWRL rules present in the Jena OWL reference model from the 'GO_CID_Implementation.owl' file. They are loaded onto the bridge and then executed by the Jess reasoning engine. The Method `infer()` contains sub methods which do the following:

- the `reset()` method, which clears all knowledge that is already present on the Jess reasoning engine;
- the `importSWRLRulesAndOWLKnowledge()` method, which imports all the SWRL rules (and relevant OWL knowledge) from the Jena OWL reference model from the 'GO_CID_Implementation.owl' file into the '`SWRLJessBridge`' (all existing bridge rules and knowledge must be cleared through the `reset()` method);
- the `run()` method, which invokes the Jess reasoning engine;

- the `writeInferredKnowledge2OWL()` method, which transfers any information classified/inferred/asserted by the Jess reasoning engine to the Jena OWL reference model from the 'GO_CID_Implementation.owl' file. Following is an example of the code describing this. (Note: the results are also stored on the bridge until they are cleared).

```
      OWLModel owlModel = null;
                owlModel = ProtegeOWL.createJenaOWLModelFromURI(uri);
                 /**creates the GO_CID_Implementation2 Jena OWL reference model/*
      SWRLFactory factory = new SWRLFactory(owlModel);
                SWRLRuleEngineBridge bridge =
BridgeFactory.createBridge("SWRLJessBridge", owlModel);
                try {
                    bridge.infer(); /**runs the infer() method on the bridge/*
                } catch (SWRLRuleEngineException ex) {

Logger.getLogger(NewJFrame1.class.getName()).log(Level.SEVERE, null, ex);
                }
```

Updating the Jena OWL reference models

The 'GO_CID_Implementation.owl' file is updated using the following piece of code. Subsequently, the GO_CID_Implementation Jena OWL reference model is saved back as the 'GO_CID_Implementation2.owl' file.

```
((JenaOWLModel) owlModel).save(new File(filename).toURI(),
FileUtils.langXMLAbbrev, errors);  (), /**the GO_CID_Implementation2 Jena OWL reference
model is saved replacing the existing the GO_CID_Implementation3.owl with a new OWL model with 'classified /
inferred / asserted' result sets from running SWRL rules /*
```

Retrieving classes from Go-CID

Ontological classes from the final Go-CID ontology are retrieved by inserting the name of the OWL class. Thus, we specify the following code:

```
OWLModel owlModel1 = null;
  try {
       owlModel1 = ProtegeOWL.createJenaOWLModelFromURI(uri1);
      /**creates the GO_CID_Implementation3 Jena OWL reference mode1, where 'uri1' is specified as
      GO_CID_Implementation3.owl/*
                } catch (OntologyLoadException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
                }
                Collection classes =
owlModel1.getUserDefinedOWLNamedClasses/**search through the GO_CID_Implementation3 Jena
OWL reference model until you find a class that is equivalent to 'y', where 'y' is specified as the 'Go-
CID_ontological_layer' named class'/*
    for (java.util.Iterator it = classes.iterator(); it.hasNext();) {
         OWLNamedClass cls = (OWLNamedClass) it.next();
          if ( y.equals(cls.getBrowserText()))/**if a class is equivalent to 'y', where 'y' is
           specified as the 'Go-CID_ontological_layer' named class, then get the name of sublcasses in a string
           format/*
                        {
                    System.out.println("Class " + cls.getBrowserText());
                    Collection z = cls.getNamedSubclasses(true);
                    for ( java.util.Iterator pt = z.iterator();pt.hasNext();)
                    {
```

```
                    OWLNamedClass cls1 = (OWLNamedClass) pt.next();
                        if ( cls1.getSubclassCount() == 0 ){
                      System.out.println("-->" + cls1.getBrowserText());
            Collection instances = cls1.getInstances(true);
                    for (java.util.Iterator jt = instances.iterator();
jt.hasNext();) {
                            OWLIndividual individual = (OWLIndividual)
jt.next();
                        System.out.println(" ---- " + ((RDFObject)
individual).getBrowserText());
                    }
                    }
                        else
                        {
                        System.out.println("->" + cls1.getBrowserText());
                        }
                    }
                    }
                }
```

Displaying the individuals retrieved from the Go-CID

The output of retrieving classes from Go-CID are generally displayed on the NetBeans 6.4 IDE. Thus, in order to display the output of retrievals onto the text box (JFrame) for displaying the results in GUI, we redirect the output of the computation above into a text file. We use the text file to print back the output of computation to the 'action event' associated to the JFrame. Thus, the text file acts as a buffer between the NetBeans 6.4 IDE and the JFrame used in our interface. This text file can be cleared as soon as the output is transferred to the interface. Below is an excerpt from the code to illustrate this:

```
File openAs = new File("c:/netbeansproject/system_out.txt"); /**a new text
file is created to hold the output/*
fos = new FileOutputStream(openAs); /**a new output stream is created/*
        PrintStream ps = new PrintStream(fos); /**PrintStream is created from the
        FileOutputStream/*
System.setOut(ps); /**The system output is set to the new PrintStream/*
```

The following code uses the text from the system_out.txt to redirect the output (i.e. the list of subclasses and individuals between them) to the '`jTextArea1`' in the interface.

```
File openAs1 = new File("c:/netbeansproject/system_out.txt"); /**opens the
text file mentioned in the path/*
FileReader in = null; /**creates a text file reader/*
        try {
            in = new FileReader(openAs1);
        } catch (FileNotFoundException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
        }
        try { /**reads the data in the text file to the jTextArea1  in the jFrame/*
            jTextArea1.read(in, openAs.toString());
        } catch (IOException ex) {

Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
        }
        FileOutputStream fop = null; /**creates the outputstream and clears the text
file/*
        try {
            fop = new
FileOutputStream("c:/netbeansproject/system_out.txt");
```

```
                } catch (FileNotFoundException ex) {
Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
                }
                try {
                    fos.write("".getBytes());
                } catch (IOException ex) {
Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
                }
                try {
                    fos.close();
                } catch (IOException ex) {
Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
                }
        }   catch (SWRLRuleEngineBridgeException ex) {
                Logger.getLogger(NewJFrame1.class.getName()).log(Level.SEVERE,
null, ex);
            }   catch (OntologyLoadException ex) {
                Logger.getLogger(NewJFrame1.class.getName()).log(Level.SEVERE,
null, ex);
            } catch (FileNotFoundException ex) {
            Logger.getLogger(NewJFrame1.class.getName()).log(Level.SEVERE,
null, ex);
        }
}
    }
```

Note: in order to provide re-usability in terms of accommodating any combination of Dr Smith's choice of data repositories and information types, the Go-CID software application uses two copies of the 'GO_CID_Implementation.owl' file. The first copy named 'GO_CID_Implementation2.owl' is used to accommodate the population of ontological individuals. The second copy named 'GO_CID_Implementation3.owl' is used to accommodate the results of running the SWRL rules. The original copy remains as it is, and ise used for replacing copies 'GO_CID_Implementation2.owl' and 'GO_CID_Implementation3.owl', if and when the 'Refresh button is selected in the Go-CID software application. Therefore, in order to revert the Go-CID software application to its original state, i.e. reverting back to an empty ontology that contains no 'classified/inferred/asserted' ontological individuals as a consequence of running the SWRL rules, we run the following code that saves copy 1 ('GO_CID_Implementation2.owl' file) and copy 2('GO_CID_Implementation3.owl' file) of the 'GO_CID_Implementation.owl' file to its original state by overriding it with the 'GO_CID_Implementation' Jena OWL reference model.

```
  private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) { ) /**if
   the 'jButton6' is clicked, then do the following/*
{
                FileOutputStream fos = null;
                try {
                    String uri =
"file:///c:/Users/Kat/Desktop/Protegeowl/GO_CID_Implementation1.owl";
                    File openAs = new
File("c:/netbeansproject/system_out.txt");
                    fos = new FileOutputStream(openAs);
                    PrintStream ps = new PrintStream(fos);
                    System.setOut(ps);
                    OWLModel owlModel = null;
                    owlModel = ProtegeOWL.createJenaOWLModelFromURI(uri);
```

```
                    /**create the GO_CID_Implementation Jena OWL reference mode1, where 'uri1' is
                    specified as GO_CID_Implementation.owl/*
                    String filename = "GO_CID_Implementation2.owl";
                    Collection errors = new ArrayList();
                    ((JenaOWLModel) owlModel).save(new File(filename).toURI(),
                    /**the GO_CID_Implementation Jena OWL reference model  is saved replacing the
                    existing the GO_CID_Implementation2.owl with a new  OWL model with NO
                    'classified / inferred / asserted' result sets from running SWRL rules /*
FileUtils.langXMLAbbrev, errors);
                String filename1 = "GO_CID_Implementation3.owl";
                Collection errors1 = new ArrayList();
                    ((JenaOWLModel) owlModel).save(new File(filename1).toURI(),
                    /**the GO_CID_Implementation Jena OWL reference model  is saved replacing the
                    existing the GO_CID_Implementation3.owl with a new  OWL model with NO 'classified
                    / inferred / asserted' result sets from running SWRL rules /*
FileUtils.langXMLAbbrev, errors);
```

5.6 Summary

In this chapter we have illustrated the implementation of our ontological and layered software architecture through a specific example of retrievals of semantically related data across repositories in pervasive healthcare environments. We have presented the heterogeneous relational schemas for the data repositories *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep*, and have described their semantically related data across information types *medical summaries*, *treatment summaries*, and *patient details* plus the various types of semantic conflicts they may generate. To that end, we exemplify the process for creating and deploying Go-CID software architectural components in order to resolve semantic conflicts.

Specifically, we have detailed the preparation of semantics for our core ontological layers by (a) translating the content and structure of data repositories into Local ontologies and the ENV_ONT and (b) storing user involvements in the USER_INP_ONT and interpreting their inputs through the ADDED_VAL_ONT. We have presented our results of running the *Selection* and *Grouping* SWRL rules in order to group semantically related data pertained within information types *medical summaries*, *treatment summaries*, and *patient details*. Hence, creating a context within which we can start comparing and resolving semantic conflicts.

Subsequently, we have detailed our core ontological layers by exemplifying our ontology mappings: *alignment*, *integration* and *merge*. We have presented our results of the running *Low-level* SWRL rules that aligns $LO_j s$ into $TO_k s$, resolving synonym based naming conflicts in the process. We have presented our results of running the *High-level* SWRL rules that integrates $TO_k s$ into $DO_g s$, resolving generalisation, specialisation, isomorphism and union incompatibility based structural conflicts in the process. We have also presented our results of running the *Post-High-Level* SWRL rules that merges $DO_g s$ into the final Go-CID ontology.

Finally we have described our full scale implementation of a Go-CID software application that successfully retrieves ontological concepts of Go-CID, ensuring semantic interoperability of heterogeneous data repositories *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep*.

Chapter 6

Case Study: Submissions of MA Applications for Medicines

Our reasoning mechanisms, i.e. ontological mappings and groupings, which have been introduced and used in the previous 2 chapters, proved to be a very effective way of managing semantically related data in heterogeneous data repositories. When resolving semantic conflicts triggered by the existence of semantically related data, we exploit the semantics stored in such data through a specific process. It allows us to understand that semantic conflicts may appear only when requests for retrievals across data repositories are issued by users. In other words, our process initially helps us to establish where to identify which semantically related data exists. This will be the first indication that semantic conflicts exist, when a request for data retrieval is issued. The process also enables the manipulation of existing semantics in data repositories by inferring or asserting a set of ontological individuals, which exhibit no semantic conflicts and produce correct results of retrievals, according to user's requests.

In this chapter we describe how the same reasoning mechanisms, which are either ontological groupings or mappings, can be used in different problem domains and in environments where we need to

(i) establish if and when we have overlapping "semantics" which creates relationship between data/information in such domains/environments, or/and

(ii) infer and/or assert a correct set of "semantics" which can support any decision making required in such domains/environments.

We have conducted various case studies to see the effectiveness and feasibility of our ontological reasoning in order to illustrate (i) and (ii) above. Therefore, our ontological mappings (i.e. alignment, options 6a and 6c, see chapter 4, section 4.3.4.2) have been used to support decision making, when addressing changes in business models/processes, which are expected to react to their environments and which, consequently, have to respond to business models/processes' external/internal factors which impose changes upon them [297 and 298]. We have also used ontological mappings (alignment option 6b and integration option 7b (see chapter 4, sections 4.3.4.2 and 4.3.4.3), when managing non-functional requirements, in terms of making decisions on the content(s) and functionality(ies) of pervasive spaces for remote patient monitoring [291 and 292]. Finally, the same ontological mappings (see chapter 4, sections 4.3.4.2 and 4.3.4.3) has been used in supporting decision making when creating virtual learning

environments [299]. However, in this last example, we have exclusively experimented with data type properties as a mechanism of matching ontological individuals in order to perform alignments.

All the examples above are available as separate publications, which carry detailed explanations on the way we re-use our ontological mappings introduced in chapter 4. Due to space restrictions, we are not in a position to include them in this thesis. Therefore, we invite readers to read all these publications in order to build a better picture on re-usability of our reasoning mechanisms when we have to deal with overlapping semantics in any problem domain.

In this chapter we choose to elaborate on only one case study (the shortest!) which is described in section 6.1. We use ontological grouping as in (i) above (introduced in chapter 4, section 4.3.3.5.2 and illustrated in chapter 5, section 5.25). The problem domain covers the submission of *applications for marketing authorisation (MA)* of medicines, where we need to ensure that the correct set of PDF documents has been submitted, according to requirements specified in the regulatory bodies' common documentation [300]. This example is an obvious case where we can semantically relate the content of each PDF document to a section of the *application for MA* where the PDF document should be located. In other words, we ensure that PDF documents are never in the wrong place.

6.1 Problems with Submissions for Marketing Authorisation of Medicines

In the current process of MA electronic submissions, the documentation required by government regulatory agencies across the world is complex, and the amount of information supplied by pharmaceuticals is multifaceted and voluminous. Each electronic submission must be done according to the electronic Common Technical Document (eCTD), which is a common **format and structure** of the submission document, which ensures that the correct data/information is supplied by pharmaceutical companies. The eCTD consists of five modules:

- Module 1 refers to regional administrative information,
- Module 2 refers to quality, non-clinical and clinical summaries,
- Module 3 refers to quality, chemical, pharmaceutical and biological data,
- Module 4 refers to non-clinical reports, and
- Module 5 refers to the clinical study reports.

Each module contains a number of 'sections' that mirror the structure of the eCTD (in the form of their required sub headings), which in turn specifies the type of information required for each sub section of a particular module. Each section further contains a number of 'contents' that specifies the exact data required in each module constituting the eCTD.

The structure of the eCTD is mirrored in a navigational structure available in the XML data file. PDF documents containing the data required for each section in the module i.e. PDF documents which constitute the eCTD, are linked together, by referencing an XML data file which positions them correctly within eCTD, according to their content.

Current software support for managing MA submissions is limited and offers only a technical validation mechanism of linking PDF files, by creating the XML data file (i.e. creating the eCTD navigational structure). The software is able to check the file format (is it a PDF?), size, missing PDF

documents, and the existence of a valid eCTD navigational structure. However, it fails to check if the content of each PDF document is correct and if it is correctly positioned within eCTD. In other words, it fails to check if the 'content' of each PDF document corresponds (is related) to a correct 'section' of a module within the eCTD. We interpret that there is a relationship between the 'content' of each PDF document and associated 'section' where the PDF document belongs (i.e. within which we have to position the PDF document). We can then claim that 'content' and 'section' are semantically related. Therefore, in order to check the validity of the eCTD it becomes important to:

a) establish the number of semantically related 'sections' and 'contents' within the eCTD navigational structure, because each module of eCTD has a different number of sections and each section is associated with a particular 'content'.

b) guarantee the correct content of eCTD and its navigational structure i.e. avoid linking 'contents' to 'sections' that may initially be seen as semantically related ("resemble each other") but actually have no similarity(ies) (errors when allocating PDF documents at a particular position within eCTD).

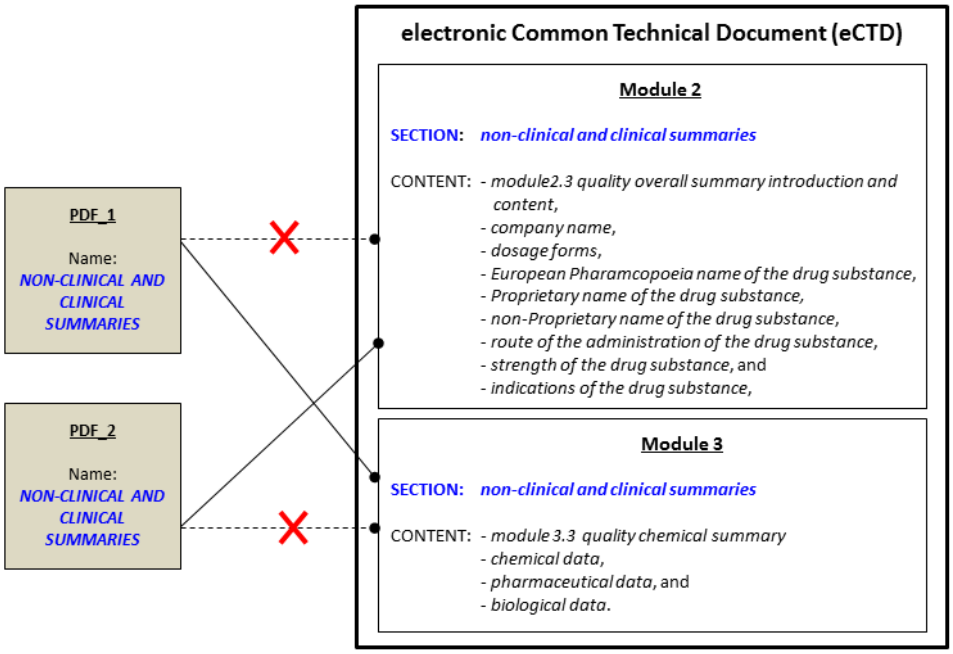

Figure 6.1 *Example of linking 'content' of a PDF document to the (wrong) 'section' in the eCTD*

Figure 6.1 gives an example where a 'content' of a PDF document may be linked to the wrong 'section' in the eCTD (**X** in the diagram). Modules 2 and 3 in the eCTD require a section on *non-clinical and clinical summaries*. Therefore, we expect to have two different PDF documents for two different modules, both dealing with *non-clinical and clinical summaries.* The naming of PDF documents is usually dictated by the eCTD, but the user may violate any of the existing recommendations and name both PDF documents as *NON-CLINICAL_AND_CLINICAL_SUMMARIES*. This is one of the most common errors discovered in eCTD. Figure 6.1 illustrates that:

- in Module 2 of the eCTD, the section on *non-clinical and clinical summaries* may require to have the following content: *module 2.3 quality overall summary introduction and content*, *company name*, *dosage forms*, *European Pharamcopoeia name of the drug substance*, *Proprietary name of the drug*

substance, *non-Proprietary name of the drug substance*, *route of the administration of the drug substance*, *strength of the drug substance*, and *indications of the drug substance*, and

- in Module 3 of the eCTD, the section on *non-clinical and clinical summaries* may require to have the following content: *module3.3 chemical summary*, *chemical data*, *pharmaceutical data*, and *biological data*.

Therefore, the PDF documents in Figure 6.1 have different contents, but their names are the same. We also know that PDF 1 document should belong to Module 3 and PDF 2 document to module 2. However, when trying to place these two PDF documents, both named as *NON-CLINICAL_AND_CLINICAL_SUMMARIES*, at the correct place within eCTD, we have no mechanism of ensuring that a correct PDF document is placed within Modules 2 (PDF 2) and Module 3 (PDF 1) in the eCTD.

It is obvious that there is a relationship between the content of PDF documents and their positioning within the eCTD. We claim that the content of each PDF document, in this particular case two PDF documents named as *NON-CLINICAL_AND_CLINICAL_SUMMARIES*, is ***semantically related*** to a particular section within Module 2 and Module 3 (because their contents are "expected" within these two modules). This is true for any other PDF documents: there is always a relationship between the content of PDF document and section in the eCTD where we need to place it.

6.2 Reasoning Mechanisms for Creating a Correct eCTD

We propose to manipulate the semantics of eCTD, and the content of PDF documents which constitute them, in order to solve the problem of positioning the wrong PDF documents within eCTD. We create two ontologies, which describe separately eCTD and PDF documents, and use our grouping reasoning mechanism (introduced in chapter 4, section 4.3.3.5.2 and illustrated in chapter 5, section 5.25), upon ontological concepts stored in them, in order to ensure that a correct PDF document (i.e. PDF content) is placed within Modules 2 and Module 3 in the eCTD.

Our grouping reasoning mechanism, exploited in the example which resolves semantic conflicts in retrievals across heterogeneous sources, is re-usable in this case study. We mirror the role and purpose of the:

- ENT_ONT ontology from chapter 5 in eCTD ontology, and
- Local Ontologies (LO_j) from chapter 5 in the PDF ontology.

However, we also re-use the USER_INP_ONT ontology in this case study, but its purpose is slightly different. We use the USER_INP_ONT ontology in order to trigger the grouping of ontological concepts in order to place PDF documents in the correct place. Therefore, triggering grouping is equal in this case study, to specifying a link between a PDF document and a particular section within eCTD where the document belongs. Consequently, the ADDED_VAL_ONT ontology is re-used in this case study to store results of running grouping rules. This means that the ADDED_VAL_ONT ontology contains ontological individuals which are moved from the subclasses of the PDF ontology, according to grouping rules that specify the exact (and expected) content of a particular section within eCTD.

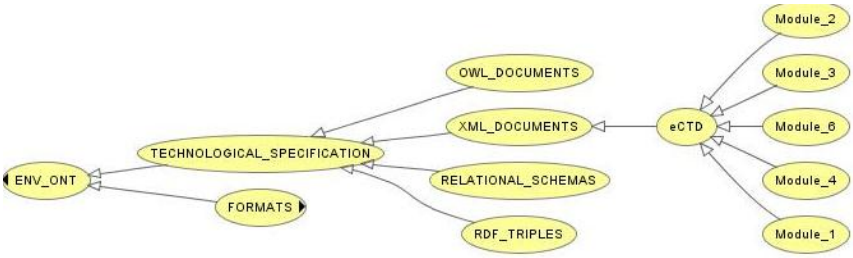

Figure 6.2 *The results of mirroring the eCTD navigational structure into the ENV_ONT*

Figure 6.2 gives us the exact ENV_ONT hierarchies, which is self-explanatory. We mirror the current XML-based eCTD navigational structure in the `eCTD` subclass of the `XML_DOCUMENTS` class of the `TECHNOLOGICAL SPECIFICATION` parent class. The `eCTD` subclass contains a number of subclasses that directly correspond to the navigation structure of the eCTD. The modelling behind the `eCTD` is guided by the content of the eCTD navigational structure and the semantics of the guidance-compliant contents of eCTD available in [301].

It is important to note that we correct our statement from bullets above. Our ENV_ONT ontology from Figure 6.2 is almost identical to the ENV_ONT ontology introduced in chapter 4, Figure 4.6. The only difference is that the ENV_ONT ontology in Figure 6.2 extends the hierarchies of `XML Documents` class in order to accommodate the semantics of the eCTD navigational structure.

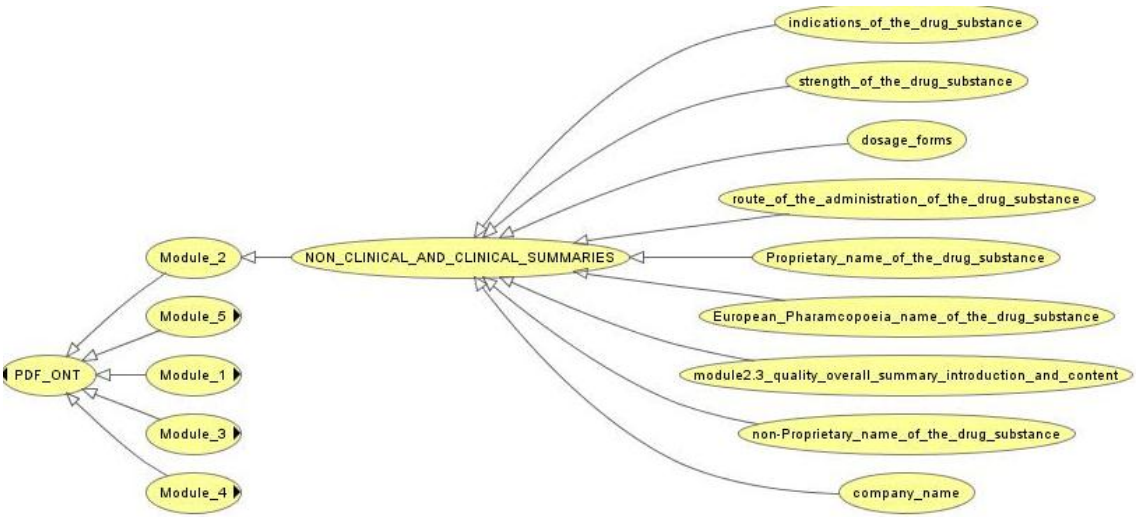

Figure 6.3 *Example of the PDF_ONT created in order to exemplify semantics of PDF documents constituting the eCTD*

Figure 6.3 gives us the exact PDF_ONT hierarchies, which is created to mirror PDF documents and their content in an ontological format. We exemplify the semantics of the PDF 2 document in Figure 6.1, which belongs to Module 2 of the eCTD, as shown in Figure 6.1. Therefore, the subclass named `NON-CLINIC_AND_CLINIC_SUMMARIES` denotes that there is a PDF document which is named *NON-CLINICAL_AND_CLINICAL_SUMMARIES* and which should be linked to the correct section in Module 32 of the eCTD. The same ontological subclass contains a number of ontological concepts that denote the 'contents' of the PDF 2 document, i.e. it contains a number of ontological individuals that make up the semantics of the 'contents' in the PDF document.

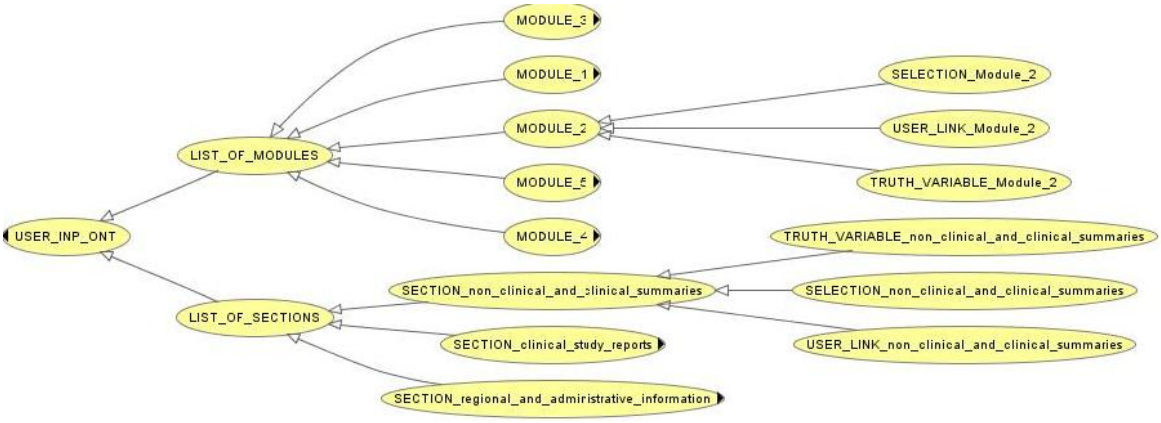

Figure 6.4 *Results of mirroring the list of 'sections' for each module in the eCTD and the list of 'contents' for each module in the eCTD navigational structure into the USER_INP_ONT*

Figure 6.4 gives us the exact USER_INP_ONT hierarchies, where we mirror the list of 'sections' and the list of 'contents' for modules in the eCTD. The modelling behind the USER_INP_ONT is guided by the eCTD ('sections') and the content of a PDF document ('contents'). The `LIST_OF_MODULES` class and the `LIST_OF_SECTIONS` class have the same role as hierarchies in the USER_INP_ONT ontology from chapter 4, Figure 4.7. The only difference is that in Figure 6.4 we establish a LINK (we are not dealing with user's clicks) which the user makes when creating eCTD (i.e. to connect PDF document to a 'section' in eCTD).

Figure 6.4 also shows that each subclass of the `LIST_OF_MODULES` and the `LIST_OF_SECTIONS` subclasses contain three further subclasses named `SELECTION`_xxx/yyy, `TRUTH_VARIABLE`_xxx/yyy and `USER_LINK`_xxx/yyy. "xxx/yyy" denotes the section/content to which these three subclasses belong to, i.e. 'xxx' may denote the modules: *module 1*, *module 2*, *module 3*, *module 4*, *module 5,* and "yyy" may denote the sections: *non-clinical and clinical summaries, regional_administrative_information*, and *clinical_study_reports.* These three subclasses `SELECTION`_xxx/yyy, `TRUTH_VARIABLE`_xxx/yyy and `USER_LINK`_xxx/yyy are essential for capturing and storing the results of "links" made by the user when trying to create eCTD.

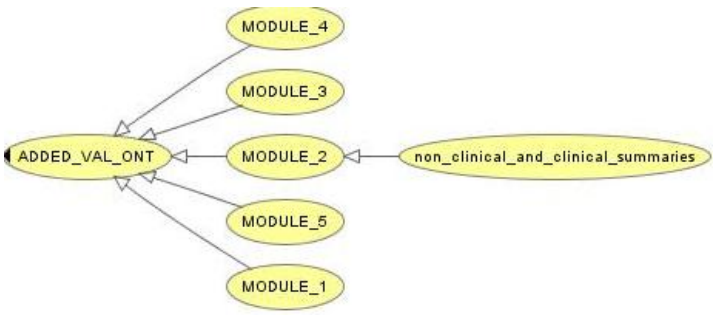

Figure 6.5 *Results of mirroring the eCTD navigational structure into the ADDED_VAL_ONT and the example of the 'non_clinical_and_clinic_summaries' subclass*

Figure 6.5 gives us the exact ADDED_VAL_ONT hierarchies, which stores the subclasses `MODULE_1`, `MODULE_2`, `MODULE_3`, `MODULE_4` and `MODULE_5`. The modelling of the ADDED_VAL_ONT concepts depends on the eCTD structure (modules) and the correct hierarchies behind each module. In

other words, the hierarchy behind `MODULE_2` class should show that its subclasses are "semantically related" to the position of Module 2 within the eCTD.

Table 6.1 *Ontological individuals that make up the correct content of the section named non-clinical and clinical summaries in module 2 of the eCTD*

- module2.3 quality overall summary introduction and content,
- company name,
- dosage forms,
- European Pharamcopoeia name of the drug substance,
- Proprietary name of the drug substance,
- non-Proprietary name of the drug substance,
- route of the administration of the drug substance,
- strength of the drug substance, and
- indications of the drug substance.

Furthermore, the `non_clinical_and_clinical_summaries` subclass of `MODULE_2` will store ontological individuals listed in Table 6.1, that make up the correct content of the section named *non-clinical and clinical summaries* in module 2 of the eCTD. Note: the hierarchies of the `MODULE_1`, `MODULE_2`, `MODULE_3`, `MODULE_4` and `MODULE_5` subclasses can be extended in order to accommodate any 'content' and 'section' within listed modules of the eCTD.

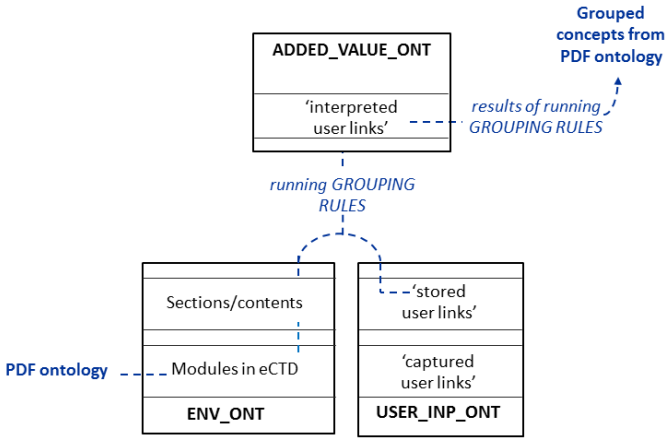

Figure 6.6 Inter-relationships between ontologies ENV_ONT, USER_INP_ONT and ADDED_VAL_ONT to secure grouping of ontological individuals from PDF_ONT

Figure 6.6 shows the inter-relationships between all four ontologies. It is similar to Figure 4.8 from chapter 4, except that we do not use *Selection* rules in this particular case study. However, in the full scale implementation of submissions for MA of medicines, we are able to completely mirror the process of capturing, storing and interpreting user's involvements, as described in chapters 4 and 5.

