Knowledge Engineering with Semantic Web Technologies for Decision Support Systems based on Psychological Models of Expertise

ASIA RAMZAN

Doctor of Philosophy

ASTON UNIVERSITY

June 2016

Aston University

'Knowledge Engineering with Semantic Web Technologies for Decision Support Systems based on Psychological Models of Expertise'

Asia Ramzan

Doctor of Philosophy, 2016

Thesis summary

Machines that provide decision support have traditionally used either a representation of human expertise or used mathematical algorithms. Each approach has its own limitations. This study helps to combine both types of decision support system for a single system. However, the focus is on how the machines can formalise and manipulate the human representation of expertise rather than on data processing or machine learning algorithms. It will be based on a system that represents human expertise in a psychological format. The particular decision support system for testing the approach is based on a psychological model of classification that is called the Galatean model of classification. The simple classification problems only require one XML structure to represent each class and the objects to be assigned to it. However, when the classification system is implemented as a decision support system within more complex real-world domains, there may be many variations of the class specification for different types of object to be assigned to the class in different circumstances and by different types of user making the classification decision. All these XML structures will be related to each other in formal ways, based on the original class specification, but managing their relationships and evolution becomes very difficult when the specifications for the XML variants are text-based documents. For dealing with these complexities a knowledge representation needs to be in a format that can be easily understood by human users as well as supporting ongoing knowledge engineering, including evolution and consistency of knowledge.

The aim is to explore how semantic web technologies can be employed to help the knowledge engineering process for decision support systems based on human expertise, but deployed in complex domains with variable circumstances. The research evaluated OWL as a suitable vehicle for representing psychological expertise. The task was to see how well it can provide a machine formalism for the knowledge without losing its psychological validity or transparency: that is, the ability of end users to understand the knowledge representation intuitively despite its OWL format. The OWL Galatea model is designed in this study to help in automatic knowledge maintenance, reducing the replication of knowledge with variant uncertainties and support in knowledge engineering processes. The OWL-based approaches used in this model also aid in the adaptive knowledge management. An adaptive assessment questionnaire is an example of it, which is dynamically derived using the users age as the seed for creating the alternative questionnaires. The credibility of the OWL Galatea model is tested by applying it on two extremely different assessment domains (i.e. GRiST and ADVANCE). The conclusions are that OWL-based specifications provide the complementary structures for managing complex knowledge based on human expertise without impeding the end users' understanding of the knowledge-base. The generic classification model is applicable to many domains and the accompanying OWL specification facilitates its implementations.

Keywords: galatea, GDSS, GRiST, ADVANCE, OWL, SWRL

Dedication

to my parents

Acknowledgments

Enormous thanks are due to my supervisors, Dr. Hai Wang and Dr. Christopher Buckingham, both of them have patiently advised me and supported me throughout my PhD studies, with great care and knowledge. I appreciate the support of both for their insightful guidance in improving my work.

Nothing in this short paragraph can express my gratitude. I have also received help and support from many colleagues and friends during my PhD studies. I am thankful to Dr. Errol Thompson for the advice about writing to Marilyn Thompson. I am also grateful of Dr. Sameera Abar in helping and guiding me regarding the corrections in thesis. I am very thankful to Ali Rezaei Yazdi in the presentation of the conference paper. I am also thankful to Dr. Diar Nasiev for helping mathematical problems, and to Khalil Al Ruqeishi and Ashraful Islam in discussions, reviews and suggestions on my research work. I am also thankful to Ishrat Hayat Malik, Irfana Sohail, Madhia Omar, Tahira, Dr. Hira, Irum Maria, Fozia Faiz, Shehella Tanveer and Irum Rasool for their companionship and patience especially when I experienced hard time.

Finally, I am thankful to my family: my parents, brother (Abdul Rehman), sisters (Rasheeda Ishaq and Asma Faisal) and my nieces and nephews for their great support and care in my life. It would have been impossible without their unremitting support that I could achieve this milestone of a doctoral degree!

Declaration

I, Asia RAMZAN, declare that this thesis titled, 'Knowledge Engineering with Semantic Web Technologies for Decision Support Systems based on Psychological Models of Expertise' and the work presented in it are my own. I confirm that this thesis has been generated by me as the result of my own original research. Parts of this work has been published as:

Asia Ramzan, Hai Wang and Christopher Buckingham. Representing Human Expertise by the OWL Web Ontology Language to Support Knowledge Engineering in Decision Support Systems, 'KES-InMed-14', Spain, 2014.

Signed: _____

Date: _____

Contents

Contents 6

List of Figures 11

List of Tables 15

1 Introduction 16
- 1.1 Research problem and challenges . . . 20
- 1.2 Aims and objectives . . . 21
- 1.3 Thesis contributions . . . 22
- 1.4 Thesis plan . . . 24

2 Intelligent Decision Support System 26
- 2.1 Introduction . . . 26
  - 2.1.1 Decision making for the domains of the psychological model . . . 28
    - 2.1.1.1 Applications of the Galatean model of classification . . . 29
    - 2.1.1.2 The limitations of current knowledge representation techniques . . . 31
- 2.2 Web-based decision support systems . . . 32
  - 2.2.1 Semantic Web . . . 34
    - 2.2.1.1 Description Logic . . . 34
  - 2.2.2 Semantic Web standards . . . 35
    - 2.2.2.1 Extensible Markup Language (XML) . . . 35
    - 2.2.2.2 Resource Description Framework (RDF) . . . 38
    - 2.2.2.3 Web Ontology Language (OWL) . . . 40
    - 2.2.2.4 Semantic Web Rule Language (SWRL) . . . 40
- 2.3 Semantic Web decision support systems . . . 41
  - 2.3.1 Requirements for Semantic Web decision support systems . . . 42
  - 2.3.2 Existing Semantic Web decision support systems . . . 43
- 2.4 Conclusions . . . 45
  - 2.4.1 Ontology / knowledge-base evolution . . . 45
  - 2.4.2 Information normalisation . . . 46
  - 2.4.3 Context sensitivity . . . 47
  - 2.4.4 Adaptation of ontology . . . 47

3 The Galatean Model of Classification 49
- 3.1 Introduction . . . 49

3.2 The Galatean Model . . . 51
3.2.1 The hierarchical Galatean model and its classification process . . 52
3.3 The knowledge structure of the domains of the Galatean model of classification . . . 55
3.3.1 XML knowledge tree . . . 55
3.3.1.1 Super Structure Tree (SST) . . . 55
3.3.1.2 Structure Tree (ST) . . . 56
3.3.1.3 Relative Influence Tree (RIT) . . . 56
3.3.1.4 Client Assessment Tree (CAT) . . . 56
3.3.1.5 Question Tree (QT) . . . 57
3.3.2 The types of nodes in XML trees . . . 57
3.3.2.1 The generic nodes . . . 57
3.3.2.2 The non-generic nodes . . . 59
3.3.3 Attributes of the nodes in XML trees . . . 59
3.3.3.1 The '*code*' attribute . . . 60
3.3.3.2 The label attribute . . . 60
3.3.3.3 The '*generic-type*' attribute . . . 60
3.3.3.4 The '*generic*' attribute . . . 62
3.3.3.5 The '*generic-datum*' attribute . . . 63
3.3.3.6 The '*populations*' attribute . . . 64
3.3.3.7 The '*prune-for*' attribute . . . 64
3.3.3.8 The '*question*' and '*filter-q*' attributes . . . 64
3.3.3.9 The '*level-question*' attribute . . . 66
3.3.3.10 The '*level-code*' attribute . . . 66
3.3.3.11 The '*multiple-tick*' attribute . . . 68
3.3.3.12 The '*ri*' attribute . . . 69
3.3.3.13 The '*value-mg*' attribute . . . 69
3.3.3.14 The '*values*' attribute . . . 69
3.4 Why OWL? . . . 72
3.5 Conclusions . . . 75

4 Overview of Semantic Web Technologies 78
4.1 Introduction . . . 78
4.2 Semantic Web . . . 78
4.2.1 Web Ontology Language (OWL) . . . 80
4.2.1.1 OWL sub-languages . . . 81
4.2.1.2 OWL individuals . . . 82
4.2.1.3 OWL Properties . . . 82
4.2.1.4 OWL classes . . . 84
4.2.2 Semantic Web Rule Language (SWRL) . . . 91
4.3 Conclusions . . . 93

5 Specifications of galateas in OWL 94
5.1 Introduction . . . 94
5.2 Ontology development methodology . . . 95
5.2.1 Existing ontology engineering methodologies . . . 95
5.2.1.1 Analysis of methodologies . . . 97

5.2.2 Design Process . . . 98
5.2.2.1 Specification . . . 98
5.2.2.2 Conceptualisation . . . 98
5.2.2.3 Implementation or formalisation . . . 99
5.2.2.4 Evaluation . . . 99
5.2.2.5 Maintenance . . . 99
5.3 The OWL Galatea model . . . 100
5.3.1 Node . . . 103
5.3.1.1 The Concept Node (‘*ConceptNode*’) . . . 107
5.3.1.2 The Datum Node (‘*DatumNode*’) . . . 109
5.3.2 Question . . . 112
5.3.2.1 FilteredQuestion . . . 112
5.3.3 Assessment-Type . . . 113
5.3.4 Value-Mg-Pair . . . 114
5.3.4.1 The MGs calculations from the value-mg lists having numerical data types . . . 118
5.3.4.2 The MGs calculations from the value-mg lists having nominal data types . . . 123
5.4 Conclusions . . . 124

6 Extensibility of the OWL Galatea model 127
6.1 Introduction . . . 127
6.2 Domains of the Galatean model of classification . . . 128
6.2.1 Mental health . . . 128
6.2.1.1 GRiST data structure . . . 129
6.2.2 Logistics . . . 129
6.2.2.1 ADVANCE data structure . . . 130
6.2.3 Commonalities and differences in the data structure of GRiST and ADVANCE tools . . . 130
6.3 Generating the SST ontology for GRiST and ADVANCE by extending the OWL Galatea model . . . 132
6.3.1 Extensibility of the ‘*Assessment-Type*’ class and the translation of the ‘*population*’ attribute . . . 132
6.3.1.1 The assessment types of the GRiST . . . 133
6.3.1.2 The assessment types of the ADVANCE . . . 135
6.3.2 Translation of the ‘*code*’ attribute . . . 137
6.3.3 Translation of the ‘*label*’ and ‘*help*’ attributes . . . 137
6.3.4 Extensibility of the subclasses of the ‘*Node*’ class and the translation of the ‘*generic*’, ‘*generic-type*’ and ‘*generic-datum*’ attributes 138
6.3.5 Extensibility of the ‘*Value-Mg-Pair*’ class and the translation of the ‘*value-mg*’ attribute . . . 140
6.3.6 Translation of the ‘*values*’ attribute . . . 142
6.3.7 Translation of the ‘*prune-for*’ attribute . . . 142
6.3.8 Extensibility of the ‘*Question*’ and ‘*FiltredQuestion*’ classes and the translation of the ‘*question*’ and ‘*filter-q*’ attributes . . . 145
6.3.9 Translation of the ‘*level*’ attribute . . . 147
6.3.10 Translation of the interim or temporary attributes . . . 148

6.4 Conclusions 149

7 Knowledge-bases Transformations by using OWL Reasoner 152
7.1 Introduction 152
7.2 Transformation of the ST ontology from the SST ontology 153
7.2.1 The transformation process for the ST ontology from the SST ontology 154
7.3 Transformation of the RIT ontology from the ST ontology 159
7.3.1 The RIs distribution or initialisation mechanism 160
7.3.2 The transformation process for the RIT ontology from the ST ontology 160
7.4 Transformation of the CAT from the RIT 166
7.4.1 The transformation process for the CAT ontology from the RIT ontology 167
7.5 Transformation of the QT from the ST or RIT ontology 168
7.5.1 The transformation of the QT ontology 169
7.6 The benefits of OWL based approach 171
7.6.1 Checking the consistency of the knowledge base of the domains of the Galatean model of classification 171
7.6.2 Automatic knowledge maintenance of complicated knowledge base 172
7.6.3 Automatical generation of the ST ontology from the SST ontology 172
7.6.4 Provision of generic and flexible knowledge-base 175
7.6.5 Provision of domain independent transformations 177
7.6.6 Provision of coherent and intelligible knowledge-base 177
7.7 Conclusions 178

8 Dynamically Generation of the Adaptive Assessment Questionnaire 180
8.1 Introduction 180
8.2 The assessment user and its attributes 182
8.3 Generation of an adaptive assessment questionnaire according to the age of the assessment user 182
8.4 Benefits of adaptation 184
8.4.1 Provide run-time adaptation 184
8.4.2 Reduce the uses of a series of complex knowledge trees 185
8.5 Conclusions 185

9 Evaluation 187
9.1 Introduction 187
9.2 Case-studies 188
9.2.1 Knowledge structure modification according to user's requirements 188
9.2.2 Knowledge management for the forensic service 190
9.3 Comparison of the present XML structures with the XML structures generated through the OWL-based specifications 191
9.3.1 SST structure 192
9.3.1.1 Path translation 192
9.3.1.2 Minimise the syntax dependency of the knowledge 194
9.3.1.3 Determination of the nodes types 195
9.3.1.4 Generic section 196

9.3.2 ST structure 198
9.3.2.1 Unsuitable knowledge chunks 198
9.3.2.2 Syntax independent knowledge management 199
9.3.3 RIT structure 200
9.3.3.1 Initialisation of uncertain attribute 201
9.3.4 CAT and QT structure 201
9.3.4.1 Elimination of Interim variables 202
9.4 Conclusions 203

10 Conclusions and Future Research **205**
10.1 Summary and Conclusions 205
10.2 Directions for future research 216
10.2.1 Machine support for classification process 216
10.2.2 Integrate adaptive assessment questionnaire with an ontology-based adaptive Information Collection Systems (ICS) 217

Bibliography **231**

List of Figures

3.1 The “suicide” risk is an individual concept in GRiST. The intention to commit suicide is the sub concept of the suicide-risk structure [1]. Concepts are portrayed as ovals and datum components are rectangles. 52
3.2 Hypothetical example of how membership grades (MGs). RIs are relative influences, which represent the weights of data and concepts [2]. 53
3.3 An example of contextual concepts in GRiST. 58
3.4 An example of ‘*code*’ attribute in SST.xml.. 60
3.5 An example of ‘*generic-type*’ attribute with ‘*g*’ value in SST.xml. 61
3.6 An example of ‘*generic-type*’ attribute with ‘*gd*’ value in SST.xml. 61
3.7 An example of a generic datum node having a fixed value-mg list in SST.xml. 62
3.8 An example of a generic distinct datum node having different value-mg lists in SST.xml. 62
3.9 An example of ‘*generic*’ attribute in RIT.xml. 62
3.10 An example of a node that has the ‘*generic*’ attribute with its definition in the CAT.xml in GRiST. 63
3.11 An example of ‘*generic-datum*’ attribute in SST.xml. 63
3.12 An example of the ‘*populations*’ attribute in ADVANCE tool. 64
3.13 An example of a node that has the ‘*prune-for*’ attribute. 64
3.14 An example of ‘*question*’ and ‘*filter-q*’ attributes in SST.xml. 65
3.15 An example of ‘*level*’ attributes in SST.xml. 65
3.16 An example of ‘*level-question*’ attribute in RIT.xml. 66
3.17 An example of the ‘*level-code*’ attribute of a node (i.e. gen-presentation) that has a generic concept ancestor (i.e. gen-depression) in the RIT.xml of the GRiST. 67
3.18 An example of the ‘*level-code*’ attribute of a node (i.e. gen-presentation) that has an individual concept ancestor (i.e. suic) in the RIT.xml of GRiST. 67
3.19 An example of ‘*multiple-tick*’ attribute in SST.xml. 68
3.20 An example of ‘*value-mg*’ attribute in SST.xml in ADVANCE. 69
3.21 An example of ‘*values*’ attribute in SST.xml. 70
3.22 An example of a root agent and two derived agents, each derived agent is obtained for a particular end user. The edges and nodes of the derived agents are depicted in dotted form. 74