6.3 Grouping Semantically Related Ontological Individuals in Module 2 of the eCTD

Let us assume that a user has made a choice of working on a section within Module 2 of the eCTD, and thus he/she is required to link a particular (and correct) PDF document with the section named *non-clinical_and_clinical_summaries.* This means that the user has created the "link" between the PDF

document named *NON-CLINICAL_AND_CLINICAL_SUMMARIES* and the section *non-clinical_and_clinical_summaries* in module 2 of the eCTD. The `non_clinical_and_clinical_summaries` subclass of the `MODULE_2` class in the ADDED_VAL_ONT stores the results of running the *Grouping* rules specific to the combination of the section named *non-clinical_and_clinical_summaries* in module 2.

Table 6.2 *Grouping rule used to move ontological individuals into the 'non_clinical_and_clinical_summaries' subclass of 'Navigational Structure' class in PDF_ONT*

```
MODULE_2.3Quality_Overall_Summary_Introduction(?Y) ∧
has_Section(?X, ?Y) ∧ has_Content(?X, company_name) ∧
has_Content(?X, dodage_forms) ∧
has_Content(?X, European_pharmacopoeia_name_of_drug_substance) ∧
has_Content(?X, non_propriety_name_of_drug_substance) ∧
has_Content(?X, propriety_name_of_drug_substance)  ∧
has_Content(?X, route_of_administration_of_drug_substance) ∧
has_Content(?X, strength_of_drug_substance) ∧
has_Content(?X, indications_of_drug_substance)
→ non_clinical_and_clinical_summaries(?X)
```

The *Grouping* rule in Table 6.2 is run against the Jess engine. It uses the `has_section` and `has content` object properties to group and move ontological individuals defined in their range values from the `content` and `section` classes in PDF_ONT according to "semantically related data" indicated in the `non_clinical_and_clinical_summaries` subclass (see above Table 6.1).

Table 6.3 *The results of running the Grouping rule to move ontological individuals into the non_clinical_and_clinical_summaries subclass*

1.	PDF_1-Module2.3_quality_overally_summary_introduction
2.	PDF_2-company_name
3.	PDF_3-dosage_form
4.	PDF_4-European_Pharamcopoeia_name
5.	PDF_5-Proprietary_name
6.	PDF_6-document
7.	PDF_7-document
8.	PDF_8-administration
9.	PDF_9-strength
10.	PDF_10-indications

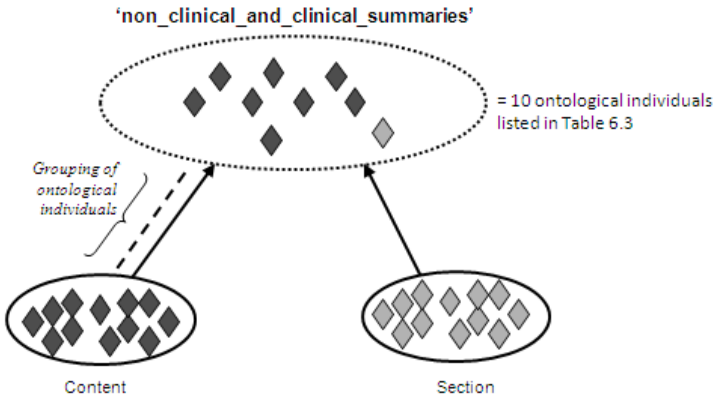

Figure 6.7 *Grouping ontological individuals from classes 'section' and 'content' into the class 'non_clinical_and_clinical_summaries' in the ADDED_VAL_ONT*

After running the *Grouping* rule in Table 6.2, 10 ontological individuals listed in Table 6.3 are moved into the `non_clinical_and_clinical_summaries` subclass of `MODULE_2` class in the

ADDED_VAL_ONT ontology. The inference created as a result of running the *Grouping* rule in Table 6.2 is graphically shown in Figure 6.7.

6.3.1 Object Properties for Securing the Correct Content in Module 2 of the eCTD

However, the grouping of ontological individuals from Figure 6.7 does not guarantee that the correct 'content' is placed under the correct 'section' in the eCTD. We use the power of OWL restrictions which allow us to create more constraints in order to secure the correct eCTD. Our set of OWL restrictions determine the set criteria for which classes are involved in a particular 'relationship' through an object properties 'domain' and 'range' values. We are in the same situation as in chapter 4, section 4.3.3.5.3 where we say that:

- ontological individuals can only be moved into a NEW ontological class if they meet the set criteria for being a member of that NEW ontological class;
- if at least one ontological individual within a particular class does not meet the set criteria for being grouped into a particular class, then NO ontological individuals are moved from that class.

Therefore, in this case study, we model the object properties `has_section` and `has_content` in order to create a relationship between the `non_clinical_and_clinical_summaries` class in the ADDED_VAL_ONT ontology and the `Section` and `Content` classes of the PDF_ONT ontology. The domain for both the object properties is set to the `non_clinical_and_clinical_summaries` ontological class, in order to specify where to move ontological individuals into. The range for the object property is set to the names of the `Section` and `Content` classes, in order to specify where to move ontological individuals from (i.e. to move the exact semantically related ontological individuals that make up the section named *non-clinical_and_clinical_summaries*).

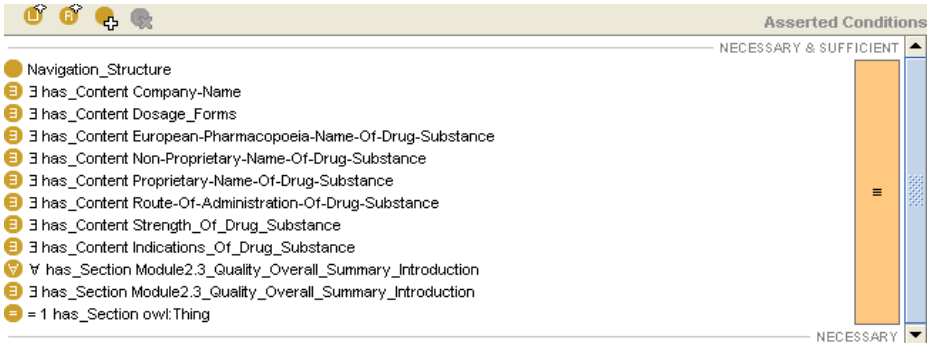

Figure 6.6 *OWL restrictions that determine the set criteria for 'non_clinical_and_clinical_summaries' class membership in the ADDED_VAL_ONT*

Figure 6.6 shows a set of OWL restrictions applied to both the object properties `has_section` and `has_content` that determine the set criteria for `non_clinical_and_clinical_summaries` class membership. We use two OWL restrictions:

- the existential restriction (∃) is used to describe that the `non_clinical_and_clinical_summaries` ontological class has **some** ontological individuals from the `Section` and `Content` class.
- the universal restriction (∀) is used to describe that the `non_clinical_and_clinical_summaries` ontological class has **only** ontological individuals from the `Section` and `Content` class.

Both restrictions are made 'necessary and sufficient' conditions to imply the concreteness of the `non_clinical_and_clinical_summaries` class.

6.4 Summary

In this chapter we have illustrated how the ontologies ENV_ONT, USER_INP_ONT and ADDED_VAL_ONT from our proposal, can be used in a completely different environment, when managing the correct content of applications for MAs of medicines. We have reused our grouping reasoning mechanism, defined in the process of resolving semantic conflicts in heterogeneous data repositories, and applied it to a completely different purpose. However, the similarities are in the way we (a) manipulate user's involvements and (b) run *Grouping* rules in order to create semantic relationships between ontological individuals. The output of using the grouping reasoning mechanism in this case study is that we can guarantee that the correct content of eCTD is submitted as a part of MAs procedures. In other words we:

- establish the number of semantically related 'sections' and 'contents' within the eCTD through user's involvements, and
- eliminate the numbers of errors occurring when submitting the wrong content of PDF document within the eCTD, by using the grouping rules as a reasoning mechanism which connects the correct content of PDF documents to their correct section within the eCTD modules.

Chapter 7

Conclusions

In this thesis, we have carried out research on resolving semantic conflicts, which arise from semantic heterogeneities triggered by retrievals across heterogonous data repositories in PCEs. We touch upon many topics, ranging from looking at the history of resolving semantic conflicts and problems associated with interoperability in software systems since the early 90s, to the use of Semantic Web technologies and OWL/SWRL enabled ontologies in particular, for the purpose of making the semantic of heterogeneous data explicit. We were particularly interested in resolving semantic conflicts through the power of OWL, mappings of their concepts and reasoning upon them through SWRL in order to achieve semantic interoperability.

In retrospect to resolving semantic conflicts through generations of software systems, we have become aware that achieving semantic interoperability in 2010 still remains a very complex task. Past solutions to resolving semantic conflicts have either failed or been short lived. We have talked about migrations, federations, global multi-database schemas, mediations and many other old fashioned solutions to database interoperability, which very often sacrifice the autonomy of data centric software systems (i.e. databases) and evolution of their individual elements. Loosing the autonomy of data repositories in modern software systems and changing their original semantics are undesirable outcomes when resolving semantic conflicts. This is because modern software systems rely on data created on an ad-hoc basis, stored in unpredictable locations in various formats and hosted by wireless and mobile technologies for securing their persistence. Autonomy and evolution of modern software systems cannot be sacrificed for the sake of removing heterogeneities across them.

The emergence of Semantic Web technologies with ontologies/mappings/reasoning, has motivated us to address semantic interoperability from a different perspective. We have streamlined our research towards an ontology based and layered SA, which supports retrievals across pervasive software systems, which are heterogeneous in their nature, and achieve semantic interoperability without sacrificing the level of data sharing and autonomy of their individual participating data repositories.

We give an example of a software application which illustrates and tests the proposed SA. We demonstrate the process for resolving semantic conflicts across heterogeneous data repositories and achieving their semantic interoperability by using ontological layering. The SA is deployable within environments created by component and Semantic Web technologies.

In this chapter we summarise the research of this thesis in section 7.1 and evaluate our proposal in section 7.2. We look at the research objectives and comment on our results. We also highlight the uniqueness of our ontological solution, highlight the additional research outcomes that have not been anticipated in our research objectives and compare our solution to similar approaches. In section 7.3 we reflect upon our research results by commenting on the complexity of computations in our proposal and impact of technologies, which pose interesting challenges for future approaches to using OWL and SWRL enabled ontologies in software engineering. Section 7.5 outlines our future works.

7.1 Research Summary

In chapter 2 we introduced the problem of resolving semantic conflicts that become an obstacle in achieving interoperability in modern computational environments, characterised by their pervasiveness and the need to share data and information available within them. We have agreed that semantic heterogeneities are inherent in such systems, but if we really want to address the interoperability problem in 2010, we have to go back 20 years, and analyse heterogeneities through various generations of software systems, which have led towards recognition of semantic conflicts. We outline that semantic heterogeneities trigger a variety of semantic conflicts that are concerned with the disagreements in the implicit meanings, perspectives and assumptions made during the creation of computational models and data repositories. This is still very much a problem in today's computational environments, regardless of which type of data repositories we create today. The problem is further aggravated by the fact that we need to guarantee ***meaningful data sharing*** across modern software systems, whilst preserving the autonomy of participating data repositories. Therefore, Semantic Web technologies might hold the answer if we wished to manipulate the meaning in our heterogeneous world. If Semantic Web technology enables us to support machine processable meaning of information over the WWW by providing a formal description of concepts, terms, and relationships within URLs, Web sources and their content [234, 302 and 303] then, the same technology should be exploited in our attempts to guarantee ***meaningful data sharing*** across modern software systems.

In chapter 3 we analyse all the works in resolving structural and semantic conflicts in the DB interoperability field since the early 90s and outline their benefits and drawbacks. Most of these works fail to address the complex nature of semantic conflicts, their impact on data sharing and the way they can preserve autonomy of data repositories. However, methods and approaches to resolving semantic conflicts based on the use of Semantic Web technologies do help us to move away from integration or centralisation in DB systems and mediations in software systems which were prevalent in the 90s. Therefore, we review examples of ontological modelling, mappings and reasoning in order to outline a new era in resolving semantic conflicts. We expect that all these "ontological" solutions will preserve the autonomy of data sources without affecting the level of data sharing and will ultimately achieve semantic interoperability.

In chapter 4 we propose a SA based on ontological layering which supports retrievals from various data repositories and resolves semantic conflicts which arise from heterogeneities inherent in them. Ontological layering is in the core of our SA, which contains different software architectural components in different layers. However, our core ontological layering is triggered by the semantic stored in the users

request for retrievals across heterogeneous data repositories. Therefore, there is a significant difference between core ontological layering, which generates Go-CID and ontologies stored within the User Request layer of our SA solution, which are responsible for capturing, storing and interpreting user's requests. They have different purposes in terms of:

(i) preparing semantics from the user's involvements in retrievals across heterogeneous data repositories in order identify semantically related data in the User Request layer,

(ii) resolving semantic conflicts, as a consequence of the existence of semantically related data, through ontological mappings in core ontological layering.

Each layer in the SA is generated through different reasoning mechanisms based on the execution of a chain of SWRL rules which enable reasoning upon the result set of the reasoning in adjacent/lower layers.

However, the existence of semantically related data, points towards the complexities in identifying semantic conflicts, their types and occurrences. To address all we use our own classification of semantically related data and their degrees of similarities that generate particular types of semantic conflicts. Therefore, ontology mappings used in (ii) are created through our specific reasoning mechanisms, guided by our classification of semantically related data and categorisation of semantic conflicts. The relationship between:

(a) the classification of semantically related data and their degrees of similarities, and

(b) the way we resolve semantic conflicts

are defined in our ontology mappings performed in the core ontological layering.

The process for resolving semantic conflicts consists of 8 steps. Steps 1-5, equivalent to (i) above, "prepare" the semantics essential for creating and deploying core ontological layers. This includes initiating the lowest ontological layer (Local Ontological layer) through the translations of the content and structure of available heterogeneous Data Repositories $\{Rep_i \mid i = 1, ... m\}$ into Local Ontologies $\{LO_j \mid j = 1, ... n\}$, and the ENV_ONT ontology. We also model user's involvements in the USER_INP_ONT and ADDED_VAL_ONT ontologies. The role of these two ontologies is to interpret the user's request for retrievals and create a context within which we can identify and resolve semantic conflicts. This allows a user to specify what is expected from heterogeneous repositories and which information from them is relevant for the retrieval.

Steps 6-8, equivalent to (ii) above, illustrate the exact way of resolving semantic conflicts through core ontological layers, which are dynamically generated from LO_j through a set of specific ontological mappings and reasoning: Target Ontologies $\{TO_k \mid k = 1, ... p\}$ are generated through ontological alignment; Derived Ontology $\{DO_g \mid g = 1, ... q\}$ through ontological integration and the final layer is a consequence of a merging all $DO_g s$ into the final Go-CID. Thus, Go-CID ontological concepts do not contain semantic conflicts, which have been carried forward from heterogeneous data repositories into local ontologies LO_j.

In chapter 5 we illustrate the implementation of our generic SA proposal from chapter 4 through a case study in the domain of pervasive healthcare [304, 305, 306 and 307]. The advances of wireless and mobile computing, and proliferation of pervasive healthcare technologies have made a huge impact on how we create healthcare computational spaces and software applications in them. We depend on enormous amounts of information and data stored in a variety of forms: from highly structured database records to multimedia data streams of medical images, which are expected to be shared across various

operational environments. However, when performing retrievals of information and data across such heterogeneous environments, we may face a number of semantic conflicts that may become an obstacle in creating the correct results of retrievals. The healthcare domain appears to be one of the best examples of heterogeneities in modern software systems, considering that records about patient(s) are created often on an ad-hoc basis, outside the patient's GP environments, stored in data repositories which do not have to be traditional databases and designed/modelled so differently that we may not be able to see that they store data with the same meaning. One of the most common problems is 'patient records' which may have different meaning in different healthcare software systems and store different data about the same patient. Therefore, we have had no problems in creating a healthcare environment in chapter 5 which shows semantically related data with all degrees of similarities introduced in chapter 4. We have also been able to show through the same case study all the types of semantic conflicts we categorise in chapter 4. Therefore, we have been in a position to illustrate and give a detailed description of the exact steps of our process for resolving semantic conflicts in chapter 5.

We start with a request issued by a medical professional, which will require retrievals across available data repositories *GP_data_rep*, *Hospital_data_rep*, *Clinic_1_data_rep* and *Clinic_2_data_rep* in order to obtain healthcare summary for a particular patient. We give the content and structure of semantics stored in data repositories, and the way we translate them into local ontologies *LO_gp*, *LO_hospital*, *LO_clinic_1*, *LO_clinic_2*, which immediately resolves the Mispelt/Case-Sensitive semantic conflicts, if they exist. We also demonstrate how the medical professional's input in terms of specifying what he/she needs for obtaining a healthcare summary for a particular patient will help us to find out exactly where other semantic conflicts exist, and how we will resolve them. By interpreting the medical professional's input through our *Grouping* reasoning mechanism, we resolve the Homonym semantic conflict. The Aggregation and Synonym semantic conflicts are resolved through *aligning* local ontologies *LO_gp*, *LO_hospital*, *LO_clinic_1*, *LO_clinic_2* into target ontologies *TO_1-TO_10*. The Generalisation, Specialisation, Isomorphism and Union Incompatibility semantic conflicts are resolved by *integrating* target ontologies *TO_1-TO_10* into derived ontologies *DO_1-DO_10*. Finally, we *merge* derived ontologies *DO_1-DO_10* into the final Go-CID in order to prepare data for retrieving a healthcare summary for a patient, which contains no semantic conflicts.

We test our proposal through a full scale implementation of a software application built in NetBeans 6.4 IDE, which manages the retrievals and triggers ontology mappings and reasoning. We use Oracle for creating healthcare databases and the DataMaster plug-in in the Protégé 3.4 ontological development environment for translating databases into OWL ontologies. We also use Protégé 3.4 for the creation of ontological concepts. We use the SWRL tab plug-in in Protégé 3.4 for creating SWRL rules and the JESS engine for running computations (SWRL rule chaining – connection created through the SWRL rule bridge in OWL API). We use OWL API for connecting to Protégé 3.4 from NetBeans 6.4 IDE and JSP and Servlets used for managing the software application.

In chapter 6 we give another case study which illustrates how a section of our process for resolving semantic conflicts can be used in a completely different domain of managing submissions for MA of medicines [300]. Therefore, we use ontologies ENV_ONT, USER_ONT and ADDED_VAL_ONT from our SA proposal and deploy our *Grouping* reasoning mechanism [46] to ensure that correct content of a PDF document has been placed at the correct position within the eCTD for MAs. The evidence of

reusability of our ontologies and *Grouping* reasoning mechanism, which manipulate the meaning of user's involvements when resolving semantic conflicts, is striking. The role of the user in this case study is the same as in our example for resolving semantic conflicts. We capture, store and interpret user's involvements in both case studies in the same way/manner, PLUS we use the same *Grouping* reasoning mechanism and type of OWL restrictions to move semantically related ontological individuals in both case studies. In this chapter we also list 3 more case studies, from another set of domains [292, 292, 297, 298 and 299], which show a high level of re-usability of our SA and its reasoning mechanisms, introduced in chapter 4 and illustrated in chapter 5.

7.2 Evaluation

7.2.1 Achieving Research Objectives

In this research, developments have been made towards the:

- understanding of the complex nature of semantic heterogeneities in software systems and the analysis of limitations of past and current approaches to resolving semantic conflicts when addressing the problem of semantic interoperability;
- investigating the importance of ontologies, their mappings and reasoning when using OWL/SWRL enabled ontologies as an instrument in resolving semantic conflicts;
- creating and implementing a SA and a specific SA style based on ontological layering which secures the implementation of our proposal and resolves semantic conflicts.

Therefore, the major part of the research has been dedicated to building ontological and layered software architectural components. Their main purpose is to accommodate a software solution which will resolve semantics conflicts, triggered by the existence of semantically related data when performing retrievals across heterogeneous data repositories. Our first research objective has been realised in chapter 3: we have come to a conclusion that ontology mappings and reasoning mechanisms defined upon ontological concepts are feasible and deployable within our architectural layers by using Semantic Web technologies.

Our second research objective has been met through the proposed ontological layering in chapter 4. Our SA which accommodates ontological layering and Go-CID, as its final result, supports retrievals in modern software systems and resolves semantic conflicts which arise from heterogeneities inherent in them.

The third research objective has been satisfied in chapter 5 through a detailed case study of retrievals across data repositories in pervasive healthcare environments. We have proved that it is feasible to achieve semantic interoperability through the proposed ontological layering in our SA. We deploy our software architectural components and its ontological layering to build a software application upon it, using Semantic Web technology and NetBeans 6.4 IDE. However, in chapter 6, we show that there are extensions to our research results, which we have not anticipated in the research objectives. We have discovered a high level of reusability of our ontological reasoning mechanisms, which are either ontological groupings or mappings, and which can be used in different problem domains and in environments, in cases when we need to:

(i) establish if and when we have overlapping "semantics" which creates relationship between data/information in such domains/environments, and/or

(ii) infer and/or assert a correct set of "semantics" which can support any decision making required in such domains/environments.

The uniqueness of our SA which accommodates ontological layering is in:

- Using the ontologies ENV_ONT, USER_INP_ONT and ADDED_VAL_ONT in manipulating the understanding of the environments where heterogeneous data repositories reside and using the power of user's involvement in retrievals across these repositories. Subsequently, the ontologies ENV_ONT, USER_INP_ONT and ADDED_VAL_ONT create the context of the interoperability being attempted and determine what actual ontology mapping is left to the core ontological layering of Go-CID.
- Using the core ontological layering which is based on a set of specific ontological mappings and reasoning performed upon ontological concepts from LO_j, TO_k, DO_g, and Go-CID, in resolving semantic conflicts and achieving data sharing and semantic interoperability in any heterogeneous environment.
- Leaving the heterogeneous data repositories intact in terms of not changing their format and semantics stored in them, i.e. preserving their autonomy, and dealing with semantic conflicts on an ad-hoc basis through ontological layers, by exploiting the meaning of user's requests for retrievals and the knowledge of the environment where retrievals take place through inference mechanisms.

7.2.2 Contribution

The main contribution of this research is in five areas:

1. Gaining knowledge and awareness of the complexity of the interoperability problem, which is still prevalent in software systems today, characterised by pervasiveness of computing environments and heterogeneities inherent in them. It has also become evident that we should strive for heterogeneities in modern software systems and achieving a high level of interoperability by using traditional approaches such as federation, mediation and global schema solutions, might not be desirable software solutions for data intensive the software systems in the 21st century.
2. Mastering the modelling and implementation of software applications, based on a specific software architectural style which accommodates ontological layering. This means that the building of a new era of software applications dependent on OWL/SWRL enabled ontologies allows us to understand and manipulate the semantics of and meaning stored within heterogeneous data repositories, and thus achieving our ultimate goal of resolving semantic conflicts when performing retrievals across them.
3. Delivering a unique approach to addressing the problem of semantic conflicts (*Semantic Interoperability*!) in modern software applications through our own specific way of creating semantic models (*ontological layering*) and imposing reasoning (*SWRL rule chaining*) upon their modelling concepts in order to identify and resolve semantic conflicts.
4. Classifying the way of creating and manipulating concepts stored in OWL/SWRL enabled ontologies in order to perform ontological mappings and store/manipulate the inference created

within them. We use our own way of classifying semantically related concepts for the purpose of identifying different types of semantic conflicts and our own way of classifying reasoning mechanisms for the purpose of resolving the identified semantic conflicts.

5. Delivering a novel approach of exploiting the meaning of user's requests for retrievals across heterogeneous data repositories for the purpose of understanding expectations of the user, and triggering ontological layering. In other words, users inputs, in the form of their requests, dictates the context in which semantically related concepts from data repositories are compared in order to identify and resolve semantic conflicts through ontological layering. Consequently, the manipulation of the meaning of users request becomes essential for the managing of software applications which secure correct results of retrievals across heterogeneous data repositories.

7.2.3 Comparison to Similar Approaches

The comparison of our solution with similar approaches available in academia and industry is divided into two parts. We firstly look at old fashioned approaches to the interoperability problem which have existed since the early 90s and compare them with our own SA solutions for resolving semantic conflicts. Secondly, we look at the solutions in the ontological world and approaches which use ontologies for explicitly dealing with the problem of resolving semantic conflicts and ontological mismatches. We compare them to our own way of ontological mapping and reasoning.

The issue of interoperability in software systems has been in the focus of the software engineering and database communities since the late 80s. It is important to note that the interoperability problem does not exist per se, i.e. it is always triggered by heterogeneities which exist within and between software systems. In other words, we can not talk about interoperability if we do not specify exactly what is heterogeneous. Consequently, heterogeneous data structures, data models and technologies in the database communities of the late 80s initiated research on semantic interoperability because they were concerned with conflicting data in databases and interpretations of their meaning [10, 19, 20, 21, 22, 32, 44, 77, 78, 79, 80, 81, 82 and 83]. A direct answer to this initiative appeared to be federated [13 and 73] and global schema approaches [177, 252, 256 and 257, 232] which had in mind a certain level of integration of either data structures or data models in order to remedy heterogeneities. Furthermore, the emergence of object technologies in the early 90s highlighted different types of heterogeneities which were consequences of technology impact on database communities and software engineering in general. We witnessed various migrations between database systems [25, 26 and 27] because we faced different types of interfaces and query languages, different platforms within which database systems operated, on top of the traditional problems of heterogeneities at the data and schema level, which persisted since the 80s.

All these solutions, federations, global conceptual schemas and migrations, which were trying to address the issue of interoperability, had always ended with a certain level of integration of either data structures or data models in order to remedy heterogeneity. This means that even in flexible federated database systems, we ended up with solutions which sacrifice autonomy of databases and systems we built around them, in order to have a global or integrated view of heterogeneous and conflicting data. This is a serious drawback, because the database autonomy plays a very important role in database

systems. Consequently, there were not too many commercially available federated database management systems which had been adopted by the database community and industry in the 90s.

The ideas of mediation in software systems [17, 28, 29, 30, 31, 32, 33, 34 and 258] and the impact of middleware and component based software development (CORBA[88], JEEE[89], DCOM[90]) have given a slightly different approach to resolving heterogeneities in software systems. They do not always offer integration of conflicting data as the answer to their heterogeneity. However, the problem of having heterogeneities at the data and schema a level, so prevalent in database systems from the late 80s onwards, has now escalated to all other dimensions of computing environments, and the issue of semantic conflicts across heterogeneous data repositories has been slightly marginalised since the late 90s. Therefore, component based and service oriented software architectures [10, 74 and 75] have been dealing mostly with heterogeneities of platforms, applications, interfaces, data accessing mechanisms, programming procedures, software components and similar.

We have learned from the research results in the interoperability field and accompanied solutions from industry since the early 90s that we will have to:

1) avoid any type of integration in our solution because we need to maintain the autonomy of data repositories and structures which store important semantics and meaning of real world concepts they model;
2) allow the evolution of heterogeneous repositories at the place where they originate, therefore any changes in structures and content of data repositories should happen independently from any interoperability solution;
3) keep original data sources and their structures intact and built component based solutions on the top of them which will deal with every type of heterogeneities we may have;
4) allow user impact, i.e. empower users to specify what they expect from heterogeneous systems and participating data repositories and exploit user's requests, their knowledge and inputs when trying to resolve conflicting data in heterogeneous data repositories and interpretations of their meaning.

None of the existing examples in our related works (chapter 3, section 3.2.1) exhibit characteristics listed in 1) - 4) above. However, in our solution we have decided to use some of the ideas from Sheth and Kashyap [44] in order to classify semantically related data and their degrees of similarities, for the purpose of identifying and resolving semantic conflicts, as described in chapter section 4, section 4.2. We have also decided to propose a specific SA style based on components and layering, which has proved to address the complexity of modern systems and deal with certain types of heterogeneities which are prevalent now, in the 21st century [10, 74, 75, 286 and 287]. Consequently, our main principles behind our proposal are in a)-d) below:

a) Our SA proposal and its particular layered architectural styles secures flexibility in terms of isolating data repositories and semantic conflicts which exist across them, from the software solution which resolves them. This means that we can build an "n" number of solutions upon the same set of repositories in order to address a different set of semantic conflicts, which are associated

[88] http://www.corba.org/
[89] http://www.oracle.com/technetwork/java/javaee/tech/index.html
[90] http://www.microsoft.com/com/

with the way we issues request for retrievals across these repositories, which in turn become independent from the software solutions built upon them.

b) The layering in our SA secures a mechanism of addressing different types of semantic conflicts at a different layer, i.e. we systemise the way we resolve semantic conflicts through layering, which is given by layered SA style. The benefits are twofold: we will always know which conflicts will be resolved at which layer – thus we will be allowed to choose and skip layers if needed, and we will always know in advance which mechanism for resolving conflicts are available. Our solution is not a "bundle" which is static and sits on the top of heterogeneous data repositories. It is a dynamic solution implemented through different layers and different instances of layering, dependent on the nature of user's requests for retrievals. Without such layering we would not be able to achieve flexibility and address a variety of user's requests.

c) User's request for retrievals dictates a context in which conflicting data in heterogeneous data repositories, and different interpretations of their meaning, may occur. Therefore, these requests are a driving force behind a particular instance of layering in our SA. In other words the user has a significant impact on the way we identify and choose to resolve semantic conflicts though a particular instance of our layering.

d) Our SA consists of software components which are generated ad-hoc, i.e. as soon as requests for retrieval have been issued. This means that a set of layering is created every time when request appears and the set of layering from the previous retrievals is deleted as soon as its results have been displayed (i.e. Go-CID's content becomes reverted to its previous "empty" state). This means that there is no burden on the existing repositories or the application built upon them, in terms of remembering or storing any intermediary results when resolving semantic conflicts.

e) Our SA does not include any integration of the original data sources (structures and values) into any other format. This means that they remain intact – to allow for database autonomy and evolution to take place. We create mirrored copies of database schemas, or any other types of repositories, and their contents in order to exploit their semantics they store and resolve possible semantic conflicts! Therefore our base SA layer i.e. the bottom most layer, consists of a set of ontologies which are translation from underlying data repositories, mirroring all underlying persistent data and their structures.

In the late 90s, at the same time as the Semantic Web initiative took off, ontologies came again into the focus of interest of research communities in terms of providing formal explicit specification of a shared conceptualisation. Apart from using formal ontologies in order to create shared vocabularies and shared domain models, we started witnessing the creation of ontological solutions which try to address semantic interoperability between heterogeneous software systems. With the standardisation of RDFs and OWL/SWRL we have discovered the enormous power of semantic Web Technologies in terms of providing a means to deal explicitly with the problem of semantic conflicts [151, 153, 170, 172, 173, 174 and 175].

There were two pre-dominate ontological approaches to addressing heterogeneities in database and information systems, used by software engineers and database communities.

The first approach relied on the use of a single ontology in the form of either: a global schema [15, 142 and 251] or a shared vocabulary [47, 48, 49 and 50]. It is obvious that global ontological schema

approach requires integrating either data structures or data models into a single ontological schema in order to provide a homogenous view of conflicting data. This is very similar to Global Schema approaches in databases from t he early 90s, but without ontologies! Shared vocabularies required the explicit description and resolution of semantic conflicts through the use of domain specific knowledge, and mapping knowledge, or rules, into a single ontological schema which becomes a shared vocabulary. In both cases we deal with single ontologies which require either (a) a certain level of integration in order to mask heterogeneity or (b) additional mappings between ontological ontologies and heterogeneous systems. In both cases, we end up in highly specialized translations, which address a range of heterogeneities from all databases into a single ontology, thus being not flexible and expensive solution in terms of both time and money.

The second ontological approach relied on the use of multiple ontologies in terms of describing each databases though its own ontology, commonly referred to as source/local ontology. Semantic conflicts were resolved through either the process of ontology based *semantic matching* between source ontologies [153 and 173], or the process of ontology based *semantic mapping* of source ontologies into a domain (or upper) ontology [51 and 52]. By using multiple ontologies (i.e. the use of source/local ontologies to represent underlying heterogeneous databases), software engineers and database communities were able to avoid the complexity and overheads of integrating databases. However, the manipulation of multiple ontologies proved to be a difficult job, because it was extremely difficult to deal with heterogeneities, which were transferred from local repositories into their ontologies. At the same time the problem of ontological mismatches and their semantic interoperability (as a consequence of the development of Semantic Web) became evident once again. However, this time it was not within the software engineering and databases communities, but within the ontological engineering world because they were concerned with heterogeneous ontological descriptions, ontological models and technologies and ontological interpretations of their meaning [54, 69, 247, 248, 249, 250 and 251]. To that end, the process of *ontology mapping* is based on any of the following three mappings [57, 59, 249, 262 and 263]:

- *schema mapping* between heterogeneous ontologies,
- *instance mapping* between heterogeneous ontologies, or
- *hybrid mapping* that uses a combination of both schema and instance based ontology mappings.

However, the need to consider multiple ontologies in environments in which different views and interpretations of ontological data (e.g. different aggregation and granularity of the ontology concepts) raise the question of semantic similarities between ontological concepts, which brings forward semantic conflicts in ontologies, as it did in any type of heterogeneous databases in the late 80s.

Our proposal uses multiple ontologies when resolving semantic conflicts and their concepts are manipulated through ideas given in schema and instance mappings. However, we had to systemise and customise mapping mechanisms in order to (i) resolve different types of semantic conflicts at different ontological layers and (ii) follow our own classification of semantic conflicts and their degrees of similarities. Furthermore, we exploit the power of OWL/SWRL enabled ontologies which in turn had also dictated the way of performing the ontology mappings by creating an instance of our layering through SWRL rule chaining.

We have not found in the research community and industry any solution which uses both: ontological layering, as a core structure of a SA style for resolving semantic conflicts and SWRL rule chaining to perform ontology mapping which eventually will resolve semantic conflicts.

Furthermore, sources like [57, 59, 249, 262 and 263] rely always on natural language processing techniques, algorithms and knowledge bases to create logical inferences in their ontologies and never exploit the power of SWRL rule chaining to achieve a flexible solution in terms of the manipulating ontological individuals, which is very powerful and available mechanism in OWL/SWRL enabled ontologies. The same sources also tend to use formal ontologies which have no place within our ontological layers. This is because our ontological layers are created on an ad-hoc bases, for a particular purpose (user's requests), they are very short lived and controlled by a software application, i.e. we offer a software engineering solution based on OWL/SWRL enabled ontologies, dictated and controlled by user's requests for retrievals. Our ontologies are very often small, automatically generated though SWRL rule chaining and easily accessible by any software application generated from our proposed SA. However, the deployment of software components from the proposed SA is dictated by technologies from NetBeans IDE, to its bridges to the JESS reasoning engine and Protégé ontological editor through the OWL API library.

At the time of writing this thesis, we have also not found any published work which exploits the power of OWL/SWRL ontologies by deploying them through component based technologies and their IDEs. However, the performance of our software applications generated from the proposed SA is satisfactory from the software engineering perspective, which has opened doors for commercial exploitation of our solution to resolve semantic conflicts in heterogeneous data repositories.

Finally, during the course of this research we had an opportunity to juxtapose our SA to one completely different architectural model named Context-Aware Data Retrieval Architecture (CADRA) which was designed for a similar purpose: sharing data and information across heterogeneous e-health systems [313]. It appeared that, when summarising the main technical characteristics of CADRA and our SA which accommodates Go-CID, we could see how the differences in creating CADRA and our core ontological layers, and similarity/differences of computations and communications within them, paved the way towards two completely different solutions, which are addressing the same problem in the e-health domain. However our solution compared to CADRA supports heterogeneous environments and allows any number of data repositories to be included into any instance of our SA. Our ontological solution is also dynamic, i.e. a set of ontological layers is created as soon as a request is issued for the retrievals across the heterogeneous data repositories. Therefore our core ontological layering is constantly changeable and responds to the semantics stored in user's requests, as a part of user's involvements in retrievals across heterogeneous software environments.

In summary, there are no published research in academia and industry which can be compared with our idea of using a specific SA style based on ontological layering, and performing SWRL rule chaining, for the purpose of creating ontological layers, which in turn will resolve semantic conflicts as the result of request for retrievals across heterogonous data repositories.

7.2.4 Lessons Learnt

We would like to point towards a few limitations or concerns which are results of our research. We itemise them in paragraphs below.

Automation in our ontology mappings

We wish to draw reader's attention to the way we automate the creation of ontological layering, i.e. mappings and reasoning. Automatically deriving ontology mapping at runtime, without having human involvements, is generally considered impossible [250 and 312]. Therefore, one of the biggest challenges in our work has been to identify a process that will maximize the automation of ontology mappings at runtime. To that end, we have a runtime solution that uses user's involvements (through reasoning upon the ontologies ENV_ONT and USER_INP_ONT and ADDED_VAL_ONT) to define a specific set of core ontological layers, i.e. creation of LO_j, TO_k, DO_g, and the final Go-CID. However, though we have demonstrated the potential power of ontologies in performing automated resolution of semantic conflicts, the automation is still based on the following factors:

- the availability of participating heterogeneous data repositories that are willing to expose their semantics, i.e. subscribe to our SA which supports retrievals accommodates ontological layering and Go-CID software applications;
- the enumeration of every possible combination of available data repositories and information types stored within them in the ontologies ENV_ONT, USER_INP_ONT and ADDED_VAL_ONT;
- the level of human intervention in identifying semantically related concepts and semantic conflicts triggered by them, during steps 1-5 in our process for resolving semantic conflicts, i.e. the amount of human intervention in preparing semantics for core ontological layering.

Classification of semantically related concepts and their identification

By placing the classification of semantically related concepts, and the various types of semantic conflicts they trigger, outside our SA layering ensures the dynamic resolution of semantic conflicts. This means that, our SA does not use an ontology that describes the semantic conflicts; instead it uses a classification of semantically related data to identify semantic conflicts and ultimately guide us in the creation of ontology mappings (including the assumptions modelled into SWRL rules). However, the preparation of semantics for core ontological layering relies heavily on human intervention and a number of technological dependencies, all of which suggest potential deficiencies in our approach. We list them below:

- The question of identifying semantically related data and the various types of semantic conflicts they trigger remains to be a manual process in our approach. The use of our classification of semantically related data is dependent on obtaining an understanding of the semantics contained within underlying heterogeneous data repositories and the semantics carried forward from ontological schema and their content (i.e. ontological classes, properties and individuals), in order to judge how semantically equivalent overlapping concepts are. Therefore, our approach can be seen to create a burden on the developer with regards to the following three tasks:
  - *The task of understanding and identifying semantically related data*. Notably, attempts in automating this process may involve additional extraction mechanisms, that similar to

algorithms that can determine semantic similarity and natural language techniques that perform ontology matching. Subsequently, such extraction mechanisms may reduce the time it takes in understanding overlapping ontological semantics. However, the need for human intervention still suggests an ongoing commitment to building tools and wizards such as Cupid [58] and Chimaera [61] to ease the process for identifying overlapping semantics.

- *The way in which our approach uses the degrees of similarity.* Our classification of semantically related data and various semantic conflicts they trigger, give rise to deficiencies in similarity thresholds, i.e. how much is necessary to qualify as enough to ignore, tolerate or resolve semantic conflicts. Again, as the dependency lies on the human understanding of semantics carried forward into local ontologies, similarity thresholds are not clear and have yet to be determined. Similarities thresholds such as weighted keyword proximity where the 'subject' of a word is mapped to a similar concept can help to guarantee that degrees of similarity can be measured according to the nature of overlapping semantics.
- *The dependency on ontology translations techniques*. Translation techniques such as DR2Q mapping [226] or Datamaster [227] dictate the way semantics from heterogeneous data repositories are mirrored in the ontological world. Although, we always have the option of adding additional semantics to local ontologies, once they have been generated as a result of translations, we are still be faced with the issue of burdening the developer with regards to the task of understanding original semantics, and the prospect of how they could be mirrored in the constraints of ontological models. The option of creating universal ontology translation techniques is a daunting task and will have to be considered in order to produce desired results of these translations.

Finally, as generic observations, users of our ontological and layered SA are diverse and prone to changing interests rapidly, therefore it is impractical for our SA to capture all possible data expectations of the end user**s**. However, by addressing our concerns in the paragraphs above we will probably achieve a balance between the semantics stored in various data repositories and user requests which actually trigger the problem of semantic heterogeneities.

In the next section we continue with the evaluation of our work, by reflecting upon aspects of our solution which shows a pathway of the development of our solution, and our experiments which have had an impact on the final version of our SA. We also reflect upon the impact of technologies on our proposal and solution, and address the reader's interest in how well the solution may perform in terms of its complexity and computational power.

7.3 Reflections

In our early attempts to create a generic SA which can accommodate our ontological layering for resolving semantic conflicts, we concentrated on the final layer, named Go-CID, more than on anything else [308]. This was in early 2007. We were more concerned if the "context awareness" [182] has an equal role and power as data sharing and interoperability within PCEs. Therefore, we insisted on taking "context awareness" into account at any stage of the process of achieving interoperability, when performing retrievals across heterogeneous data repositories. However, it had become very obvious in

2008 that we had to replace "context" with "situations" in modern computing [309, 310, and 311]. This is because all our ontological layers which were created on an ad-hoc basis, generated nothing else but a "SITUATION", once a request for retrievals across heterogeneous data repositories had been issued. Therefore, there was no need to insist on modelling "context awareness" as a separate dimension of either our SA or retrievals it supports. However, we kept "C" within the name of Go-CID to remind the reader that we have managed to replace the explicit modelling of context awareness, with a "situation" through our ontological layering. In other words an instance of ontological layers in our SA is equivalent to a "situation" in PCEs.

- The names of ontological concepts in Go-CID, which is the final result of layering (i.e. the result which contains no semantic conflicts), will always correspond to 'real world' concepts which have been modelled initially in heterogeneous data repositories. Our rationale for such naming is to retain the maximum number of concepts from original data repositories and avoid the creation of new concepts while resolving semantic conflicts.
- In our case study in chapter 5, we use relational databases as examples of heterogeneous data repositories. However, our SA, ontological concepts and reasoning mechanisms (i.e. layering) are fully capable of accommodating any type and format of heterogeneous data repositories, i.e. XML and HTML documents, Web services, data repositories that store sensor derived data, and many more. We have given examples of relational databases because they have proven to be the easiest way of showing semantic conflicts in heterogeneous data. However, pervasive healthcare today depends on a range of data repositories, which are still very likely to be relational, but created on ad-hoc basis and accommodating various types of data through a range of devices and computers. What is important to note is that all these data repositories might exhibit various levels of persistence, they might not be stored at 'fixed' locations at all, they can even find their space on mobile devices which collects and processes data. This is what the pervasiveness of modern healthcare is about: we have to agree that data is generated "as we go", stored at the most convenient "places" and processed by any device which happens to be available. If we need to exploit such data repositories, we have to have a mechanism of performing meaningful retrievals across them.

We are not aware of any new research that comprehensively looks at the problem of managing semantic heterogeneities and resolving semantic conflicts in order to achieve semantic interoperability, which could enable us to evaluate our SA which accommodates ontological layering further. We have examples of using ontologies within software frameworks to provide data sharing, because the ontologies provided similar functionalities as that of a middleware layer or federated schema between heterogeneous computing environments/underlying data repositories. Thus, allowing ontologies to behave like a mechanism that can mask heterogeneities provides a means to deal explicitly with semantic interoperability challenges. We are not aware of any other work that uses ontological and layered software architectural styles for resolving semantic conflicts and achieving semantic interoperability in heterogeneous data repositories.

After evaluating our research and summarising its contribution, a number of new issues that pose interesting challenges to the results of this research have arisen. Some will be the subject of future

research, but some need to be reflected upon to outline our concerns and views resulting from this research.

7.3.1 Complexity of Computations

We have already addressed the feasibility of implementing our proposal from this research as a component based software application derived from our SA, using NetBeans IDE and bridges to OWL API. In this section we would like to summarise its main characteristics by itemising the issues of complexity in our solution, and commenting on the nature of computations stored in our software components. Both of them are summarised from three different perspectives: software engineering, reasoning rules and technology used.

Complexity of our solution from the **software engineering** point of view is minimal because our:

- Ontological layers and set of ontologies which store them are generated ad-hoc, as a consequence of the request for retrievals.
- Ontologies are generated automatically according to the semantics of user's request, but reasoning upon each ontological layer creates a new ontological layer which contains a smaller number of ontological concepts and individuals than the layer before. Therefore, the number of individuals in our ontologies decreased as we proceed towards the final layer.
- Ontologies are dynamically created for a particular user's request. This means that they are deleted, i.e. they seize to exist when the results of retrievals are displayed on the screen. There is no need to save any ontological layer (including ontological concepts and individuals) and make them persistent because they are tailored for a particular moment.
- Software applications built upon our ontological layers are responsible for reverting ontologies back to their original status (empty state!) after the results of retrievals have been displayed. This means that no ontological individuals are stored in any of the ontological layers and ontologies within them after the retrieval, thus they are ready to accept a new set of individuals which will be populating ontologies as soon as the new request for retrieval is issued.
- The issue of *the number of ontologies* created and mapped to one another does not affect the complexity of the solution because of its strict layering. Please note that the issue of layering in software architectures [286 and 287] directly reduces the level of complexity of software solutions built upon layered architectures, thus making them more flexible and reusable and less complex.
- The issue of storing intermediary results of our ontological layering while reasoning and creating new layers does not impose any burden on the storage and management of our ontologies. This is because the number of local ontologies will probably never exceed the number which would affect the performance of our application and the initial number of Local Ontologies. If we take into account that we work in the healthcare domain, which exhibits almost the highest level of heterogeneities today, then how likely is it that patient records (about the same person) will be stored across 50-100 databases at the same time? Furthermore, translations of such repositories into Local Ontologies are parts of the preparation for our layering and it does not influence the reasoning process at all. At the moment the automated translation through well know tools of 4 databases,

which contain a substantial amount of records, into Local Ontologies takes no more than four minutes.

- Preparing the semantics for the reasoning and layering can be complex to a certain extent in terms of understanding the user's requests, choosing appropriate repositories and data in them relevant for the request, and discovering semantic conflicts which may appear as the result of the request. However, we have a very strict process which distinguishes between the preparation of semantics for ontological layering and layering itself. The preparation, which is provided through the specific ontologies (ENV_ONT, USER_INP_ONT and ADDED_VAL_ONT) reduces the complexity of reasoning and layering later, by providing a correct taxonomical structures needed for reasoning and core layering. These three ontologies always exist in the same format, i.e. their format is not influenced by the type of user's request. However, the content is dependent on our interpretation of user's request.

Complexity of our solution from the **reasoning rules** point of view:

- In the thesis we illustrate the proposal of resolving all 9 semantic conflicts (Appendix A.1, Table 4.1) in one single example of seriously heterogeneous set of 4 databases in the healthcare domain. In the example all possible heterogeneities, number of SWRL rules (79), and number of $LO_j s$, $TO_k s$, $DO_g s$ and Go-CID (which is 21 in total) are managed by the software application which runs on a moderate size machine (laptop) in 3 minutes and 55 seconds. This must be considered as good "application performance" considering that we have never had any proper hardware environment available for out testing.
- Our example mimics one of the most complex cases of heterogeneities, which is resolved in its entirety! The likeliness that we will constantly have all 9 semantics conflicts present in all retrievals across heterogeneous repositories in our modern applications is remote! Aggregation, Union Incompatibility and Isomorphism are the most common semantic conflicts, and they are probably always considered to be the main reasons for detecting structural conflicts in general [10]. Therefore, in reality we will always have fewer semantic conflicts than in our main example which will reduce dramatically the number of rules and ontological individuals moved around our layers. More importantly we might be able to reduce the number of rules in a) and b) above, if the type of semantic conflict will not require any alignment or integration (see chapter 4, section 4.1.2 and section 4.3.4.5).

Complexity of our solution dictated by the **technology** used:

- Expressivity of OWL is a double-edge sword! The freedom OWL offers might be an obstacle in creating the desired representation of heterogeneous data structures/values in OWL models when translating database relational schemas, and exploiting OWL models through additional constraints/reasoning mechanisms at the level of and between ontological classes, properties and individuals. However, our strict software layering, which houses all ontologies, is dependent on formally defined steps and tasks in our process of resolving semantic conflicts, that nay violation of the software engineering process might not give an adequate result. Furthermore, by having a well-

defined process in software engineering you automatically reduce complexities and utilise the technology you need to its maximum.

- Ontology mapping is performed through SWRL rules and because the rules are there to move ontological individuals in order to create the result of retrievals according to user's request. However, the rules perform mapping of ontologies which are not formal ontologies which are dependent on persistence (we do not wish to save anything!) or ontologies which are overburden with heavy logic! Our ontologies and their concepts are simple for the sake of mirroring underlying heterogeneous repositories and semantic conflicts we discovered within them therefore they cannot have any additional constraints modeled with them. This is one of the most important software engineering principles when resolving complex problems and achieving reusability and maintainability.
- Our ontologies are not formal ontologies and therefore certain issue of computational complexity is immaterial in the SE world. This is particularly true in this research because we use the power of semantic technologies for achieving software engineering solutions. Our computational complexity has been resolved through:
  (i) systemising semantic conflicts and their degrees of similarity,
  (ii) defining and creating software architectural layers,
  (iii) systemising the process of creating layers and resolving conflicts, and
  (iv) defining a reasoning mechanism where each layer is generated through a strictly defined ontological mapping (i.e. ontological alignment, integration and merge).

In light of (i)-(iv) above, the computational complexity of our solution is minimal: we have java code which is lean, small and highly re-usable, effective connection from Java IDE with external engines (JESS) in order to trigger the creation of ontological layering and running SWRL rules and finally effective user interfaces which disseminate user's input and results of retrievals, which use data from heterogeneous data repositories in an adequate manner! The total running of 79 rules within such an application architecture in less than 4 minutes, which resolve all possible 9 semantic conflicts is more than impressive. It is feasible because we reduce the computational complexity though (i)-(iv) above.

7.3.2 Impact of Technology

We use a software engineering method, which is our process for identifying and resolving semantic conflicts across heterogonous data repositories which range from databases to XML documents and web pages. The techniques used within the process are our reasoning mechanisms which perform ontological grouping, alignment, integration and merge. However, modern software architectures cannot be deployed without specifying the technology used for the deployment in the first place. Therefore, the technology dictates the way we exercise techniques create implementations from our architectural models. Note: we cannot talk about techniques (ii) above) before we choose a technology for the deployment of our architecture.

Technology used in the thesis is a part of Semantic Web technology stack, which dictates the deployment of our software architectural components and the application built from them. Therefore, our

technology dictates that our reasoning mechanisms are performed upon OWL/SWRL enable ontologies and no other!

7.3.2.1 OWL DL Versus OWL Full

In this research we use OWL DL as the language for creating and manipulating ontological concepts. *OWL DL* provides maximum expressiveness while retaining computational completeness (i.e. all conclusions in ontologies and reasoning rules are guaranteed to be computable) and decidability (i.e. all computations will finish in finite time) [42].

However, OWL DL is not the only sub language of OWL. OWL Lite supports a simple classification hierarchy and constraints, and *OWL Full* provides maximum expressiveness and the syntactic freedom of RDF with no computational guarantees [42]. Therefore, OWL Full allows free mixing of OWL with RDF Schema modelling constructs and subsequently, like RDF schemas, it does not enforce a strict separation of ontological classes, properties and individuals. This freedom may result in:

1. the desired representation of heterogeneous data structures/values in OWL models when translating database relational schemas and data values within a database into ontological concepts (the same applies to any other format of data repositories including XML), and
2. exploiting OWL models through additional constraints/reasoning mechanisms at the level of and between ontological classes, properties and individuals.

OWL Full may ultimately provide more 'semantic power' in the ontological world because it allows us to manipulate any concepts in the OWL model across all and between levels: classes / individuals / properties. However, using OWL Full may have a severe implication towards the consistency of OWL models, and subsequently, affect the reasoning rules we run upon them [314 and 315]. Having too much freedom, in terms of adding constraints at any level within OWL models, may result in rigid ontological solutions, which show (a) no flexibilities in accommodating changes in problem domains and ultimately (b) weak consistencies, which can make OWL model unsuitable for retaining computational completeness. Furthermore, we have experienced that reasoning engines in general cannot handle OWL models overloaded with constraints at all levels.

From the discussion above, we can see that we should strike a balance between:

(i) utilising the power of OWL Full and the freedom it gives us in exploiting the semantics of OWL models, as itemised in 1. and 2. above, and

(ii) ensuring the consistency of OWL models which may guarantee successful reasoning mechanisms for the purpose of creating inference as itemised in (b) above.

With respect to 1. above, we have to emphasise that when translating relational schemas into ontological concepts, we use tools, such as DataMaster [227] which enable automatic representations of relational concepts within the ontological world. However, as soon as we start using a software tool, we become dependent on functionalities offered by the tool and the subsets of OWL available within the tool (which is always the decision made by developers of such tools). In our research, the use of DataMaster dictated the following choices:

- we had to use option 2 (out of three) when translating data repositories into local ontologies, offered by DataMaster [294 and 296]. This is because we have found and reported errors in the logical consistency of the ontologies produced through translation option 3 (available at:

http://www.nabble.com/DataMaster:-OWL-models-generated-from-Relational-schemas-to2191).

Option 3 would deliver for us the desired representation of relational schemas within ontologies, as outlined in 1 above, and unfortunately could not be used. Option 1 required frame-based modelling in accordance to the Open Knowledge Base Connectivity[91] and it was automatically eliminated because it is not OWL based;

- option 2 in DataMaster is based on OWL DL in terms of dictating the way modeling constructs from relational databases are translated within ontological world. In other words we cannot assume that we will have a desired representation of heterogeneous data structures/values in OWL models as outlined in 1 above, when using option 2.

Therefore, in our research we use OWL DL. However, this is not the only reason of "preferring" OWL DL over OWL full.

With respect to 2. above, OWL DL requires constraints on disjointness of ontological classes, properties and individuals, which in turn validates the consistency of OWL models and guarantees reasoning mechanisms upon OWL modeling constructs. These constraints are essential if we expect OWL DL to comply with DL [316] and hence, allowing a reasoning engine (e.g. Jess) to make decisions that "OWL modelling constructs are decidable" [213]. This means that we can:

- decide whether an ontological individual IS of a particular class,
- check object and data type properties between ontological classes,
- check if a an ontological class is a subset of another class
- check if an OWL ontological model is consistent.

None of the bullets above can be done with OWL Full, because we will have to use SWRL rules for securing "decidability" (see the bullets above). Additionally SWRL rules will not be successful if OWL models are inconsistent, because they will not be able to guarantee successful reasoning mechanisms for the purpose of creating inference.

Finally, we have to draw the reader's attention to the role of SWRL rules in our proposal, in order to justify the use of OWL DL further. We use SWRL rules in order to:

a. establish semantically related data in heterogeneous data repositories (i.e. grouping reasoning mechanism);
b. execute ontology mappings (alignment, integration and merge) to resolve semantic conflicts, which automatically generates ontological layering;
c. provide domain-specific ontological solutions (e.g. resolving semantic conflicts in healthcare);
d. avoid overloading of OWL models with built-in semantics, because SWRL rules can add expressivity to and strengthen the semantics of OWL models;
e. leave local ontologies LO_j intact while resolving semantic conflicts, i.e. preserving the original semantics in local ontologies LO_j carried forward from data repositories Rep_i. In other words semantic conflicts detected within local ontologies LO_j are carried forward in and resolved through ontological layering;
f. add expressivity to OWL models in terms of modeling CWA, i.e. define assumption in which everything that is not derivable from the OWL model is assumed to be false (see chapter 3, section

[91] http://www.ai.sri.com/~okbc/

3.3.2), in order to retain computational completeness of OWL models and reduce the level of OWA (i.e. assumption in which conclusions cannot be derived from an ontological model (see chapter 3, section 3.3.2)). For example, the names of range values or ontological individuals in our SWRL rules are categorised as CWA that guarantee right inferences/assertions at the right place in our ontological layering.

Therefore, in our research, striking a balance between the freedom of utilising the power of OWL Full and ensuring the consistency of OWL models, means that we have to trade-off and sacrifice the freedom of exploiting semantics across and between all levels of OWL models in OWL Full, in order to secure reasoning mechanisms (running SWRL rules upon OWL DL model), which ultimately generate ontological layering and resolve semantic conflicts.

7.3.2.2 Reasoning Rules and Hard Coding

When reflecting upon the nature of our SWRL rules used in a. and b. above, we can argue that we "hard code" the way we resolve semantic conflicts. "Hard coding" per se may:

- restrict the level of 're-usability' of *Low-Level*, *High-Level* and *Post-High-Level* reasoning mechanisms (i.e. our reasoning mechanism may be seen as being "specific" to the problem of resolving semantic conflicts, which appear within a particular retrievals across heterogeneous data repositories),
- have impact on how generic our ontological layering is (i.e. it is most likely that ontologies are too domain specific), and
- increase the number of SWRL rules used in in our reasoning mechanisms.

The alternative to our way of using SWRL rules could be to create additional constraints within our OWL models instead of creating SWRL rules, which add expressivity to them. However, this would have a severe impact on autonomy of our data repositories and the semantics carried forward to our local ontologies. Our proposal is based on the decision to keep data repositions INTACT (autonomous) and a build a solution which resolves semantic conflicts in ontological layering, without changing concepts and semantics of data repositories.

Therefore, we promote "hard coding" through the names of range values or ontological individuals in our SWRL rules because it allows us to:

- express and strengthen the semantics stored in heterogeneous data repositories by fully exercising SWRL rules,
- resolve semantic conflicts through SWRL rules and NOT through additional constraints modeled within OWL models and
- provide re-usability in our solution in terms of using the same set of local ontologies across various user's requests for retrievals imposed on them. Hence, the whenever the user issue a request for a different retrieval, the only aspect of our proposal which changes is a set of SWRL rules. In other words, each request for a retrieval will result in a different set of ontology mappings which are generated according to the types of semantic conflicts we try to resolve and NOT according to the semantics stored in local ontologies. Hence, different ontology mappings can be used through different set of SWRL rules without changing the semantics of data repositories.

7.3.2.3 Computations in Java Versus SWRL rules

In chapter 5, section 5.4, we describe a software application which manages user's requests and triggers ontological layering which ultimately resolves semantic conflicts. It is obvious that we had to use integrated development environments which secures front end GUI and access to the results of our ontological layering. (NetBeans 6.4 IDE and JSP, OWL API), Therefore, we must ask how much we should rely on Java in future and how much we can overload SWRL to do computations. Thus, our idea to distinguish between computations created in Java and computations that rely on execution of SWRL rules though reasoning engines has become important because we wanted to achieve:

- a high level of level of automation when creating ontological layering,
- reduction in the amount java code which manipulates data from repositories
- isolation of reasoning rules from the main stream computations (java) to allow us to run any number and combination of SWRL rules, upon constantly changing ontological individuals stored in ontological layers.

There are no restrictions in using methods from OWL API libraries when using NetBeans, therefore our choice to import methods from OWL API libraries is dictated by their roles in:

- accessing and manipulating ontological concepts and
- triggering reasoning mechanisms upon ontological concepts.

The co-existence of NetBeans 6.4 IDE and Protégé 3.4 has helped us to generate the software application built upon ontological layering and decide exactly where "computations" (Java versus SWRL) should take place.

By relying on the power of SWRL and flexibility of our OWL models we have minimized java coding and created a software application which in its back-end manipulates semantics of data stored in ontologies and NOT in classical (relational?) data repositories. Furthermore our java code remains intact regardless which request is being issued upon heterogeneous repositories. This might be the first step towards the creation of *semantic software applications* which manipulate ontological concepts.

A list of future works applicable solely to the example of retrievals of semantically related data across repositories in pervasive healthcare environments is below.

7.4 Future Research

We give below a list of future works which will either enhance or evaluate our proposal further.

1. We have not experimented with all of the sub-options in our *Low-level* and *High-level* reasoning mechanisms outlined in chapter 4 and illustrated in chapter 5, when creating matches/links between ontological individuals and moving/transferring them to a different ontological class (pages 56, 71 and 74). In other words, we have used sub-options 6a, 6b, 6c and 7b but have had no opportunity to experiment with other sub-options 6d, 7a, 7c and 7d (see chapter 4. section 4.3.4.2-4.3.4.3). The reasons for this are twofold. Firstly, the sub-options we used have proven to be sufficient to establish semantically related data and resolve all 9 semantic conflicts they generate. Secondly, the semantic richness of the data in repositories and consequently in local ontologies did not require the use off all the sub-options in our reasoning mechanisms. However, unused sub-options are there for

any other situation where we may use our ontological layering, i.e. use our reasoning mechanisms across domains and case studies. If for any reason we are in future faced with new types of semantic conflicts then our reasoning mechanisms will have provisions for handling them through all unused sub-options. Therefore, problem domains where unused matching/linking options can be illustrated, remains to be found.

2. We have elaborated in section 7.2.4, that the level of automation in our proposal depends on our classification of semantically related data and their degree of similarities. We use the classification to identify semantic conflicts and perform ontology mappings in order resolve semantic conflicts. Thus, the next step would be to build an new ontology for our classification of semantically related data and their degrees of similarities, which in turn can be used for:
   i. mappings between the ontology and core ontological layering in order to fully automate the process for identifying and resolving semantic conflicts, and
   ii. adding semantic weightings to each degree of similarity through OWL annotations in order to evaluate if our degrees of similarities are an acceptable measure for identifying semantic conflicts.
3. We are currently extending our categorisation of semantic conflicts in order make provisions for other types of semantic heterogeneities in modern computational environments. However, we should deal in future with ontology mismatches in terms of the differences in the number of restrictions applied to ontological concepts, and the number of properties used in restrictions. Consequently, the most challenging task would be to concentrate on semantic heterogeneities of completely unstructured data!
4. We have managed to create layered ontologies capable of building machine interpretable knowledge. However, our ontological solution is NOT there to build a knowledge base. Our ontological layering is a software engineering mechanism for resolving semantic conflicts, which exists only for the purpose of retrievals across heterogeneous data repositories. Therefore, we should open doors to anyone who is ready to (re)use our reasoning mechanisms for creating a knowledge-base (in any domain and for any purpose). Furthermore, the inference in our ontological solution is based on ontological mappings which move/transfer ontological individuals through ontological layers. It would be not appropriate for us to "keep" the content of these ontological layers in any format of persistence. More research should be done in order to see which type of knowledge-base we could generate through our reasoning mechanisms.
5. We have tested the performance of our software application built upon ontological layering, in terms of the time it takes to run the request for retrievals across heterogeneous data repositories. However, we are aware that we can increase performance by saving/distributing SWRL rules across several .owl files, hence adhering to Tbox and Abox philosophy in DL[92].
6. We have not used RDF for resolving semantic conflicts because of the nature of data repositories and heterogeneities we carry forward in our ontological layering. It is obviously easier to mirror structures and semi-structures of data repositories such as databases in ontological concepts than in RDF triples. However, we leave for future works the possibility of building ontologies at the RDF

[92] http://www.w3org/2001/sw/

level in order to support the Semantic Web Stack (introduced in chapter 2, section 2.5) and guarantee machine interpretable knowledge.

7. We have to draw the reader's attention to our own list of future works applicable solely to the problem domain of pervasive healthcare and technologies used in such environments. They are itemised below:
   a. *Component based solution and MVC pattern* - although we have demonstrated that we are capable of running SWRL rules within an .owl file, we must analyse technologies which can support and manage the persistence of .owl files so that ontological concepts within such solutions can correspond to the "model" of the MVC pattern[93].
   b. *The power of users* – although we deliver rich user interfaces to medical professionals who are using our software application described in section 5.4, we have to evaluate the possibility of giving them more freedom in specifying the exact data required in retrievals across heterogeneous data repositories, e.g. the choice of more radio buttons in order to select the exact data that constitutes *information types*. However, this will naturally imply an increase in the number of SWRL *Grouping* rules used in order to group semantically related data in information types specified in retrievals, as a consequence of storing and interpreting user requests. Consequently, we must investigate the impact of these additional rules on the performance of our software application.
   c. *Methods within OWL API library* – although our choice of library methods works perfectly well in all our experiments, we have to consider, for the purpose of improving the functionality of our application from section 5.4, which other 'methods' within the OWL API library may be more suitable than the current ones.

[93] http://www.oracle.com/technetwork/java/mvc-140477.html

Appendices

Appendix A.1

Table 4.1 Different types of semantic conflicts based on Naming and Structural conflicts that our SA resolves through core ontological layering.

Types of *Naming* based conflicts:	**Definition:**	**Example:**
MISSPELT/ CASE-SENSITIVE semantic conflict.	Incorrect spellings/ Upper and lower cased letters used to describe named concepts.	Consider the example of two concepts called `MEDICAL_SUMMARY` and `MMMedical_Summary` that model a patient's summary of previous diagnosis in a general practitioner and hospital environment respectively. The `MEDICAL_SUMMARY` and `MMMedical_Summary` concepts are semantically related and have the same meaning, i.e. they both model a patient's summary of previous diagnosis. However, there is a conflict between them due to the difference in their spelling and case sensitivity of characters.
HOMONYM semantic conflict.	A named concept that sounds alike another named concept but have different meanings to each other.	Consider the example of a concept called `Report` that models the combinations of treatments and diagnosis per patient for a general practitioner environment, and a concept called `Report` that represents the list of patients who have positively reacted to a particular treatment within a hospital environment. The concepts `Report` have the same name in both environments, and appear to be semantically related. However, there is a conflict between them due to the difference in their meaning, i.e. these two concepts do not have any similarities in respect to their interpretation in a given context.
SYNONYM based semantic conflict.	A named concept having the same or nearly the same meaning as another named concept.	Consider the example of a concept called `Electronic health record` that models a patient's health summary in a general practitioner environment and a concept called `Computational record` that models the same patient's health summary in a hospital environment. Both concepts `Electronic health record` and `Computational record` are semantically related and have semantic similarities between them as they model the same patient's health summary. However, there is a conflict between them due to the difference in their intended use, i.e. a hospital patient semantics might have different structures compared to a general practitioner's patient semantics.