4.1 Semantic Web Layer Cake. 80

5.1 The architecture of the OWL Galatea model. The arrows of the edges are indicated is-a or sub and super class relationships. The Thing is a predefined class. The Node, Question, Assessment-Type and Value-Mg-Pair are core classes of the OWL Galatea model that are depicted in different background colours. The names of these classes are represented in specified dark colours and their constraints and relationships are represented in specified light colours. The Question class is depicted in red colour, and the Node class and its subclasses are represented in blue colour. The Value-Mg-Pair class is depicted in green colour and the Assessment-type class is represented in yellow colour. 102
5.2 An example of a basic tree structure with universal restriction in the SST. 104
5.3 An example of an OWL node. The C is a super class of the c1 and c2. . . 105
5.4 An example of a tree structure in the SST. The subclass or is-A relationship is depicted in blue colour edges and black colour edges are represented has-A relationship. 106
5.5 An example of the sh-specific concept that having a filtered question. . . . 113
5.6 A hypothetical example of a pattern of N-ary relationship. 115
5.7 An abstract view of value-mg list. 116
5.8 An example of value-mg list in the OWL version of SST ontology of GRiST tool. 117
5.9 A graphical representation of a value-mg list having numerical data. . . . 118
5.10 A graphical representation of a value-mg list having integer type datum values. 122

6.1 Common XML trees of GRiST and ADVANCE tools. 131
6.2 The demonstration of the extensibility of the '*Assessment-Type*' class in the SST ontology of the GRiST. The constraints of the OWL Galatea model are represented in the gray background colour and the constraints of the SST ontology of GRiST are depicted in the yellow background colour. 134
6.3 The demonstration of the extensibility of the Assessment-Type class in the SST ontology of the ADVANCE. The constraints of the OWL Galatea model are represented in the gray background colour and the constraints of the SST ontology of ADVANCE are depicted in the yellow background colour. 136
6.4 An example of a GD-DatumNode (i.e. gen-gender) with n number of value-mg lists in the SST.owl of the GRiST tool. 143
6.5 An example of a hasPruneFor constraint in the risk-dep concept in the SST.owl of the GRiST. 144
6.6 An example of the generic question of the concept of the GRiST tool. . . 147
6.7 An example of the context based question of the concept of the GRiST tool. 148

7.1 An example of the nodes have the hasPruneFor constraints related to a specified assessment type (i.e. child-adolescent) in the SST ontology. The nodes are represented in pink colour. The edges and necessary conditions are represented in black colour. 156
7.2 An Illustration of the concepts having $hasDirSubNode$ constraints in the ST ontology. 157

7.3 An example of the nodes have relationships from sub node to super node via using $hasDirSupNode$ property in the SST ontology. 157
7.4 An example of a sub node (i.e. B), that have two super nodes (A and X) in the SST ontology. The edges between the sub and super nodes are represented in gray colour and the edges between sub and super classes are represented in black colour. 158
7.5 An example of a sub node (i.e. B), that have two super nodes (A and X) in the ST ontology. The constraints in the nodes of the SST ontology are represented in gray colour and the constraints in the nodes of the ST ontology are represented in black colour. The edges between the sub and super nodes are represented in gray colour and the edges between sub and super classes are represented in black colour. 159
7.6 An example of the sub nodes of G-ConceptNode (i.e. gen-hopeless), that are represented in pink colour. The edges sub and super nodes are represented in gray colour. The classes created in the RIT ontology are represented in dark blue colour. 162
7.7 An example of the sub nodes of gen-curnt-bhvr (i.e. GD-ConceptNode) having context based RIs in the GRiST. 165
7.8 An example of a GD-ConceptNode (i.e. gen-curnt-bhvr) having the 'hasLevel value 1' constraint. 168
7.9 The s-spec node has two RIs (i.e. 0.50 and 0.33) and an OWL reasoner inferred the knowledge-base as inconsistent. 172
7.10 An example of the sub nodes of G-ConceptNode (i.e. sn-aprance) having RIs. The super classes are represented in gray color and subclasses are depicted in light blue color. The inferred subclasses are depicted in yellow color. The subclass of the sn-hygiene is depicted in blue color in the RIT ontology. 173
7.11 An example of a logical tree in the SST.OWL. 174
7.12 An example of a logical tree in the ST.OWL. The ignored nodes and edges are depicted in gray color. 175
7.13 An example of flexibility of the constraints in the RIT ontology of the GRiST tool. 176

8.1 The generation of an an adaptive assessment questionnaire from the SST ontology. 183

9.1 Initial version of SST mind map file. 188
9.2 Modified version of SST mind map file. 188
9.3 Initial version of OWL file. 189
9.4 Modified version of OWL file. 190
9.5 The present SST.xml file has a path for representing sub and super node relationships. Note: the highlighted text is depicted in yellow background colour. 193
9.6 The SST.xml file generated through OWL-based specification facilitates the logical correct representation of the knowledge in the form of sub and super node relationships which does not need to be modified in its next state or structure. Note: the highlighted text is depicted in blue background colour. 193

9.7 The “hto-target” node have the “multiple-tick” attribute in the SST.xml file. Note: the highlighted text is depicted in yellow background colour. 194
9.8 The “hto-target” node and its sub nodes without using the “multiple-tick” attribute in the SST.xml file derived from the SST.owl. Note: the highlighted text is depicted in blue background colour. 195
9.9 A repeating or generic node (i.e. “gen-presentation”) has a path in an individual risk. Note: the highlighted text is depicted in yellow background colour. 196
9.10 A repeating or generic node (i.e. “gen-presentation”) has “level-code” attribute. Note: the highlighted text is depicted in blue background colour. 197
9.11 A repeating or generic node (i.e. “gen-presentation”) is inside an individual risk (i.e. “suic”) with its complete definition in the SST.xml generated from SST.owl. Note: the highlighted text is depicted in blue background colour. 197
9.12 The “gen-cog-think-mem” is a unsuitable node for the working-age user but still it is a part of the present ST.xml file. Note: the highlighted text is depicted in yellow background colour. 198
9.13 The “sh-change-mth” node is removed from the present ST.XML due to partially judgments of its core contents. Note: the highlighted text is depicted in yellow background colour. 199
9.14 The “sh-change-mth” node is added in the ST.XML which is derived from ST.owl. Note: the highlighted text is depicted in blue background colour. . 200
9.15 New RIT XML having RIs. 201
9.16 The “gen-presentation” node has “level-question=To what extent does the person’s behavioural presentation during assessment match that of a person who would give you maximum concern about suicide risk?” attribute and contents in the present RIT.xml. Note: the highlighted text is depicted in yellow background colour. 202

List of Tables

2.1 Description logic constructors and DL-syntax 36

3.1 Examples of mandatory attributes of domains of Galatean model of classification in XML specifications [2]. 60

Chapter 1

Introduction

Over the past few decades, the decision support systems [3, 4] have found tremendous significance in the knowledge domains. Traditionally, the knowledge embedded in the decision support system is mainly based on human expertise and recommendations particularly in uncertain situations. They use their common sense and experience to make decisions in ambivalent situations and present a great creativity through their responses. They are also able to explain the logic and reasoning behind a decision, and respond even if the data is not available. The issue is that human experts cannot be available all the time and so, it makes sense to put all knowledge into the machines.

The machines or computers are excellent in terms of speed, accuracy, consistent performance, minimizing the costs, reducing the number of errors and transferring knowledge into remote locations and we can call these characteristics as the machine expertise. The decision support system [5] is an interactive computer application that solves problems and makes decisions. The data-driven decision support system [6] is a class chosen from the wider taxonomy of decision support system that is used for the manipulation of internal and sometimes external data within the problem domains. Such types of decision support systems provide limited functionality and response specific answers within the context of particular purposes. However, the data belongs to a purely mathematical domain can be efficiently computed and calculated by the data-driven decision support systems. For example, the information system of a bakery shop can calculate their customers' bills and keep the record of the stocks and purchase information by

using computational calculations. Still, the manager of the bakery shop may also be interested in making decisions to identify delivery routes that guarantee on-time delivery. This example demonstrates that the involvement of human expertise is required besides the machine computations. Therefore, we cannot ignore the importance of human expertise in better understanding the relationships of data over the computed or calculated outputs. Human experts may use those machine calculated outputs as their inputs and provide final decisions on the basis of their innate reasoning power. For better understanding how the human brain manages the information and infers the required knowledge, the scientific study of human cognitive abilities is needed. Human thought and behaviour has emerged to formulate various psychological theories. Different theoretical or hypothetical theories are used for supporting different psychological models. However, these psychological model applications have quite complex knowledge structures.

Psychological [7] knowledge structures encapsulate varying approaches regarding the study of human mental processes and behaviours. The psychodynamic, behavioural and cognitive are different examples of psychological theories. However, the cognitive theories and their supporting psychological models more concretely present the scientific study of human cognition abilities. Human experts make classification of things or objects on the basis of their shared qualities or characteristics into the appropriate categories or classes. The Galatean model of classification (a psychological model based on classification theory) is employed to capture human expertise. This model uses a similar approach of being a cognitive model that captures human expertise for the classification of objects into their appropriate classes or categories. It also uses some uncertain variables for the classification process, which are needed some mathematical computations. It means the knowledge domains based on this model need computational and human expertise together and any stand-alone decision support system cannot support diverse knowledge engineering processes. Therefore, it will be ideal to fill the gap between the human and machine expertise by combining the data-driven and knowledge-driven decision support system techniques.

A decision support system cannot completely rely merely on the human expertise. It is difficult for the human experts to elicit results from large unprocessed data in a limited

time. They need a consensus for evoking their memory and expertise while compiling such multidimensional data. The human-centric data structures represent the cognitive abilities of a persons' psychology, which is primarily based on the mental and cognitive functions and behaviours of human beings. For example, the investor companies use stock market data to raise their money, which helps making a plan for the future investments. The investors use human expertise for analysing the stock market data, but the human experts need some pre-processed data, as a base to invoking their skills and enhancing level of competence. This example demonstrates that decision making is not possible without seamlessly merging the data driven algorithms and processes, and human expertise. The implementation of data-driven decision support system alone does not provide the required functionalities in the knowledge domains. In a similar way the human knowledge-driven decision support system individually cannot make accurate predictions. Therefore, it is a good idea to combine the data-driven techniques and knowledge-driven systems together in such a way that can aid in overcoming their limitations thereby integrating their mutual strengths.

Various decision support systems can become out-dated due to the non-availability of the human experts or their specification manuals are not properly managed with time such as: Quick Medical Reference (QMR)[1] [8]. However, such systems traditionally assume a description and use keywords, attributes and interfaces, therefore the decision making process depends on match making mechanism based on these. Since they can only support exact syntax matches thus, any modifications in the form of model enhancement cannot be adapted by these systems. Current applications of Galatean model of classification employs Extensible Markup Language (XML)[2] representation for knowledge management along with a supporting document. It has been observed that although the XML code is both human and machine readable but it is only suitable for a basic knowledge codification task. It is not possible to make dynamic discovery of the knowledge and to transform knowledge automatically by using such representations. Such issues may become more prominent in the domains that embed psychologically complex and complicated knowledge structures.

[1] `http://www.openclinical.org/aisp_qmr.html`
[2] `http://www.w3.org/XML/`

Further, with the popularity and evolution of the World Wide Web thousands of Web based decision support systems originating from diverse sources in various forms and complexities, are becoming prevalent. With the high volume of decision support systems currently available, a static and mismanaged internal knowledge structure can decrease the usability, efficiency and popularity of such systems. This raised the demands to formalise an accurate, consistent, flexible, extendable, transformable and adaptive knowledge structure underlying within the intelligent applications. Hence, a facility that can automatically manage a well-defined structure of knowledge and adapt to certain modifications that suit to the users' requirements are increasingly becoming a demand in the Web-based decision support system arena. More recently, the adoption of Semantic Web representation and inference techniques to overcome the limitations of traditional decision support systems, has gained considerable attention. Several emerging frameworks for Web-based decision support system (e.g. OntoQuest [9], MASON [10], O_2DSS [11]) follows a Semantic Web line of research whereby the concepts or instances with formally defined semantics can be dynamically reasoned and deduced.

By analysing the limitations present in the existing knowledge management, extension and transformation efforts, a semantic decision support approach is devised to facilitate the effective discovery of knowledge resources. This thesis presents the proposed semantic solution, which provides a pragmatic approach to predict correctly in the context of psychological models. The proposed approach also facilitates the automatic knowledge maintenance, reducing the replication of knowledge with variant uncertainties and support in knowledge engineering processes. A flexible and expressive approach can effectively describe various aspects of humanly represented knowledge including the evolution of knowledge as the expertise changes. Loosely structured psychological knowledge is maintained in a way so that it is not only interpretable by a machine but also easily understandable by a human being. Moreover it supports reasoning even if the context of the knowledge is changed and when a new service or user is added in the system. The proposed framework is intelligent enough in the sense that it can support to formalise knowledge within diverse domains based on a single psychological conceptualisation without having to explore their syntax details. The functionality and capabilities of different services are reasoned along with the knowledge associated to

target population against various application contexts. The proposed solution can effectively support the knowledge transformation process which change the form of the knowledge constructed in the source repository to a form that is usable in the target context of a specified user.

The proposed semantic framework solution is implemented in certain psychological models to formalise their well-defined formal structures. This implementation has been used systematically to evaluate the complex and complicated structures and the transformation process is demonstrated with the support of some empirical results.

1.1 Research problem and challenges

The rationale is that humans and machines have different characteristics, roles, and constraints while processing knowledge. Humans are more flexible in the way knowledge is represented and communicated as compared to the machines, which have problems with interpreting ambiguity. The challenge is to see whether human expertise can formally be specified to remove ambiguity without losing the semantics of the human representation. The research question is to explore the role of Semantic Web Technologies especially Web Ontology Language (OWL) and Semantic Web Rule Language (SWRL) representations in achieving this and whether their represented knowledge can improve the knowledge-engineering process for decision support systems based on a psychological model.

The research questions addressed in this thesis are: (i) how human expertise can be represented by a formal machine specification? (ii) can OWL represent a psychological model of classification for maintaining the flexibility in the knowledge-base? (iii) can OWL portray a psychological model within formal specifications? and (iv) does ontological represented knowledge improve the knowledge engineering process of the domains of the Galatean model of classification? More specifically, we have formalised hypotheses whether a psychological model of classification that is, by definition, based

on human cognition can be systematically and efficiently represented in OWL for machine processing.