Types of *Structural* based conflicts:	Definition:	Example:
GENERALISATION semantic conflict.	A named concept having a 'super types' of their 'characteristics' in terms of describing the same meaning as another named concept.	Consider the example of a concept called `Summary of treatments` that models a patient's treatments over the six months in a general practitioner's environment and the concepts called `Previous treatment summaries` and `Current treatment summaries` that model the same patient's treatments over the last year in a clinic environment. The three concepts are semantically related and have semantic similarities because they all model the same patient's treatments. However, there is a conflict between them due to the difference in their structures. The `Summary of treatments` concept is a super-type of `Previous treatment summaries` and `Current treatment summaries` concepts, i.e. `Summary of treatments` is a generalized concept which contains subsets of both concepts `Previous treatment summaries` and `Current treatment summaries`.
SPECIALISATION semantic conflict.	A named concept having a 'sub types' of their 'characteristics' in terms of describing the same meaning as another named concept.	Consider the example of concepts called `Previous prescription summary` and `Current prescription summary` that model a patient's prescription over the last six months in a hospital environment, and a concept called `Summary of prescription` that also models the same patient's prescription over the last year in a general practitioner's environment. The three concepts are semantically related and have semantic similarities because they all model the same patient's prescriptions. However, there is a conflict between them due to the difference in their structures. The `Previous prescription summary` and `Current prescription summary` concepts are sub-types of the `Summary of prescription` concept i.e. `Previous prescription summary` and `Current prescription summary` are specialized concepts which are contained within' the `Summary of prescription` concept.
ISOMORPHISM semantic conflict.	A named concept having a 'different numbers' of their 'characteristics' in terms of describing the same meaning as another named concept.	Consider the example of concepts called `Labtest name`, `Labtest type` and `Labtest date` that model a patient's lab tests in a hospital environment, and the concepts called `Labtest name`, `Labtest type`, `Labtest data` and `Labtest date` that also model the same patient's lab test in a clinic environment. The seven concepts are semantically related and have semantic similarities because they all model the same patient's lab tests results. However, there is a conflict between them due to the difference in the number of structures, i.e. concepts in the clinic

		environment have a different number of 'characteristics'[94] compared to the concepts in the hospital environment (`Labtest data` exists only in the hospital environment).
UNION INCOMPATIBILITY semantic conflict.	A named concept having a 'different structures' of their 'characteristics' in terms of describing the same meaning as another named concept.	Consider the example of concepts called `Medication name`, `Medication description` and `Manufacturing address` that models a patient's prescribed medications in a hospital environment, and the concepts called `Medication name`, `Medicine description` and `Manufacturing description` that also model the same patient's prescribed medications in a clinic environment. These six concepts are semantically related and have semantic similarities because they all model the same patient's prescribed medicine. However, there is a conflict between them due to the difference in their structures, i.e. the hospital environment uses the concept `Manufacturing address` and the clinic environment uses the concept `Manufacturing description` to cover the same meaning.
AGGREGATION semantic conflict.	A named concept having a 'different aspects' of their 'characteristics' in terms of describing the same meaning as another named concept.	Consider the example of concepts called `Patient name`, `Patient address` and `Patient contact number` that model a patient's personal details in a hospital environment, and the concepts called `Patient first name`, `Patient last name`, `Patient address` and `Patient contact number` that also models the same patient's personal details in a general practitioners environment. These seven concepts are semantically related and have semantic similarities because they all model the same patient's personal details. However, there is a conflict between them due to the difference in their structures, i.e. patient's demographic data are modeled differently because the hospital environment uses the concept `Patient name` and the clinic environment uses the concepts `Patient first name` and `Patient last name`. The aggregation of the two concepts from the clinic environment will create a concept equivalent to `Patient name` in the hospital environment.

[94] 'characteristics' of the concept may be either: 'labels/titles' given to concepts, 'data structures' that make up concepts, or 'data instances/values' of concepts in any data model, schema or meta-data levels.

Appendix A.2

The relational schemas for the data repositories *GP_Rep*, *Hospital_Rep*, *Clinic_2_Rep*, and *Clinic_2_Rep*, including the 'insert' SQL statements for patient 'JANE FLEE' in the Persistent layer of the software architecture for Go-CID software applications:

Relational schema for GP_Rep

Create statement for GP.Patient table:

```
CREATE TABLE GP_DB.PATIENT (
PATIENT_ID VARCHAR (6),
FIRST_NAME VARCHAR (10),
LAST_NAME VARCHAR (20),
SEX CHAR(1) CHECK (Gender IN ('M', 'F')),
DOB VARCHAR(20),
ADDRESS VARCHAR (100),
REGION VARCHAR (100),
TELEFFONNE  VARCHAR (20),
NEXT_OF_KIN  VARCHAR (30),
EMERGENCY_CONTACT VARCHAR (100),
NO_OF_CHILDREN VARCHAR (10),
BMI VARCHAR (20),
HEIGHT VARCHAR (30),
PRIMARY KEY (PATIENT_ID)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Insert values for GP.Patient table:

```
INSERT INTO GP_DB.PATIENT VALUES ('P3344A', 'JANE', 'FLEE', 'F',
'JULY_04_1970', '167_BOULEVARD_RD_W1W_5TU', 'LONDON', '02075698899',
'NEMANJA_FLEE', '07965896456', '0', 'NORMAL', '5_feet_8_inches');

INSERT INTO GP_DB.PATIENT VALUES ('P2255B', 'JULIA', 'FOX', 'F',
'JUNE_17_1971', '27_HANGER_LANE_RD_N1E_6SU', 'LONDON', '02078691369',
'PETER_FOX', '07949538498', '1', 'NORMAL_PER_BMI', '5_feet_4_inches');

SELECT * FROM GP_DB.PATIENT;
```

Create statement for GP.Prescription table:

```
CREATE TABLE GP_DB.PRESCRIPTION (
PRESCRIPTION_NO VARCHAR (6),
PATIENT_ID VARCHAR (6),
PRESCRIPTION_DATE DATE,
PRIMARY KEY (PRESCRIPTION_NO),
FOREIGN KEY (PATIENT_ID) REFERENCES PATIENT(PATIENT_ID)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Insert values for GP.Prescription table:

```
INSERT INTO GP_DB.PRESCRIPTION VALUES('PP1245', 'P3344A', '12-03-09');

INSERT INTO GP_DB.PRESCRIPTION VALUES('PP4569', 'P2255B', '14-03-09');

SELECT * FROM GP_DB.PRESCRIPTION;
```

Create statement for GP.Medication table:

```
CREATE TABLE GP_DB.MEDICATION(
MEDICINE_NUM VARCHAR (6),
MEDICINE_NAME VARCHAR (30),
VENDOR VARCHAR (100),
MNF_DESC VARCHAR (100),
PRIMARY KEY (MEDICINE_NUM)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Insert values for GP.Medication table:

```
INSERT INTO GP_DB.MEDICATION VALUES('M0031', ' M0031_Capzasin',
'M0031_Xhing_Ltd', ' M0031_China_pharmaceuticals');

SELECT * FROM GP_DB.MEDICATION;
```

Create statement for GP.Medication_Prescribed table:

```
CREATE TABLE GP_DB.MEDICATION_PRESCRIBED (
PRESCRIPTION_NO VARCHAR (6),
MEDICINE_NUM VARCHAR (6),
MEDICATION_DESC VARCHAR (200),
DOSAGE_AMOUNT VARCHAR(200),
PRIMARY KEY (PRESCRIPTION_NO, MEDICINE_NUM),
FOREIGN KEY(PRESCRIPTION_NO) REFERENCES PRESCRIPTION(PRESCRIPTION_NO),
FOREIGN KEY(MEDICINE_NUM) REFERENCES MEDICATION(MEDICINE_NUM)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Insert values for GP.Medication_Prescribed table:

```
INSERT INTO GP_DB.MEDICATION_PRESCRIBED VALUES('PP1245', 'M0031',
'M0031_Pain killer_Perindopril', ' M0031_2_tablets_per_day');

INSERT INTO GP_DB.MEDICATION_PRESCRIBED VALUES('PP4569', 'M0031',
'M0031_Pain killer_Perindopril_Ebrumine', ' M0031_3_tablets_per_day');

SELECT * FROM GP_DB.MEDICATION_PRESCRIBED;
```

Create statement for GP.Treatment table:

```
CREATE TABLE GP_DB.TREATMENT (
TREATMENT_NO VARCHAR(6),
PRESCRIPTION_NO VARCHAR (6),
MEDICINE_NUM VARCHAR (6),
TREATMENT_OVERVIEW VARCHAR (200),
DATE VARCHAR(20),
PRIMARY KEY (TREATMENT_NO),
FOREIGN KEY(PRESCRIPTION_NO) REFERENCES PRESCRIPTION(PRESCRIPTION_NO),
FOREIGN KEY(MEDICINE_NUM) REFERENCES MEDICATION(MEDICINE_NUM)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Insert values for GP.Treatment table:

```
INSERT INTO GP_DB.TREATMENT VALUES ('TT1989', 'PP1245', 'M0031',
'TT1989_Patient_is_suffering_from_aches_in_lower_limbs_and_has_minor_swell
ing_to_ankle_pain_support_through_chronic_pain_recovery_is_suggested',
'TT1989_12-03-09');
INSERT INTO GP_DB.TREATMENT VALUES('TT4563', 'PP4569', 'M0031',
'TT4563_Patient_is_suffering_from_pains_in_lower_limbs_and_has_minor_swell
ing_to_ankle_see_prescription_given', 'TT4563_14-03-09');

SELECT * FROM GP_DB.TREATMENT;
```

Relational schema for Hospital_Rep

Create statement for Hospital.Patient table:

```
CREATE TABLE HOSPITAL_DB.PATIENT (
PATIENT_NO VARCHAR (6),
NAME VARCHAR (10),
SEX CHAR(1) CHECK (Gender IN ('M', 'F')),
DOB DATE,
H_MEDICAL_SUMMARY VARCHAR (200),
MAJOR_ILLNESS VARCHAR (100),
CHRONIC_DISEASE  VARCHAR (100),
PRIMARY KEY (PATIENT_NO)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Insert values for Hospital.Patient table:

```
INSERT INTO HOSPITAL_DB.PATIENT VALUES ('P0001', 'JANE_FLEE', 'F', '1970-
07-04',
'Mrs_Flee_complains_of_severe_pain_in_left_ankle_Minor_swelling_evident_an
d_xrays_taken_admitted_as_overnight_stay_and_found_to_have_acute_COPD_exac
erbation', 'no_major_illnness_evident', 'no_chronic_disease_evident');

INSERT INTO HOSPITAL_DB.PATIENT VALUES ('P0002', 'JULIA FOX', 'F', '1971-
06-17',
'Mrs_Fox_complains_of_severe_pain_in_left_ankle_and_xrays_taken_admitted_a
s_overnight_stay_and_found_to_have_acute_COPD_exacerbation',
'no_major_illnness', 'no_chronic_disease');

SELECT * FROM HOSPITAL_DB.PATIENT;
```

Create statement for Hospital.Treatment table:

```
CREATE TABLE HOSPITAL_DB.TREATMENT (
TREATMENT_NO VARCHAR (6),
PATIENT_NO VARCHAR (6),
TREATMENT_TYPE VARCHAR (100),
TREATMENT_NAME VARCHAR (100),
REPORT VARCHAR (100),
DATE VARCHAR (20),
PRIMARY KEY (TREATMENT_NO),
FOREIGN KEY(PATIENT_NO) REFERENCES PATIENT(PATIENT_NO)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Insert values for Hospital.Treatment table:

```
INSERT INTO HOSPITAL_DB.TREATMENT VALUES ('T09851', 'P0001',
'T09851_COPD_Chronic_pain_recovery', 'T09851_COPD_exacerbation',
‘file_24456tt’, 'T09851_17-04-09');

INSERT INTO HOSPITAL_DB.TREATMENT VALUES ('T01245', 'P0002',
'T01245_COPDT_Chronic_pain_recovery', 'T01245_COPDT_exacerbation',
‘file_24466tt’, 'T01245_18-04-09');

SELECT * FROM HOSPITAL_DB.TREATMENT;
```

Create statement for Hospital.Medication table:

```
CREATE TABLE HOSPITAL_DB.MEDICATION(
MEDICINE_NO VARCHAR (6),
MEDICINE_NAME VARCHAR (30),
VENDOR VARCHAR (100),
MNF_ADDRESS VARCHAR (100),
PRIMARY KEY (MEDICINE_NO)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Insert values for Hospital.Medication table:

```
INSERT INTO HOSPITAL_DB.MEDICATION VALUES('M222p', 'M222p_NAPROXEN', '
M222p_Risedronate', ' M222p_Andheri_east_India');

INSERT INTO HOSPITAL_DB.MEDICATION VALUES('M225i', 'M225i_EHOSUXIMIDE',
'M225i_Emeside', 'M225i_South_coast_Canada');

SELECT * FROM HOSPITAL_DB.MEDICATION;
```

Create statement for Hospital.Medication_Prescribed table:

```
CREATE TABLE HOSPITAL_DB.MEDICATION_PRESCRIBED (
TREATMENT_NO VARCHAR (6),
MEDICINE_NO VARCHAR (6),
MEDICATION_DESC VARCHAR (200),
DOSAGE_AMOUNT VARCHAR(200),
PRIMARY KEY (TREATMENT_NO, MEDICINE_NO),
```

```
FOREIGN KEY(TREATMENT_NO) REFERENCES HOSPITAL_DB.TREATMENT(TREATMENT_NO),
FOREIGN KEY(MEDICINE_NO) REFERENCES HOSPITAL_DB.MEDICATION(MEDICINE_NO)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Insert values for Hospital.Medication_Prescribed table:

```
INSERT INTO HOSPITAL_DB.MEDICATION_PRESCRIBED VALUES('T09851', 'M222p', '
M222p_Anti_inflammatory_drugs', '
M222p_1_or_2_tablets_to_be_taken_4_times_a_day');

INSERT INTO HOSPITAL_DB.MEDICATION_PRESCRIBED VALUES('T09851', 'M225i', '
M225i_Anti_convulsant_drugs', '
M225i_1_tablets_to_be_taken_4_times_a_day');

INSERT INTO HOSPITAL_DB.MEDICATION_PRESCRIBED VALUES('T01245', 'M225i',
'M225i_Anti_convulsant', 'M225i_1_tablets_to_be_taken_2_times_a_day');

SELECT * FROM HOSPITAL_DB.MEDICATION_PRESCRIBED;
```

Relational schema for Clinic_1_Rep

Create statement for Clinic_1.Patient table:

```
CREATE TABLE CLINIC_1_DB.patient (
PATIENT_NO VARCHAR (6),
NAME VARCHAR (10),
SEX CHAR(1) CHECK (Gender IN ('M', 'F')),
TELEPHONE  VARCHAR (20),
DOB DATE,
PREVIOUS_MEDICAL_SUMMARY VARCHAR (200),
CURRENT_MEDICAL_SUMMARY VARCHAR (200),
MAJOR_ILLNESS VARCHAR (100),
CHRONIC_DISEASE  VARCHAR (100),
PRIMARY KEY (PATIENT_NO)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Insert values for Clinic_1.Patient table:

```
INSERT INTO CLINIC_1_DB.PATIENT VALUES ('PP9985', 'JANE_FLEE', 'F',
'02075698899', '1970-07-04',
'Mrs_Flee_has_a_regular_cervical_smear_test_results_appear_normal',
'Mrs_Flee_has_come_into_the_clinic_for_a_blood_test_complains_of_fatigue',
'none_found', 'none');

INSERT INTO CLINIC_1_DB.PATIENT VALUES ('PP1125', 'JULIA FOX', 'F',
'02078691369', '1971-06-17',
'Mrs_Fox_has_a_regular_cervical_smear_test_results_appear_normal',
'Mrs_Fox_has_come_into_the_clinic_for_a_blood_test_complains_of_fatigue',
'no_trace', 'no');

SELECT * FROM CLINIC_1_DB.PATIENT;
```

Create statement for Clinic_1.Labtest table:

```
CREATE TABLE CLINIC_1_DB.LABTEST (
LABTEST_ID VARCHAR (6),
PATIENT_NO VARCHAR (6),
LABTEST_TYPE VARCHAR (100),
LABTEST_NAME VARCHAR (100),
LABTEST_RESULTS VARCHAR (100),
REPORT VARCHAR (100),
LABTEST_DATE VARCHAR(20),
PRIMARY KEY (LABTEST_ID),
FOREIGN KEY(PATIENT_NO) REFERENCES PATIENT(PATIENT_NO)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Insert Values for Clinic_1.Labtest table:

```
INSERT INTO CLINIC_1_DB.LABTEST VALUES ('LT256', 'PP9985',
'LT256_Smear_test', 'LT256_Cervical_Type_3', 'LT256_Normal',
'LT256_file_0066', 'LT256_16-01-08');

INSERT INTO CLINIC_1_DB.LABTEST VALUES ('LT123', 'PP9985',
'LT123_Pathology', 'LT123_Blood_test_Type_4123', 'LT123_anaemia_level_46',
'LT123_file_0098', 'LT123_16-02-08');

INSERT INTO CLINIC_1_DB.LABTEST VALUES('LT659', 'PP1125',
'LT659_Smear_test_1', 'LT659_Cervical_T2', 'LT659_Normal_ABt',
'LT659_file_0000',  ' LT659_16-01-08');

SELECT * FROM CLINIC_1_DB.LABTEST;
```

Relational schema for Clinic_2_Rep

Create statement for Clinic_2.Patient table:

```
CREATE TABLE CLINIC_2_DB.PATIENT (
PATIENT_NO VARCHAR (6),
LAST_NAME VARCHAR (10),
FIRST_NAME VARCHAR (10),
SEX CHAR(1) CHECK (Gender IN ('M', 'F')),
DOB DATE,
MEDICAL_SUMMARY VARCHAR (200),
MAJOR_ILLNESS VARCHAR (100),
CHRONIC_DISEASE  VARCHAR (100),
PRIMARY KEY (PATIENT_NO)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Insert values for Clinic_2.Patient table:

```
INSERT INTO CLINIC_2_DB.PATIENT VALUES ('Pt8895', 'FLEE', 'JANE', 'F',
'1970-07-04',
'Mrs_Flee_complains_of_shortness_of_breath_and_feels_intervals_of
pain_in_chest_area', 'No_MJ.', 'no_cd_found');

INSERT INTO CLINIC_2_DB.PATIENT VALUES ('Pt1123', 'FOX', 'JULIA', 'F',
'1971-06-17',
'Mrs_Fox_complains_of_shortness_of_breath_and_feels_intervals_of
pain_in_chest_area', 'MJ_not_found.', 'CD_not_found');

SELECT * FROM CLINIC_2_DB.PATIENT;
```

Create statement for Clinic_2.Labtest table:

```
CREATE TABLE CLINIC_2_DB.LABTEST (
LABTEST_ID VARCHAR (6),
PATIENT_NO VARCHAR (6),
LABTEST_OVERVIEW VARCHAR (200),
LABTEST_DATA VARCHAR (200),
LABTEST_TYPE VARCHAR (100),
LABTEST_NAME VARCHAR (100),
LABTEST_RESULTS VARCHAR (100),
REPORT VARCHAR (100),
LABTEST_DATE VARCHAR(20),
PRIMARY KEY (LABTEST_ID),
FOREIGN KEY(PATIENT_NO) REFERENCES PATIENT(PATIENT_NO)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Insert values for Clinic_2.Labtest table:

```
INSERT INTO CLINIC_2_DB.LABTEST VALUES('LL456', 'Pt8895',
'LL456_Used_to_identify_lung_diseases', 'LL456_data_aa2',
'LL456_Radiation', 'LL456_Xray', 'LL456_fileID_wavelength908',
'LL456_file_0001',  'LL456_28-04-09');
```

```
INSERT INTO CLINIC_2_DB.LABTEST VALUES('LLl4569', 'Pt1123',
'LLl4569_Used_to_identify_Pneumonia_lung_cancer_fluid_collection_the_lungs
', 'LL14569_data_aa1',  ' LLl4569_Radiation_143', ' LLl4569_X_ray', '
LLl4569_fileID_wavelength203', ' 'LLl4569_file_1401',  ' LLl4569_29-03-
09');

SELECT * FROM CLINIC_2_DB.LABTEST;
```

Appendix A.3

Can be found on CD-ROM attached to thesis.

Appendix A.4

Table 5.1 The complete set of domain and range constraints for the datatype properties in local ontology *LO_gp*.

Datatype Property:	**Domain Value:** (***Class name and associated ontological indvidual***)	**Range Value:** (***Literal values***)
db1:patient.PATIENT_ID	ontological individual: db:patient_Instance_1 of the: db1:patient ontological class.	P3344A
db1:patient.FIRST_NAME	ontological individual: db:patient_Instance_1 of the: db1:patient ontological class.	JANE
db1:patient.LAST_NAME	ontological individual: db:patient_Instance_1 of the: db1:patient ontological class.	FLEE
db1:patient.ADDRESS	ontological individual: db:patient_Instance_1 of the: db1:patient ontological class.	167_BOULEVARD_RD_W1W_5TU
db1:patient.REGION	ontological individual: db:patient_Instance_1 of the: db1:patient ontological class.	LONDON
db1:patient.TELEFFONNE	ontological individual: db:patient_Instance_1 of the: db1:patient ontological class.	02085698899
db1:patient.NEXT_OF_KIN	ontological individual: db:patient_Instance_1 of the: db1:patient ontological class.	NEMANJA_FLEE
db1:patient.EMERGENCY_CONTACT	ontological individual: db:patient_Instance_1 of the: db1:patient ontological class.	07965896456
db1:patient.NO_OF_CHILDREN	ontological individual: db:patient_Instance_1 of the: db1:patient ontological class.	0
db1:patient.BMI	ontological individual: db:patient_Instance_1 of the: db1:patient ontological class.	NORMAL
db1:patient.HEIGHT	ontological individual: db:patient_Instance_1 of the: db1:patient ontological class.	5_feet_8_inches

Appendix A.5

Table 5.2 The complete set of domain and range constraints for the object properties in local ontology *LO_gp*.

Datatype Property:	**Domain Value:** (***Class name and associated ontological indvidual***)	**Range Value:** (***Literal values***)
gp-patient-PATIENT_ID	ontological individual: gp-patient_Instance_1 of the: LO_gp-patient_records ontological class.	the ontological individual: P3344A of the: LO_gp-patient_instances ontological class.
gp-patient-FIRST_NAME	ontological individual: gp-patient_Instance_1 of the: LO_gp-patient_records ontological class.	the ontological individual: JANE of the: LO_gp-patient_instances ontological class.
gp-patient-LAST_NAME	ontological individual: gp-patient_Instance_1 of the: LO_gp-patient_records ontological class.	the ontological individual: FLEE of the: LO_gp-patient_instances ontological class.
gp-patient-ADDRESS	ontological individual: gp-patient_Instance_1 of the: LO_gp-patient_records ontological class.	the ontological individual: 167_BOULEVARD_RD_W1W_5TU of the: LO_gp-patient_instances ontological class.
gp-patient-REGION	ontological individual: gp-patient_Instance_1 of the: LO_gp-patient_records ontological class.	the ontological individual: LONDON of the: LO_gp-patient_instances ontological class.
gp-patient-TELEPHONE	ontological individual: gp-patient_Instance_1 of the: LO_gp-patient_records ontological class.	the ontological individual: 02085698899 of the: LO_gp-patient_instances ontological class.
gp-patient-NEXT_OF_KIN	ontological individual: gp-patient_Instance_1 of the: LO_gp-patient_records ontological class.	the ontological individual: NEMANJA_FLEE of the: LO_gp-patient_instances ontological class
gp-patient-EMERGENCY_CONTACT	ontological individual: gp-patient_Instance_1 of the: LO_gp-patient_records ontological class.	the ontological individual: 07965896456 of the: LO_gp-patient_instances ontological class.
gp-patient-NO_OF_CHILDREN	ontological individual: gp-patient_Instance_1 of the: LO_gp-patient_records ontological class.	the ontological individual: 0 of the: LO_gp-patient_instances ontological class.
gp-patient-BMI	ontological individual: gp-patient_Instance_1 of the: LO_gp-patient_records ontological class.	the ontological individual: NORMAL of the: LO_gp-patient_instances ontological class.

gp-patient-HIEGHT	ontological individual: gp-patient_Instance_1 of the: LO_gp-patient_records ontological class.	the ontological individual: 5_feet_8_inches of the: LO_gp-patient_instances ontological class.

Appendix A.6

Can be found on CD-ROM attached to thesis.

Appendix A.7

Table 5.3. Examples of semantic similarities and conflicts that have been carried forward into the ontological concepts of the local ontologies *LO_gp*, *LO_hospital*, *LO_clinic_1* and *LO_clinic_2*.

Semantic similarities carried forward into ontological concepts of local ontologies *LO_gp, LO_hospital, LO_clinic_1* and *LO_clinic_2*:	**Type of semantic conflicts:**	**Degrees of similarity:**
The attribute MEDICINE_NUM from the MEDICATION table in the relational schema for the *GP_Rep* and the attribute MEDICINE_NO from the MEDICATION table in the relational schema for the *Hospital_Rep* have been carried forward into: - object property gp-medication-MEDICATION_NUM of LO_gp-medication_instances ontological class in the local ontology *LO_gp*, and - object property hospital-medication-MEDICATION_NO of LO_hospital-medication_instances ontological class in the local ontology *LO_hospital*. Both object properties resemble each other and do have some semantic similarities between them.	The domain (i.e. ontological individuals) of the object properties may generate the SYNONYM based naming conflict in the retrievals of *medical summaries*.	Semantic Likeness (3)
The attribute NAME from the PATIENT table in the relational schema for the *Hospital_Rep* and the attributes FIRST_NAME, and FIRST_NAME from the PATIENT table in the relational schema for the *Clinic_2_Rep* have been carried forward into: - object property hospital-patient-NAME of the LO_hospital-patient_instances ontological class in the local ontology *LO_hospital*, and - object properties clinic_2-patient-FIRST_NAME and clinic_2-patient-LAST_NAME of LO_clinic_2-patient_instances ontological class in the local ontology *LO_clinic_2*. All the object properties resemble each other and there is some kind of aggregation between them, i.e. the aggregation of the object properties clinic_2-patient-FIRST_NAME, and clinic_2-patient-LAST_NAME may create a concept equivalent to hospital-patient.NAME object property.	The domain (i.e. ontological individuals) of the object properties may generate the AGGREGATION based structural conflict in the retrievals of *patient details*.	Semantic Likeness (3)
The attribute MEDICAL_SUMMARY from the PATIENT table in the relational schema for the *Hospital_Rep* and the attributes CURRENT_MEDICAL_SUMMARIES and the PREVIOUS_MEDICAL_SUMMARIES from the patient table in	The domain (i.e. ontological individuals) of the object properties may generate the GENERALISATION	Semantic Subset - *contains* (4)

the relational schema for the *Clinic_1_Rep* have been carried forward into: - object property hospital-patient-H_MEDICAL_SUMMARY of the LO_hospital-patient_instances ontological class in local ontology *LO_hospital*, and - object properties clinic_1-patient-CURRENT_MEDICAL_SUMMARIES and the clinic_1-patient-PREVIOUS_MEDICAL_SUMMARIES of LO_clinic_1-patient_instances ontological class in the local ontology *LO_clinic_1*. All object properties have semantic similarities through the generalisation of their characteristics, i.e. the object property hospital-patient-H_MEDICAL_SUMMARY is a super-type for the object properties clinic_1-patient-CURRENT_MEDICAL_SUMMARIES and the clinic_1-patient-PREVIOUS_MEDICAL_SUMMARIES.	based structural conflict in the retrievals of *medical summaries*.	
The attributes TREATMENT_TYPE, TREATMENT_NAME and TREATMENT_DATE from the TREATMENT table in the relational schema for the *Hospital_Rep* and the attributes TREATMENT_OVERVIEW and TREATMENT_DATE from the TREATMENT table in the relational schema for the *GP_Rep* have been carried forward into: - object properties hospital-treatment-TREATMENT_TYPE and hospital-treatment-TREATMENT_NAME of LO_hospital-treatment_instances ontological class in the local ontology *LO_hospital*, and - object properties 'gp-treatment-TREATMENT_OVERVIEW' and 'gp-treatment-DATE' of the 'LO_gp-treatment_instances' ontological class in the local ontology *LO_gp*. All object properties have semantic similarities through the specialisation of their characteristics, i.e. the object properties hospital-treatment-TRETAMENT_TYPE and hospital-treatment-TREATMENT_NAME are sub-types for the object property gp-treatment-TREATMENT_OVERVIEW.	The domain (i.e. ontological individuals) of the object properties may generate the SPECIALISATION based structural conflict in the retrievals of *treatment summaries*.	Semantic Subset - *contained within* (4)
The attributes MEDICINE_NUM, MEDICINE_NAME, VENDOR, and MNF_DESC from the MEDICATION table in the relational schema for the *GP_Rep* and the attributes MEDICINE_NO, MEDICINE_NAME, VENDOR, and MNF_ADDRESS from the MEDICATION table in the relational schema for the *Hospital_Rep* have been carried forward into: - object properties gp-medication-MEDICINE_NUM, gp-medication-MEDICINE_NAME, gp-medication-MNF_DESC and gp-medication-MNF_DESC of LO_gp-medication_instances ontological class in the local ontology *LO_gp*, and - object properties hospital-medication-MEDICINE_NO, hospital-medication-MEDICINE_NAME, hospital-medication-VENDOR and hospital-medication-MNF_DESC of	The domain (i.e. ontological individuals) of the object properties may generate the UNION INCOMPATIBILTY based structural conflict in the retrievals of *medical summaries*.	Semantic Overlapping (6)

LO_hospital-medication_instances ontological class in the local ontology *LO_hospital*. All object properties resemble each other and have some semantic similarities between them, but are not semantically equivalent to each other, i.e. modelling of medication results in having two different structures.		
The attributes LABTEST_ID, PATIENT_NO, LABTEST_TYPE, LABTEST_NAME, LABTEST_RESULTS, REPORT and DATE from the LABTEST table in the relational schema for the *Clinic_1_Rep* and the attributes LABTEST_ID, PATIENT_NO', 'LABTEST_OVERVIEW, LABTEST_DATA, LABTEST_TYPE, LABTEST_NAME, LABTEST_RESULTS, REPORT and DATE from the LABTEST table in the relational schema for the *Clinic_2_Rep* have been carried forward into: - object properties clinic_1-labtest-LABTEST_ID, clinic_1-labtest-PATIENT_NO, clinic_1-labtest-LABTEST_OVERVIEW, clinic_1-labtest-LABTEST_TYPE, clinic_1-labtest-LABTEST_NAME and clinic_1-labtest-LABTEST_RESULTS of LO_clinic_1-labtest_instances ontological class in the local ontology *LO_clinic_1*, and - object properties clinic_2-labtest-LABTEST_ID, clinic_2-labtest-PATIENT_NO, clinic_2-labtest-LABTEST_OVERVIEW, clinic_2-labtest-LABTEST_DATA, clinic_2-labtest-LABTEST_TYPE, clinic_2-labtest-LABTEST_NAME and clinic_2-labtest-LABTEST_RESULTS of LO_clinic_2-labtest_instances ontological class in the local ontology *LO_clinic_2*. All object properties resemble each other but the number of object properties used to describe lab tests carried out for a particular patient are different, i.e. the object property clinic_2-labtest-LABTEST_DATA does NOT belong to the LO_clinic_1-labtest_instances ontological class in the local ontology *LO_clinic_1*.	The domain (i.e. ontological individuals) of the object properties may generate the ISOMORPHISM based structural conflict in the retrievals of *treatment summaries*.	Semantic Overlapping (6)
The attribute REPORT from the TREATMENT table in the relational schema for the *Hospital_Rep* and the LABTEST table in the relational schema for the *Clinic_1_Rep* have been carried forward into : - object property hospital-treatment-REPORT of LO_hospital-treatment_instances ontological class in the local ontology *LO_hospital*, and - object property clinic_1-treatment.REPORT of LO_clinic_1-treatment_instances ontological class in the local ontology *LO_clinic_1*. Both object properties have the similar names, but actually model different subjects.	The domain (i.e. ontological individuals) of the object properties may generate the HYMONYM based naming conflict.	Semantic False Likeness (2),

Appendix A.8

Can be found on CD-ROM attached to thesis.

Appendix A.9

The SWRL *Selection* rules 1-8 for storing the results user involvements in the USER_INP_ONT ontology in the User Request layer of the software architecture for Go-CID software applications:

Selection rule 1:

```
USER_CLICK_gp_rep(?A) ∧ TRUTH_VARIABLE_gp_rep(?B)
→ SELECTION_gp_rep(?B)
```

Selection rule 2:

```
USER_CLICK_hospital_rep(?A) ∧ TRUTH_VARIABLE_hospital_rep(?B)
→ SELECTION_hospital_rep(?B)
```

Selection rule 3:

```
USER_CLICK_clinic_1_rep(?A) ∧ TRUTH_VARIABLE_clinic_1_rep(?B)
→ SELECTION_clinic_1_rep(?B)
```

Selection rule 4:

```
USER_CLICK_clinic_2_rep(?A) ∧ TRUTH_VARIABLE_clinic_2_rep(?B)
→ SELECTION_clinic_2_rep(?B)
```

Selection rule 5:

```
USER_CLICK_patient_details(?A) ∧ TRUTH_VARIABLE_patient_details(?B)
→ SELECTION_patient_details(?B)
```

Selection rule 6:

```
USER_CLICK_patient_details(?A) ∧ TRUTH_VARIABLE_patient_details(?B)
→ SELECTION_patient_details(?B)
```

Selection rule 7:

```
USER_CLICK_treatment_summaries(?A) ∧
TRUTH_VARIABLE_treatment_summaries(?B)
→ SELECTION_treatment_summaries(?B)
```

Selection rule 8:

```
TEXT_ENTERED_jane_flee(?A) ∧ TRUTH_VARIABLE_jane_flee(?B)
→ SELECTION_jane_flee(?B)
```

Appendix A.10

Can be found on CD-ROM attached to thesis.

Appendix A.11

SWRL *Grouping* rules 10-30 used for interpreting Dr. Smith's involvements (i.e. clicks) within the ADDED_VAL_ONT ontology in the User Request layer of our software architecture for Go-CID software applications:

Grouping rule 10:

```
SELECTION_gp_rep(?A) ∧ has_variable_gp_rep(?A, true) ∧
SELECTION_patient_details(?B) ∧ has_variable_patient_details(?B, true) ∧
gp-patient-FIRST_NAME(?C, ?D) ∧ gp-patient-LAST_NAME(?E, ?Ff) ∧
gp-patient-SEX(?G, ?H) ∧ gp-patient-DOB(?I, ?J) ∧
gp-patient-ADDRESS(?K, ?L) ∧ gp-patient-REGION(?M, ?N) ∧
gp-patient-NEXT_OF_KIN(?O, ?P) ∧ gp-patient-EMERGENCY_CONTACT(?Q, ?R) ∧
gp-patient-NO_OF_CHILDREN(?S, ?T) ∧ gp-patient-BMI(?U, ?V) ∧
gp-patient-HEIGHT(?W, ?X) ∧gp-patient-TELEFFONNE(?Y, ?Z)
→ patient_details-FROM-gp_rep (?D) ∧ patient_details-FROM-gp_rep(?Ff) ∧
patient_details-FROM-gp_rep(?H) ∧ patient_details-FROM-gp_rep(?J) ∧
patient_details-FROM-gp_rep(?L) ∧ patient_details-FROM-gp_rep(?N) ∧
patient_details-FROM-gp_rep(?P) ∧ patient_details-FROM-gp_rep(?R) ∧
patient_details-FROM-gp_rep(?T) ∧ patient_details-FROM-gp_rep(?V) ∧
patient_details-FROM-gp_rep(?X) ∧ patient_details-FROM-gp_rep(?Z)
```

Grouping rule 11:

```
SELECTION_hospital_rep(?A) ∧ has_variable_hospital_rep(?A, true) ∧
SELECTION_patient_details(?B) ∧ has_variable_patient_details(?B, true) ∧
hospital-patient-NAME(?C, ?D) ∧ hospital-patient-SEX(?E, ?Ff) ∧
hospital-patient-DOB(?G, ?H) ∧
→ patient_details-FROM-hospital_rep(?D) ∧ patient_details-FROM-
hospital_rep(?Ff) ∧ patient_details-FROM-hospital_rep(?H)
```

Grouping rule 12:

```
SELECTION_clinic_1_rep(?A) ∧ has_variable_clinic_1_rep(?A, true) ∧
SELECTION_patient_details(?B) ∧ has_variable_patient_details(?B, true) ∧
clinic_1-patient-NAME(?C, ?D) ∧ clinic_1-patient-SEX(?E, ?Ff) ∧
clinic_1-patient-TELEPHONE(?G, ?H) ∧ clinic_1-patient-DOB(?I, ?J) ∧
→ patient_details-FROM-clinic_1_rep(?D) ∧
patient_details-FROM-clinic_1_rep(?Ff) ∧
patient_details-FROM-clinic_1_rep(?H) ∧
patient_details-FROM-clinic_1_rep ?J)
```

Grouping rule 13:

```
SELECTION_clinic_2_rep(?A) ∧ has_variable_clinic_2_rep(?A, true) ∧
SELECTION_patient_details(?B) ∧ has_variable_patient_details(?B, true) ∧
clinic_2-patient-LAST_NAME(?C, ?D) ∧
clinic_2-patient-FIRST_NAME(?E, ?Ff) ∧
clinic_2-patient-SEX(?G, ?H) ∧ clinic_2-patient-DOB(?I, ?J)
→ patient_details-FROM-clinic_2_rep(?D) ∧
patient_details-FROM-clinic_2_rep(?Ff) ∧
patient_details-FROM-clinic_2_rep(?H) ∧
patient_details-FROM-clinic_2_rep(?J)
```

Grouping rule 14:

```
SELECTION_gp_rep(?A) ∧ has_variable_gp_rep(?A, true) ∧
SELECTION_hospital_rep(?B) ∧ has_variable_hospital_rep(?B, true) ∧
SELECTION_clinic_1_rep(?C) ∧ has_variable_clinic_1_rep(?C, true) ∧
SELECTION_clinic_2_rep(?D) ∧ has_variable_clinic_2_rep(?D, true) ∧
SELECTION_patient_details(?E) ∧ has_variable_patient_details(?E, true) ∧
gp-patient-FIRST_NAME(?fF, ?G) ∧ gp-patient-LAST_NAME(?H, ?I) ∧
```

```
gp-patient-SEX(?J, ?K) ∧ gp-patient-DOB(?L, ?M) ∧
gp-patient-ADDRESS(?N, ?O) ∧ gp-patient-REGION(?P, ?Q) ∧
gp-patient-NEXT_OF_KIN(?R, ?S) ∧ gp-patient-EMERGENCY_CONTACT(?T, ?U) ∧
gp-patient-NO_OF_CHILDREN(?V, ?W) ∧ gp-patient-BMI(?X, ?Y) ∧
gp-patient-HEIGHT(?Z, ?a) ∧ gp-patient-TELEPHONE(?b, ?c)
→ patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep(?G) ∧
patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep(?I) ∧
patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep(?K) ∧
patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep(?M) ∧
patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep(?O) ∧
patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep(?Q) ∧
patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep(?S) ∧
patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep(?U) ∧
patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep(?W) ∧
patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep(?Y) ∧
patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep(?a) ∧
patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep(?c)
```

Grouping rule 15:

```
SELECTION_gp_rep(?A) ∧ has_variable_gp_rep(?A, true) ∧
SELECTION_hospital_rep(?B) ∧ has_variable_hospital_rep(?B, true) ∧
SELECTION_clinic_1_rep(?C) ∧ has_variable_clinic_1_rep(?C, true) ∧
SELECTION_clinic_2_rep(?D) ∧ has_variable_clinic_2_rep(?D, true) ∧
SELECTION_patient_details(?E) ∧ has_variable_patient_details(?E, true) ∧
hospital-patient-NAME(?fF, ?G) ∧ hospital-patient-SEX(?H, ?I) ∧
hospital-patient-DOB(?J, ?K)
→ patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep(?G) ∧
patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep(?I) ∧
patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep(?K)
```

Grouping rule 16:

```
SELECTION_gp_rep(?A) ∧ has_variable_gp_rep(?A, true) ∧
SELECTION_hospital_rep(?B) ∧ has_variable_hospital_rep(?B, true) ∧
SELECTION_clinic_1_rep(?C) ∧ has_variable_clinic_1_rep(?C, true) ∧
SELECTION_clinic_2_rep(?D) ∧ has_variable_clinic_2_rep(?D, true) ∧
SELECTION_patient_details(?E) ∧ has_variable_patient_details(?E, true) ∧
clinic_1-patient-NAME(?fF, ?G) ∧ clinic_1-patient-SEX(?H, ?I) ∧
clinic_1-patient-TELEPHONE(?J, ?K) ∧ clinic_1-patient-DOB(?L, ?M)
→ patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep(?G) ∧
patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep(?I) ∧
patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep(?K) ∧
patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep(?M)
```

Grouping rule 17:

```
SELECTION_gp_rep(?A) ∧ has_variable_gp_rep(?A, true) ∧
SELECTION_hospital_rep(?B) ∧ has_variable_hospital_rep(?B, true) ∧
SELECTION_clinic_1_rep(?C) ∧ has_variable_clinic_1_rep(?C, true) ∧
SELECTION_clinic_2_rep(?D) ∧ has_variable_clinic_2_rep(?D, true) ∧
SELECTION_patient_details(?E) ∧ has_variable_patient_details(?E, true) ∧
clinic_2-patient-LAST_NAME(?fF, ?G) ∧
clinic_2-patient-FIRST_NAME(?H, ?I) ∧
clinic_2-patient-SEX(?J, ?K) ∧ clinic_2-patient-DOB(?L, ?M)
→ patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep(?G) ∧
patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep(?I) ∧
patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep(?K) ∧
patient_details-FROM-gp--hospital--clinic_1--clinic_2_rep(?M)
```

Grouping rule 18:

```
SELECTION_gp_rep(?A) ∧ has_variable_gp_rep(?A, true) ∧
SELECTION_medical_summaries(?B) ∧
```

```
has_variable_medical_summaries(?B, true) ∧ NO_information_retrievals(?C)
→ medical_summaries-FROM-gp_rep(?C)
```

Grouping 19:

```
SELECTION_hospital_rep(?A) ∧ has_variable_hospital_rep(?A, true) ∧
SELECTION_medical_summaries(?B) ∧
has_variable_medical_summaries(?B, true) ∧
hospital-patient-MEDICAL_SUMMARY(?C, ?D) ∧
hospital-patient-MAJOR_ILLNESS(?E, ?Ff) ∧
hospital-patient-CHRONIC_DISEASE(?G, ?H)
→ medical_summaries-FROM-hospital_rep(?D) ∧
medical_summaries-FROM-hospital_rep(?Ff) ∧
medical_summaries-FROM-hospital_rep(?H)
```

Grouping rule 20:

```
SELECTION_clinic_1_rep(?A) ∧ has_variable_clinic_1_rep(?A, true) ∧
SELECTION_medical_summaries(?B) ∧
has_variable_medical_summaries(?B, true) ∧
clinic_1-patient-PREVIOUS_MEDICAL_SUMMARY(?C, ?D) ∧
clinic_1-patient-CURRENT_MEDICAL_SUMMARY(?E, ?Ff) ∧
clinic_1-patient-MAJOR_ILLNESS(?G, ?H) ∧
clinic_1-patient-CHRONIC_DISEASE(?I, ?J)  ∧
clinic_1-labtest-LABTEST_TYPE(?K, ?L) ∧
clinic_1-labtest-LABTEST_NAME(?M, ?N) ∧
clinic_1-labtest-LABTEST_RESULTS(?O, ?P) ∧
clinic_1-labtest-LABTEST_DATE(?Q, ?R)
  → medical_summaries-FROM-clinic_1_rep(?D) ∧
medical_summaries-FROM-clinic_1_rep(?Ff) ∧
medical_summaries-FROM-clinic_1_rep(?H) ∧
medical_summaries-FROM-clinic_1_rep(?J) ∧
medical_summaries-FROM-clinic_1_rep(?L) ∧
medical_summaries-FROM-clinic_1_rep(?N) ∧
medical_summaries-FROM-clinic_1_rep(?P) ∧
medical_summaries-FROM-clinic_1_rep(?R)
```

Grouping rule 21:

```
SELECTION_clinic_2_rep(?A) ∧ has_variable_clinic_2_rep(?A, true) ∧
SELECTION_medical_summaries(?B) ∧
has_variable_medical_summaries(?B, true) ∧
clinic_2-patient-MEDICAL_SUMMARY(?C, ?D) ∧
clinic_2-patient-MAJOR_ILLNESS(?E, ?Ff) ∧
clinic_2-patient-CHRONIC_DISEASE(?G, ?H) ∧
clinic_2-labtest-LABTEST_OVERVIEW(?I, ?J) ∧
clinic_2-labtest-LABTEST_TYPE(?K, ?L) ∧
clinic_2-labtest-LABTEST_NAME(?M, ?N) ∧
clinic_2-labtest-LABTEST_RESULTS(?O, ?P) ∧
clinic_2-labtest-LABTEST_DATE(?Q, ?R)  ∧
clinic_2-labtest-LABTEST_DATA(?S, ?T)
→ medical_summaries-FROM-clinic_2_rep(?D)
∧ medical_summaries-FROM-clinic_2_rep(?Ff) ∧
medical_summaries-FROM-clinic_2_rep(?H) ∧
medical_summaries-FROM-clinic_2_rep(?J) ∧
medical_summaries-FROM-clinic_2_rep(?L) ∧
medical_summaries-FROM-clinic_2_rep(?N) ∧
medical_summaries-FROM-clinic_2_rep(?P) ∧
medical_summaries-FROM-clinic_2_rep(?R) ∧
medical_summaries-FROM-clinic_2_rep(?T)
```

Grouping rule 22:

```
SELECTION_gp_rep(?A) ∧ has_variable_gp_rep(?A, true) ∧
SELECTION_hospital_rep(?B) ∧ has_variable_hospital_rep(?B, true) ∧
SELECTION_clinic_1_rep(?C) ∧ has_variable_clinic_1_rep(?C, true) ∧
SELECTION_clinic_2_rep(?D) ∧ has_variable_clinic_2_rep(?D, true) ∧
SELECTION_medical_summaries(?E) ∧
has_variable_medical_summaries(?E, true) ∧
hospital-patient-MEDICAL_SUMMARY(?fF, ?G) ∧
hospital-patient-MAJOR_ILLNESS(?H, ?I) ∧
hospital-patient-CHRONIC_DISEASE(?J, ?K)
→ medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?G) ∧
medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?I) ∧
medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?K)
```

Grouping rule 23:

```
SELECTION_gp_rep(?A) ∧ has_variable_gp_rep(?A, true) ∧
SELECTION_hospital_rep(?B) ∧ has_variable_hospital_rep(?B, true) ∧
SELECTION_clinic_1_rep(?C) ∧ has_variable_clinic_1_rep(?C, true) ∧
SELECTION_clinic_2_rep(?D) ∧ has_variable_clinic_2_rep(?D, true) ∧
SELECTION_medical_summaries(?E) ∧
has_variable_medical_summaries(?E, true) ∧
clinic_1-patient-PREVIOUS_MEDICAL_SUMMARY(?fF, ?G) ∧
clinic_1-patient-CURRENT_MEDICAL_SUMMARY(?H, ?I) ∧
clinic_1-patient-MAJOR_ILLNESS(?J, ?K) ∧
clinic_1-patient-CHRONIC_DISEASE(?L, ?M) ∧
clinic_1-labtest-LABTEST_TYPE(?N, ?O) ∧
clinic_1-labtest-LABTEST_NAME(?P, ?Q) ∧
clinic_1-labtest-LABTEST_RESULTS(?R, ?S) ∧
clinic_1-labtest-LABTEST_DATE(?T, ?U)
→ medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?G) ∧
medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?I) ∧
medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?K) ∧
medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?M) ∧
medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?O) ∧
medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?Q) ∧
medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?S) ∧
medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?U)
```

Grouping rule 24:

```
SELECTION_gp_rep(?A) ∧ has_variable_gp_rep(?A, true) ∧
SELECTION_hospital_rep(?B) ∧ has_variable_hospital_rep(?B, true) ∧
SELECTION_clinic_1_rep(?C) ∧ has_variable_clinic_1_rep(?C, true) ∧
SELECTION_clinic_2_rep(?D) ∧ has_variable_clinic_2_rep(?D, true) ∧
SELECTION_medical_summaries(?E) ∧ has_variable_medical_summaries(?E, true)
∧
clinic_2-patient-MEDICAL_SUMMARY(?fF, ?G) ∧
clinic_2-patient-MAJOR_ILLNESS(?H, ?I) ∧
clinic_2-patient-CHRONIC_DISEASE(?J, ?K) ∧
clinic_2-labtest-LABTEST_OVERVIEW(?L, ?M) ∧
clinic_2-labtest-LABTEST_TYPE(?N, ?O) ∧
clinic_2-labtest-LABTEST_NAME(?P, ?Q) ∧
clinic_2-labtest-LABTEST_RESULTS(?R, ?S) ∧
clinic_2-labtest-LABTEST_DATE(?T, ?U)  ∧
clinic_2-labtest-LABTEST_DATA(?V, ?W)
→ medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?G) ∧
medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?I) ∧
medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?K) ∧
medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?M) ∧
medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?O) ∧
```

```
medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?Q) ∧
medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?S) ∧
medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?U) ∧
medical_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?W)
```

Grouping 25:

```
SELECTION_gp_rep(?A) ∧ has_variable_gp_rep(?A, true) ∧
SELECTION_treatment_summaries(?B) ∧
has_variable_treatment_summaries(?B, true) ∧
gp-medication-MEDICINE_NAME(?C, ?D) ∧ gp-medication-VENDOR(?E, ?Ff) ∧
gp-medication-MNF_DESC(?G, ?H) ∧
gp-medication_prescribed-DOSAGE_AMOUNT(?I, ?J) ∧
gp-treatment-TREATMENT_OVERVIEW(?K, ?L) ∧ gp-treatment-DATE(?M, ?N) ∧
gp-medication-MEDICINE_NUM(?O, ?P)
→ treatment_summaries-FROM-gp_rep(?D) ∧
treatment_summaries-FROM-gp_rep(?Ff) ∧
treatment_summaries-FROM-gp_rep(?H) ∧
treatment_summaries-FROM-gp_rep(?J) ∧
treatment_summaries-FROM-gp_rep(?L) ∧
treatment_summaries-FROM-gp_rep(?N) ∧
treatment_summaries-FROM-gp_rep(?P)
```

Grouping 26:

```
SELECTION_hospital_rep(?A) ∧ has_variable_hospital_rep(?A, true) ∧
SELECTION_treatment_summaries(?B) ∧
has_variable_treatment_summaries(?B, true) ∧
hospital-medication-MEDICATION_NAME(?C, ?D) ∧
hospital-medication-VENDOR(?E, ?Ff) ∧
hospital-medication-MNF_ADDRESS(?G, ?H) ∧
hospital-medication_prescribed-DOSAGE_AMOUNT(?I, ?J) ∧
hospital-treatment-TREATMENT_TYPE(?K, ?L) ∧
hospital-treatment-TREATMENT_NAME(?M, ?N) ∧
hospital-treatment-DATE(?O, ?P) ∧ hospital-medication-MEDICINE_NO(?Q, ?R)
→ treatment_summaries-FROM-hospital_rep(?D) ∧
treatment_summaries-FROM-hospital_rep(?Ff) ∧
treatment_summaries-FROM-hospital_rep(?H) ∧
treatment_summaries-FROM-hospital_rep(?J) ∧
treatment_summaries-FROM-hospital_rep(?L) ∧
treatment_summaries-FROM-hospital_rep(?N) ∧
treatment_summaries-FROM-hospital_rep(?P) ∧
treatment_summaries-FROM-hospital_rep(?R)
```

Grouping rule 27:

```
SELECTION_clinic_1_rep(?A) ∧ has_variable_clinic_1_rep(?A, true) ∧
SELECTION_treatment_summaries(?B) ∧
has_variable_treatment_summaries(?B, true) ∧ NO_information_retrievals(?C)
→ treatment_summaries-FROM-clinic_1_rep(?C)
```

Grouping 28:

```
SELECTION_clinic_2_rep(?A) ∧ has_variable_clinic_2_rep(?A, true) ∧
SELECTION_treatment_summaries(?B) ∧
has_variable_treatment_summaries(?B, true)∧ NO_information_retrievals(?C)
→ treatment_summaries-FROM-clinic_2_rep(?C)
```

Grouping rule 29:

```
SELECTION_gp_rep(?A) ∧ has_variable_gp_rep(?A, true) ∧
SELECTION_hospital_rep(?B) ∧ has_variable_hospital_rep(?B, true) ∧
SELECTION_clinic_1_rep(?C) ∧ has_variable_clinic_1_rep(?C, true) ∧
SELECTION_clinic_2_rep(?D) ∧ has_variable_clinic_2_rep(?D, true) ∧
SELECTION_treatment_summaries(?E) ∧
```

```
has_variable_treatment_summaries(?E, true) ∧
gp-medication-MEDICINE_NAME(?fF, ?G) ∧ gp-medication-VENDOR(?H, ?I) ∧
gp-medication-MNF_DESC(?J, ?K) ∧
gp-medication_prescribed-DOSAGE_AMOUNT(?L, ?M) ∧
gp-treatment-TREATMENT_OVERVIEW(?N, ?O) ∧ gp-treatment-DATE(?P, ?Q) ∧
gp-medication-MEDICINE_NUM(?R, ?S)
→ treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?G) ∧
treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?I) ∧
treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?K) ∧
treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?M) ∧
treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?O) ∧
treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?Q) ∧
treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?S)
```

Grouping rule 30:

```
SELECTION_gp_rep(?A) ∧ has_variable_gp_rep(?A, true) ∧
SELECTION_hospital_rep(?B) ∧ has_variable_hospital_rep(?B, true) ∧
SELECTION_clinic_1_rep(?C) ∧ has_variable_clinic_1_rep(?C, true) ∧
SELECTION_clinic_2_rep(?D) ∧ has_variable_clinic_2_rep(?D, true) ∧
SELECTION_treatment_summaries(?E) ∧
has_variable_treatment_summaries(?E, true) ∧
hospital-medication-MEDICATION_NAME(?fF, ?G) ∧
hospital-medication-VENDOR(?H, ?I) ∧
hospital-medication-MNF_ADDRESS(?J, ?K) ∧
hospital-medication_prescribed-DOSAGE_AMOUNT(?L, ?M) ∧
hospital-treatment-TREATMENT_TYPE(?N, ?O) ∧
hospital-treatment-TREATMENT_NAME(?P, ?Q) ∧
hospital-treatment-DATE(?R, ?S) ∧ hospital-medication-MEDICINE_NO(?T, ?U)
→ treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?G) ∧
treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?I) ∧
treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?K) ∧
treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?M) ∧
treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?O) ∧
treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?Q) ∧
treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?S) ∧
treatment_summaries-FROM-gp--hospital--clinic_1--clinic_2_rep(?U)
```

Appendix A.12

Can be found on CD-ROM attached to thesis.

Appendix A.13

Can be found on CD-ROM attached to thesis.

Appendix A.14

SWRL *Low-Level* rules 31- 40 used for aligning semantically related data into Target Ontologies in the Target ontological layer of our software architecture for Go-CID software applications:

Low-level reasoning rule 31:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
```

```
LO_gp-patient_instances(JANE) ∧ LO_gp-patient_instances(FLEE) ∧
LO_gp-patient_instances(FEMALE) ∧ LO_gpatient_instances(JULY_04_1970) ∧
LO_hospital-patient_instances(JANE_FLEE.)∧
LO_hospital-patient_instances(Female....) ∧
LO_hospital-patient_instances(JULY_04_1970..) ∧
has_same_FIRST_NAME(?fF, "JANE") ∧ has_same_LAST_NAME(?G, "FLEE")
→ TO_1(JANE) ∧ TO_1(FLEE) ∧ TO_1(FEMALE) ∧ TO_1(JULY_04_1970) ∧
TO_1(Female....) ∧ TO_1(JULY_04_1970..)
```

Low-level reasoning rule 32:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true)  ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
LO_gp-patient_instances(JANE) ∧ LO_gp-patient_instances(FLEE) ∧
LO_gp-patient_instances(FEMALE) ∧
LO_gppatient_instances(JULY_04_1970) ∧
LO_gp-patient_instances(TEL_02075698899) ∧
LO_clinic_1-patient_instances(JANE_FLEE) ∧
LO_clinic_1-patient_instances(Female......) ∧
LO_clinic_1-patient_instances(JULY_4_1970) ∧
LO_clinic_1-patient_instances(Tel_02075698899.) ∧
has_same_FIRST_NAME(?fF, "JANE") ∧ has_same_LAST_NAME(?G, "FLEE")
→ TO_2(JANE) ∧ TO_2(FLEE) ∧ TO_2(FEMALE) ∧ TO_2(JULY_04_1970) ∧
TO_2(TEL_02075698899) ∧ TO_2(Female......) ∧ TO_2(JULY_4_1970) ∧
TO_2(Tel_02075698899.)
```

Low-level reasoning rule 33:

```
SELECTION_jane_flee(?A)  ∧ has_variable_jane_flee(?A, true)  ∧
SELECTION_gp_rep(?B)  ∧ has_variable_gp_rep(?B, true)  ∧
SELECTION_hospital_rep(?C)  ∧ has_variable_hospital_rep(?C, true)  ∧
SELECTION_clinic_1_rep(?D)  ∧ has_variable_clinic_1_rep(?D, true)  ∧
SELECTION_clinic_2_rep(?E)  ∧ has_variable_clinic_2_rep(?E, true)  ∧
LO_hospital-patient_instances(JANE_FLEE.)  ∧
LO_hospital-patient_instances(Female....)  ∧
LO_hospital-patient_instances(JULY_04_1970..)  ∧
LO_clinic_2-patient_instances(JANE.)  ∧
LO_clinic_2-patient_instances(FLEE.)  ∧
LO_clinic_2-patient_instances(FEMALE...)  ∧
LO_clinic_2-patient_instances(July_04_1970...)  ∧
has_same_FIRST_NAME(?fF, "JANE") ∧ has_same_LAST_NAME(?G, "FLEE")
→ TO_3(JANE.)  ∧ TO_3(FLEE.)  ∧ TO_3(Female....)  ∧ TO_3(JULY_04_1970..)
∧ TO_3(FEMALE...)  ∧ TO_3(July_04_1970...)
```

Low-level reasoning rules 34:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true)  ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
LO_clinic_1-patient_instances(JANE_FLEE) ∧
LO_clinic_1-patient_instances(Female......) ∧
LO_clinic_1-patient_instances(JULY_4_1970) ∧
LO_clinic_2-patient_instances(JANE.) ∧
LO_clinic_2-patient_instances(FLEE.) ∧
LO_clinic_2-patient_instances(FEMALE...) ∧
LO_clinic_2-patient_instances(July_04_1970...) ∧
```

```
has_same_FIRST_NAME(?fF, "JANE") ∧ has_same_LAST_NAME(?G, "FLEE")
→ TO_4(JANE.) ∧ TO_4(FLEE.) ∧ TO_4(Female......) ∧
TO_4(JULY_4_1970) ∧ TO_4(FEMALE...) ∧ TO_4(July_04_1970...)
```

Low-level rule 35:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
LO_hospital-
patient_instances(Mrs_Flee_complains_of_severe_pain_in_left_ankle_Minor_sw
elling_evident_and_xrays_taken_admitted_as_overnight_stay_and_found_to_hav
e_acute_COPD_exacerbation) ∧
LO_hospital-patient_instances(no_major_illnness_evident) ∧
LO_hospital-patient_instances(no_chronic_disease_evident) ∧
LO_clinic_1-
patient_instances(Mrs_Flee_has_a_regular_cervical_smear_test_results_appea
r_normal) ∧
LO_clinic_1-
patient_instances(Mrs_Flee_has_come_into_the_clinic_for_a_blood_test_compl
ains_of_fatigue) ∧
LO_clinic_1-patient_instances(none) ∧
LO_clinic_1-patient_instances(none_found)
→
TO_5(Mrs_Flee_complains_of_severe_pain_in_left_ankle_Minor_swelling_eviden
t_and_xrays_taken_admitted_as_overnight_stay_and_found_to_have_acute_COPD_
exacerbation ∧
TO_5(Mrs_Flee_has_a_regular_cervical_smear_test_results_appear_normal) ∧
TO_5(Mrs_Flee_has_come_into_the_clinic_for_a_blood_test_complains_of_fatig
ue) ∧ TO_5(no_major_illnness_evident) ∧ TO_5(no_chronic_disease_evident) ∧
TO_5(none) ∧ TO_5(none_found)
```

Low-level rule 36:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
LO_clinic_1-
patient_instances(Mrs_Flee_has_a_regular_cervical_smear_test_results_appea
r_normal) ∧
LO_clinic_1-
patient_instances(Mrs_Flee_has_come_into_the_clinic_for_a_blood_test_compl
ains_of_fatigue) ∧
LO_clinic_1-patient_instances(none) ∧
LO_clinic_1-patient_instances(none_found) ∧
LO_clinic_2-patient_instances(Mrs_Flee_complains_of_shortness_of_breath) ∧
LO_clinic_2-patient_instances(No_MJ.) ∧
LO_clinic_2-patient_instances(no_cd_found)
→ TO_6(Mrs_Flee_has_a_regular_cervical_smear_test_results_appear_normal)∧
TO_6(Mrs_Flee_has_come_into_the_clinic_for_a_blood_test_complains_of_fatig
ue) ∧
TO_6(Mrs_Flee_complains_of_shortness_of_breath) ∧
TO_6(none) ∧ TO_6(none_found) ∧ TO_6(No_MJ.) ∧ TO_6(no_cd_found)
```

Low-level rule 37:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
```

```
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
LO_clinic_1-labtest_instances(LT256_Smear_test) ∧
LO_clinic_1-labtest_instances(LT256_Cervical_Type_3) ∧
LO_clinic_1-labtest_instances(LT256_Normal) ∧
LO_clinic_1-labtest_instances(LT123_16-02-08) ∧
LO_clinic_1-labtest_instances(LT123_Pathology) ∧
LO_clinic_1-labtest_instances(LT123_Blood_test_Type_4123) ∧
LO_clinic_1-labtest_instances(LT123_anaemia_level_46) ∧
LO_clinic_1-labtest_instances(LT256_16-01-08) ∧
LO_clinic_2-labtest_instances(LL456_Used_to_identify__lungs_diseases) ∧
LO_clinic_2-labtest_instances(LL456_Radiation) ∧
LO_clinic_2-labtest_instances(LL456_X_ray) ∧
LO_clinic_2-labtest_instances(LL456_fileID_wavelength908) ∧
LO_clinic_2-labtest_instances(LL456_28-04-09) ∧
LO_clinic_2-labtest_instances(LL456_data_aa2)
→ TO_7(LT256_Smear_test) ∧ TO_7(LT256_Cervical_Type_3) ∧
TO_7(LT256_Normal) ∧ TO_7(LT123_16-02-08) ∧
TO_7(LT123_Pathology) ∧ TO_7(LT123_Blood_test_Type_4123) ∧
TO_7(LT123_anaemia_level_46) ∧ TO_7(LT256_16-01-08) ∧
TO_7(LL456_Used_to_identify__lungs_diseases) ∧ TO_7(LL456_Radiation) ∧
TO_7(LL456_X_ray) ∧ TO_7(LL456_fileID_wavelength908) ∧
TO_7(LL456_28-04-09) ∧ TO_7(LL456_data_aa2)
```

Low-level rule 38:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
LO_gp-medication_instances(M0031) ∧
LO_hospital-medication_instances(M222p) ∧
LO_hospital-medication_instances(M225i) ∧
has_same_UNIQUE_IDENTIFIER1(?fF, "M0031") ∧ has_same_UNIQUE_IDENTIFIER2(?G,
"M222p") ∧ has_same_UNIQUE_IDENTIFIER3(?H, "M225i")
→ TO_8(M0031) ∧ TO_8(M222p) ∧ TO_8(M225i)
```

Low-level rule 39:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
LO_gp-
treatment_instances(TT1989_Patient_is_suffering_from_aches_in_lower_limbs_
and_has_minor_swelling_to_ankle_pain_support_through_chronic_pain_recovery
_is_suggested) ∧
LO_gp-treatment_instances(TT1989_12-03-09) ∧
LO_hospital-treatment_instances(T09851_COPD_Chronic_pain_recovery) ∧
LO_hospital-treatment_instances(T01245_COPDT_exacerbation) ∧
LO_hospital-treatment_instances(T09851_17-04-09)
→
TO_9(TT1989_Patient_is_suffering_from_aches_in_lower_limbs_and_has_minor_s
welling_to_ankle_pain_support_through_chronic_pain_recovery_is_suggested)
∧ TO_9(TT1989_12-03-09) ∧ TO_9(T09851_COPD_Chronic_pain_recovery) ∧
TO_9(T01245_COPDT_exacerbation) ∧ TO_9(T09851_17-04-09)
```

Low-level reasoning rule 40:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
```

```
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
LO_gp-medication_instances(M0031_Capzasin) ∧
LO_gp-medication_instances(M0031_Xhing_Ltd) ∧
LO_gp-medication_instances(M0031_China_pharmaceuticals) ∧
LO_gp-medication_prescribed_instances(M0031_2_tablets_per_day) ∧
LO_hospital-medication_instances(M222p_Andheri_east_India) ∧
LO_hospital-medication_instances(M225i_EHOSUXIMIDE) ∧
LO_hospital-medication_instances(M225i_Emeside) ∧
LO_hospital-medication_instances(M225i_South_coast_Canada) ∧
LO_hospital-medication_instances(M222p_NAPROXEN) ∧
LO_hospital-medication_instances(M222p_Risedronate) ∧
LO_hospital-
medication_prescribed_instances(M222p_1_or_2_tablets_to_be_taken_4_times_a
_day) ∧
LO_hospital-
medication_prescribed_instances(M225i_1_tablets_to_be_taken_4_times_a_day)
→ TO_10(M0031_Capzasin) ∧ TO_10(M0031_Xhing_Ltd) ∧
TO_10(M0031_China_pharmaceuticals) ∧ TO_10(M0031_2_tablets_per_day) ∧
TO_10(M222p_Andheri_east_India) ∧ TO_10(M225i_EHOSUXIMIDE) ∧
TO_10(M225i_Emeside) ∧ TO_10(M225i_South_coast_Canada) ∧
TO_10(M222p_NAPROXEN) ∧ TO_10(M222p_Risedronate) ∧
TO_10(M222p_1_or_2_tablets_to_be_taken_4_times_a_day) ∧
TO_10(M225i_1_tablets_to_be_taken_4_times_a_day)
```

Appendix A.15

Can be found on CD-ROM attached to thesis.