The particular psychological model under consideration for investigating the research questions is the Galatean model of classification. This basically serves as the nucleus of a decision support system for its application within diverse knowledge domains. The translation of the Galatean model into its machine formalisation has been driven by the detailed specification documents that require reading and understanding by application developers delivering the functionality. The idea is to replace a detailed specification document by an OWL specification to determine whether (i) it can fully encapsulate the written specification and underlying cognitive model; (ii) improve the knowledge engineering processes embedded in the decision support system that are required for its continuing evolution during use; and (iii) facilitate the application of the model to new decision domains.

1.2 Aims and objectives

The core aim of the study is to investigate how to fill the gap between human and machine expertise. A classification (cognitive) model is selected for exploring how much human expertise and advice are required for the continuous evolution of knowledge within specific knowledge domains, and how a formal language representation can formalise loosely represented knowledge in a machine interpretable form. The more specific objectives of the research are:

- To construct a generic intelligent process that can provide support, without the advice of human expert in decision making and knowledge engineering processes.
- To replace a paper manual with a machine interpretable specification that can directly adapt within its application context as the expertise changes.
- To simplify the replication of knowledge and provide shared interfaces, programs and resources.
- To instantiate the uncertain variables of various knowledge chunks in a diverse application contexts.

- To manage consistent and logically correct knowledge structures of the decision domains based on the idea of classification theory.
- To build formal specifications that support the creation of multiple views of core knowledge structure for the target end-user and associated services.
- To present a flexible, generic and robust approach that supports adaptive generation of knowledge structures for new purpose and context.

1.3 Thesis contributions

This thesis presents a semantic framework to facilitate the effective description and discovery of resources in psychological models. The proposed approach addresses several limitations present in the existing knowledge structure models designed in XML specification. The core contributions of this research could be summarised as below:

- **OWL Galatea model:** This provides a general framework for Galatean model of classification that describes its core components and their functionalities, characteristics and relationships with other components. This framework encapsulates the written manual used for the applications of underlying cognitive model. It also provides machine-centric predictions without losing psychological semantics of the knowledge-base based on the idea of classification theory. The Galatean model of classification is based on classification theory and the classification process is triggered by the user given data. The user provided data is used to determine the associated uncertain data. For sake of this, certain machine-interpretable rules are designed in our proposed model by employing description logic. This intelligent framework not only support in making high level decisions on uncertain knowledge but also facilitates consistency checking of the knowledge-bases by using the inference engine. Initially, this model was designed by adding on only few core components, however as the model is tested and verified on some real time case studies, then further components are included in the under consideration model. This thesis has also demonstrated how the extensibility of the investigated model is possible. Two real world applications of Galatean model

of classification are used for this purpose. These case studies provide an assessment of the usefulness and usability of the model. It has also shown how this model coupled with DL reasoner can dramatically improve the results of dynamic knowledge management without having to explore the syntax details.

- **Formal specifications for transformation processes:** The applications of the Galatean model of classification have evolutionary or progressive data structures. Therefore, some formal transformation processes are designed for the derivation of knowledge-base into its different states. The consistency and stability of the knowledge-bases are considered on high priority while the transformation processes are in progress. The inference engine employed for sake of this not only supports the transformation processes but also confirms the internal conflicts of the knowledge-base. The distinction of our presented approach with the present approach is that the knowledge engineering process is only required in the first state or in the core repository. Next states are generated by adding some constraints and importing the previous state of knowledge-base. In other words, the required or requested views of the main repository are only created. Thus, the multiple views can be created for different users, services, contexts and circumstances. The benefits of our presented approach are that no conflicts between different structures or states of the knowledge-bases exist. If any changes are occurred in the core repository then these modifications lead to subsequent states. Wrighton and Buckingham [12] have also emphasised the importance of the machine interpretable structures. They have also designed a specific ontological structure for the GRiST (an application of Galatean model of classification). In this paper, they have described the Super Structure Tree (SST), Structure Tree (ST), Relative Influence Tree (RIT), Question Tree (QT) and Classification Assessment Tree (CAT) structures separately, these are independent of each other which indicates the similar issue that if an expert want to modify some contents in SST then it does not leads to ST, RIT, QT and CAT structures.

- **Verification and evaluation of the structures derived through the OWL Galatea model:** The verifications and evaluations of the OWL Galatean model are achieved by using some real time case studies. One real world issue was that to restructure the main repository according to the requirements of the end user. It

was impossible to modify the structure of the knowledge without losing its characteristics. The proposed approach not only facilitates the requirements of the multiple users in different contexts but also adapt the new knowledge structures without losing its associated attributes and data. The second challenge was to transform the knowledge into its final state by considering the suitability of the knowledge to a specified end user besides taking care of the knowledge associated to a service. It was a challenge for the current applications of the Galatean model of classification to translate a customised knowledge structure that not only keeps information associated to a specified user but also maintain a specified service information without changing core knowledge structure. The benefits of our proposed approach is that the knowledge engineer does not need to replicate a service related information in each customised structure and any changes regarding the services in core repository get reflected into customised structures. Other benefit is that the modifications are not required in the existing core structure and the rest of other structures.

- **Generation of adaptive assessment questionnaire:** The dynamic generation of an adaptive assessment questionnaire was achieved through this study. It has been discussed above that for the maintenance and management of present knowledge structures, a specification document is required. This document keeps some information (i.e. age, gender, ethnicity, marital status) which was not designed in the present knowledge trees. We have designed such information within the core structure of under considered decision domain while importing the proposed intelligent model. These types of information provided flexibility and robustness to the knowledge-base. Such information not only supports in derivation of an adaptive assessment questionnaire but also reduces the five numbers of states or knowledge trees into merely two states or knowledge trees.

1.4 Thesis plan

Chapter 2 introduces the background of decision support system, it expresses the traditional approaches for decision frameworks used for decision making. A brief introduction is presented on existing Semantic Web driven decision support system and also

discussed their limitations. This also discusses the core requirements for developing a decision support framework by using Semantic Web Technologies in the psychological domains.

Chapter 3 presents the overview of the Galatean model of classification and its inherent knowledge representations in XML format. This chapter also elaborates different types of XML nodes and their attributes in different knowledge trees with suitable examples and this chapter also concludes the importance of OWL in intelligent decision support systems.

Chapter 4 explores the overview of the semantic web layer architecture and with a brief description of OWL and SWRL and their conceptualisations.

In chapter 5, the OWL Galatea (a proposed) model and its core components with some suitable test-cases are discussed in detail.

Chapter 6 demonstrates the extension of the OWL Galatea model, and the generation of the SST ontology by using the case studies of the GRiST and ADVANCE tools.

Chapter 7 discusses on how ST, RIT, QT and CAT can be transformed for a single user within the domain of interest. This chapter ends by pointing out some advantages that a knowledge-base can gain by adopting the OWL-based approach.

Chapter 8 demonstrates the automatic adaptive questionnaire from the OWL based knowledge base.

Chapter 9 presents the comparisons of the present XML structures with the XML structure derived through OWL-based specification.

Chapter 10 contains the concluding remarks and discusses the possible future directions for this research.

Chapter 2

Intelligent Decision Support System

2.1 Introduction

The use of Intelligent Decision Support Systems (IDSSs) [13, 14] has been increasing rapidly over the past decades. The IDSS is interdisciplinary in nature; it works as a bridge among artificial intelligence, decision science and information systems. The decision making is a cognitive process through which appropriate decisions are predicted among several alternative possibilities. This process is embedded within all kinds of the decision support systems such as communication-driven, data-driven, document-driven, knowledge-driven and model-driven ones. Each typical decision support system has four components: (1) data management, (2) model management, (3) knowledge management and (4) user interface management.

1. The data component is used for managing and maintaining information that comes from organisational, external and personal sources.
2. The model or language component describes or declares the constructs used for creating relationships that support the decision making process.
3. Predicting and decision making is performed by the knowledge management component. It consists of rules and plays a role as an expert system.

4. The user interface management component allows to communicate with the decision support system.

The design and implementation of each component has an impact on the others; for instance, the formal descriptions, and the rules and structures of communal and personal information are decided through the selection of an appropriate formal language model. Therefore, it is quite important to select a suitable standard for the knowledge representation.

The deployment and uses of intelligent decision support systems are very common in various domains including business, engineering, logistics, military, medicine, and risk or disaster management. These domains need computational processing, as well as human expertise. For instance, the Preterm Birth Risk Assessment is a clinical decision support system [15] for determining the preterm birth risk of pregnant women. This decision support system uses numerous confounding variables for predicting the preterm labour from the gestation period and it also employs statistical and validation techniques besides certain rules for analysing large data sets. It reflects that this expert system needs both human and machine skills. DXplain [16, 17], is another clinical decision support system [18, 19], that is designed to suggest a list of diseases, which are associated with a set of clinical findings entered by a health profession practitioner. This tool assists clinicians by generating stratified diagnoses based on the user input of patient signs and symptoms, laboratory results, and other clinical findings. This intelligent system also rigorously requires human skills as compared to machine expertise.

The implementations and uses of different decision support systems has demonstrated that certain knowledge domains exploit human expertise and a good way of doing this is to build human knowledge and reasoning within the system. The DXplain, IDSSGL [20], disaster management in buildings [21] and other decision support systems, have used human expertise beside machine expertise for knowledge acquisition, analysis and predictions. Humans have innate mental skills that can manage information and deduce the information when it is timely required. The study of human cognitive abilities has emerged to various psychological theories. Divergent hypothetical theories have been employed for supporting different psychological models.

2.1.1 Decision making for the domains of the psychological model

Psychology [7] encircles a big domain and includes varying approaches to the study of human mental processes and behaviours. Cognitive theories are one from the class of psychological propositions. These explain how the human brain thinks and makes reasoning and they present the scientific study of human cognitive abilities. Cognitive abilities mean mental skills related to knowledge attention, memory, judgments, reasoning, problem solving, decision making and language processing. There are many expert (rule-based reasoning) systems that work in a similar manner. For example, MYCIN [22] is a rule-based system that recommends drugs to the patients by determining their indicated symptoms.

Human cognitive abilities become more mature by different experiences, training or exercises and one can quickly solve the problem if it is similar to a previous one. ACT-R [23] is a cognitive theory, which presents an integrated account of many aspects of human cognition and portrays how to make inference from already existing factual knowledge. This theory presents the similar idea for making predictions or decisions, as the human experts use some prior computed data, which they employ as the parameters in their reasoning process, and elicit the outcomes in the form of interpreting results. For example, a mechanic fixes a motorbike by recalling another motorbike repair experience that once exhibited similar symptoms and issues. Similar approaches can also be utilised in intelligent decision support systems that can investigate the size and volume of an abnormal lump in kidneys by processing ultrasound scan images, and another that can inspect the magnitude and size of a tumour and cyst in the breast through processing ultrasound scan images. To construct the formal structures for the systems based on human expertise, it is helpful to study how human brain or mind architecture manages knowledge and makes reasoning for deducing the required knowledge.

Soar [24, 25], is a cognitive architecture, that provides views of cognition and its symbolic implementation for general intelligence. It keeps three components: (1) state, (2) operator and (3) result, for reflecting the representation of problems solving and finally displays the results by using an operator on a state. An expert system works in a similar

way for extracting the results from the given knowledge-base.

Decision support systems are actually human mind-support systems that establish a symbiosis of human mind and computer by facilitating a high degree of human-computer interaction. ACT-R and Soar cognitive theories also support this conceptualisation of human brain skills. Cognitive orientation or mental models play an important role in a decision maker's understanding of the problem domain. For example, a speech recognition decision support system needs intensive human expertise for providing different patterns of a word, which help in the recognition and translation of spoken words. The machine also uses some computations for reducing the accents or dialects factors, which do not modify the semantics of the actual word, but make it harder in understanding and recognition. This example demonstrates that it is necessary to combine the human and machine expertise altogether.

The implementation of the data-driven decision support system alone does not provide the required functionalities in the knowledge domains and in a similar way the knowledge-driven decision support system individually cannot demonstrate its performance perfectly. It is essential that a decision support system can combine the expertise of both human and machine, and can support in machine interpretations and reasoning without losing the actual or psychological semantics of the knowledge. The Galatean model of classification belongs to cognitive psychological theory that requires both human and machine expertise. In the next section, a brief introduction of the applications of the Galatean model is to be presented.

2.1.1.1 Applications of the Galatean model of classification

The Galatean model of classification is used as the heart of the decision support system for its application to multiple knowledge domains [26]. The purpose of the exploration of different applications of the Galatean model of classification is to highlight the actual issues in its application domains and invoke a sense how to improve the knowledge engineering process. This discussion supports that the type of knowledge representation related can be beneficial to its knowledge domains. The chosen knowledge representation should not only reduce the replication of knowledge in various knowledge structures

but should also give support in reasoning and deducing the required knowledge. The Galatean model of classification was used for the first time in the horse racing domain [27], where there was only one user, who can be a winner or loser. The knowledge structure of this domain was quite small in size, consistently of two decision classes, one for the winner and other for the loser, and aimed at users who want to bet on horse races.

This model was later used in the psychodynamic psychotherapy domain. This decision support system was used to guide the clinician to which services of the psychodynamic psychotherapy are suitable for what client(s). The model of psychotherapy assessment was designed through generating a questionnaire based on 76 questions. 24 clients were assessed by that questionnaire and their responses were filled in by a consultant psychotherapist [28]. This decision support was not designed for a variety of populations and practitioners. This knowledge domain had a very small size of data and aimed at benefiting a very low number of users. So, the experts have used a less expressive language to model the decision making process for the said knowledge domain.

At present, this model is used in the health service and transportation domains. The Galatean Risk Safety Tool (GRiST) [29] belongs to the health service domain and Advanced Predictive-Analysis-Based Decision-Support Engine for Logistics (ADVANCE) is affiliated to the logistics domain. GRiST is used to determine the mental health risks using empirical evidences alongside clinical judgments. On the other hand, ADVANCE is employed to predict accurate and timely deliveries of goods and packages. The data for GRiST tool is captured from a number of mental health experts and it applies to a number of patients according to their age ranges and learning disabilities. GRiST and ADVANCE both have a big and complex knowledge structure with a variety of different versions and these assessment tools are used in different remote locations via their Web-based decision support systems. Therefore, applying a decision support system for multiple users with diverse contexts complicates the knowledge engineering process.

The knowledge-bases of the horse racing and psychodynamic psychotherapy domains

have smaller and less complicated data than that of the GRiST and ADVANCE tools. A less expressive knowledge representation language can easily manage the knowledge-bases, when a knowledge domain has only one decision class and a single group of the users. Thus, a more expressive knowledge representation language is not required in this case. A knowledge domain such as GRiST or ADVANCE with a complex knowledge structure, needs a more expressive knowledge representation language that makes it feasible to manage the psychological representations into consistent formal structures and support in deducing the required knowledge. Therefore, it is a fact that the increasing complexity of deploying decision support systems in the real world means the knowledge engineering process becomes more complex.