Appendix A.16

SWRL *High-Level* rules 41-50 used for integrating semantically similar data into Derived Ontologies in the Derived ontological layer of our software architecture for Go-CID software applications:

High-level reasoning rule 41:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true)  ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true)  ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true)  ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true)  ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true)  ∧
TO_1(JANE) ∧ TO_2(JANE)  ∧ TO_3(JANE.) ∧ TO_4(JANE.)
→ DO_1(JANE) ∧ DO_1(JANE) ∧ DO_1(JANE.) ∧ DO_1(JANE.)
```

High-level reasoning rule 42:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
TO_1(FLEE) ∧ TO_2(FLEE) ∧ TO_3(FLEE.) ∧ TO_4(FLEE.)
→ DO_2(FLEE) ∧ DO_2(FLEE) ∧ DO_2(FLEE.) ∧ DO_2(FLEE.)
```

High-level reasoning rule 43:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
```

```
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
TO_1(FEMALE) ∧ TO_1(Female....) ∧ TO_2(FEMALE) ∧
TO_2(Female......) ∧ TO_3(FEMALE...) ∧ TO_3(Female....) ∧
TO_4(FEMALE...) ∧ TO_4(Female......)
→ DO_3(FEMALE) ∧ DO_3(Female....) ∧ DO_3(FEMALE) ∧ DO_3(Female......) ∧
DO_3(FEMALE...) ∧ DO_3(Female....) ∧
DO_3(FEMALE...) ∧ DO_3(Female......)
```

High-level reasoning rule 44:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
TO_1(JULY_04_1970) ∧ TO_1(JULY_04_1970..) ∧ TO_2(JULY_04_1970) ∧
TO_2(JULY_4_1970) ∧ TO_3(JULY_04_1970..) ∧ TO_3(July_04_1970...) ∧
TO_4(JULY_4_1970) ∧ TO_4(July_04_1970...)
→ DO_4(JULY_04_1970) ∧ DO_4(JULY_04_1970..) ∧ DO_4(JULY_04_1970) ∧
DO_4(JULY_4_1970) ∧ DO_4(JULY_04_1970..) ∧ DO_4(July_04_1970...) ∧
DO_4(JULY_4_1970) ∧ DO_4(July_04_1970...)
```

High-level reasoning rule 45:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
TO_5(Mrs_Flee_complains_of_severe_pain_in_left_ankle_Minor_swelling_eviden
t_and_xrays_taken_admitted_as_overnight_stay_and_found_to_have_acute_COPD_
exacerbation) ∧
TO_5(Mrs_Flee_has_come_into_the_clinic_for_a_blood_test_complains_of_fatig
ue) ∧
TO_6(Mrs_Flee_has_come_into_the_clinic_for_a_blood_test_complains_of_fatig
ue) ∧
TO_6(Mrs_Flee_complains_of_shortness_of_breath)
  →
DO_5(Mrs_Flee_complains_of_severe_pain_in_left_ankle_Minor_swelling_eviden
t_and_xrays_taken_admitted_as_overnight_stay_and_found_to_have_acute_COPD_
exacerbation) ∧
DO_5(Mrs_Flee_has_come_into_the_clinic_for_a_blood_test_complains_of_fatig
ue) ∧
DO_5(Mrs_Flee_complains_of_shortness_of_breath)
```

High-level reasoning rule 46:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
TO_5(no_major_illnness_evident) ∧ TO_5(none) ∧ TO_6(none) ∧
TO_6(No_MJ.)
→ DO_6(no_major_illnness_evident) ∧ DO_6(none) ∧ DO_6(none) ∧
DO_6(No_MJ.)
```

High-level reasoning rule 47:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
```

```
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
TO_5(no_chronic_disease_evident) ∧ TO_5(none_found) ∧
TO_6(none_found) ∧ TO_6(no_cd_found)
→ DO_7(no_chronic_disease_evident) ∧ DO_7(none_found) ∧
DO_7(none_found) ∧ DO_7(no_cd_found)
```

High-level reasoning rule 48:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
TO_7(LT256_Smear_test) ∧ TO_7(LT256_Cervical_Type_3) ∧
TO_7(LT256_Normal) ∧ TO_7(LT123_16-02-08) ∧ TO_7(LT123_Pathology) ∧
TO_7(LT123_Blood_test_Type_4123) ∧ TO_7(LT123_anaemia_level_46) ∧
TO_7(LT256_16-01-08) ∧ TO_7(LL456_Radiation) ∧ TO_7(LL456_X_ray) ∧
TO_7(LL456_fileID_wavelength908) ∧ TO_7(LL456_28-04-09)
→ DO_8(LT256_Smear_test) ∧ DO_8(LT256_Cervical_Type_3) ∧
DO_8(LT256_Normal) ∧ DO_8(LT123_16-02-08) ∧ DO_8(LT123_Pathology) ∧
DO_8(LT123_Blood_test_Type_4123) ∧ DO_8(LT123_anaemia_level_46) ∧
DO_8(LT256_16-01-08) ∧ DO_8(LL456_Radiation)∧ DO_8(LL456_X_ray) ∧
DO_8(LL456_fileID_wavelength908) ∧ DO_8(LL456_28-04-09)
```

High-level reasoning rule 49:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
TO_9(TT1989_12-03-09) ∧ TO_9(T09851_17-04-09)
→ DO_9(TT1989_12-03-09) ∧ DO_9(T09851_17-04-09)
```

High-level reasoning rule 50:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B)  ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
TO_10(M0031_Capzasin) ∧ TO_10(M0031_Xhing_Ltd) ∧
TO_10(M0031_2_tablets_per_day) ∧ TO_10(M225i_EHOSUXIMIDE) ∧
TO_10(M225i_Emeside) ∧ TO_10(M222p_NAPROXEN) ∧
TO_10(M222p_Risedronate) ∧
TO_10(M222p_1_or_2_tablets_to_be_taken_4_times_a_day) ∧
TO_10(M225i_1_tablets_to_be_taken_4_times_a_day)
→ DO_10(M0031_Capzasin) ∧ DO_10(M0031_Xhing_Ltd) ∧
DO_10(M0031_2_tablets_per_day) ∧ DO_10(M225i_EHOSUXIMIDE) ∧
DO_10(M225i_Emeside) ∧ DO_10(M222p_NAPROXEN) ∧
DO_10(M222p_Risedronate) ∧
DO_10(M222p_1_or_2_tablets_to_be_taken_4_times_a_day) ∧
DO_10(M225i_1_tablets_to_be_taken_4_times_a_day)
```

Appendix A.17

Can be found on CD-ROM attached to thesis.

Appendix A.18

SWRL *Post-High-Level* rules 51-61 used for merging semantically equivalent into Go-CID in the Go-CID ontological layer of our software architecture for Go-CID software applications:

Post-High-Level rule 51:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
DO_1(JANE)
→ FIRST_NAME(JANE)
```

Post-High-Level rule 52:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
DO_2(FLEE)
→ LAST_NAME(FLEE)
```

Post-High-Level rule 53:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
DO_3(FEMALE)
→ SEX(FEMALE)
```

Post-High-level rule 54:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
DO_4(JULY_04_1970)
→ DOB(JULY_04_1970)
```

Post-High-level rule 55:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
LO_gp-patient_instances(ADD_167_BOULEVARD_RD_W1W_5TU)
→ ADDRESS(ADD_167_BOULEVARD_RD_W1W_5TU)
```

Post-High-level rule 56:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
LO_gp-patient_instances(LONDON)
→ REGION(LONDON)
```

Post-High-Level rule 57:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
LO_gp-patient_instances(NEMANJA_FLEE)
→ NEXT_OF_KIN(NEMANJA_FLEE)
```

Post-High-level rule 58:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
LO_gp-patient_instances(TEL_07965896456)
→ EMERGENCY_CONTACT(TEL_07965896456)
```

Post-High-level rule 59:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
LO_gp-patient_instances(CHILDREN_0)
→ No_OF_CHILDREN(CHILDREN_0)
```

Post-High-level rule 60:

```
SELECTION_jane_flee(?A)  ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
LO_gp-patient_instances(NORMAL)
→ BMI(NORMAL)
```

Post-High-level rule 61:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
LO_gp-patient_instances(H_5_feet_8_inches)
→ HEIGHT(H_5_feet_8_inches)
```

Post-High-level rule 62:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
DO_5(Mrs_Flee_complains_of_severe_pain_in_left_ankle_Minor_swelling_eviden
t_and_xrays_taken_admitted_as_overnight_stay_and_found_to_have_acute_COPD_
exacerbation) ∧
DO_5(Mrs_Flee_has_come_into_the_clinic_for_a_blood_test_complains_of_fatig
ue) ∧ DO_5(Mrs_Flee_complains_of_shortness_of_breath)
→
SUMMARIES(Mrs_Flee_complains_of_severe_pain_in_left_ankle_Minor_swelling_e
vident_and_xrays_taken_admitted_as_overnight_stay_and_found_to_have_acute_
COPD_exacerbation) ∧
```

```
SUMMARIES(Mrs_Flee_has_come_into_the_clinic_for_a_blood_test_complains_of_
fatigue) ∧ SUMMARIES(Mrs_Flee_complains_of_shortness_of_breath)
```

Post-High-level rule 63:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
DO_6(no_major_illnness_evident)
→ MAJOR_ILLNESS(no_major_illnness_evident)
```

Post-High-level rule 64:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
DO_7(no_chronic_disease_evident)
→ CHRONIC_DISEASE(no_chronic_disease_evident)
```

Post-High-level rule 65:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
DO_8(LT123_Blood_test_Type_4123) ∧ DO_8(LL456_Radiation) ∧
DO_8(LT256_Smear_test)
  → LABTEST_TYPE(LT123_Blood_test_Type_4123) ∧
LABTEST_TYPE(LL456_Radiation) ∧ LABTEST_TYPE(LT256_Smear_test)
```

Post-High-level rule 66:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
DO_8(LT123_Pathology) ∧ DO_8(LT256_Cervical_Type_3) ∧ DO_8(LL456_X_ray)
→ LABTEST_NAME(LT123_Pathology) ∧ LABTEST_NAME(LT256_Cervical_Type_3) ∧
LABTEST_NAME(LL456_X_ray)
```

Post-High-level rule 67:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
DO_8(LL456_fileID_wavelength908) ∧ DO_8(LT123_anaemia_level_46) ∧
DO_8(LT256_Normal)
→ LABTEST_RESULTS(LL456_fileID_wavelength908) ∧
LABTEST_RESULTS(LT123_anaemia_level_46) ∧ LABTEST_RESULTS(LT256_Normal)
```

Post-High-level rule 68:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
```

```
DO_8(LL456_28-04-09) ∧ DO_8(LT123_16-02-08) ∧ DO_8(LT256_16-01-08)
→ LABTEST_DATE(LL456_28-04-09) ∧ LABTEST_DATE(LT123_16-02-08) ∧
LABTEST_DATE(LT256_16-01-08)
```

Post-High-level rule 69:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
TO_5(Mrs_Flee_has_come_into_the_clinic_for_a_blood_test_complains_of_fatig
ue)
→
PREVIOUS_MEDICAL_SUMMARIES(Mrs_Flee_has_come_into_the_clinic_for_a_blood_t
est_complains_of_fatigue)
```

Post-High-level rule 70:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
TO_7(LL456_Used_to_identify__lungs_diseases)
→ LABTEST_OVERVIEW(LL456_Used_to_identify__lungs_diseases)
Post-High-level rule 71:
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
TO_7(LL456_data_aa2)
→ LABTEST_DATA(LL456_data_aa2)
```

Post-High-level rule 72:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
DO_9(T09851_17-04-09) ∧ DO_9(TT1989_12-03-09)
→ TREATMENT_DATE(T09851_17-04-09) ∧ TREATMENT_DATE(TT1989_12-03-09)
```

Post-High-level rule 73:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
DO_10(M222p_NAPROXEN) ∧ DO_10(M225i_EHOSUXIMIDE) ∧ DO_10(M0031_Capzasin)
→ MEDICINE_NAME(M222p_NAPROXEN) ∧ MEDICINE_NAME(M225i_EHOSUXIMIDE) ∧
MEDICINE_NAME(M0031_Capzasin)
```

Post-High-level rule 74:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
DO_10(M225i_Emeside) ∧ DO_10(M222p_Risedronate) ∧ DO_10(M0031_Xhing_Ltd)
```

```
→ VENDOR(M225i_Emeside) ∧ VENDOR(M222p_Risedronate) ∧
VENDOR(M0031_Xhing_Ltd)
```

Post-High-level rule 75:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
DO_10(M0031_2_tablets_per_day) ∧
DO_10(M225i_1_tablets_to_be_taken_4_times_a_day) ∧
DO_10(M222p_1_or_2_tablets_to_be_taken_4_times_a_day)
→ DOSAGE_AMOUNT(M0031_2_tablets_per_day) ∧
DOSAGE_AMOUNT(M225i_1_tablets_to_be_taken_4_times_a_day) ∧
DOSAGE_AMOUNT(M222p_1_or_2_tablets_to_be_taken_4_times_a_day)
```

Post-High-level rule 76:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
TO_8(M0031) ∧ TO_8(M225i) ∧ TO_8(M222p)
→ MEDICINE_NUMBER(M0031) ∧ MEDICINE_NUMBER(M225i) ∧ MEDICINE_NUMBER(M222p)
```

Post-High-level rule 77:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
TO_9(TT1989_Patient_is_suffering_from_aches_in_lower_limbs_and_has_minor_s
welling_to_ankle_pain_support_through_chronic_pain_recovery_is_suggested)
→
TREATMENT_OVERVIEW(TT1989_Patient_is_suffering_from_aches_in_lower_limbs_a
nd_has_minor_swelling_to_ankle_pain_support_through_chronic_pain_recovery_
is_suggested)
```

Post-High-level rule 78:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true) ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
TO_9(T01245_COPDT_exacerbation) ∧   TO_9(T09851_COPD_Chronic_pain_recovery)
→  TREATMENT_NAME(T01245_COPDT_exacerbation) ∧
TREATMENT_NAME(T09851_COPD_Chronic_pain_recovery)
```

Post-High-level rule 79:

```
SELECTION_jane_flee(?A) ∧ has_variable_jane_flee(?A, true) ∧
SELECTION_gp_rep(?B) ∧ has_variable_gp_rep(?B, true)  ∧
SELECTION_hospital_rep(?C) ∧ has_variable_hospital_rep(?C, true) ∧
SELECTION_clinic_1_rep(?D) ∧ has_variable_clinic_1_rep(?D, true) ∧
SELECTION_clinic_2_rep(?E) ∧ has_variable_clinic_2_rep(?E, true) ∧
TO_10(M222p_Andheri_east_India) ∧ TO_10(M225i_South_coast_Canada)  ∧
TO_10(M0031_China_pharmaceuticals)
→ MEDICATION_DETAILS(M222p_Andheri_east_India) ∧
MEDICATION_DETAILS(M225i_South_coast_Canada)  ∧
MEDICATION_DETAILS(M0031_China_pharmaceuticals)
```

Appendix A.19

Can be found on CD-ROM attached to thesis.

Appendix A.20

Can be found on CD-ROM attached to thesis.

References

[1] WEISER, M. (1993) “Some computer science problems in ubiquitous computing”, Communications of the ACM, 36(7), pp. 75–84.

[2] ABOWD, G.D. (1999) “Software Engineering Issues for Ubiquitous Computing”, In: *Proceedings of the 21st International Conference on Software Engineering, (LA, USA, May 16-22)*, pp. 75-84.

[3] SATAYANARAYAN, M. (2001) “Pervasive Computing: Vision and Challenges”, IEEE Personal Communications, 8(4), pp. 10–17.

[4] GUPTA, P.; MOITRA, D. (2004) “Evolving a pervasive IT infrastructure: a technology integration approach”, Journal of Personal and Ubiquitous Computing, 8(1), pp. 31–41.

[5] FUENTES, L.; JIMENEZ, D.; PINTO, M. (2004) “Towards the development of ambient Intelligence Environments using Aspect-Oriented techniques”, In: *Proceedings of International Conference on Aspect-Oriented Software Development, Workshop in Aspects, Components, and Patterns for Software Infrastructure, (Lancaster, UK, March 22-26)*, pp. 22-26.

[6] HAGRAS, H. (2007) “Embedding Computational Intelligence in Pervasive Spaces”, IEEE Pervasive Computing, 6(3), pp. 85-89.

[7] HELLENSCHMIDT, M. (2006) “Some Issues on Requirements for Pervasive Software Infrastructures”, In: Fraunhofer-Institstute for Computer Graphics, (Darmstadt, Germany), available at http://www.igd.fhg.de/igd-a1/RSPSI/papers/RSPSI-Hellenschmidt.pdf, [accessed June 2009].

[8] BECKER, M.; KARSHMARA, A.; LAMM, R.; NEHMER, J. (2006) “Living Assistance Systems – An Ambient Intelligence Approach”, In: *Proceeding of the International Conference on Software Engineering, (Shanghai, China, May 20-28)*, pp. 43-50.

[9] CABRI, G.; FERRARI, L.; LEONARDE, L.; ZANBONELI, F. (2005) “The LAICA Project: Supporting Ambient Intelligence via Agents and Ad-Hoc Middleware”, In: *Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise, (Sweden, June 13-15)*, pp. 39-44.

[10] SHETH A.P. (2003) “Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics”, Interoperating Geographic Information Systems, M. Goodchild, M. Egenhofer, R. Fegeas, and C. Kottman (eds.), Kluwer Academic Publishers: Norwell, MA, pp. 5-30.

[11] GARLAN, D.; SIEWIOREK, D.P.; SMAILAGIC, A.; STEENKISTE, P. (2002) “Project Aura: toward distraction free pervasive computing”, IEEE Pervasive Computing, 1(2), pp. 22-31.

[12] HURSON, A. R.; BRIGHT, M. W.; PAKZAD, S. H. (1994) “Multidatabase Systems: an advanced solution for global information sharing”, IEEE Computer Society Press, USA.

[13] SHETH, A.P.; LARSON, J.A. (1990) “Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases”, ACM Computing Surveys, 22(3), pp. 183-236.

[14] SCHEUERMANN, P.; YU, C.; ELMAGARMIND, A.; GARCIA-MOLINA, H.; MANOLA, F.; MCLEOD, D.; ROSENTHAL, A.; TEMPLETON, M. (1990) “Report on the workshop on heterogeneous database systems held at Northwestern University Evanston, Illinois, December 11-13, 1989 sponsored by NSF”, ACM SIGMOD Record, 19(4) pp.23-31.

[15] WACHE, H.; VOEGELE, T.; VISSER, U’; STUCKENSCHMIDT, H.; SCHUSTER, G.; NEUMANN, H.; HUEBNER, S. (2001) “Ontology-based integration of information - a survey of existing approaches”. In:

Proceedings of the International Joint Conferences on Artificial Intelligence, *Workshop on Ontologies and Information Sharing, (Seattle, Washington, USA, Aug 4–5),* pp. 108-117.

[16] BISHR Y.A.; PUNDT H.; KUHN W.; RDWAN M. (1999) "Probing the Concepts of Information Communities – A First Step Towards Semantic Interoperability", M. Goodchild, M. Egenhofer, R. Fegeas, and C. Kottman (eds.), Interoperating Geographic Information Systems, Kluwer Academic Publishers: Norwell, MA, pp. 55-70.

[17] WIEDERHOLD, G.; (1999) "Mediation to deal with Heterogeneous Data Sources", Interoperating Geographic Information Systems, LNCS 1580, pp. 1-16.

[18] SHETH, A.; KASHYAP, V. (1996) "Semantic and schematic similarities between database objects: A context based approach", Very Large Database Journal, 5(4), pp. 276-304.

[19] HULL, R. (1997) "Managing Semantic Heterogeneity in Databases: A Theoretical Perspective", In: *Proceedings of the ACM Symposium on Principles of Database Systems, (Tucson, USA, May 12-14),* pp. 51-61.

[20] HAMMER, J.; MCLEOD, D. (1993) "An Approach to Resolving Semantic Heterogeneity in a Federation of Autonomous, Heterogeneous DB Systems", International Journal of Intelligent and Cooperative Information Systems, 2(1), pp. 51-83.

[21] KIM, W.; CHOI, I.; GALA, S.; SCHEEVEL, M. (1993) "On Resolving Schematic Heterogeneity in Multidatabase Systems", Distributed and Parallel Databases, 1(3), pp. 251-279.

[22] CERCONE, N.; MORGENSTERN, M.; SHETH, A.; LITWIN, W.; (1990) "Resolving semantic heterogeneity", In: *6th International Conference on Data Engineering (panel), (Los Angeles, California, February).*

[23] BATANI, C.; LENZERINI, M.; NAVATHE, S.B. (1986) "A comparative analysis of methodologies for database schema integration", ACM Computing Surveys, 18(4), pp.323-364.

[24] ZIEGLER, P.; DITTRICH, R, K. (2004) "User-Specific Semantic Integration of Heterogeneous Data: The SIRUP Approach", Semantics of a Networked World,LNCS 3226, pp. 44-46.

[25] REDDY, M. P.; PRASAD, B. E.; REDDY, P. G.; GUPTA, A. (1994) "A methodology for integration of heterogeneous databases", IEEE Transactions on Knowledge Data Engineering, (6)6, pp. 920–933.

[26] SHETH, A.P.; GALA, S.K.; NAVATHE, S.B. (1993) "On automatic reasoning for schema integration", International Journal on Intelligent and Cooperative Information Systems 2(1), pp. 23–50.

[27] HAYNE, S.; RAM, S. (1990) "Multi-user view integration system (MUVIS): An expert system for view integration", In: *Proceedings of the 6th International Conference on Data Engineering (Los Angeles, CA, Feb. 5–9),* pp. 402–409.

[28] OUKSEL, A.M. (1999) "A framework for a scalable agent architecture of cooperating heterogeneous knowledge sources", Intelligent Information Agents: Cooperative, Rational and Adaptive Information Gathering in the Internet. M. Klusch (Ed.) Springer, Berlin, Germany, pp. 100–124.

[29] GENESERETH, M.R.; KELLER, A.M.; DUSCHKA, O.M.; (1997) "Infomaster: An information integration system", In: *Proceedings of the ACM SIGMOD International Conference on Management of Data (Tucson, USA, May 13–15),* pp. 39–542.

[30] LEVY, A.Y.; RAJARAMAN, A.; ORDILLE, J.J. (1996) "Querying Heterogeneous Information Sources Using Source Descriptions", In: *Proceedings of the 22nd International Conference on Very Large Data Bases, (Mumbai, India),* pp. 251–262.

[31] PAPAKONSTANTINOU, Y.; GARCIA-MOLINA, H.; ULLMAN, J. (1996) "Medmaker: A mediation system based on declarative specifications", In: *Proceedings of the 12th IEEE International Conference on Data Engineering (New Orleans, LA, Feb. 26–March 1),* pp. 132–141.

[32] CAREY, M.J.; HAAS, L.M.; SCHWARZ, P.M.; ARYA, M.; CODY, W.F.; FAGIN, R.; FLICKNER, M.; LUNIEWSKI, A.; NIBLACK, W.; PETKOVIC, D.; THOMAS, J.; WILLIAMS, J.H.; WIMMERS, E.L. (1995) "Towards Heterogeneous Multimedia Information Systems: The Garlic Approach". In: *Proceedings of the 5th*

Workshop on Research Issues in Data Engineering-Distributed Object Management, (Taipei, Taiwan, March 6-7), pp. 124–131.

[33] CHAWATHE, S.; GARCIA-MOLINA, H.; HAMMER, J.; IRELAND, K.; PAPAKONSTANTINOU, Y.; ULLMAN, J.; WIDOM, J. (1994) "The TSIMMIS Project: Integration of Heterogeneous Information Sources", In: *Proceedings of Information Processing Society of Japan Conference*, (Tokyo, Japan, October), pp. 7-18.

[34] WIEDERHOLD, G. (1992) "Mediators in the Architecture of Future Information Systems", IEEE Computer, 25(3), pp. 38-49.

[35] JURIC, R.; BEUS-DJUKIC, L. (2005) "COTS components and DB interoperability", In: *Proceedings of the 4th International Conference COTS-based software systems, (Bilbao, Spain, February 7-11)*, pp. 77-89.

[36] JURIC, R.; KULJIS, J.; PAUL, R. (2004) "Software Architecture to Support Interoperability in Multiple Database Systems", In: *Proceedings of the 22nd International Conference on Software Engineering, (Innsbruck, Austria, February 11-14)*, pp. 71-77.

[37] JURIC, R.,; KULJIS, J.; PAUL, R. (2004a) "Software Architectural Style for Interoperable Databases", In: *Proceedings of the 26th International Conference on Information Technology Interfaces, (Croatia, June 7-10)*, pp. 159-166.

[38] JURIC, R.; KULJIS, J.; PAUL, R. (2004b) "Contextualising components when addressing database interoperability", In: *Proceedings of the 8th International Conference on Software Engineering and Applications, (MIT, Cambridge, MA, USA, November 9-11)*, pp. 690-695

[39] FISH, D.A. (2006) "An Emergent Perspective on Interoperation in Systems of Systems", Technical Report, CMU/SEI-2006-TR-003, ESC-TR-2006-003, available at, http://www.sei.cmu.edu/reports/06tr003.pdf, accessed January 2009].

[40] MORRIS, E.; LEVINE, L.; MEYERS, C.; PLACE, P.; PLAKOSH, D. (2004) "System of Systems Interoperability (SOSI): Final Report", Technical Report, CMU/SEI-2004-TR-004, ESC-TR-2004-004, available at, http://www.sei.cmu.edu/reports/04tr004.pdf, accessed January 2009].

[41] GRIMM, R.; DAVIS, J.; LEMAR, E.; MACBETH, A.; SWASON, S.; GRIBBLE, S.; ANDERSON, T.; BERSHAD, B.; BORRIELLO, G.; WETHERALL, D. (2001), "Programming for Pervasive Computing Environments", University of Washington, available at http://one.cs.washington.edu/papers/tr01-06-01.pdf, [accessed in March 2009].

[42] MCGUINESS, D.; HARMELEN, F. (2004) "OWL Web Ontology Overview/Guide", W3C Recommendation, available at http://www.w3.org/TR/owl-features/, [accessed January 2009].

[43] HORROCKS, I.; PATEL-SCHNEIDER, P. F.; BOLEY, H.; TABET, S.; GROSOF, B.; Dean, M. (2004) "SWRL: Semantic Web Rule Language- Combining OWL and RuleML Overview and Guide", W3C Member Submission, available at http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/, [accessed June 2009].

[44] SHETH, A.; KASHYAP, V. (1992) "So far (schematically) yet so near (semantically)", In: *Proceedings of the Conference on Semantics of Interoperable Database Systems, (Lome, Australia, November 16-20)*, pp. 238-312.

[45] SCIORE, E.; SIEGAL, M.; ROSENTHAL, A. (1994) "Using Semantic Values to Facilitate Interoperability Among Heterogeneous Information Systems", ACM Transactions on Database Systems, 19(2), pp.254-290.

[46] KATARIA, P.; JURIC, R. (2010) "Creating Semantics From User Inputs Through Ontological Reasoning", In: *Proceedings of the 15th International Conference on System Design and Process Science, (Texas, Dallas, US, June 6-11)*, CD-ROM.

[47] LIU. Q.; HUANG. T.; LIU, S.H.; ZHONG., H. (2007) "An Ontology-Based Approach for Semantic Conflict Resolution in Database Integration". Journal of Computer Science and Technology, 22(2), pp.218-227.

[48] TRINH, Q.; BARKER, K.; and ALHAJJ, R. (2007) "Semantic Interoperability Between Relational Database Systems", In: *Proceeding of the 11th International Database Engineering and Applications Symposium, (Banff, Alberta, Canada, September 6-8)*, pp. 208-215.

[49] RAM, S.; PARK, J. (2004) “Semantic Conflict Resolution Ontology (SCROL): An Ontology for Detecting and Resolving Data and Schema-Level Semantic Conflicts”, IEEE Transactions on Knowledge and Data Engineering, 16(2), pp.189-202.

[50] GOH, C.H, (1997) “Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Sources”, Ph.D. thesis, Massachusetts Institute of Technology, Sloan School of Management, available at http://web.mit.edu/smadnick/www/wp/1997-01.pdf, [accessed in March 2009].

[51] HAJMOOSAEI, A.; ABDUL-KAREEM, S. (2007) “An ontology-based approach for resolving semantic schema conflicts in the extraction and integration of query-based information from heterogeneous web data sources”, In: *Proceedings of the 3rd Australasian Ontology Workshop, (Gold Coast, Australia, December 2-6)*, pp.35-43.

[52] STOIMENOV, L.; STANIMIROVIC, A.; DJORDJEVIC-KAJAN, S. (2006) “Discovering mappings between ontologies in semantic integration process”, In: *Proceedings of the 9th AGILE International Conference on Geographic Information Science, (Visegrád, Hungary, April 19-22)*, pp. 213-219.

[53] WANG, C.; LU, J.; ZHANG, G. (2006) “Integration of Ontology Data through Learning Instance Matching”, In: *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, (Hong Kong, December 18-22)*, pp. 536-539

[54] BREITMAN, K. K.; FELICISSIMO, C. H.; CYSENEIROS, L. M. (2003) “Semantic Interoperability by Aligning Ontologies”, World Energy Research supported in part by CNPq under contract ESSMA-552068/2002-0, by Faperj under Student Grade 10 scholarship and by CAPES, available at http://wer.inf.puc-rio.br/WERpapers/artigos/artigos_WER03/karin_breitman.pdf, [accessed January 2009].

[55] THANH-LE, B.; DIENG-KUNTZ R.; GANDON, F. (2004) “On Ontology Matching Problems: for building a corporate Semantic Web in a multi-communities organization”, In: *Proceedings of the 6th International Conference on Enterprise Information Systems, (Porto, Portugal, April 14-17)*, pp. 236-243.

[56] GIUNCHIGLIA, F.; SHVAIKO P. (2003) “Semantic Matching”, The Knowledge Engineering Review, 18(3), pp.265-280.

[57] DOAN, A.; MADHAVAN, J.; DOMINGOS, P.; HALEVY, A. (2003) “Learning to map ontologies on the semantic web”, In: *Proceedings of the 11th International World Wide Web Conference (Honolulu, Hawaii, USA, May 7-11)*, pp. 662–673.

[58] MADHAVAN, J.; BERNSTEIN, P. A.; RAHM, E.; (2001) “Generic Schema Matching with Cupid”, In: *Proceedings of the 27th Conference on Very Large Databases, (Roma, Italy, September 11-14)*, pp. 49-58.

[59] EHRIG, M.; STAAB, S. (2004) “QOM: Quick ontology mapping”, In: *Proceedings of the 3rd International Semantic Web Conference, (Hiroshima, Japan, November 7-11)*, pp. 683–697.

[60] NOY, N.F.; MUSEN, M. A.; (2004) “Ontology Versioning in an Ontology-Management Framework”, *IEEE Intelligent Systems*, 19(4), pp. 6-13.