2.1.1.2 The limitations of current knowledge representation techniques

The current knowledge domains of Galatean model of classification use Extensible Markup Language (XML) for managing the human represented knowledge. The experts' represented concepts are stored in the form of XML nodes in the form of hierarchical structures. A chain of XML trees known as: Super Structure Tree (SST), Structure Tree (ST), Relative Influence Tree (RIT), Question tree (QT) and Client Assessment Tree (CAT) are defined within a domain for maintaining the concerned knowledge for multiple users and experts. These knowledge trees also keep and maintain some repeating nodes with a number of uncertainties. These complex series of XML trees in the current representation of knowledge are maintained and further developed by some bespoke programs, which are used by the experts to interact with the trees. These programs all have to ensure consistency with a paper manual or XML specification document. Thus, it is difficult to ensure the consistency between the programs and their outputs. For example, the ST is derived from the SST and if the programmer modifies any thing in the derived structure then it is difficult to map back changes in the core structure. Maintaining the consistency of the knowledge structures of these trees by using complex and long paper based specifications is very difficult. The consistent knowledge management of repeating (i.e. generic and generic distinct[1]) concepts with uncertain attributes in a series of hierarchical data structure for different end users is

[1]The generic and generic distinct concepts and datum nodes are used to distinguish nodes that are located in more than one place with exactly the same values (generic) or repeat with variations in their values (generic distinct).

quite a laborious task when only human expertise with a specification document are utilised. A generic method is required that works as a bridge between human and machine expertise and provides a diverse range of solutions for various knowledge domains. This is rationally a knowledge engineering problem. The knowledge engineering [30] processes are required for extracting the knowledge from human experts and constructing their formal structure into the system.

In the last few decades, Web-based decision support systems [31] have attracted so many artificial intelligence communities due to their all-time availability on the Internet. Such types of decision support systems are flexible and have no time frame issues. In the next section, a brief overview of Web-based decision support systems, the current work and emergent technologies are going to be elaborated.

2.2 Web-based decision support systems

The rapid advancement of Internet and Web technologies in recent years has increased the demands of Web-based decision support systems. A Web-based decision support system [32] is an interactive software-based system that is available online on the Internet. The Web-based decision support framework is deployed in a wide variety of decision domains such as health, inventory, manufacturing, resource management, career development and education. Ozan and Mustafa [33] have utilised JAVA Servlets, MySQL technologies besides the modelling principles of fuzzy Analytic Hierarchy Process (AHP) methodology for the development of a Web-based decision support system for multi-criteria inventory classification system. The limitation of this system is that it is well applicable to only pure production firms. This system needs more flexible structures that help in its applicability to many other samples inventories.

Vassilis et al. [34] have designed a three tier architecture for Web-based decision support system to provide the information about higher education studies of the users in Greece and also guide in choosing their vocational prospects. This Web-based decision support system used PHP, MySQL, JavaScript, Ajax and Geographic Information System (GIS) technologies and the Google Maps Application Programming Interface

(API) library that makes it easy for the users to visualise the geographical location of the higher education departments. This Web-based system meets the user's functional requirements, but does not cover the other context-based information relevant to career development like the labour market overview for the placement of graduates and recommendations for the selection of the right career.

Pl@netInfo [35] is another decision support system that was developed to provide advises and decision support for crop management. Farmers and advise-subscribers are dedicated users of this decision support system but this system still has certain limitations such as: it is not able to store users' personal information and access the production data from the client side. Herman and Clay [36] have designed a Web-based decision support system for the disposal and recycling of household waste. They used very simple and classical technologies such as: HTML and relational database management system. For the decision making purpose, they utilised some mathematical algorithms, and their designed system makes prediction on the users' provided data by computing mathematical equations. A Web-based decision support system was developed by G. Banias et al. for the management of the construction and demolition waste [37]. Java code, GWT Framework, JavaScript, HTML, CSS and Google Web Toolkit technologies are used for developing this decision support system. The decision making activities implemented in these systems are based on their syntactic information.

The decision support systems discussed in this section are available on the Internet. The technologies employed in these systems do not support the semantic description of the research problem. Therefore, it is difficult to make machine-centric decisions and predictions. The information is searched and retrieved through these systems by exploring their syntax details. The technologies used in these systems do not support retrieving information on the basis of resources or data semantics. These issues lead towards the demands of a Semantic Web driven intelligent framework. The next section presents a brief overview of the Semantic Web, description logic, Semantic Web standards and recommendations.

2.2.1 Semantic Web

The Semantic Web[2] is considered as the extension of existing conventional Web. It has emerged as a prominent role model in the Web by providing the facilities of machine readability, understanding and reasoning. These Web pages are normally used to store, display and retrieve the information. The problem is that the machine does not understand and comprehend the meaning of the contents of these documents. The Semantic Web technologies help the machine to understand the underlying meaning embedded within the Web pages. Semantic Web technologies aid in representing the human represented knowledge in a machine-centric form and also support automatic manipulation of the Web-based information.

For encoding and describing Web contents and resources, a suitable formal language is required to realise the Semantic Web. Such language should have a well-defined semantic format and must support in creating and developing relationships, constraints and restrictions among the Web objects. Semantic Web Layered architecture [38] represents several languages such as: XML [39], RDF [40], RDF Schema [41], DAML+OIL [42] and OWL [43]. DAML+OIL and OWL provide a natural way to describe the class and subclass relationships or taxonomies. These representations are in fact based on the description logic layer.

2.2.1.1 Description Logic

Description Logic (DL) [44] is a family of logic based knowledge representation languages which are suitable for representing and deducing about the knowledge resources. Biomedical informatics applications mostly use formal languages that belong to the description logic family. DL is originated from the frame-based systems and semantic networks which express knowledge in the form of classification of objects and inter-relationships among them. The concepts (unary predicates or classes), roles (binary relations) and individuals (constants) are the knowledge models in the description logic format. The relationships between these models are developed for reasoning and

[2] `http://www.w3.org/standards/semanticweb/`

deducing about the knowledge. A description logic knowledge-base consists of the terminological (TBox) and assertional (ABox) formalisms. TBox terminologies introduce concept definitions and axioms, while on the other hand ABox terminologies contain instances of the concepts and the specific instances are utilised in relationships via incorporating their roles. A DL system does not only stores TBox and ABox terminologies but also support reasoning services over the formally represented knowledge constraints and restrictions.

An interesting thing related to the description logic language is that when the formal expressivity of a language increases then its reasoning efficiency decreases. There are a number of description logic based language models which are distinguished by the constructors they provide. For instance, $\mathcal{AL}$ (attributive language) is a description language that uses minimal expressive constructors such as: conjunction, disjunction, negation, existential, intersection and value restrictions. The negation is only applied to the atomic concepts in $\mathcal{ALC}$. Table 3.1 portrays the constructors available in the description logic language. The $\mathcal{SHIONN}$ and $\mathcal{SHIQ}(\mathcal{D})$ are DL variants that provide expressiveness in terms of the constructors.

2.2.2 Semantic Web standards

The Semantic Web has inspired and motivated knowledge engineers to create innovative semantic based technologies and machine understandable applications and solutions. XML, RDF, OWL and SWRL are the standards of Semantic Web that help the knowledge engineers to create applications which are easy to understand for a human being and also machine interpretable. Some brief details of these standards are presented next.

2.2.2.1 Extensible Markup Language (XML)

Extensible Markup Language (XML) [45] is a simple knowledge representation format that is easily understandable for the humans and machines, however it does not facilitate the reasoning and inferring of knowledge. Jichang et al. [46] have developed

TABLE 2.1: Description logic constructors and DL-syntax

Name/Constructor	DL Syntax	Symbol
Concept	A	$\mathcal{AL}$
Top/Thing	$\top$	$\mathcal{AL}$
Bottom/Nothing	$\bot$	$\mathcal{AL}$
Intersection	C $\sqcap D$	$\mathcal{AL}$
Value restriction	$\forall R.C$	$\mathcal{AL}$
Union	C $\sqcup D$	$\mathcal{U}$
Negation	$\neg C$	$\mathcal{C}$
Existential quantification	$\exists R.C$	$\mathcal{E}$
Unified number restriction	$\geq nR$ $\leq nR$ = n R	$\mathcal{N}$
Qualified number restriction	$\geq nR.C$ $\leq nR.C$ = n R.C	$\mathcal{Q}$
Role-value-map (Subsumption) Role-value-map (Equivalence)	R $\subseteq S$ R = S	$\mathcal{H}$
Nominal	I	$\mathcal{O}$
Data Property Universal datatype restriction Existential datatype restriction	T $\forall T.d$ $\exists T.d$	$(\mathcal{D})$
Agreement Disagreement	u1 = u2 u1 $\neq u2$	$\mathcal{F}$

Web-based decision support system for portfolio selection by using XML representation. This XML-based system is constituted on a three-tier structure (1) GUI-tier, (2) Middle-tier (includes Web servers, application logic and decision switchers) and (3) Back-End application-tier. They documented XML technology as a suitable language for the representation of heterogeneous data in the decision processing system, but data security is not guaranteed due to its inherent openness. They also highlighted that the natural representation of the real world objects through its tree-like hierarchical data structure is easily possible, but this characteristic also creates issues in communicating with the existing databases that are relational or hierarchical.

Present knowledge structures related to the decision domains of the Galatean model are developed by using XML representation. The limitation of the XML is that there is no intended meaning[3] associated with the nesting of tags.

[3]what is meant by a word, text, concept, or action used inside the tag(s)?

For instance, the “electronic-pad”, “note-pro” and “ipad-pro” are the nodes in a knowledge-base. The information given in this example can be interpreted either where the “electronic-pad” is a super class with an “is-a” relationship with the “note-pro” and “ipad-pro” or where the relationship between them is a “has-a” one. A specification document is required for elaborating or explaining the correct structure of this knowledge.

```
<electronic-pad>
    <note-pro>
        <price>425.99</price>
        <released-date>01/10/2015</released-date>
        <released-by>Samsung</released-by>
    </note-pro>
    <ipad-pro>
        <price>550.49</price>
        <released-date>06/06/2015</released-date>
        <released-by>Apple</released-by>
   </ipad-pro>
</electronic-pads>
```

Or if certain knowledge chunks are associated to some domain users then the information can only be extracted from the knowledge-base by manipulating its syntactic details. For instance, following knowledge-base is designed for multiple users and certain knowledge chunks are particularly associated to some specific users, e.g. older or child. So, it is difficult to infer the required knowledge associated to the “child” user without mining the text given in the “has-app” attribute of the example. It is not possible to extract accurate information with its complete or correct semantics through the XML representation.

```
<electronic-pad>
    <note-pro
has-apps="(older, child) keeps specific applications for all users">
        <price>425.99</price>
        <released-date>01/10/2015</released-date>
        <released-by>Samsung</released-by>
```

```
    </note-pro>
    <ipad-pro
has-apps="(child) keeps specific applications for children only">
        <price>550.49</price>
        <released-date>06/06/2015</released-date>
        <released-by>Apple</released-by>
   </ipad-pro>
</electronic-pads>
```

This issue increased further if there is a massive amount of knowledge that has more than one expert or service or context.

For the manipulation and maintenance of different XML knowledge trees within the applications of the Galatean model of classification, a specification document is required. It is challenging for the developers to read and then confirm the consistency of the knowledge-base. It is also difficult to match or compare the modifications, and to check whether the driven outputs are correct or not. Any new demand or requirement argues must be updated in the paper manual and also in the implemented series of logical hierarchies. A machine-centric specification can be useful for the maintenance and manipulation of XML trees. The benefit of a machine-oriented specification is that any modification in it can directly be manifested to the implemented knowledge-base.

2.2.2.2 Resource Description Framework (RDF)

Resource Description Framework (RDF) [47] is a basic data model. Object, attribute and value triplets are the basic building blocks for an RDF statement. It helps creating Web objects and relationships among them. This knowledge representation is domain independent, thus no assumptions about a particular domain of interest are made. In contrast with XML, this representation provides intended meaning associated with a subclass of relationship.

RDF is a World Wide Web Consortium (W3C) standard that is based on the XML technology. It describes resources about anything but its main use is to provide meta-data

descriptions in the Semantic Web architecture [48]. RDFS (RDF Schema) Schema makes the information machine readable and understandable. RDFS extends basic primitives for modelling RDF classes and properties hierarchically with clearly defined semantics. Friend of a Friend (FOAF)[4] is a well-known example of RDF ontology. FOAF is a machine-readable ontology that was developed using RDF representation. It is also seen as the first social Semantic Web application that integrates social network with the Semantic Web Technology.

RDF facilitates in creating machine-readable concepts, roles, constraints and restrictions, which are also understandable to the human beings. Although this technology is able to codify the human represented knowledge, the challenge is to see that if the data structure embeds within the human–centric knowledge then besides requiring certain mathematical computations to make accurate predictions, it is not possible to construct the descriptive mathematical algorithms or rules over the RDF represented knowledge. As it is for the case of the Galatean model applications, their data structure is basically double-fold. One segment consists on the human centric expertise and advise, and other fold requires computations over the mathematical data for classification and prediction of knowledge. The first segment or fold of the knowledge-base can be easily developed in the RDF representation by creating subsumption and has-a relationships but the said standard does not provide support to construct mathematical algorithms over its represented knowledge. It is because, RDF representation has limited expressivity [43]. Hence, it is not possible to design the knowledge-base through the RDF language and then create the semantic rules by using the SWRL-based representation. The RDF represented data does not directly support the implementation of semantic rules in the knowledge-base. A knowledge representation is required that can facilitate the development of formal description of human represented knowledge and further support to construct the semantic rules (including mathematical notations) over the represented knowledge. DAML-OIL (DARPA Agent Markup Language – Ontology Inference Layer)[5] and Web Ontology Language (OWL) are Web ontology language based on description logic [44] and these build on RDF. The Semantic Web Layer architecture[6] portrays a layer of OWL between the RDF and SWRL representations.

[4] `http://xmlns.com/foaf/spec/`
[5] `http://www.w3.org/TR/daml+oil-reference`
[6] `http://www.w3.org/2006/Talks/0718-aaai-tbl/Overview.html`

2.2.2.3 Web Ontology Language (OWL)

Web Ontology Language (OWL) [49] is a useful specification language for enabling machines to process knowledge representations. It provides explicit meaning, which makes easier for the machines to automatically process and integrate information available on the Web. It builds on the low level semantics of RDF to define a more precise vocabulary support structure in which the relationships between the classes and their properties can be defined. It is more expressive than RDF representation. OWL provides the machine-readable contents (i.e. Web pages), where a machine can process knowledge itself similar to the human deductive reasoning and inference. The advantage of ontological represented knowledge is that the description of the mathematical notations and equations can be easily mapped on such knowledge.

The knowledge belonging to the applications of the Galatean model of classification has many transitions and folds, and a number of different users use it within a domain. The classification process of the said model is triggered by employing the user given data which eventually helps in the calculation of uncertain data by using some mathematical algorithm (see details in Chapter 3). Semantic rule language accommodates mathematical notations and functions besides encoding conditional knowledge, which is difficult to express through the Web ontology language.

2.2.2.4 Semantic Web Rule Language (SWRL)

Semantic Web Rule Language (SWRL)[7] is another knowledge representation language that is used to define the custom rules. It is compatible with OWL therefore semantic rules can easily be constructed or designed by employing the OWL classes and properties. OWL expresses the problem classification by using description logic and first order logic. The choice of SWRL language suits well where it becomes difficult to demonstrate the logical representation in the form of If and Else statements for an ontology.

[7] `http://www.w3.org/Submission/SWRL/`

An SWRL rule [50, 51] contains two parts, (1) antecedent part and (2) consequent part. The antecedent part of a rule is called the body and the consequent part of a rule is called the head. Both body and head consist of conjunctions of some atoms. If all the atoms in the body are true, then the head must also be true. OWL classes and properties are used as functions or methods. The parameters of the method can be constants or variables or individuals. A question mark *?* is pre-affixed to the variable name [50], that makes it distinguishable from a constant of the SWRL rules. For example, If a player has played some test and T-twenty matches and if the total number of played matches is greater than or equal to 5 matches, then one will be selected to play in next coming world cup match. This information is designed in the following SWRL rule.

```
Rule:
    Player(?p),  hasWon(?p, ?numberofMatchesPlayed),
    Test(?p, ?testplay), T-twenty(?p, ?twntyplay),
    add(?numberofMatchesPlayed, ?testplay, ?twntyplay),
    greaterThan(?numberofMatchesPlayed, 5)
     -> hasSelectedToPlay-Worldcup(?p, true)
```

Note: the `hasSelectedToPlay-Worldcup(?p, true)` axiom is the head part of the above rule and the rest of the axioms are the segments of the body.

IDSS are employed in a wide variety of applications using a number of techniques, technologies and programming languages. Semantic Web technologies are one of the most promising state-of-art technologies used for evolution of decision support systems in the recent years.

2.3 Semantic Web decision support systems

Semantic Web Technologies aid in the creation of intelligent and machine interpretable Web pages and online applications. These standards can also perform a vital role in developing an intelligent decision support system which facilitates decision makers by providing information from documents and user provided data. In order to develop

an intelligent decision support system using the Semantic Web representations, it is necessary to find out what are the core requirements for it.

2.3.1 Requirements for Semantic Web decision support systems

For the experts of the psychology domains, it is quite challenging to estimate the actual requirements of an intelligent decision support system, due to the complexities existing in the non-formal or natural structures of information. Therefore, it becomes difficult to discover cutting-edge technologies suitable to adapt to the problem of interest. Lars and David [52] have combined the technical, psychological and philosophical research ideas and as a result of their survey, they have outlined a set of requirements for the development of decision support systems employing the Semantic Web technologies.

- **Ontology evolution by knowledge exchange:** Human-centric tasks and advice are essential for the evolution of the ontological represented knowledge. There are many disciplines for the ontology change management. Asad et al. [53] have highlighted, ontology engineering, evolution, merging, integration, importing and maintenance areas regarding the evolution of the ontological represented knowledge. Thus, it is required that the decision support system should itself restructure or modify the ontological represented knowledge dynamically and automatically.

- **Information normalisation:** This refers to the ability of Semantic Web technologies that should allow for a normalised form of information sequences and its relationships. Lule and Edmond [54] have developed an approach for building normalised relations in the Semantic Web systems. Humans usually learn information that directs from the physical structure movements besides other communication skills. Therefore, the normalisation of human-centric information should be codified in any communication platform by using a reference model.

- **Thought-accompanying storage and retrieval:** The intelligent system should exploit the whole functionality of the decision processes rather than putting unnecessary effort on designing the number of the concepts that are not obvious. The main purpose of the knowledge-base should to serve as the formal specification of the actions that support to express, store and retrieval of the knowledge.

- **Natural language representations:** The framework should allow the use of natural language rich tokens which not only ease the reading and understanding by human expert, but also support making machine-centric predictions.
- **Free information serialisation:** Decision support system should support dynamic information retrieval and browsing besides the information serialisation for the readers.
- **Context-sensitivity:** Intelligent framework should provide support to create the customised views on context dependent information.
- **Semantic adaption of ontology:** It is difficult to adapt the contents of the manual specifications for unknown or unfamiliar situations or contexts. So, the challenge is for the intelligent system is that it should have the ability to utilise existing knowledge for a new purpose or situation.
- **Personal information store:** It should not only facilitate managing collective information of contexts, persons and circumstances, but also support formal codification of personal subjective information.
- **Subjective ontology:** Intelligent model should facilitate the correct formalisation of personal, emotional and mental states of knowledge management processes in a functional subjective knowledge-base or ontology.
- **Intention explication:** The decision support system should facilitate the codification of widely explicated intentional characteristics of personal factual knowledge.

All these requirements are not essential to have in single decision support system. These can be implemented or added into the system according to its specific need. Section 2.4 presents the requirements utilised for developing an intelligent decision support system for the decision domains of the Galatean model of classification.

2.3.2 Existing Semantic Web decision support systems

Decision support is a classical research field that was affiliated to the Semantic Web technologies a decade ago for inferring and retrieving information on the basis of meaning and interpretation. A wide variety of Semantic Web based decision support frameworks are developed in multiple domains. For example, Valery DONFACK et al. [55]

have designed an ontology-driven decision support system for medical diagnosis of malignant disorders. They utilise OWL and SWRL technologies for the concept reorganisation, concept addition, medical reports and creating the prototypical case methods. These methods are used for determining the distinction between multiple Myeloma stages in guiding therapy, evaluating and validating the system by using real time clinical reports and storing the knowledge into a MYSQL database. They try to determine the likelihood of a given diagnosis through the number of observed signs that help predicting the disease. The system they have developed is particularly designed for a single type of patient and only dedicated to providing clinical judgments on a malignant disorder. Thus, such types of knowledge-bases are quite easy to model and their knowledge management effort is also not a troublesome issue.

Kamran Farooq et al. [56] have developed a decision support framework by using ontological techniques and Bayesian Network for the cardiovascular domain. They employ Bayesian Network for uncertainty modelling using cardiovascular decision support system based on ontological knowledge. Their developed system starts working by loading the profile ontology, and the Bayesian Network inference engine calculates the probability of each risk factor associated to the patient's clinical history through the variable elimination algorithm. Another decision support framework (i.e. SybillaTUC) was developed using Semantic Web Technologies for the prediction of evolution of Bipolar Depression by [57]. For designing SybillaTUC, they employed different technologies such as: OWL for developing knowledge-base, Temporal representation for designing ternary relationships and an algorithm for constant monitoring for patients, and for deciding best medical treatment based on the patient's condition.

Kyoung [58] has developed a decision support system to apply to the physical structure of engineered products. They demonstrated their idea by illustrating a fixture for holding parts inside a machining center that has four clamping arms. They said that the two different methods for connecting or joining arms to other components are too ambiguous to be expressed through geometrical form. The same issues can easily be addressed by using description logic approaches where the machine is also able to understand and interpret the code. In the manufacturing process, manufacturing workflow is a crucial part. A very simple ontology with certain basic semantic rules for giving an idea

on how these representations can help in the automatic generation of manufacturing workflow, was presented in [59]. They combined the semantic rules and data mining techniques for the on demand manufacturing workflow reasoning framework.

Hendro et al. [60] have demonstrated some intelligent methods that are used to construct and process the ontology. They also used SWRL rules over domain specific ontology in discrete manufacturing process. They generated SWRL rules by employing some machine learning algorithms for the intelligent energy management in the discrete manufacturing domains. Their algorithms provide inputs and instruction for the generation of the SWRL rules.

The decision support systems discussed in this section have many issues which are need to be address. For example, [57] have designed a static ontology which is dedicated to only a single type of user and make predictions about a specific type of disorder.

2.4 Conclusions

A number of recent research efforts on the decision support systems were discussed in the previous section that possess certain weaknesses in certain areas, such as maintenance of uncertain data by using semantic rules, knowledge evolution, information normalisation, natural language representative, context sensitivity and adaptation of ontology.

2.4.1 Ontology / knowledge-base evolution

One of the main objectives of knowledge engineer are to support the evolution of the knowledge. The evolution of the knowledge-base is not possible in the system presented by [57] because they developed a static ontology and maintain the uncertain data through temporal techniques. The expert systems lapse due to the non-availability

of experts and specification documents become out of date. However, in these situations a machine-interpretable specification can play a significance role. So, if the expertise change, then it directly has an impact on the implementation application. The said specifications can be in the form of concepts, roles, constraints and restrictions. For example, an ontology (i.e. O1) have “Animal” and “Cat” classes. An animal can have two, four or eight legs and a cat can have merely kitten offspring which has four legs. This information can be modelled in the O1 ontology in the following way.

Animal $\equiv$ ($\exists$ hasLegs.Two) $\sqcup$($\exists$ hasLegs.Four) $\sqcup$($\exists$ hasLegs.Eight)
Cat $\equiv$ ($\exists$ hasLegs.Four)

If we want to engineer further knowledge about Kitten, Sparrow and Spider concepts in the O1 ontology then we can model following knowledge for auto integration with existing knowledge.

Kitten $\subseteq$ ($\exists$ hasLegs.Four)
Sparrow $\subseteq$ ($\exists$ hasLegs.Two)
Spider $\subseteq$ ($\exists$ hasLegs.Eight)

The constraints mentioned above not only help in auto classification of the knowledge but also help to evolve the existing knowledge. The new information can be modelled in another ontology which can easily be integrated with the O1 ontology.

2.4.2 Information normalisation

The information normalisation has a significance importance for managing and maintaining a right or correct knowledge-base. [54] have demonstrated the normalisation of relations for supporting schema refinement. Their conclusions of research were that the relations and ontology normalisation techniques support the machine understanding of normalisation algorithm and aid in creating better relations and constraints for making correct decisions. If the relationships are developed by using some other techniques, then the knowledge base can become static. For instance, the ternary relationships in the SybillaTUC are developed by employing the Temporal representation techniques

[57], which contributes in static representation of ontology and any modifications are not adjustable. An intelligent approach can be utilised that can aid in the development of ternary relationships by using description logic techniques. For example, a bus (i.e. B) has route sequence (i.e. RS1) and each route has a bus stop (i.e. Bstop). This information can be modelled in the description logic form as:

B $\subseteq \exists hasRouteSequence.RS1$
$RS1 \subseteq \exists hasStop.Bstop$

2.4.3 Context sensitivity

Certain approaches discussed in the previous sections dealt with the problem related to how one single category of users or only single contextual information is stored, reasoned and retrieved. For instance, an intelligent approach can be developed to maintain the context sensitive information. Therefore, this strategy can enhance the chance to create views of the knowledge base by considering its contexts. Such customised views can be modified and generated according to the users' requirements without modifying the main repository of the knowledge-base. For example, a decision support system was designed for the medical diagnosis of malignant disorders by [55]. The mentioned decision framework can only make clinical judgments on the presence and absence of a disease. A system is required that supports knowledge management in a wide variety of decision domains for a number of different users within a domain without exploring the domain specific terminologies.

2.4.4 Adaptation of ontology

The ability to adapt the knowledge according to a new situation or for new use attracts many knowledge engineers. Today, the research communities are not only looking for knowledge management, storage, retrieval and reasoning, but they are also concerned on how to adapt the existing knowledge base for new context(s) and situations(). Human beings have innate mental power to utilise learned skills or knowledge into new situations. For example, psychotherapy is used to treat patients with mental health disorder and emotional problems. So, if a psychotherapy service is available for an

emotionally disturb patient in a general hospital, then the same service can be suggested to a person who has career difficulties although the disease conditions of both patients are different to each other; but still the doctor or adviser suggests the same service to them. It is because both diseases have links to mental disorder but the doctor still needs to analyse the condition and severity of the diseases. Therefore, an intelligent system is required that can itself estimate the issues and try to predict a suitable solution.

Chapter 3

The Galatean Model of Classification

3.1 Introduction

The Galatean model is a psychological model that is used to capture human expertise. The knowledge domains of this model capture their data directly from human experts. The Galatean model decision support system makes the decisions and predictions through its classification process. The classification of things or objects into their natural categories is a generic form of decision making. This process helps and advises the people about which categories to choose. For instance, the Galatean model was previously applied in horse racing and psychodynamic psychotherapy domains. A winner or loser were the categories of the horse racing domain and whether a psychotherapy is suitable or not suitable were the categories of the psychodynamic psychotherapy domain. In a similar way the Galatean model can be applied for diagnosis of diseases, where the domain could have cold, flu, asthma, stroke and meningitis categories. The Galatean decision support system is used to provide advice on which class to choose.

Each decision class is represented by a single classification model and it is represented by an XML structure termed the Classification Assessment Tree (CAT). CAT alone is sufficient to manage complete knowledge of the knowledge domains having one category or class for making decision for exactly one population. For instance, when the

Galatean model is implemented in GRiST for one risk (say suicide) then a single CAT XML for suicide risk is adequate for managing high and low suicide risk categories. The decision making or classification process is quite simple and easy for such data. The user's provided data trigger the classification process and the first uncertain variable (i.e. membership grade) (see details in Section 3.2) of the model is initialised in the leaf nodes of the CAT knowledge hierarchy. A category is automatically predicted that has a high degree of membership grade.

When the model is expanded to many risks many of the concepts are duplicated across all of them. For example, if the categories are suicide, self harm, harm to others and self neglect, then the feeling/emotions, social contexts and general demographics concepts repeat in the above said categories. These things make maintaining consistency of the model between the risks for more than one user more difficult. Different types of knowledge must be maintained across different risks, including concept structures and the relative importance of each tree branch for assessing risk. The management of the structure and uncertainty values requires separate trees that are then eventually combined into the single CAT for each class. These things complicate the knowledge engineering process and increase difficulties in decision making.

Finally, the number of trees and their deployment exploded with different types of patients (older, children, etc), assessment circumstances (community, secondary care, primary care) and end users (patients, clinicians, carers). Managing all of these knowledge deployments was extremely difficult and created similar issues to the explosion of the MYCIN rule base. The MYCIN expert system was used for the diagnosis of various diseases like, bacteremia, meningitis and blood clotting diseases [61] by various clinicians on various types of patients. Hence, the need for a machine representation of the XML that helped maintain the validity of the knowledge and the ability to evolve them flexibly.

It is challenging, for a decision support framework to make machine-interpretable decisions that has knowledge structure based on XML representation. XML is a simple

language, and suitable for designing basic applications. Therefore, it is difficult to manage and maintain consistent knowledge into diverse hierarchical structures or trees for multiple users within different context by using XML standard. These are completely knowledge engineering issues and XML cannot provide the required expertise. The heavy demands on extendibility, transformations, correct and precise maintenance of the knowledge-base of different domains of the Galatean model of classification increases the need for strong machine interpretable structures which provide the required expertise.

In this chapter, the details of the Galatean model and its classification or decision making process will be elaborated. This chapter will also render the detail review of literature on different XML knowledge trees, variant types of nodes and the attributes used for maintaining knowledge structure of the XML nodes. This chapter will end on discussion why OWL is a reasonable choice for such complex knowledge structures and how the expertise can be evolved by using the ontological structures.