[61] MCGUINESS, D.L.; FIKES, R.; RICE, J.; WILDER, S. (2000) “The chimaera ontology environment”, In: *Proceedings of the 17th National Conference on Artificial Intelligence, (Austin, Texas, July 30-August 3)*, pp. 1123-1124.

[62] GRANATIR, N.; JURIC, R.; KULJIS, J.; TESANOVIC, I. (2007) “Supporting Interoperability Frameworks in the UK Public Sector”, In: *Proceedings of the 10th International Conference on Integrated Design and Process Technology, (Antalya, Turkey, June 3-8, 2007)*, CD-ROM.

[63] WILLIAMS, S.; JURIC, R.; MILLIGAN, P. (2005) “Design patterns for automation of marketing authorisations in pharmaceutical industry”, In: *Proceedings of the 27th International Conference on Information Technology Interfaces,(Cavtat, Croatia, June 20-23)*, pp. 565-570.

[64] JURIC, R.; SLEVIN, L.; SHOJANOORIE, R.; WILLIAMS, S. (2005) “Software support in automation of medicinal product evaluations”, In: Studies in health technology and informatics, Bos, Lodewijk and Laxminarayan, Swamy and Marsh, Andy, (eds.) Medical and care compunetics, 2(114), pp. 298-306.

[65] BUSSE, S.; KUTSCHE, R.D.; LESER, U.; WEBER, H. (1999) "Federated Information Systems: Concept, Terminology and Architectures", Technical report No 99-9, Technische Universität (TU) Berlin, Germany, available at http://citeseer.ist.psu.edu/busse99federated.html, [accessed in March 2009].

[66] GARCIA-MOLINA, H.; YEMENI, R., (1999) "Coping with Limited Capabilities of Sources", In: *Proceedings of the 8th GI Facht-agung Datenbamksysteme (Verlag, Buero, Frieburg, Germany),* pp. 1-19.

[67] TORK, M.; ROTH, P.; SCHWARZ, M. (1997) "Don't Scrap it, Wrap it – A Wrapper Architecture for Legacy Data Sources", In: *Proceedings of the 23rd International Conference on Very Large Databases, (Athens, Greece, August 25-29),* pp. 266-275.

[68] TURKER, C.; SAAKE, G.; CONRAD, S. (1997) "Modelling Federations in Terms of Evolving Agents", In: *Poster Proceedings of the 10th International Symposium on Methodologies for Intelligent Systems, (Charlotte, North Carolina, US, October 15-18),* pp. 197-208.

[69] VISSER, P.R.S.; JONES, D.M.; BENCH-CAPON, T.J.M.; SHAVE, M.J.R. (1997) "An Analysis of Ontology Mismatches; Heterogeneity Versus Interoperability", In: *Proceedings of the American Association for Artificial Intelligence (AAAI) Spring Symposium on Ontological Engineering (California, US, March 24-26),* pp.164-172.

[70] GRUBER, T.R. (1995) "Toward principles for the Design of Ontologies Used for Knowledge Sharing", International Journal of Human-Computer Studies, 43(5-6), pp. 907-928.

[71] FAGIN, R. (1996) "Combining Fuzzy Information from Multiple Systems", In: *Proceedings of ACM Symposium on Principles of Database Systems, (Montreal, Canada, June 3-5)* pp. 216-226.

[72] SAMET, H.; AREF, W.G. (1995) "Spatial Data Models and Query Processing", Addison Wesley, Reading, MA, US, pp. 338-360.

[73] COLOMB, R. M. (1997) "Impact of Semantic Heterogeneity on Federated Databases", Computer Journal, 40(5), pp. 235-244.

[74] O'SULLIVAN, D.; POWER, R. (2003) "Bridging heterogeneous, autonomous, dynamic knowledge at runtime", In: *Proceedings of the 1st International Workshop on Managing Ubiquitous Communications and Services, (Waterford, Ireland, December 11),* available at http://www.m-zones.org/deliverables/d234_2/papers/1-02-tcd-bridging-knowledge.pdf, [accessed January 2007].

[75] DECKER, S.; MELNIK, S.; VAN-HARMELEN, F.; FENSAL, D.; KLIEN, M.; BROEKSTRA, J.; ERDMANN, M.; HORROCKS, I. (2000) "The Semantic Web: the roles of XML and RDF", IEEE Internet Computing, 4(5), pp.63-73.

[76] CHUNG, J.Y.; LIN, K.J.; MATHIEU, R.G. (2003) "Web Services Computing: Advancing Software Interoperability", IEEE Computer, 36(10), pp.35-37.

[77] HALEVY, A. Y. (2005) "Why Your Data Won't Mix: Semantic Heterogeneity". Que, feature Q focus: semi-structured data, 3(8), pp. 50-58.

[78] ABDULLA, K. (1998) "A new approach to the integration of heterogeneous databases and information systems", Ph.D. thesis, University of Miami, Florida.

[79] GOH, C. H.; MADNICK, S. E.; SIEGAL, M. D. (1994) "Context inter-change: overcoming the challenges of large-scale interoperable database systems in a dynamic environment", In: *Proceeding of the International Conference on Information and Knowledge Management, (New Orleans, Louisiana, USA, November 2-8),* pp.337-346.

[80] MILLER, R. J. (1998) "Using schematically heterogeneous structures", ACM SIGMOD Record, 27(2), pp.189-200.

[81] GARCIA-SOLACE, M.; SALTOR, F.; CASTELLANOUS, M. (1996) "Semantic Heterogeneity in Multidatabase Systems", In: O. A. Burkhres and A. K. Elmagarmid, editors, Object Orientated Multidatabase Systems, Prentice-hall, pp.129-202.

[82] HOLOWCZAK, R.D.; LI, W.S. (1996) "A survey on attribute correspondence and heterogeneity metadata representation", In: *Proceedings of the 1st IEEE Metadata Conference, (April 16-18)*, available at http://www.computer.org/conferences/meta96/li/paper/html, [accessed in March 2009].

[83] CERI, S.; WIDOM, J. (1993) "Managing Semantic Heterogeneity with Production Rules and Persistent Queues", In: *Proceedings of the 19th Very Large Databases Conference (VLDB 1993), (Dublin, Ireland, August 24-27)*, pp. 108-119.

[84] WEISER, M. (1991). "The Computer for the 21st Century", The Scientific American, 265(3), pp.94-104.

[85] WEISER, M.; BROWN, J.S. (1997) "The coming of calm technology". Xerox PARC, Beyond calculation: the next fifty years, available at www.johnseelybrown.com/calmtech.pdf, [accessed January 2009].

[86] WEISER, M. (1993a) "Hot Topics: Ubiquitous Computing", IEEE Computer, available at http://www.ubiq.com/hypertext/weiser/UbiCompHotTopics.html, [accessed January 2009].

[87] WEISER, M.; BROWN, J.S.(1996) "Designing Calm Technology", Powergrid Journal, 1(1), pp. 75 -85.

[88] GRIMM, R.; DAVIS, J.; HENDRICKSON, B.; LEMAR, E.; MACBETH, A.; SWANSON, S.; ANDERSON, T.; BERSHAD, B.; BORRIELLO, G.; GRIBBLE, S.; WETHERELL, D. (2001) "Systems Directions for Pervasive Computing", In: *Proceedings 8th Workshop on Hot Topics in Operating System, (Elmau, Oberbayern, Germany, May 20-23)*, pp.147-151.

[89] SMAILAGIC, A; SIEWIOREK, D. (2002) "Application design for wearable and context-aware computers", IEEE Pervasive Computing, 1(4), pp. 20-29.

[90] MIDKIFF, S.F. (2002) "Mobile and wireless networks and applications", IEEE Pervaisve Computing, 1(4), pp. 9-11.

[91] BANAVAR, G.; BECK, J.; GLUZBERG, E.; MUNSON, J.; SISSMAN, J.; ZUKOWSKI, D. (2000) "Challenges: An application Model for Pervasive Computing", In: *Proceedings of 6th Annual International Conference on Mobile Computing and Networking (MOBICOM 2000), (Boston, Massachusetts, United States, August 6-11)*, pp. 266-274.

[92] HELAL, S.; MANN, W.; EL-ZABADANI, H.; KING, J.; KADDOURA Y.; JANSEN, E. (2005). *"The Gator Tech Smart House: A Programmable Pervasive Space"*, IEEE Computer Society, 38(3), pp. 50-60.

[93] LEE, C.; NORDSTEDT, D.; HELAL, S. (2003) "Enabling Smart Spaces with OSGi", IEEE Computer Society, 2(3), pp. 89- 94.

[94] MEYER, S.; RAKOTONIRAINY, A. (2003) "A Survey of Research on Context-Aware Homes", In: *Proceedings of the Workshop Conference on Wearable, Invisible, Context-Aware, Ambient, Pervasive and Ubiquitous Computing, (Adelaide, South Australia, February 5)*, pp.159-168.

[95] INTILLE, S.S. (2002) "Designing a Home of the Future' MIT School of Architecture and Planning", IEEE Educational Activities Department, 1(2), pp. 76 – 82.

[96] KUMAR, M.; SHIRAZI, B.A.; DAS, S.K.; SUNG, B.Y.; LEVINE, D.; SINGHAL, M. (2003) "PICO: a middleware framework for pervasive computing", IEEE Pervasive Computing, 2(3), pp.72-79.

[97] PHILIPOSE, M.; FISHKIN, K.P.; PATTERSON, D.J.; FOX, D.; KAUTZ, H.; HAHNEL, D. (2004) "Inferring activities from interaction with objects", IEEE Pervasive Computing, 3(4), pp. 50-57.

[98] BECKWITH, R. (2003) "Design for ubiquity: the perception of privacy", IEEE Pervasive Computing, 2(2), pp.40 – 46.

[99] SENGERS, P.; KAYE, J.; BOEHNER, K.; FAIRBANK, J.; GAY, G.; MEDYBSKIY, Y.; WUCHE, S. (2004) "Culturally embedded computing", IEEE Pervasive computing, 3(1), pp. 14-21.

[100] BURRELL, J.; BROOKE, T.; BECKWITH, R. (2004) "Vineyard computing: sensor networks in agricultural production", IEEE Pervasive computing, 3(1), pp. 38-45.

[101] KELLER, I.; VAN-DER-HOOG, W.; STAPPERS, P.J. (2004) "Gust of me: reconnecting mother and son", IEEE Pervasive Computing, 3(1), pp. 22-27.

[102] JOSEPH, A.D. (2004) “Work in progress- Gaming, fine art, and familiar strangers”, IEEE Pervasive Computing, 3(1), pp. 35-37.

[103] HAUPTMANN, A.G.; GAO, J.; YAN, R.; QI, Y.; YANG, J.; WACTLAR, H.D. (2004) “Automated analysis of nursing home observations”, IEEE Pervasive computing, 3(2), pp. 15-21.

[104] YAU, S.S.; HUANG, D.; GONG, H.; SETH, S. (2004) “Development and runtime support for situation-aware application software in ubiquitous computing environments”, In: *Proceedings of the 28th Annual International Conference on Computer Software and Applications, (Hong Kong, September 28-30)*, pp. 452-457.

[105] ASHBROOK, D.; LYONS, K.; CLAWSON, J. (2006) “Capturing Experiences Anytime, Anywhere”, IEEE Pervasive Computing, 5(2), pp.8-9.

[106] ISLAM, N. (2004) “From smart to autonomous phone”, IEEE Pervasive Computing, 3(3), pp. 102-104.

[107] EAGLE, N.; PENTLAND, A. (2005) “Social serendipity: mobilizing social software”, IEEE Pervasive Computing, 4(2), pp. 28-34.

[108] BEALE, R. (2005) “Supporting social interaction with smart phones”, IEEE Pervasive Computing, 4(2), pp. 35-41.

[109] CABITZA, F.; SARINI, M.; DAL-SENO, B. (2005) “DJess – A Context-Sharing Middleware to Deploy Distributed Inference Systems in Pervasive Computing Domains”, In: *Proceedings of the IEEE International Conference on Pervasive Services (ICPS 2005), (Santorini, Greece, July 11- 14)*, pp. 229-238.

[110] DAVIES, N.; LANDAY, J.; HUDSON, S.; SCHMIDT, A. (2005) “Introduction: Rapid Prototyping for Ubiquitous Computing”, IEEE Pervasive Computing, 4(4), pp. 15-17.

[111] DAVIES, N.; FRIDAY, A.; STORZ, O. (2004) “Exploring the grid’s potential for ubiquitous computing”, IEEE Pervasive Computing, 3(2), pp.74-75.

[112] YAU, S.; KARIM, F. (2004) “An Adaptive Middleware for Context-Sensitive Communications for Real-Time Applications in Ubiquitous Computing Environments”, Real-Time Systems, 26(1), pp.29-61.

[113] GU, T.; XHANG, D.Q. (2003) “Towards an OSGI based Infrastructure for Context aware Applications”, IEEE Pervasive Computing, 3(4), pp. 66-74.

[114] YAU, S.S.; KARIM, F.; YU. W.; BIN, W.; GUPTA, S.K.S. (2002) “Reconfigurable context-sensitive middleware for pervasive computing”, IEEE Pervasive Systems, 1(3), pp.33-44.

[115] GRIMM, R.; DAVIS, J.; HENDRICKSON, B.; LEMAR, E.; MACBETH, A.; SWANSON, S.; ANDERSON, T.; BERSHAD, B.; BORRIELLO, G.; GRIBBLE, S.; WETHERELL, D. (2004) “System support for pervasive applications”, ACM Transactions on Computer Systems, 22(4), pp. 421-486.

[116] JOHANSON, B.; FOX, A.; (2002), “The Event Heap: a coordination infrastructure for interactive workspaces”, In: *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications, (Callicoon, New York, 20-21 June)*, pp. 83-93.

[117] ROMAN, M.; HESS, C. K.; CERQUEIRA, R.; RANGANTHAN, A.; CAMPBELL, R. H.; NAHRSTEDT, K. (2002) “Gaia: A Middleware Infrastructure to Enable Active Spaces”, IEEE Pervasive Computing, 1(4), pp. 74-83.

[118] SOUSA, J. P.; GARLAN, D. (2002) “Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments”, In: *Proceedings of the 3rd Working IEEE/IFIP Conference on presented at Software Architecture: System Design, Development, (Montreal, Canada, August 25-30)*, pp. 29-43.

[119] FOK, C.L. (2009) “Adaptive Middleware for resource Constrained Mobile Ad Hoc and Wireless Sensor Network”, Ph.D. Thesis, Washing University in St Louis, School of Engineering and Applied Sciences, Department of Computer Science and Engineering, available at http://cse.wustl.edu/Research/Lists/Technical%20Reports/Attachments/905/fok_dissertation.pdf, [accessed in March 2009].

[120] RIVA, O. (2007) "Middleware for Mobile Sensing Applications in Urban Environments", Academic Dissertation, Faculty of Science, University of Helsinki, Finland. Available at http://www.doria.fi/bitstream/handle/10024/27194/middlewa.pdf?sequence=1, [accessed January 2009].

[121] HADIM, S.; MOHAMED, N. (2006) "Middleware for wireless sensor networks: A survey", In: *Proceedings of the 1st International Conference on Communication System Software and Middleware, (Delhi, India, January 8-12)*, pp. 1-7.

[122] BYUN, H.E.; CHEVEREST, K. (2003) "Supporting Proactive 'Intelligent' Behaviour: The Problem of Uncertainty", In: *Proceedings of the Workshop on User Modeling for Ubiquitous Computing (UM03), (Pittsburgh, PA, USA, June 22-23)*, available at http://www.di.uniba.it/~ubium03/byun-8.pdf, [accessed January 2009].

[123] ESTRIN, D.; CULLER, D.; PPSITER, K.; SUKHATME, G. (2002) "Connecting the Physical World with Pervasive Networks", IEEE Pervasive Computing, 1(1), pp. 59-69.

[124] WANT, R. (2007), "Sensor-Driven Computing Comes of Age", IEEE Pervasive Computing, 6(2), pp. 4-6.

[125] BAEK, S.H.; CHOI, E.C.; HUH, J.D. (2007) "Design of Information Management Model for Sensor Based Context-Aware Service in Ubiquitous Home", In: *Proceedings of the International Conference on Convergence Information Technology (ITCS 2007), (Gyeongju, Korea, November 21-23)*, pp.1040-1047.

[126] MA, J.; YANG, L.T.; APDUHAN, B.O.; HUANG, R.; BAROLLI,L.; TAKIZAWA, M.; SHIH, T.K. (2005) "A Walkthrough from Smart Spaces to Smart Hyperspaces towards a Smart World with Ubiquitous Intelligence", In: *Proceeding of the 11th International Conference on Parallel and Distributed Systems, (Fuduoka, Japan, July 22)*, pp. 370-376.

[127] SIXSMITH, A; JOHNSON, N. (2004) "A smart sensor to detect the falls of the elderly", IEEE Pervasive Computing, 3(2), pp.42-47.

[128] BALAZINSKA, M.; DESHPANDE, A.; FRANKLIN, M.J.; GIBBONS, P.B.; GRAY, J.; NATH, S.; HANSEN, M.; LIEBHOLD, M. (2007) "Data Management in the Worldwide Sensor Web", IEEE Pervasive Computing, 6(2), pp. 30-40.

[129] COOK, D.J. (2007) "Making Sense of Sensor Data", IEEE Pervasive Computing, 6(2), pp. 105-108.

[130] SCHMOHL, R.; BAUMGARTEN, U. (2008) "A Generalized Context-aware Architecture in Heterogeneous Mobile Computing Environments", In: *Proceedings of the 4th International Conference on Wireless and Mobile Communications, (Athens, Greece, July 27-August 1)*, pp.118-124.

[131] KOWALSKI,G.; MAYBURY, M.T. (2000) "Information Storage and Retrieval Systems - Theory and Implementation", 2nd ed. Kluwer Academic Publishers.

[132] BAEZA-YATES, R.; and RIBEIRO-NETA, B. (1999), "*Modern information retrieval",* New York, ACM Press.

[133] BONTO-KANE, M.V.A;; CHIN, A.; MCCARTHY, S.; SRIKULWONG, M.; TIMMINS, P.J. (2007) "Pervasive 2007: It's about the User", IEEE Pervasive Computing, 6(4), pp. 95-96.

[134] MOSTOWFI, F.; FOTOUHI, F.; ARISTAR, A. (2005) "Ontogloss: an ontology-based annotation tool", In: *Proceedings of the Electronic Metastructure for Endangered Languages Data Workshop on Morphosyntactic Annotation and Terminology, (Boston, USA, July 1-3)*, pp.11.

[135] GANESH, S.; JAYARAJ, M.; KALYAN, V.; MURTHY, S.; AGHILA, G. (2004) "Ontology-based web crawler", In: *Proceedings of the International Conference on Information Technology: Coding and Computing, (Las Vegas, Nevada, April 5-7)*, pp. 337–341.

[136] ERDMANN, M.; MAEDCHE, A.; SCHNURR, H. P.; STAAB, S. (2001) "From manual to semiautomatic semantic annotation", In: *Proceedings of the 18th International Conference on Computational Linguistics Workshop on Semantic Annotation and Intelligent Content (COLING 01), (Luxembourg, August 5-6)*, available at http://www.aifb.uni-karlsruhe.de/WBS/sst/Research/, [accessed in March 2009].

[137] VARGRAS-VERA, M.; DOMINGUE, J.; MOTTA, E.; SHUM, S.B.; LANZONI, M. (2001) "Knowledge extraction by using an ontology-based annotation tool". In: *Proceedings of Workshop on Knowledge Markup & Semantic Annotation, held in association with the 1st International Conference on Knowledge Capture, (Victoria Canada, October 21-23),* pp. 5-12.

[138] CHATTERJEE, A.; SEGEV, A. (1995) "Rule-based joins in heterogeneous databases" Decision Support Systems, 13(3-4), pp. 313-333.

[139] BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. (2001) "The Semantic Web", Scientific American, 284(5), pp. 34-43.

[140] CHEN, H.; FININ, T.; JOSHI. A. (2003) "Semantic Web in a Pervasive Context Aware Architecture", In: *Proceedings of the Artificial Intelligence in Mobile System Conference, (Seattle, USA, October 12-15),* pp. 33-40.

[141] LU, H.; OOI, B.C.; GOH, C.H. (1992) "On global multidatabase query optimization", ACM SIGMOD Record, 20(4), pp. 6-11.

[142] USCHOLD, M.; GRUNINGER, M. (2004), "Ontologies and Semantics for Seamless Connectivity", SIGMOD Record, 33(4), pp. 58-64.

[143] HITZLER, P.; KROTZSCH, M.; PARSIA, B.; PATEL-SCHNEIDER, P.F.; RUDOLPH, S. (2009) "OWL 2 Web Ontology Language Primer", W3C Working Draft 2009, available at http://www.w3.org/TR/2009/WD-owl2-primer-20090421/, [accessed June 2009].

[144] WONGTHONGTHAM, P.; CHANG, E.; SOMMERVILLE, I. (2006) "Software Engineering Ontology for Software Engineering Knowledge Management in Multi-site Software Development Environment", Journal of Systems Architecture: the EUROMICRO Journal - Special issue: AGILE methodologies for software production, 52 (11), pp. 640 – 653.

[145] GRUBER, T.R. (2003) "Collective Knowledge Systems: Where the Social Web meets the Semantic Web", Journal of Web Semantics, 6(1), pp. 4-13.

[146] PEIGUANG, L.; XU, R.; LY, C.; ZHANG, N. (2008) "An Ontology-based and Distributed Service for EG", In: *Proceedings of the International Conference on Internet Computing in Science and Engineering, (Harbin, China, January 28-29),* pp.485-491.

[147] BICER, V.; LALECI, G.B.; DOGAC, A.; KABAK, Y. (2005) "Providing Semantic Interoperability in Healthcare Domain through Ontology Mapping*", In: Proceedings of the eChallenges 2005, (Ljubljana, Slovenia, October 19-21), available at www.srdc.metu.edu.tr/.../2005healthcareSemanticInteroperability.doc, [accessed January 2009].

[148] JAMADHVAJA, M.; SENIVONGSE, T. (2005) "An Integration of Data Sources with UML Class Models Based on Ontological Analysis", In: *Proceedings of the 1st International Workshop on Interoperability of Heterogeneous Information Systems, associated to the Conference on information and Knowledge Management, (Bremen, Germany, October 31-November 5),* available at http://portal.acm.org/citation.cfm?id=1096967.1096969, [accessed in March 2009].

[149] BRAUNER, D.F.; CASANOVA, M.A.; LUCENA, C.J.P.; (2004) "Using Ontologies as Artifacts to Enable Databases Interoperability", University of Hamburg, Germany, available at http://swt-www.informatik.uni-hamburg.de/conferences/oopsla%202004%20-%20accepted/20040815_Brauner_OOPSLA_oopsla_2004_DFB.pdf, [accessed January 2009].

[150] OBERLE, D.; EBEHERT, A.; STAAB S.; VOLZ, R. (2004) "Developing and Managing Software Components in an Ontology-based Application Server", In: *Proceedings of the 5th International Conference Middleware, (Toronto, Ontario, Canada, October 18-22),* pp. 459-478.

[151] PARK, J.; RAM, S. (2004) "Semantic Conflict Resolution Ontology (SCROL): An Ontology for Detecting and Resolving Data and Schema-level Semantic Conflicts", IEEE Transactions on Knowledge and Data Engineering, 16(2), pp.189-201.

[152] SIDHU, A.S.; CHANG, E.; SIDHU, B.S. (2005) "Protein Ontology: Vocabulary for Protein Data", In: *Proceedings of the 3rd International Conference on Information Technology and Applications, (Sydney, Australia, July 4-7),* pp.465-469.

[153] DOU, D.; LEPENDU, P.; KIM, S.; QI, P. (2006) "Integrating Databases into the Semantic Web through an Ontology-based Framework", In: *Proceedings of the 22nd International Conference on Data Engineering Workshops, (Atlanta, Georgia, USA, April 3-7)*, pp. 54.

[154] ALANI, H.; KIM, S.; MILLARD, D.E.; WEAL, M. J.; HALL, W.; LEWIS, P. H. SHADBOLT, N. R. (2003) "Automatic Ontology-Based Knowledge Extraction from Web Documents", IEEE Intelligent Systems, 18(1), pp. 14-21.

[155] RAZMERITA, L.; ANGENHRN, A.; MAEDCHE, A. (2003) "Ontology-based User Modeling for Knowledge Management Systems", User Modeling, LNCS 2702, pp. 213- 217.

[156] CORAZZON, R. (2007) "Formal ontology: ONTOLOGY", A Resource for Philosophers, available at http://www.formalontology.it/, [accessed in March 2009].

[157] BORST, W. N. (1997) "Construction of Engineering Ontologies". PhD thesis, Center for telematica and information technology, University of Tweently, Enschede, NL, available at doc.utwente.nl/17864/1/t0000004.pdf, [accessed January 2009].

[158] GRUBER, T. R. (1993) "A translation approach to portable ontology specifications", Knowledge Acquisition, 5(2), pp.199–220.
[159] MCCARTHY, J. (1980) "Circumscription - A Form of Non-Monotonic Reasoning", Artificial Intelligence, 5(13), pp. 27-39.

[160] GRUBER, T.R. (2003) "It Is What It Does: The Pragmatics of Ontology", In: *invited talk at International Conference on Sharing the Knowledge, CIDOC CRM Symposium, (Washington, DC, March 26-27)*, available at http://tomgruber.org/writing/cidoc-ontology.htm, [accessed in March 2009].

[161] SOWA, J.F. (2007) "Building, Sharing, and Merging Ontologies", available at http://www.jfsowa.com/ontology/ontoshar.htm, [accessed January 2009].

[162] HUI, B.; YU, E. (2005) "Extracting conceptualizations from specialized documents", Data and Knowledge Engineering, 54(1), pp. 29-55.

[163] FENSEL, D. (2002) "Ontology based Knowledge Management" IEEE Computing, 35(11), pp. 56-59.

[164] STOJANOVIC, L.; MAEDCHE, A.; MOTIK, B.; STOJANOVIC, N. (2002) "User-Driven Ontology Evolution Management", Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web, LNCS 2473, pp. 285–300.

[165] LENAT, D. B.; GUHA, R. V. (1990) "Building Large Knowledge Bases", Addison-Wesley, Reading, MA.

[166] NECHES, R.; FIKES, R. E.; FININ, T.; GRUBER, T. R.; PATIL, R.; SENATOR, T.; SWARTOUT, W.R. (1991), "Enabling technology for knowledge sharing", AI Magazine, 12(3), pp. 16-36.

[167] HUMPHREYS, B.L.; LINDBERG, D. A. (1998) "The Unified Medical Language System: an informatics research collaboration", Journal of American Medical Association, 5(1), pp. 1-11.

[168] FENSAL, D.; ANGELE, J.; DECKER, S.; ERDMANN, M.; SCHNURR, H.P.; STAAB, S.; STUDER, R.; WITT, A. (1999) "On2broker: Semantic-based access to information sources at the www", . In: *Proc. of the World Conference on the WWW and Internet, (Honolulu, Hawaii, USA, October 24-30)*, pp.7.

[169] HESSE, W. (2006) "Ontologies in the Software Engineering Process", Faculty of Mathematics and Informatics, University of Marburg, available at citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.84.5229&rep, [accessed June 2009].

[170] APARICIO, A.S.; FARIS, O.L.M.; DOS-SANTOS, N. (2005) "Applying Ontologies in the Integration of Heterogeneous Relational Databases", In: *Proceedings of the Australian Ontology Workshop (AOW 2005), (UTS, Sydney, Australia, December 6)*, pp. 11-16.

[171] LUBTYE, L. (2007) "Reusing Relational Sources for Semantic Information Access", In: *Proceedings of the ACM first Ph.D. workshop in the Conference on Information and Knowledge Management, (Lisbon, Portugal, November 6-10)*, pp. 9-16.

[172] PARK, J.; RAM, S. (2004a) “Information Systems interoperability: What Lies Beneath?”, ACM Transactions on Information Systems, 22(4), pp. 595-632.

[173] SUNG, S.; MCLEOD, D. (2006) “Ontology-Driven Semantic Matches between Database Schemas”, In: *Proceedings of the 22nd IEEE International Conference on Data Engineering Workshops, (Atlanta, GA, USA, April 3-7)*, pp. 6.

[174] HAKIMPOUR, F.; GEPPERT, A. (2001) “Ontologies: an Approach to Resolve Semantic Heterogeneity in Databases”, project funded by Swiss national science Foundation (SNSF), Project number 2100-053995, available at http://www.ifi.uniz.zh/dbtg/Projects/MIGI, [accessed June 2009].

[175] HAKIMPOUR, F.; TUMPF, S. (2001) “Using Ontologies for Resolution of Semantic Heterogeneities in GIS”, In: *Proceedings of the 4th AGILE Conference in Geographic Information Science, (Brno, Czech Republic, April 19-21)*, pp. 385-395.

[176] MASUOKA, R.; PARSIA, B.; LABROU, Y. (2003) “Ontology-Enabled Pervasive Computing Applications”, IEEE Intelligent Systems, 18(5). pp. 68-72.

[177] STRIZH, I.G. (2006) “Ontologies for data and knowledge sharing in biology: plant ROS signalling as a case study”, BioEssays Wileys Periodicals Inc, 28(2), pp. 199-210.

[178] JOSHI, H.; SEKER, J.; BAVRAK, C.; RAMASWAMY, S.; CONNELLY, J.B. (2007) “Ontology for Disaster Mitigation and Planning”, In: *Proceedings of the summer computation simulation conference, (San Diego, California, July 15-18)*, Article No. 6.

[179] GARDENER, S.P. (2005) “Ontologies and Semantic data integration”, Drug Discovery Today, 10(14), pp. 1001-1007.

[180] ESTRELLA (2004) “The European project for Standardized Transparent Representations in order to Extend Legal Accessibility”, available at http://www.estrellaproject.org/index.php/Main_Page, [accessed in March 2009].

[181] MORK, P.; HALEVY, A.; TARCZY-HORNOCH, P. (2001) “A Model for Data Integration Systems of Biomedical Data Applied to Online Genetic Databases”, In: *Proceedings of the Symposium of the American Medical Informatics Association, (Washington DC, May 23)*, pp. 473-477.

[182] DEY, K. A. (2001) “Understanding and Using Context”, Personal and Ubiquitous Computing, 5(1), pp. 4-7.

[183] ABOWD, G.D.; DEY, K. A.; BROWN, J. P.; DAVIES, N.; SMITH, M.; STEGGLES, P. (1999) “Towards a Better understanding of Context and Context-Awareness”, Handheld and Ubiquitous Computing, LNCS 1707, pp. 304-304.

[184] BU, Y.; CHEN, S.; LI, J.; TAO, X.; LU, J. (2006) “Context Consistency Management Using Ontology Based Model”, Trends in Database Technology, LNCS 4254, pp. 741-755.

[185] SCHIMIDT, A. (2006) “Ontology-Based User Context Management: The Challenges of Imperfection and Time-Dependence”, On the Move to Meaningful Internet Systems: CoopIS, DOA, GADA, and ODBASE, LNCS 4275, pp. 995-1011.

[186] YING, X.; FU-YAUN, X. (2006) “Research on Context Modeling Based on Ontology”, In: *Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation, and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, (Sydney, Australia, November 28-December 1)*, pp. 188-188.

[187] TROUG, B.A.; LEE, Y.K.; LEE, S.Y. (2005) “Modeling Uncertainty in Context-Aware Computing”, In: *Proceedings of the 4th Annual ACIS International conference on Computer and Information Science, (Jeju, Korea, July 14-16)*, pp. 676- 681.

[188] STRANG, T.; LINNHOFF-POPIEN, C.; FRANK, K.; (2004) “CoOl: A Context Ontology Language to enable Contextual Interoperability”, Distributed Applications and Interoperable Systems, LNCS 2893, pp. 236-247.

[189] CROWLEY, J.L.; COUTAZ, J.; REY, G.; REIGNIER, P. (2002) "Perceptual Components for Context Aware Computing", In: *Proceedings of the 4th International Conference on Ubiquitous Computing (UbiComp 2002), (Gvteborg, Sweden, September 29-October 1)*, pp. 117-134.

[190] FLURY, T.; PRIVAT, G.; RAMPARANY, F. (2004) "OWL-based location ontology for context aware services", In: *Proceedings of the Artificial Intelligence in Mobile Systems, (Nottingham, UK,September 7)*, pp. 52-57

[191] BERNERS-LEE, T. (2009) "Linked Data", the TED conference 2009, The Great Unveiling in Long Beach, CA, USA, available at http://www.ted.com/index.php/talks/tim_berners_lee_on_the_next_web.html, [accessed January 2009].

[192] BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. (2001a) "The Semantic Web: A New Form of Web Content that is meaningful to Computers Will Unleash a Revolution of New Possibilities", available at http://www.informatics-review.com/thoughts/semantic.html, [accessed January 2009].

[193] BERNERS-LEE, T. (1998) "Semantic web road map", available at http://www.w3.org/DesignIssues/Semantic.html, [accessed January 2009].