3.2 The Galatean Model

The Galatean model is a human psychological model of classification. The formal underpinning of the Galatean model of classification is to classify the real-world ideas or concepts into their appropriate categories and also support for different decision support systems for making correct decisions. Most of the time, people recognise, differentiate and predict the things, concepts and ideas on the basis of their categorisation and classification, which lies at the heart of decision support system. The Galatean model of classification provides a unifying framework within which to conceptualise decision making processes as classification behavior. The Galatean model of classification is necessary for facilitating in the classification and decision making of different concepts.

FIGURE 3.1: The "suicide" risk is an individual concept in GRiST. The intention to commit suicide is the sub concept of the suicide-risk structure [1]. Concepts are portrayed as ovals and datum components are rectangles.

3.2.1 The hierarchical Galatean model and its classification process

The Galatean model offers a novel theory and its classification theory is a process through which the concepts are classified in their related categories. Classification divides the set of entities in hierarchical form. An abstract root node is a core component of a domain knowledge hierarchal structure as that root node has some individual concepts. The individual concepts break down into some essential concepts through a successive top-down iteration process. The leaf or datum nodes are the final concept of the hierarchy (see Figure 3.1).

The Galatean model is totally a theoretical model. It uses the abstract theory to represent classes and it concerns on exceptional and hypothetical perfect one member (called a "galatea" after the Pygmalion's statue of his perfect women) [62]. The objects of the Galatean model of classification show uncertainty in terms of fuzzy sets or set memberships. The object is classified or categorised as a potential member and that member in any category is given by the degree of membership. This amount is called Membership Grades (MGs). The list of value-mg [62] is stored in every datum component of the hierarchy. A value-mg list is a combination of a set of pairs and each pair has a datum value and an associated MG. The $((5\ 0)(10\ 0.5)(15\ 1))$ is an example of a value-mg list. The 5, 10 and 15 define the MG distribution that associates MGs with

object values (0, 0.5 and 1) for that node in the value-mg list. When a potential user is assessed under the Galatean model of classification then his given or selected data is compared with the datum values of the value-mg list and the MG is generated by comparing the object value with the graph of MGs and values. Data input to the tree can be of any type (e.g. string, integer, double, boolean etc) but they are converted into an MG, from zero to one. Zero represents no support for the root decision class and one represents maximum support without taking account of any other variables. The MGs convert real-world user data to membership grades that can be uniformly processed by the galateas as shown in Figure 3.2 by the MG row of the datum nodes: it defines a distribution of MGs matching the range of potential input data values. Values above or below the range take the MG associated with the maximum or minimum value respectively; values within the range are found by linear interpolation.

FIGURE 3.2: Hypothetical example of how membership grades (MGs). RIs are relative influences, which represent the weights of data and concepts [2].

Sometimes, such as for 'days since the last attempt', the user's given data is passed through a function, *f(data)*, before matching and calculating the value-mg distribution: two dates, in this case, which the function uses to produce the number of days between them, 20. The 0.4 MG is calculated from the data (i.e. 20) given by the user. Every node of the Galatean model has a weighting or Relative Influence (RI). Each MG is then multiplied by the RI associated with the datum, as shown in the RIs row, to give the MG contribution to the parent concept. The parent concept MG is the sum of its children's contributions, which is how 0.52 is assigned to the concept node in Figure 3.2. The MGs percolate in this manner through to the root node to produce the overall class

membership and thus the specified domain based evaluation. Equation 3.1 formalizes the process

$$MG_C = \sum_{i=1}^{n} MG_i RI_{pi} \tag{3.1}$$

where C is a concept, MG is the membership grade generated at each datum node, i, of the concept, and RI_{pi} is the product of all the RIs along the path, p, from the datum node to the concept. Eventually the membership grades are linked with all parts of the Galatea hierarchy or the logical tree structure. A potential member is selected as a perfect galatea that has high degree of membership.

The data were collected and obtained by a team of experts and the best decisions were obtained by coinciding their work with the knowledge domains of the Galatean model of classification. Some open-ended interviews [63, 64] were applied to particular samples of the population (i.e. experts) of these domains [1]. The responses of the domain experts were analysed and concepts and subsequent concepts extracted from these responses, and later modelled in the mind maps [2] in the form of structure tree. Mind maps [65, 66] are quite good to represent the knowledge into graphical form and these are a natural psychological representation of knowledge and machine representations that are easy to convert into hierarchal XML structures. Therefore, the knowledge of these domains at present is managed in different XML knowledge trees. The XML structures become more and more complicated with the variety of classification circumstances, types of objects being classified (i.e. different patient types), and types of end users of the classification system. All these variations had to be documented in a paper specification that got more and more complex, which is why a machine specification became important. For understanding the complexities of the knowledge, it is important to explore variant XML knowledge trees used in knowledge domains of the Galatean model of classification. In the next section, the details of the XML knowledge trees will be discussed.

3.3 The knowledge structure of the domains of the Galatean model of classification

The current knowledge structure of the domains of the Galatean model of classification consists of some XML knowledge trees. Variant knowledge trees are used to maintain the hierarchical structure of the domains' knowledge, uncertainty attributes and decision making process. Different types of nodes with their numerous characteristics and features are managed in these knowledge trees. In this section, different types of XML knowledge trees, nodes and the attributes of the nodes will be discussed.

3.3.1 XML knowledge tree

As the demands on a domain of the Galatean model of classification functionality grew with its diversity of deployment, a number of variant XML trees were used in the knowledge engineering process. Some were based around the structure of the knowledge, others for representing uncertainty, and some were required for driving the decision support software tools. The core or source tree was the Super Structure Tree (SST) that defined the generic structure underlying all variants. Each domain of the Galatean model of classification has one root tree and some derived trees. The SST (a root) tree keeps the complete knowledge of a domain for all of its populations. The SST was then transformed by server-side tools into the specific trees required for particular populations. First, the Structure Tree (ST) for the particular population was extracted from the SST. The Galatean uncertainty values were placed in an expanded Relative Influence Tree (RIT), and the fully-expand Client Assessment Tree (CAT) and a Question Tree (QT) were generated to drive the decision support tools. More details of the process can be found elsewhere [2, 67] but brief details of these trees are given next.

3.3.1.1 Super Structure Tree (SST)

The Super Structure Tree (SST) is a fundamental or root tree. It is a universal tree for incorporating the structural knowledge for all populations of a domain. It contains all structural information for the nodes, populations and the questions attached to them with associated values.

3.3.1.2 Structure Tree (ST)

The Structure Tree (ST) is generated from the SST for a specified population. The structural information of the nodes, list(s) of paired datum values and MGs in the ST are similar to the SST. Every branch of the SST tree is not required in the ST for a potential population. The nodes and their subsequent nodes are excluded from the ST those have the '*prune-for*' attribute (see details in Section 3.3.3.7) for a specified population during the transformation. The removal of unwanted nodes and the transformation of this logical tree is achieved through some bespoke programs and the consultation of a specification document.

3.3.1.3 Relative Influence Tree (RIT)

The Relative Influence Tree (RIT) is generated from the ST tree. Each node of the RIT tree must have an RI value. The domains of the Galatean model of classification have two different categories of nodes (1) generic nodes and (2) non-generic nodes. The generic nodes are actually repeating concepts and datum nodes and non-generic nodes are opposite to them. In the RIT, the generic nodes are initialised with their complete definition in the contexts or locations where the paths or pointers were given in the SST.xml and ST.xml.

3.3.1.4 Client Assessment Tree (CAT)

The Client Assessment Tree (CAT) is generated from the RIT, because it needs the RIs for each concept and datum node [68]. It represents the complete classification process of the cues. The CAT is specifically derived according to the experience level of the population. It has all concepts fully expanded in all locations. It has no separate generic concepts or locations (detail see in Chapter 6) in its logical tree structure as the SST, ST and RIT. The associated questions are not part of this tree.

3.3.1.5 Question Tree (QT)

The QT is also generated from the RIT and has all information required for displaying questions related to a specified population. This tree has a flat tree structure, all the nodes are the direct sub nodes of the root node.

3.3.2 The types of nodes in XML trees

There are two types of nodes in XML trees, generic and non-generic.

3.3.2.1 The generic nodes

The generic nodes are repeating concepts and datum nodes. The repeating concepts and datum nodes are only fully defined in one place, under the generic concept or context, where their complete definitions are contained. These nodes have pointers to the locations of their complete definitions under variant contexts. The contexts are the individual concepts inside a root node. For example, the '*mental-health-risk*' is a root node and the '*suic*', '*sn*', '*hto*', '*generic*' (or generic section or context), etc are the contextual or individual concepts of GRiST (see Figure 3.3).

The generic section usually keeps repeating nodes which hold the path information in the generic, generic-type(g), generic-type(gd) and generic-datum attributes. The detail explanation of repeating nodes that have such attributes is given next.

1. **The generic-type '*g*'**

   The generic-type ('*g*') nodes are repeating nodes. The label of these nodes is '*generic*' or '*g*'. In such case, these are the concepts, then their existence as a super node influences on the data structure of their sub nodes. Their sub nodes must keep fixed or constant '*RIs*' in different contexts. For example, '*gen-hopelessness*' is a '*g*' concept and it has two subcomponents, '*plans for the future*' and '*life not worth living*' in the knowledge hierarchy of '*mental health risk*' domain. The data structure (i.e. '*RIs*') of its sub concepts and datum nodes does not change in different contexts ('*suicide*', '*self-neglect*', '*risk-dep*', etc.), because of the '*generic*' or '*g*' super node (i.e. '*gen-hopelessness*'). In such case, the

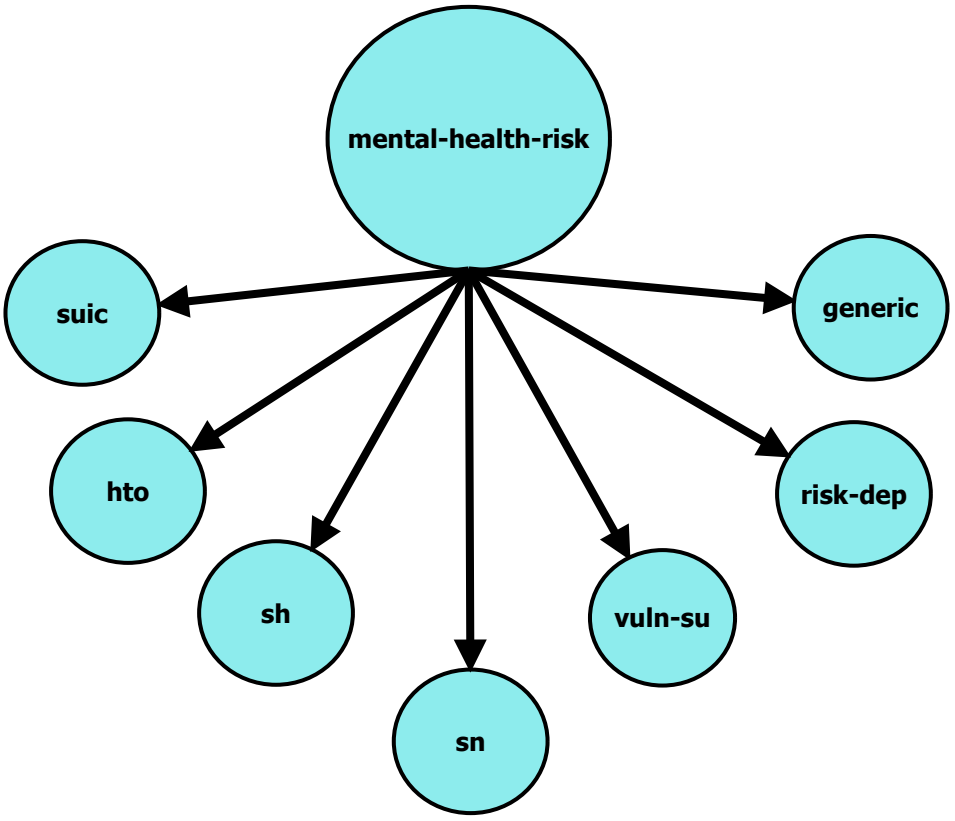

FIGURE 3.3: An example of contextual concepts in GRiST.

'*generic*' or '*g*' datum nodes, then these must keep a single value-mg list in different contexts.

2. **The generic-type ('*gd*')**

   The generic-type ('*gd*') concepts also keep repeating structure, but these are not clearly homogeneous. The tag of such nodes is '*generic distinct*' or '*gd*' or heterogeneous concepts and datum nodes. If such nodes are concepts, then their appearance effect the data structure of their sub nodes. The '*RIs*' of their sub nodes can vary in different contexts. If such nodes are datum components, then they can have different value-mg lists in different contexts.

3. **The generic-datum (path to generic node)**

   The '*generic-datum*' is actually an attribute of some repeating datum nodes that keep a path as a reference or pointer of the location, where the associated datum node has complete definition. For example, the '*gen-sh-cuts*' datum node has

a complete definition in the '*self-harm*' context and this datum node keeps the reference in the '*generic-datum*' attribute in '*suicide*' context in the SST.xml. The conceptualisation of such repeating datum nodes is completely distinctive from other repeating datum nodes. The value-mg lists of such repeating datum nodes do not vary in different contexts, but there is a possibility that the RIs of such datum nodes can vary context to context. The reason for the management of such attribute (i.e. '*generic-datum*') is to keep constant knowledge in every context, where these datum nodes are initialised with their complete definitions in RIT and CAT xml.

3.3.2.2 The non-generic nodes

The non-generic nodes are defined only in one context and such nodes are non-redundant. The variation or stability of the data (i.e. RIs) can be seen in such concepts and datum nodes, if they are the sub nodes of a generic concept. The non-generic datum nodes have a single value-mg list. No special attributes are defined for the identification of such non-generic concepts and datum nodes.

The data structure of the generic (i.e. repeating) and non-generic (i.e. non-repeating) nodes are managed by employing some attributes in the XML trees.

3.3.3 Attributes of the nodes in XML trees

For understanding the rationale of the nodes of the XML trees (such as SST, ST, RIT, QT and CAT), it is necessary to know their semantics and associated attributes. Some attributes are used only in specific tree(s) such as, the '*prune-for*' attribute (see details in Section 3.3.3.7) in the SST tree for discarding nodes from a population ST. Table 3.1 shows the main attributes used to define the trees, their transformation and processing.

TABLE 3.1: Examples of mandatory attributes of domains of Galatean model of classification in XML specifications [2].

3.3.3.1 The '*code*' attribute

The '*code*' attribute is used to keep the identification description (id) or short name of the nodes in XML trees (see Figure 3.4).

<node code="**suic**" label=" **suicide**" value-mg="**((0 0) (10 1))**" >

FIGURE 3.4: An example of '*code*' attribute in SST.xml..

3.3.3.2 The label attribute

The '*label*' attribute is used to describe the contents of the '*code*' attribute of the concepts and datum components (see Figure 3.4).

3.3.3.3 The '*generic-type*' attribute

The '*generic-type*' attribute is used in the XML trees to represent the generic and generic distinct concepts and datum nodes. The generic and generic distinct concepts and datum nodes are repeated in different contexts. The purpose of the attribute is to

support in initialising the RIs of the sub nodes of the generic and generic distinct concepts. The existence of this attribute indicates that the specified datum node either has a single value-mg list or has different value-mg lists for different contexts.

<node code="**sn-appearnce**" label="**appearance indicators of self neglect**" question="**(((learning-disabilities older working-age) "Are you concerned about the person being at risk of self neglect or neglect by others?") ((child-adolescent) "Are you concerned about the young person being at risk of self neglect or neglect by others?") ((iapt) "(Therapist question) Are you concerned about the person being at risk of self neglect?") ((service-user) "Are you concerned that you are not looking after yourself?"))**" layer="**((service-user 0)(working-age 0)(child-adolescent 0)(older 0)(iapt 0)(learning-disabilities 0))**" generic-type="**g**">

FIGURE 3.5: An example of '*generic-type*' attribute with '*g*' value in SST.xml.

The value of the '*generic-type*' attribute in the datum node must be either '*g*' or '*gd*' (see Figures 3.5 and 3.6).

<node code="**gen-feel-emot**" label="**feelings/emotions**" question="**(((child-adolescent) "Are you concerned about risks due to the young person's feelings/emotions?") ((iapt learning-disabilities older working-age) "Are you concerned about risks due to the person's feelings/emotions?") ((service-user) "Are you concerned about the way you are feeling?"))**" generic-type="**gd**">

FIGURE 3.6: An example of '*generic-type*' attribute with '*gd*' value in SST.xml.

In such case, if the contents of the '*generic-type*' attribute is '*g*' then it indicates that the specified node is a generic concept or datum node. The sub nodes of a generic concept must have fixed or constant '*RIs*' in different contexts. On the other hand a generic datum node has a fixed value-mg list in different contexts (see Figure 3.7).

In such case, if a concept has the '*generic-type*' attribute with '*gd*' contents, then it renders a generic distinct concept or datum node. The sub nodes of a generic distinct concept can vary their '*RIs*' in context to context, and the generic distinct datum nodes have different value-mg lists in different contexts. For example, the '*gen-gender*' is a generic distinct datum node in the GRiST. It has different value-mg lists for different contexts (see Figure 3.8).

<node code="**sn-hygiene**" label="**personal hygiene**" question="**(((child-adolescent) "To what extent does the person have poor young personal hygiene (eg smell, dirty hair and nails)?") ((iapt learning-disabilities older working-age) "To what extent does the person have poor personal hygiene (eg smell, dirty hair and nails)?") ((service-user) "How much do you think you have stopped keeping yourself as clean as usual?"))**" value-mg ="**((0 0) (10 1))**" generic-type="**g**">

FIGURE 3.7: An example of a generic datum node having a fixed value-mg list in SST.xml.

<node code=" **gen-gender** " label="**gender**" value-mg="**(([suic >> gen-demog >> gen-gender] ((MALE 1) (FEMALE 0))) ([sh >> gen-demog >> gen-gender] ((MALE 0) (FEMALE 1))) ([sn >> gen-demog >> gen-gender] ((MALE 1) (FEMALE 0))) ([hto >> gen-demog >> gen-gender] ((MALE 1) (FEMALE 0))) ([risk-dep >> gen-demog >> gen-gender] ((MALE 1) (FEMALE 0))) ([vuln-su >> gen-demog >> gen-gender] ((MALE 0) (FEMALE 1))))**" values="**nominal**" generic-type="**gd**">

FIGURE 3.8: An example of a generic distinct datum node having different value-mg lists in SST.xml.

3.3.3.4 The '*generic*' attribute

The '*generic*' attribute represents the generic concepts. The generic concepts are also repeated concepts. This attribute keeps the path of the specified concept (see Figure 3.9) where the full definition of the concept, or if it has sub nodes or branches is found.

<node code=" **sn-appearnce**" label=" **appearance indicators of self neglect** " generic=" **sn >> sn-app-behavr >> sn-appearnce**"/>

FIGURE 3.9: An example of '*generic*' attribute in RIT.xml.

In the CAT.xml, a generic concept with its whole inside branch(es) repeats in different contexts, but this attribute is not included in the CAT.xml, because such nodes are initialised with their complete definitions in different contexts, where the references were mentioned in the form of the paths in the '*generic*' attribute in the RIT.xml (see Figure 3.10).

- <node code="**sn**" label="**self neglect**" value-mg="**((0 0) (10 1))**" values="**scale**" question="**(((child-adolescent) "In your judgement, to what extent is the young person at risk of self-neglect?") ((iapt learning-disabilities older working-age) "In your judgement, to what extent is the person at risk of self-neglect?") ((service-user) "On a scale of 0-10, to what extent are you failing to look after myself?"))**">
    - <node code="**sn-app-behavr**" label="**person's appearance and behaviour during assessment indicating self-neglect**" value-mg="**((0 0) (10 1))**" values="**scale**" level="**1**">
        - <node code="**sn-appearnce**" label="**appearance indicators of self neglect**" question="**(((learning-disabilities older working-age) "Are you concerned about the person being at risk of self neglect or neglect by others?") ((child-adolescent) "Are you concerned about the young person being at risk of self neglect or neglect by others?") ((iapt) "(Therapist question) Are you concerned about the person being at risk of self neglect?") ((service-user) "Are you concerned that you are not looking after yourself?"))**" layer="**((service-user 0)(working-age 0)(child-adolescent 0)(older 0)(iapt 0)(learning-disabilities 0))**" generic-type="**g**">

</node>

</node>

FIGURE 3.10: An example of a node that has the '*generic*' attribute with its definition in the CAT.xml in GRiST.

3.3.3.5 The '*generic-datum*' attribute

The '*generic-datum*' attribute indicates that the associated node is a repeated datum component. The rationale of the '*generic-datum*' attribute is to keep the path of the specified datum node. The complete definition of the datum node can be accessed from the given path (see Figure 3.11).

<node code="**gen-sh-cuts**" label="**self-harming injuries**" generic-datum=" **sh >> sh-specific >> sh-past-curr-ep >> sh-occ-of-ep >> gen-sh-cuts**"/>

FIGURE 3.11: An example of '*generic-datum*' attribute in SST.xml.

3.3.3.6 The '*populations*' attribute

The '*populations*' attribute is used to represent different end users of a domain. This attribute is only associated with the root node of the assessment tool (see Figure 3.12).

<node populations="**(hub-depot hub depot)**" label="**lorry/pallet balance**" code="**lorry-pallet-balance**">

FIGURE 3.12: An example of the '*populations*' attribute in ADVANCE tool.

3.3.3.7 The '*prune-for*' attribute

The '*prune-for*' attribute is used to discard or remove concepts and datum nodes that are prohibited for a potential population (see Figure 9.12).

<node help="**(((service-user) "0 = no dependants or no risk at all, 10 = I pose an extremely serious risk"))**" code="**risk-dep**" label="**(((iapt learning-disabilities older child-adolescent service-user working-age) "risk to dependents"))**" value-mg="**((0 0) (10 1))**" values="**scale**" question="**(((child-adolescent) "In your judgement, to what extent does the young person put dependents at risk, if any (consider both children and adults but answer zero if there are no dependents)?") ((service-user) "On a scale of 0-10 to what extent do you think you are a risk to your dependents? (think about the safety of both children and adults, but answer 0 if you have no dependents)"))**" prune-for="**(child-adolescent)**">

FIGURE 3.13: An example of a node that has the '*prune-for*' attribute.

3.3.3.8 The '*question*' and '*filter-q*' attributes

The '*question*' and '*filter-q*' attributes are used for holding the contents of the questions in xml trees. A question has some contents and its contents are applied to the particular population(s). This attribute keeps the population information and its associated question's content (see Figure 3.14).

- <node code="**suic**" label="**(((service-user) "ending your own life") ((iapt learning-disabilities older child-adolescent working-age) "suicide"))**" order="**((learning-disabilities 4))**" value-mg="**((0 0) (10 1))**" values="**scale**" question="**(((child-adolescent) "In your judgement, to what extent is the young person at risk of suicide?") ((iapt learning-disabilities older working-age) "In your judgement, to what extent is the person at risk of suicide?") ((service-user) "On a scale of 0-10 how likely is it that you will try to end your own life?"))**">
  - <node code="**sui-specific**" label="**(((service-user) "questions specifically about taking your own life") ((iapt learning-disabilities older child-adolescent working-age) "suicide specific questions"))**" filter-q="**(((iapt) "Do you ever have any thoughts about ending your life?"))**" layer="**((iapt 0))**" persistent="**hard**">

FIGURE 3.14: An example of '*question*' and '*filter-q*' attributes in SST.xml.

The '*level*' attribute

The '*level*' attribute represents the experience level of the population. The contents of this attribute is determined during the transformation of the CAT.xml. In this case, if the value is 0 then a naive user of the domain is assessed and in such case, if the value is 1 then a user who has some basic knowledge or is an expert of the domain is assessed (see Figure 3.15).

<node code="**suic-int-p-trig**" label="**potential triggers for prospective suicide**" value-mg="**((0 0) (10 1))**" values="**scale**" question="**(((iapt) "IAPT: Do your current feelings and circumstances make you feel like ending it all?") ((service-user) "Is there anything specific that is making you feel like ending it all?") ((learning-disabilities older child-adolescent working-age) "Are you concerned about anything that could trigger suicide attempts?"))**" layer="**((service-user 0)(working-age 0)(child-adolescent 0)(older 0)(iapt 0)(learning-disabilities 0))**" level="**1**">

FIGURE 3.15: An example of '*level*' attributes in SST.xml.

3.3.3.9 The '*level-question*' attribute

The '*level-question*' attribute is only used in the RIT.xml (see Figure 3.16). The contents of this attribute is created through some criteria. If a concept has '*level*' attribute, then the contents of the '*question*' attribute are copied to the '*level-question*' attribute during the transformation from the ST.xml to the RIT.xml. This attribute helps in transformation of the CAT.xml. The contents of this attribute are again copied to the '*question*' attribute during the transformation from the RIT.xml to CAT.xml and this attribute is not transformed into the CAT.xml. It is because all the generic and generic distinct concepts are initialised with their complete definitions in specified contexts or locations by determining the '*generic*' attribute value (i.e. path).

<node code="**gen-presentation**" label="**person's behavioural presentation during assessment**" question="**Are you concerned about the young person's behavioural presentation with respect to potential risks (eg verbal and physical behaviour, uneasy `gut' feeling in yourself)?**" value-mg="**((0 0) (10 1))**" values="**scale**" layer="**0**" level="**1**" generic="**gd**" level-code="**suic-gen-presentation**" level-question="**To what extent does the person's behavioural presentation during assessment match that of a young person who would give you maximum concern about suicide risk?**">

FIGURE 3.16: An example of '*level-question*' attribute in RIT.xml.

3.3.3.10 The '*level-code*' attribute

The '*level-code*' attribute and its contents are created during the transformation of the RIT.xml from the ST.xml. This attribute supports in initialisation of the complete definition of repeated nodes and their inside branch(es) into different contexts in the CAT.xml, where the paths were mentioned for references in the source (i.e. RIT.xml) tree. The contents of this attribute are also created through some criteria. If a generic distinct concept has '*level*' attribute and it also has a generic concept as an ancestor node, then the content format of the said attribute is like: '*level-code=gen-depression-gen-presentation*'. The '*gen-depression*' is a generic concept and the '*gen-presentation*' is a specified node in the given example from GRiST (see Figure 3.17).

- <node code="**gen-depression**" label="**depression**" persistent="**hard**" filter-q="**Does the young person have any history of depression (past or present)?**" generic-type="**g**">
  - <node code="**gen-presentation**" label="**person's behavioural presentation during assessment**" question="**Are you concerned about the young person's behavioural presentation with respect to potential risks (eg verbal and physical behaviour, uneasy `gut' feeling in yourself)?**" layer="**0**" level="**1**" generic="**gd**" level-code="**gen-depression-gen-presentation**" level-question="**Are you concerned about the young person's behavioural presentation with respect to potential risks (eg verbal and physical behaviour, uneasy `gut' feeling in yourself)?**">

FIGURE 3.17: An example of the '*level-code*' attribute of a node (i.e. gen-presentation) that has a generic concept ancestor (i.e. gen-depression) in the RIT.xml of the GRiST.

If a generic distinct concept has a '*level*' attribute and it has no generic concept as an ancestor node, then the content format of the said attribute is like: '*level-code = suic-gen-presentation*'. In this case, the '*suic*' is a contextual concept and the '*gen-presentation*' is a specified node in the given example from GRiST (see Figure 3.18).

- <node code="**suic**" label="**suicide**" question="**In your judgement, to what extent is the young person at risk of suicide?**" value-mg="**((0 0) (10 1))**" values="**scale**">
  - <node code="**gen-presentation**" label="**person's behavioural presentation during assessment**" question="**Are you concerned about the young person's behavioural presentation with respect to potential risks (eg verbal and physical behaviour, uneasy `gut' feeling in yourself)?**" value-mg="**((0 0) (10 1))**" values="**scale**" layer="**0**" level="**1**" generic="**gd**" level-code="**suic-gen-presentation**" level-question="**To what extent does the person's behavioural presentation during assessment match that of a young person who would give you maximum concern about suicide risk?**">

FIGURE 3.18: An example of the '*level-code*' attribute of a node (i.e. gen-presentation) that has an individual concept ancestor (i.e. suic) in the RIT.xml of GRiST.

3.3.3.11 The '*multiple-tick*' attribute

In the SST and ST XML trees, the '*multiple-tick*' attribute is used in the datum components, which have multiple answers (see Figure 3.19). The '*multiple-tick*' attribute is not included into the RIT.xml, however the specified datum components are translated as generic distinct concepts into the RIT.xml and the contents of the '*multiple-tick*' attribute are used as the sub nodes of the specified node. An algorithm is utilised for this purpose to convert the datum components into the concepts during the transformation from the ST.xml to RIT.xml [62]. The purpose of this attribute is to keep the associated nodes, their generic knowledge and sub nodes in one location of the SST and ST trees and then transform them with their complete generic knowledge structure or definition and sub nodes into different locations for initialising their uncertain attribute (i.e. '*RI*').

<node code="**hto-targets**" label="**targets of harm to others**" value-mg="**((CARERS-FAMILY 1)(SHARED-ACCOM-NEIGHBOURS 1) (FRIENDS-PEERS 1) (HEALTH-WORKERS 1) (AUTHORITY-FIGS 1) (ETHNIC 1))**" values="**nominal**" question="**(((older child-adolescent working-age iapt learning-disabilities) "Has the young person targeted any particular group of people rather than complete strangers?") ((service-user) "Have you in the past, harmed any particular group of people?"))**" layer="**((service-user 0)(working-age 0)(child-adolescent 0)(older 0)(iapt 0)(learning-disabilities 0))**" persistent="**soft**" prune-for="**(working-age older service-user iapt learning-disabilities)**" multiple-tick="**((CARERS-FAMILY "Has the young person harmed anyone within the domestic setting?" [carers or family]) (SHARED-ACCOM-NEIGHBOURS "Has the young person harmed any neighbours or people sharing the living space?" [neighbours or people sharing living space])(FRIENDS-PEERS "Has the young person harmed any friends or colleagues?" [friends or peers]) (HEALTH-WORKERS "Has the young person harmed any health workers?" [health workers]) (AUTHORITY-FIGS "Has the young person harmed any authority figures?" [authority figures]) (ETHNIC "Has the young person harmed anyone from a different ethnic background?" [those from a different ethnic background]))**"/>

FIGURE 3.19: An example of '*multiple-tick*' attribute in SST.xml.