[194] JANEV, V.; VRANES, S. (2009) "Semantic Web Technologies: Ready for Adoption?" IEEE IT Professional, 11(5), pp. 8-16.

[195] GRUBER, T.R. (2009) "Ontology in the Encyclopedia of database Systems", Ling Liu and M. Tamer Özsu (Eds.), Springer-Verlag, 2009.

[196] DOBSON, G.; SAWYER, P. (2006) "Revisiting Ontology based Requirements Engineering in the Age of Semantic Web" In: *International Seminar on Dependable Requirements Engineering of Computerised Systems at NPPs, (Halden, Norway, November 27-29)*, available at http://www.comp.lancs.ac.uk/~dobsong/papers/dre06, [accessed in March 2009].

[197] SHETH, A.; RAMAKRISHNAN, C. (2003) "Semantic (Web) Technology in Action: Ontology Driven Information Systems for Search, Integration and Analysis", IEEE Data Engineering Bulletin, Special issue on Making the Semantic Web Real, 26(4), pp. 40-48.

[198] CHAARI, T.; EJIGU, D.; LAFOREST, F.; SCUTURICI, V.M. (2006) "Modeling and Using Context in Adapting Applications to Pervasive Environments", In: *Proceedings of the International Conference on Pervasive Services (ICPS 2006), (Lyon, France, June 26-29)*, pp. 111-120.

[199] STRANG, T.; LINNHOFF-POPIEN, C.; FRANK, K.; (2004a) "Applications of a Context Ontology Language", In: *Proceedings of International Conference on Applications of a Context Ontology Language Software, Telecommunications and Computer Networks, (Split, Croatia, October 10-13)*, pp 14-18.

[200] CHRISTOPOULOU, E.; GOUMOPOULOS, C.; ZAHARAKIS, I.; KAMEAS, A. (2004) "An Ontology-based Conceptual Model for Composing Context Aware Applications", In: *Proceedings of the 6th International Conference on Ubiquitous Computing (UbiComp 2004), Workshop on Advanced Context Modeling, Reasoning and Management, (Nottingham, England, September 7-10)*, available at: http://pace.dstc.edu.au/cw2004/Paper26.pdf, [accessed in March 2009].

[201] TRIFAN, M.; IONESCU, B.; IONESCU, D.; PROSTEAN, O.; PROSTEAN, G. (2008) "An Ontology based Approach to Intelligent Data Mining for Environmental Virtual Warehouses of Sensor Data", In: *Proceedings of the IEEE International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems, (Istanbul, Turkey, July 14-16)*, pp.125 – 129.

[202] HONG, C.S.; KIM, H.; KIM, H.S.; LEE, H.C. (2006) "An Approach for Configuring ontology-Based Application Context Model", In: *Proceedings of the ACS/IEEE International Conference of Pervasive Services, (Lyon, France, June 26-29*), pp. 337- 340.

[203] WANG, X.; DONG, J.S.; CHIM, C.Y.; HETTIARACHCHI, S.R.; ZHANG, D. (2004) "Semantic Space: An Infrastructure for Smart Spaces", IEEE Computer Society, 3(3), pp. 32-39.

[204] WANG, X. H;, GU, T;, ZHANG, D.Q.; PUNG, H. K. (2004a), "Ontology Based Context Modelling and Reasoning using OWL", In: *Proceedings of the IEEE International Conference on Pervasive Computing and Communication Workshop on Context Modelling and Reasoning, (Orlando, Florida, March 14)*, pp. 18-24.

[205] CURINO, C.; QUINTARELLI, E.; TANCA, L. (2006) “Ontology based information tailoring”, In: *Proceedings of the 22nd International Conference on Data Engineering Workshops (ICDE 2006), (Atlanta, Georgia, USA, March 14-17)*, pp. 5.

[206] RACK, C.; ARBANOWSKI, S.; STEGLICH, S. (2005) “Context-aware, Ontology-based Recommendations”, In: *Proceedings International Symposium on Applications and the Internet Workshop, (San Diego, CA, USA, January 23-27)*, pp. 7-104.

[207] CHEN, H.; FINN, T.; PERICH, F.; JOSHI, A. (2004) “SOUPA: Standard ontology for Ubiquitous and Pervasive applications”, In: *Proceedings of the 1st Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous 2004), (Boston, Massachusetts, USA, August 22-26)*, pp. 258-267.

[208] XU. Y.; XU F. (2004) “Research on Context modeling Based on Ontology”, In: *Proceedings of the International Conference on Intelligent Agents, Web Technologies and Internet Commerce, (Gold Coast, Australia, July 12-14)*, pp. 188.

[209] CHEN, H.; FININ, T.; JOSHI. A. (2003a) “An Ontology for Context-Aware Pervasive Computing Environments”, The Knowledge Engineering Review, 18(3), pp. 197-207.

[210] O’SULLIVAN, D.; LEWIS, D. (2004) “Semantically Driven Service Interoperability for Pervasive Computing”, In: *Proceedings of the 3rd ACM International Workshop on Data Engineering for Wireless and Mobile Access, (San Diego, CA, US, September 19)*, pp. 17–24.

[211] OU, S.; GEORGALAS, N.; AZMOODEH, M.; YANG, K.; SUN, X. (2006) “A Model Driven Integration Architecture for Ontology-Based Context Modelling and Context-Aware Application Development”, Model Driven Architecture – Foundations and Applications, LNCS 4066, pp. 188-197.

[212] CONNOLLY, D.; HARMELEN, F.; HORROCKS, I.; MCGUINESS, D. L.; PATEL-SCHEIDER, P.; STEIN, L.A.; (2001) “DAML+OIL Reference Description”, W3C Recommendation, available at http://www.w3.org/TR/daml+oil-reference, [accessed in March 2009].

[213] HORROCKS, I.; PATEL-SCHNEIDER, P. F. (2004) “A proposal for an owl rules language”. In: *Proceedings of the 13th International World Wide Web Conference, (New York, USA, May 17-20)*, pp. 723–731.

[214] KIFER, M. (2008) “Rule Interchange Format: The Framework”, In: *Proceedings of the 2nd International Conference on Web Reasoning and Rule Systems, (Karlsruhe, Germany, 31 October - 1 November)*, pp. 1-11.

[215] BOLEY, H.; HALLMARK, G.; KIFER, M.; PASCHKE, A.; POLLERES, A.; REYNOLDS, D. (2010) “RIF Core Dialect”, W3C Recommendation, available at http://www.w3.org/TR/rif-core/, [accessed January 2009].

[216] BECKETT D.; MCBRIDE, B. (2004) “RDF/XML Syntax Specification (Revised)”, W3C Recommendation, available at http://www.w3.org/TR/REC-rdf-syntax/, [accessed January 2009].

[217] BRICKLEY, D.; GUHA, R.V. (2004) “RDF Vocabulary Description Language 1.0: RDF Schema”, W3C Recommendation, available at http://www.w3.org/TR/rdf-schema/, [accessed January 2009].

[218] PRUD’HOMMEAUX, E.; SEABOURNE, A. (2008), “SPARQL Query Language for RDF”, W3C Recommendation, available at http://www.w3.org/TR/rdf-sparql-query/, [accessed January 2009].

[219] SURE, Y.; ERDMANN, M.; ANGELE, J.; STAAB, S.; STUDER, R.; WENKE, D. (2002) “OntoEdit: Collaborative Ontology Engineering for the Semantic Web”, The Semantic Web, LNCS 2342, pp. 221-235.

[220] HAARSLEV, V.; MOLLER, R. (2003), “Racer: A core inference engine for the semantic web”. In: *Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools, Sure, Y., Corcho, O. (eds.)*, pp. 27–36.

[221] HA, Y. G.; SOHN, J.C.; CHO, Y.J. (2005) “Owler: a semantic web ontology inference engine”, In: *Proceedings of the 7th International Conference on Advanced Communication Technology, (Phoenix Park, Republic of Korea, February 21-23)*, pp. 1077-1080.

[222] DENNY, M. (2002) “Ontology Building: A Survey of Editing Tools”, XML Editing Ontologies, available at http://www.xml.com/pub/a/2002/ 11/06/ontologies.html, [accessed in March 2009].

[223] KIFER, M.; LAUSEN, G.; WU, J. (1995) “Logical Foundations of Object Orientation and Frame-Based Languages”, Journal of Association for Computing Machinery, 42(4), pp 741-843.

[224] HORRIDGE, M. (2010) “Protégé: OWL viz”, Protégé Wiki, available at http://protegewiki.stanford.edu/index.php/OWLViz, [accessed June 2009].

[225] STOREY, M. A.; MUSEN, M.; SILVA, J.; BESTI, C.; ERNST, N.; FERGERSON, R.; NOY, N. (2001) “Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé”, In: *Workshop on Interactive Tools for Knowledge Capture, (Victoria B.C., Canada, October 2001)*, available at: http://www.csr.uvic.ca/~mstorey/research/../papers, [accessed January 2009].

[226] BIZER, C. (2007) “D2RQ V0.5 - Treating Non-RDF Databases as Virtual RDF Graphs”, Freie Universitat Berlin, available at http://sites.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/, [accessed January 2009].

[227] NYLUS, C.; O’CONNOR, M.; TU, S. (2007), “Datamaster – a Plug-in for Importing Schemas and Data from Relational Databases into Protégé”, In: *Proceedings of 10th International Protégé Conference, (Budapest, Hungary, July 15-18)*, available at http://protege.stanford.edu/conference/20& 07/presentations/10.01_Nyulas.pdf, [accessed January 2009].

[228] GOLBREICH, C.; ZHANG, S. BODENREIDER, O. (2006) “The foundational model of anatomy in OWL: Experience and perspectives”, Journal of Web Semantics, 4(3), pp. 181-195.

[229] DERRIERE, S.; RICHARD, A.; PREITE-MARTINEZ, A. (2006) “An ontology of astronomical object types for the virtual observatory”, Highlights of Astronomy, 14, pp. 603-603.

[230] GOODWIN. J. (2005) “Experiences of using OWL at the ordnance survey”, In: *Proceedings of the 1st OWL Experiences and Directions Workshop, (Galway, Ireland, November 11-12)*, available at http://www.ordnancesurvey.co.uk/oswebsite/partnerships/research/publications/docs/2005/experiencesofusing OWL_JGoodwin_sem.pdf, [accessed in March 2009].

[231] LACY, L.; AVILES, G.; FRASER, K.; GERBER, W.; MULVEHILL, A.; GASKILL. R. (2005) “Experiences using OWL in military applications”, In: *Proceedings of the 1st OWL Experiences and Directions Workshop, (Galway, Ireland, November 11-12)*, available at http://www.mindswap.org/2005/OWLWorkshop/sub27.pdf, [accessed January 2009].

[232] SIDHU, A.; DILLON, T.; CHANG, E.; SIDHU. B.S. (2005) “Protein ontology development using OWL”, In: *Proceedings of the 1st OWL Experiences and Directions Workshop, (Galway, Ireland, November 11-12)*, available at http://sunsite.informatik.rwth-aachen.de/PublicationDBLP, [accessed January 2009].

[233] SOERGEL, D.; LAUSER, B.; LIANG, A.; FISSEHA, F.; KEIZER, J.; KATZ. S. (2004) “Reengineering thesauri for new applications: The AGROVOC example”, Journal of Digital Information, 4(4), available at http://journals.tdl.org/jodi/article/viewarticle/112/www4.fao.org/asfa, [accessed January 2009].

[234] FENSEL, D. (2008) “Semantic Technology - More Than Just an Appendix of the Web?”. Semantic Technology Institute (STI), available at http://www.sti-innsbruck.at/fileadmin/documents/Semantic_Technology.pdf, [accessed in March 2009].

[235] SPACCAPIETRA, S.; PARENT, C.; DUPONT, Y. (1992) “Model independent assertions for integration of heterogeneous schemas”, Very Large Database Journal, 1(1), pp. 81-126.

[236] FANG, D.; HAMMER, J.; MCLEOD, D. (1992) “An approach to behavior sharing in federated database systems”, In: *Proceedings of the International Workshop on Distributed Object Management, (Edmonton, Alberta, Canada, August 19-21)*, pp. 334-346.

[237] OUKSEL, A.M.; NAIMAN, C. (1992) “Towards the design of a semantic communication protocol”, In: *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, (Chicago, IL, USA, 18-21 October 18-21)*, pp. 1271 – 1276.

[238] KIM, W.; SEO, J. (1991) “Classifying schematic and data heterogeneity in multidatabase systems”, IEEE Computer, 24(12), pp.12.

[239] KRISHNAMURTHY, R.; LITWIN, W.; KENT, W. (1991) “Language features for interoperability of databases with schematic discrepancies”, ACM SIGMOD Record, 20 (2) pp. 40 – 49.

[240] NAIMAN, C.F; OUSKEL, A. M. (1995) “A classification of semantic conflicts in heterogeneous database systems”, Journal of Organizational Computing, 5(2), pp. 167-193.

[241] BITTNER, T.; DONNELLY, M.; WINTER, S. (2005) “Ontology and semantic interoperability”. Large-Scale 3D Data Integration: Problems and Challenges, CRC Press, Boca Raton, pp. 37-48.

[242] CRUZ, I.; XIAO, H. (2005) “The Role of Ontologies in Data Integration”, Journal of Engineering Intelligent Systems, 13(4), available at http://www.cs.uic.edu/~advis/publications/dataint/, [accessed in March 2009].

[243] OBRST, L. (2003) “Ontologies for semantically interoperable systems”, In: *Proceedings of the 12th International Conference on Information and Knowledge Management, (New Orleans, LA, USA, November 5-10)*, pp. 366-369.

[244] PARTRIDGE, C. (2002) "The role of ontology in integrating semantically heterogeneous databases", Technical Report 05/02, National Research Council, Institute of Systems Theory and Biomedical Engineering (LADSEB-CNR), available at http://74.125.155.132/scholar?q=cache:wlYO3nNGHbkJ:scholar.google.com/+The+role+of+ontology+in+integrating+semantically+heterogeneous+databases&hl=en&as_sdt=2000&as_vis=1, [accessed January 2009].

[245] CHATTERJEE, N.; KRISHNA, M. (2007) “Semantic Integration of Heterogeneous Database on the Web”, In: *Proceedings of the International Conference on Computing: Theory and Applications (ICCTA 2007), (Kolkata, India, March 5-7)*, pp.325-329.

[246] BARRESI, S.; REZGUI, Y.; LIMA, C.; MEZANIE, F.; (2005) “Architecture to Support Semantic Resources Interoperability”, In: *Proceedings of the 1st International Workshop on Interoperability of Heterogeneous Information Systems (IHIS 2005), (Oldenburg, Germany, November 4)*, pp. 79-82.

[247] CHOI. N.; SONG, I.Y.; HAN, H. (2006) “A Survey on Ontology Mapping - Deals with ontology mapping between heterogeneous ontologies”, ACM SIGMOD Record, 35(3), pp. 34-41.

[248] SALEEM, A. (2006) “Semantic Web Vision: survey of ontology mapping systems and evaluation progress”, Masters Thesis, Intelligent Software Systems, Thesis No:MCS-2006:13, Blekinge Institute of Technology, Sweden, available at: http://www.bth.se/fou/cuppsats.nsf/all/a72fe543e31b0cb7c125722c007edcc8/$file/MCS-2006-13.pdf, [accessed January 2009].

[249] KALFOGLOU, Y.; SCHORELMMER, M. (2003), “Ontology Mapping: The State of the Art”, The Knowledge Engineering Review, 18(1), pp. 1-31.

[250] KLEIN, M. (2001) “Combining and Relating Ontologies: An Analysis of Problems and Solutions”, In: *Proceedings of the 17th International Joint Conference Artificial Intelligence Workshop on Ontologies and Information Sharing, (Seattle, WA, April 4-5)*, pp. 53–62.

[251] DING, L.; KOLARI, P.; DING, Z.; AVANCHE, S.; FININ, T., JOSHI, A. (2005) “Using Ontologies in the Semantic Web: A Survey”, in TR-CS-05-07 (2005) UMBC

[252] TEMPLETON, M.; BRILL, D.; CHEN, A.; DAO, S.; LUND, E.; MACGREGOR, R.; WARD. P. (1987) “Mermaid - a front-end to distributed heterogeneous databases”, In: IEEE Special issue on distributed database issues, 75(12), pp. 695 - 708.

[253] JACOBSEN, G.; PIATETSKY-SHAPIRO, G.; LAFOND, C.; RAJNIKANTH, M.; HERNANDEZ, J.; CALIDA, A. (1988) “Knowledge-based System for Integrating Multiple Heterogeneous Databases”, In: *Proceedings of the 3rd International Conference on Data and Knowledge Bases, (Jerusalem, Israel, June 28-30)*, pp. 3-18.

[254] LITWIN, W.; ABDELLATIF, A.; (1986) “Multidatabase Interoperability”, IEEE Computer, 19(12), pp. 10-18.

[255] RAM, S.; RAMESH, V. (1999) “Schema Integration: Past, Current and Future, Management of Heterogeneous and Autonomous Database Systems”, San Francisco: Morgan Kaufmann, A. Elmagarmid, M. Rusinkeiwicz, and Amit P. Sheth, ed., pp. 119-155.

[256] LITWIN, W.; (1985) “An overview of the Mutidatabase System”, In: *Proceedings of the* ACM *annual C*onference *on the Range of Computing: mid-80's perspective, (*Denver, *Colorado, United States)*, pp. 524 - 533.

[257] AHMED, R.; DESMEDT, P.; DU, W.; KENT, W.; KETABCHI, M.; LITWIN, W.; RAFII, A.; SHAN, M.C. (1991) “The Pegasus Heterogeneous Multidatabase System”, 24(12), pp. 19-27.

[258] CHIRATHANJAREE, C. (2008) “A Data Model for Heterogeneous Data Sources”, In: *Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE 2008), (Xi’an, China, October 22-24)*, pp.121-127.

[259] DONELSON, L.; TARCZY-HORNOCH, P.; MORK, P.; DOLAN, C.; MITCHELL, J.A.; BARRIER, M.; MEI, H. (2004) “The BioMediator System as a Data Integration Tool to Answer Diverse Biologic Queries”, MEDINFO, M. Fieschi et al. (Eds), Amsterdam: IOS Press, pp. 768-72, available at http://www.biomediator.org/publications/donelson-2004-medinfo-data_integration_tool.pdf, [accessed in March 2009].

[260] NOY, N.F. (2004) “Semantic integration: A Survey of Ontology Based Approaches”, In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, (Paris, France, June 13-18)*, pp. 65-70.

[261] NOY, N.F.; MCGUINESS D.L. (2001) “Ontology Development 101: A Guide to Creating your first Ontology”. Technical Report Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, Stanford University, available at http://protege.stanford.edu/publications/ontology_development/ontology101.pdf, [accessed January 2009].

[262] EHRIG, M.; SURE, Y. (2004) “Ontology mapping - an integrated approach”, The Semantic Web: Research and Applications, Springer LNCS 3053, pp. 76–91.

[263] SHVAIKO, P.; EUZENAT, J. (2004) “A Survey of Schema based Matching Approaches”, Technical Report DIT-04-087, Informatica e-Telecomunicazioni, University of Trento, available at http://www.dit.unitn.it/~p2p/RelatedWork/Matching/JoDS-IV-2005_SurveyMatching-SE.pdf, [accessed January 2009].

[264] VAPNIK, V.N. (1999) “The nature of statistical learning theory”, Statistics for engineering and information science. New York, 2nd edition, Springer.

[265] WANG, J.T.L.; SHAPIRO, B.A.; SHASHA, D.; ZHANG, K.; CURREY, K.M. (1998) “an Algorithm for Finding the Largest Approximately Common Substructures of Two Trees”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(8), pp. 889-895.

[266] MARQUES, D. (2005) “A Survey of Recent Research in Ontology Mapping”, Simon Fraser University, school of Interactive Arts and Technology, available at http://www.sfu.ca/~mhatala/iat881/2005/DM-OntologyMapping.pdf, [accessed January 2009].

[267] MADHAVAN, J.; BERNSTEIN. P. A.; DOMINGOS. P.; HALEVY, A.H. (2002) “Representing and Reasoning about Mappings between Domain Models”, In: *Proceedings of the 18th International Conference on Artificial intelligence, (Edmonton, Alberta, Canada, July 28-August 1)*, pp. 80-86.

[268] GREINER, R.; DARKEN, C.; SANTOSO, I. “Efficient reasoning”, Computing Surveys, 33(1), pp. 30.

[269] STRACCIA, U. (2008) “Managing Uncertainty and Vagueness in Description Logics, Logic Programs and Description Logic Programs”, Reasoning Web, LNCS 5224, pp. 54-103.

[270] BOLEY, H.; KIFER, M.; PATRANJAN, P.L.; POLLERES, A. (2007), “Rule Interchange on the Web”, Reasoning Web, LNCS 4636, pp. 269-309.

[271] EITER, T.; IANNI, G.; POLLERES, A.; SCHINDLAUER, R.; TOMPITS, H. (2006) “Reasoning with rules and ontologies”, Reasoning Web, LNCS 4126, pp. 93-127.

[272] ROSATI, R. (2006) “Integrating Ontologies and Rules: Semantic and Computational Issues”, Reasoning Web, LNCS 4126, pp. 128–151.

[273] EITER, T.; IANNI, G.; POLLERES, A.; SCHINDLAUER, R.; TOMPITS, H. (2008) “Rules and Ontologies for the Semantic Web”, Reasoning Web, LNCS 5424, pp. 1-53.

[274] LUTHER, M.; MROHS, B.; WAGNER, M.; STEGLICH, S.; KELLERER, W. (2005) “Situation reasoning - a practical OWL use case”, In: *Proceedings of the International Symposium on* Autonomous Decentralized, (*Chengdu, China,* April *4-8),* pp. 461-468.

[275] SHOJANOORI, R.; JURIC, R.; LOHI, M. (2010) “Towards Balanced Distribution of Computations through Automated Reasoning”, In: *Proceedings of the 15th International Conference on System Design and Process Science, (Dallas, US, June 6-11),* CD-Rom.

[276] OGNYANOV, D.; KIRYAKOV, A. (2002) “Tracking Changes in RDF(S) Repositories”, Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web, LNCS 2473, pp. 373– 378.

[277] HEFLIN, J.; HENDLER, J. (2000) “Dynamic Ontologies on the Web”, In: *Proceedings of the 17th National Conference on Artificial Intelligence, (Austin, Texas, USA,* July 30-August 3*),* pp. 443–449.

[278] MAEDCHE, A.; MOTIK, B.; STOJANOVIC, L.; STUDER, R.; VOLZ, R. (2002) “Managing Multiple Ontologies and Ontology Evolution in Ontologging”, In: Proceedings of the 17th Conference on Intelligent Information Processing (Montréal, Québec, Canada, August 25-30), pp. 51–63.

[279] ARPIREZ, J.C.; CORCHO, O.; FERNANDEZ-LOPEZ, M.; GOMEZ-PEREZ, A. (2001) “WebODE: A scalable ontological engineering workbench”, Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web, LNCS 2473 pp. 295-310.

[280] NOY, N.F.; FERGERSON, R. W.; MUSEN, M. A. (2000) “The knowledge model of Protege-2000: Combining interoperability and flexibility” In: *Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management, (Juan-les-Pins, France),* pp. 69-82-82.

[281] EKLUND, P.; ROBERTS, N.; GREE, S. (2002) “OntoRama: Browsing RDF Ontologies using a Hyperbolic-style Browser”, In: *Proceedings of the 1st* International Symposium on Cyber Worlds, *(Tokyo, Japan, November 6-8),* pp. 405 – 411.

[282] SWARTOUT, B.; PATIL, R.; KNIGHT, K.; RUSS, T. (1996) “Towards Distributed use of large-scale ontologies”, In: Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, *(Banff, Canada, November 9-14),* pp. 138-145.

[283] FARQUHAR, A.; FIKES, R.; RICE, J. (1997) “The Ontolingua Server: A Tool for Collaborative Ontology Constructions”, *International Journal of Human-Computer Studies*, 46(6), pp. 707-728.

[284] AHMAD, M.N.; COLOMB, R.M. (2007) “Overview of Ontology Server Research”, Journal of Webology 4(2), pp. 1-7.

[285] VISSER, P.R.S.; CUI, Z. (1998) “Heterogeneous ontology structures for distributed architectures”, In: *Proceedings of the 13th European Conference on Artificial Intelligence,* Workshop on Applications of Ontologies and Problem-Solving-Methods, *(Brighton, England, August 24-25),* pp. 112–119.

[286] BASS, L.; CLEMENTS, P.; KAZMAN, P. (2003) “Software Architecture in Practice”, Addison and Wesley, 2nd Edition, SEI Series in Software Engineering.

[287] CLEMENTS, P.; KAZMAN, R.; KLIEN, M. (2001) “Evaluating Software Architectures: Methods and Case Studies”, Addison and Wesley, SEI Series in Software Engineering.

[288] BASS, L.; KAZMAN, R. (1999) “Architecture-Based Development”, Technical Report, CMU/SEI-99-TR-007, ESC-TR-99-007, available at http://www.sei.cmu.edu/reports/99tr007.pdf, [accessed January 2009].

[289] MARC, E.; STEFFEN, S. (2004) “QOM- Quick Mapping”, The Semantic Web, LNCS 3298, pp. 683-697.

[290] KNUBLAUCH, H.; FERGERSON, R.W.; NOY, N.; MUSEN, A.; (2004) “The Protégé OWL Plugin: An open Development Environment for Semantic Web Applications”, Stanford Medical Informatics, Stanford

School of Medicine, available at: http://protege.stanford.edu/plugins/owl/publications/ISWC2004-protege-owl.pdf, [accessed January 2009].

[291] KOAY, N.; KATARIA, P.; JURIC, R. (2010) "Semantic Management of Non-Functional Requirements in e-Health Systems", Telemedicine and e-Health Journal, 16(4), pp. 461-471.

[292] KOAY, N.; KATARIA, P.; JURIC, R.; TERSTYANSKY, G.; OBERNDORF, P. (2009) "Ontological Support for Managing Non-Functional Requirements in Pervasive Healthcare", In: *Proceedings of the 42nd Hawaii International Conference on System Science, (Waikoloa, Big Island, Hawaii, January 5-8)*, pp. 1-10.

[293] KATARIA, P.; JURIC, R. (2010a) "Automated Reasoning in Resolving Semantic Conflicts across Heterogeneous Repositories", In: *Proceedings of the 17th Automated Reasoning Workshop - – Bridging the gap between theory and practice, (University of Westminster, Harrow, Middlesex, UK, March 30-31)*, available at http://www2.wmin.ac.uk/bolotoa/ARW/arw-2010.html, [accessed January 2009].

[294] KATARIA, P.; JURIC, R. (2009) "Sharing Healthcare Data by Manipulating Ontological Individuals", In: *Proceedings of the 12th International Conference System Design and Process Science, (Montgomery, Alabama, US, November 1-5)*, CD-ROM.

[295] GEGOV, A. (2009) "Complex systems modelling by rule based networks", In: *Proceedings of the 8th World Scientific and Engineering Academy and Society International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases, (Cambridge, UK, February 21-23)*, pp.122-127.

[296] SHOJANOORI, R.; NEMITSAS, P.; KRISHNAMURTH, A. (2008), "Generation of Ontologies from Relational Databases: An Experience", In: P*roceedings of the 11th International Conference on Integrated Design and Process Technology, (Taichung, Taiwan, June 1-6)*, CD-ROM.

[297] JAKIMAVICIUS, T.R.; KATARIA, P.; JURIC, R. (2009) "Semantic Support for Dynamic Changes in Enterprise Business Models", In: *Proceedings of the 12th International Conference on Integrated Design and Process Technology, (Alabama Montgomery, USA, November 1-5)*, CD-ROM.

[298] FILIP, E.A.; KATARIA, P.; JURIC, R. (2009) "Intelligent Business Process Improvement", In: *Proceedings of the 12th International Conference on Integrated Design and Process Technology, (Alabama Montgomery, USA, November 1-5)*, CD-ROM.

[299] FILIP, E.A.; KATARIA, P.; JURIC, R. (2009) "Intelligent Business Process Improvement", In: *Proceedings of the 12th International Conference on Integrated Design and Process Technology, (Alabama Montgomery, USA, November 1-5)*, CD-ROM.

[300] SAAIDI, S.R.; KATARIA, P.; JURIC, R. (2009), "Semantic Management of the Submission Process for Medicinal Products Authorisation", In: *Proceedings of the 12th International Conference on Integrated Design and Process Technology, (Alabama Montgomery, USA, November 1-5)*, CD-ROM.

[301] OVEREND, G.E. (2008), "Introduction to eCTD and first principles. TOPRA", The Organisation for Professionals in Regulatory Affairs, available at http://www.topra.org/files/focus1_0.pdf, [accessed January 2007].

[302] BRATT, S. (2007) "Semantic Web and other Technologies to Watch", W3C, available at http://www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb/#(2), [accessed December 2009].

[303] FENSEL, D.; MUSEN, M.A. (2001) "The Semantic Web: A Brain for Humankind", IEEE Intelligent Systems, 16(2), p.24-25.

[304] KATARIA, P.; JURIC, R. (2010b) "Sharing Healthcare Data through Ontological Layering" In: *Proceedings of the 43rd Annual Hawaii International Conference on System Sciences, (Kauai, Hawaii, January 5-8)*, pp. 1-10.

[305] KATARIA, P.; KOAY, N.; JURIC, R.; MADANI, K.; TESANOVIC, I. (2008) "Ontology for Interoperability and Data Sharing in Healthcare", In: *Proceedings of the 4th International Conference on Advances in Computer Science and Technology (Langkawi, Malaysia, April 2-4)*, CD-ROM.

[306] KATARIA, P.; JURIC, R.; PAUROBALLY, S.; MADANI, K. (2008b) "Implementation of Ontology for Intelligent Hospital Wards", In: *Proceedings of the 41st Hawaii International Conference on System Science, (Hawaii, Big Island, January 7-10)*, pp.1-8.

[307] KATARIA, P.; JURIC, K.; MADANI, K.; CROFT, J. (2007) "Building Ontology for Intelligent Software Applications in Hospitals", In: *Proceedings of the 10th International Conference on Integrated Design and Process Technology, (Antalya, Turkey, June 3-8)*, CD-ROM.

[308] KATARIA, P.; JURIC, K.; MADANI, K.; (2007a) "Go-CID: Generic Ontology for Context Aware, Interoperable and Data Sharing Applications", In: *Proceedings of the 11th International Conference on Software Engineering Applications (Cambridge, MA, US, November 19-21)*, CD-ROM.

[309] KATARIA, P.; MACFIE, A.; JURIC, K.; MADANI, K.; (2009) "Ontology for Supporting Context Aware Applications for the Intelligent Hospital Ward", Transaction of Integrated Design & Process Science Tran-Disciplinary International Journal, 12 (3), pp. 35-44.

[310] BAHRAMI, A.; YUAN, J.; SMART, P.R.; SHADBOLT, N. R. (2007) "Context-Aware Information Retrieval for Enhanced Situation Awareness", In: Military Communications Conference (MILCOM 2007), (Orlando, Florida, USA, October 29-31), pp. 29-31.

[311] Gauvin, M.; Boury-Brisset A.; Auger, A. (2004) "Context, Ontology and Portfolio: Key Concepts for a Situational Awareness Knowledge Portal", In: *Proceedings of the 37th Hawaii International Conference on System Sciences (HICSS 37), (Big Island, HI, USA, January 5-8)*, pp.10.

[312] O'SULLIVAN, D.; LEWIS, D.; WADE, V. (2004), "Enabling Adaptive Semantic Interoperability for Pervasive Computing", Knowledge and Data Engineering Group, Trinity College Dublin, available at http://www.scss.tcd.ie/Dave.Lewis/files/04m.pdf, [accessed January 2009].

[313] GANGULY, S.; KATARIA, P.; JURIC, R.; ERTAS, A.; TANIK, M. M. (2009) "Sharing Information and Data across Heterogeneous e-Health Systems", Tele-Medicine and e-Health Journal, 15(5), pp. 454-464.

[314] PARSIA, B.; SIRIN, E.; KALYANPUR, A. (2005) "Debugging OWL Ontologies", In: *Proceedings of the 14th International Conference on World Wide Web, (Chiba, Japan, May 10-14)*, pp. 633 – 640.

[315] HOHAN, A.; HARTH, A.; POLLERES, A.; (2008) "Scalable Authoritative OWL Reasoning on a Billion Triples", In: *Proceedings of 7th International Semantic Web Conference, Billion Triple Semantic Web Challenge (Karlsruhe, Germany, October 26-30)*, available at http://sw.deri.org/~aidanh/docs/saor_billiontc08.pdf, [accessed January 2009].

[316] HORROCKS, I. (2002) "DAML+OIL: a description logic for thesemantic web", IEEE Data Engineering Bulletin, 25(1), pp. 4–9.

Index