3.3.3.12 The '*ri*' attribute

The '*ri*' attribute is employed for holding the '*RIs*' of the nodes in the RIT. The knowledge experts have defined the data range of this attribute that is greater than or equal to 0.0 and less than or equal to 1.0 [62]. This information is mentioned in the paper manuals of the problem domains, but such information is not designed in RIT.xml.

3.3.3.13 The '*value-mg*' attribute

The '*value-mg*' attribute represents the list of the value-mg of the datum components (see Figure 3.20). The pairs having the datum values and their associated '*MGs*' in the list are used as the contents of this attribute. The knowledge experts have defined the data range of this attribute that is greater than or equal to 0.0 and less than or equal to 1.0 [62]. This information is mentioned in the specification document of the problem domains, but such information is not designed in RIT.xml.

<node label="**cost of company outside network delivering**" code="**cost-company-outside-net-deliv**" ri="**0.2**" values="**real**" question="**What is the cost per billing unit for the external company delivering your pallets?**" value-mg="**((0 1)(5 0))** "/>

FIGURE 3.20: An example of '*value-mg*' attribute in SST.xml in ADVANCE.

3.3.3.14 The '*values*' attribute

The '*values*' attribute is used in association with the '*value-mg*' attribute (see Figure 3.21). The contents of this attribute represent the data type of the datum values of the value-mg list. The data types of the data can be '*scale*', '*nominal*', '*integer*', '*real*', '*date-year*', '*date-month*', '*date-week*' and '*date-day*'. A brief details of these data types are given in the next.

1. **The '*scale*'**

   A standard numerical '*scale*' is a rating scale that is used to measure or identify quantitative data. The default value of a '*scale*' is from 0 to 10.

- <node code="**gen-emot-abse**" label="**emotional abuse**" order="**((service-user 0))**" value-mg="**((0 0) (10 1))**" values="**scale**" question="**(((child-adolescent) "To what extent has the young person been emotionally and/or racially abused?") ((service-user) "To what extent have you been emotionally and/or racially abused?") ((older working-age iapt learning-disabilities) "To what extent has the person been emotionally and/or racially abused?"))**" filter-q="**(((working-age iapt older learning-disabilities) "Has the person ever been emotionally abused due to, for example, race, religion, sexual orientation, appearance?") ((child-adolescent) "Has the young person ever been emotionally abused due to, for example, race, religion, sexual orientation, appearance?") ((service-user) "Have you ever been emotionally abused due to, for example, race, religion, sexual orientation, appearance?"))**" persistent="**hard**" level="**1**">
  - <node code="**gen-emot-abse-last**" label=" **most recent episode of emotional abuse"))**" value-mg="**((0 1) (24 0.2))**" values="**date-month**" question="**(((service-user) "When was the most recent episode of emotional abuse?") ((iapt learning-disabilities older child-adolescent working-age) "When was the most recent episode of emotional?"))**" persistent="**soft**"/>

  </node>

- <node code="**gen-financial-abuse**" label="**financial abuse**" order="**((service-user 3))**" value-mg="**((YES 1) (NO 0))**" values="**nominal**" question="**(((child-adolescent) "Has the young person ever been financially abused?") ((service-user) "Have you ever been the victim of financial abuse?") ((older working-age iapt learning-disabilities) "Has the person ever been financially abused?"))**" persistent="**hard**"/>

FIGURE 3.21: An example of '*values*' attribute in SST.xml.

2. **The '*integer*'**

   An '*integer*' is a data type that represents whole numbers (not fraction). These quantities can be positive or negative or zero. The 0, 10 and 18 are different examples of the '*integer*' in the $((0\ 1)(10\ 0.8)(18\ 0))$ value-mg list.

3. **The '*real*'**

   A '*real*' data type represents a quantity that can be expressed as a finite or infinite decimal expansion. It includes all the counting numbers, integers, rational numbers, and irrational numbers. For example, the 0.0 and 5.0 are '*real*' numbers in

the $((0.0\ 0)(5.0\ 1))$ value-mg list.

4. **The '*date-year*'**

   The '*date-year*' refers the units of measurement for the time between the given and current date of the year. The $((0\ 1)(5\ 0.3))$ value-mg-list keeps the '*date-year*' values (i.e. 0 and 5).

5. **The '*date-month*'**

   The '*date-month*' refers the units of measurement for the time between the given and current date of the month. The 3, and 36 are examples of the '*date-month*' in the $((3\ 1)(36\ 0))$ value-mg list.

6. **The '*date-week*'**

   The '*date-week*' refers the units of measurement for the time between the given and current date of the week. The $((0\ 1)(30\ 0))$ value-mg-list keeps the '*date-week*' values (i.e. 0 and 30).

7. **The '*date-day*'**

   The '*date-day*' refers the units of measurement for the time between the given and current date of the day. The 0 and 26 are examples of the '*date-day*' in the $((0\ 1)(26\ 0))$ value-mg list.

8. **The '*nominal*'**

   The '*nominal*' data are categorical data where the order of the categories is arbitrary or it is a type of data in which there are limited categories but no order. The '*YES*' and '*NO*' are '*nominal*' values in the '*((YES 1) (NO 0))*' value-mg list.

The generic and generic distinct concepts and datum nodes are used to distinguish nodes that are located in more than one place with exactly the same values (generic) or repeat with variations in their values (generic distinct). The sub nodes of generic concepts keep fixed '*RIs*' in different contexts, but the '*RIs*' of the sub nodes of the generic distinct concepts can vary from context to context. Similarly, the value-mg list of a generic datum node is fixed, but not for generic distinct datum nodes. It is difficult, but not impossible, to make a consistent data structure from repeating nodes in different contexts or locations. The consistency checking is complex and vexatious,

especially if the manipulation of different XML trees is governed by using a paper manual. The consistency and reliability checking of the XML nodes, their relationships and attributes with a chain of trees through a specification document is strenuous. The knowledge management of such complex data structure is difficult in simple knowledge representation languages (i.e. XML). The variant trees with their uncertainty attributes for different users requires an OWL based decision support system, which can help in making decisions and keeping their knowledge-bases consistent.

3.4 Why OWL?

The main focus of the Galatean decision support system is on the specification of galateas or classes and how they represent classification knowledge into one of two classes. For example, the original application of the Galatean model of classification was to pick out winners and losers in horse race [27]. In this simple one-class problem (where the losers are just the inverse of the winners) and a single type of end user (i.e. a foolish gambler), there is no need for OWL specifications. The demand of the ontology-centric decision support system [69] is increased when the approach is applied to a more complex domain, such as mental health risk assessments and logistics. The developers of GRiST and ADVANCE (Advanced Predictive-Analysis-Based Decision-Support Engine for Logistics) tools also eschewed OWL [67] because they wanted to keep close to the psychological semantics of galateas rather than the machine language. However, as the work progressed and it became clear that different tree structures were required for different populations. The manipulation and maintenance of their variant XML knowledge trees were much more difficult. The need of OWL was more pressing and the motivation for my PhD.

Different applications of the Galatean model of classification can be seen in many knowledge domains, where decision making is difficult due to their complex knowledge structures, like GRiST and ADVANCE. It will be better to develop a generic intelligent process that can be applied to any problem domain, which is constituted on the concept of the classification notion. The OWL is a knowledge representation language that provides the opportunity to define explicit classes, properties and constraints. The OWL

reasoner provides the facility to confirm the internal stability of the relationships, which are maintained by using different OWL properties. It classifies the classes into their correct taxonomies by interpreting their explicit constraints without intending to know the names of classes or properties. This thing motivate us to choose OWL for constructing an intelligent decision support system for continues evolving the knowledge engineering process for the knowledge domains of the Galatean model of classification.

The current knowledge domains of the Galatean model of classification have a series of XML knowledge trees for maintaining their complex knowledge [67]. The root tree (i.e. SST) is an all-inclusive tree that keeps knowledge for all populations within a domain. The rest of other knowledge trees are derived for a single population. The issues are that it is hard to transform only required knowledge into derived tree structures for a specified population and then confirm the consistency of the knowledge between the source and derived XML trees by reading a specification document. The level of complexities increases when another agent is obtained from the derived agent for initialising the RIs or weightings of the concepts and datum nodes. The client assessment agent is derived to yield to the experience level of the end user and it is the tail end of the derivative agents' process, where the actual classification process is started. A root or source agent specifies a number of different agents that work together in a computational intelligence environment. This is a plausible knowledge engineering problem where the source and derivative agents knowledge structures are trees (see example of root and derived agents in Figure 3.22).

These things are not simple and straightforward, and cannot be expected from the knowledge representation languages, which are more human-centric. It is necessary to define explicit rules and constraints for maintaining logical tree hierarchies and inherited structures. The attributes and their constraints in the concepts and datum nodes are needed not only to help in knowledge maintenance and management, but also to support in the reasoning and deducing of required knowledge. It is also necessary to define knowledge for the concepts and datum nodes that is mentioned in the paper manual, but XML knowledge trees lacks such information. For example, the data range of the RIs and MGs is defined by the knowledge experts and this information is mentioned in the specification document, but not managed in the XML trees. These

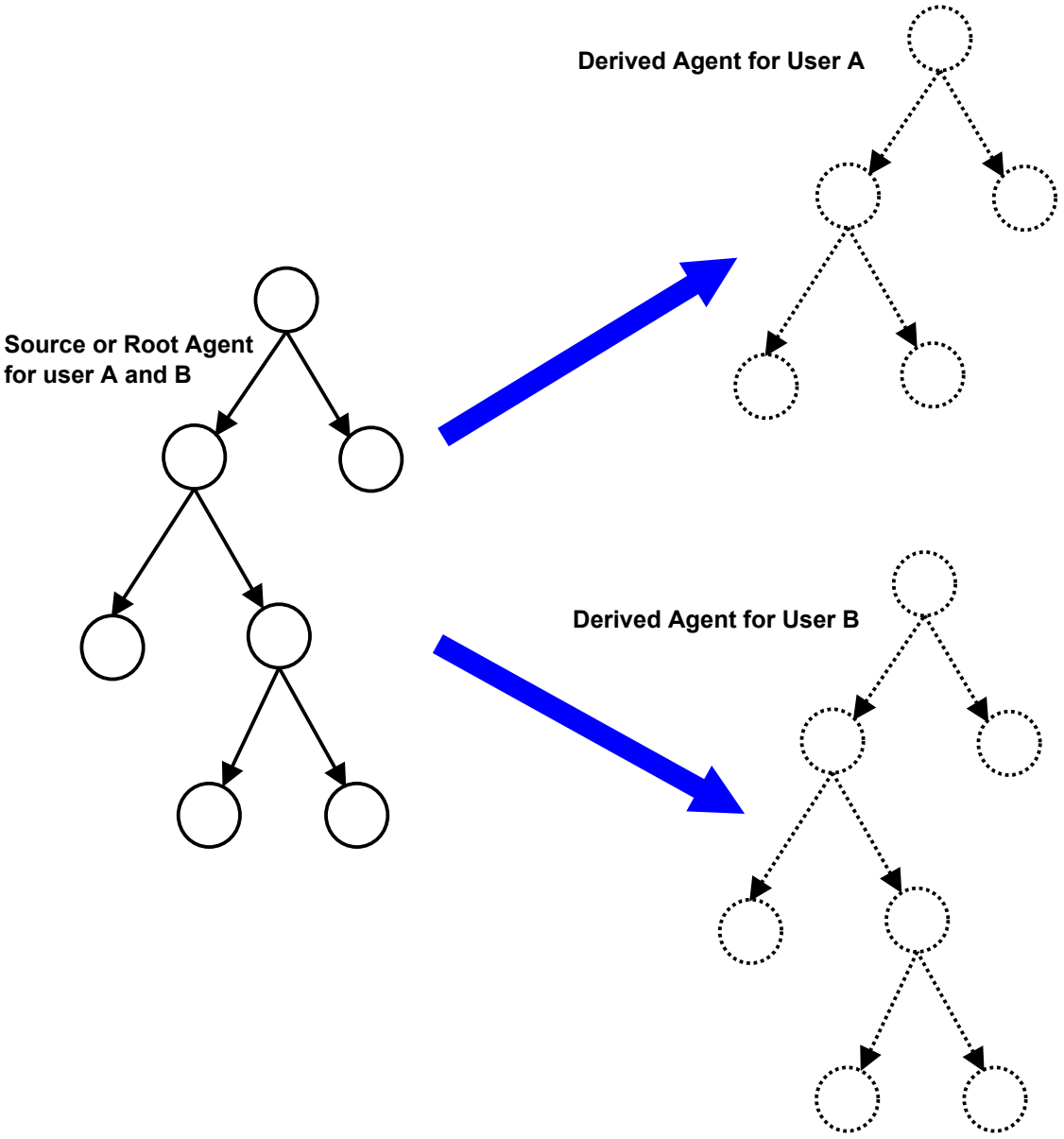

FIGURE 3.22: An example of a root agent and two derived agents, each derived agent is obtained for a particular end user. The edges and nodes of the derived agents are depicted in dotted form.

issues motivate us to use a knowledge representation language that can provide structures, which are close to human expertise and also provide an engineering solution. Therefore, the development of a knowledge engineering framework based on OWL can support the construction of explicit relationships and constraints in the root and derived agents and this framework can also bolster the knowledge engineering process of an

assessment tool of different domains.

The ultimate focus of the study is to develop a generic intelligent knowledge engineering environment based on OWL through which the manipulation and maintenance of different agents or tree hierarchies can be possible without human interpretations. It can also be possible to confirm knowledge consistency of the knowledge base, which have a progressive data structure. It can be only feasible through the machine-centric knowledge engineering process, and OWL is a good choice in managing, organizing and manipulating the knowledge-base of different domains (see more detail of OWL in chapter 4). An ontology-centric decision support system can help in making decisions for heterogeneous trees and users, and also support in the knowledge engineering. Hence, the specific objective of this study is to replace the specification document with OWL-based specifications.

The classification process of the Galatean model of classification is activated by the user provided data, which is compared to the stored data in the value-mg lists. The MGs for the concepts and datum nodes are initialised by using some mathematical functions. Semantic Web Rule Language (SWRL)[1] is an extension of OWL that can be used for creating rules, which can calculate mathematical data by employing some built-in functions. The SWRL rules can be defined for generating the MGs for the concepts and datum nodes of the whole Galatean hierarchy.

3.5 Conclusions

The Galatean model is a cognitive psychological model that attempts to encapsulate human expertise. It is based on a theory of classification that implements decision making, where the decisions depend on which classes should be assigned the objects. For making decisions in more than one category for more than one population in a knowledge-base creates complexities in the knowledge engineering process. The CAT alone is not sufficient to hold knowledge of a knowledge domain that has more than

[1] http://www.w3.org/Submission/SWRL/

one category for decision making for various populations. An ST is required to structure the knowledge for a specified population. There is still a possibility that this logical tree may have more than one categories of knowledge, which may repeat in different locations. For initialising the RIs another hierarchical structure is derived, where the RIs of the sub concepts and datum nodes are initialised by determining certain attributes of their repeating and non-repeating concept nodes. Finally, a CAT is derived where the classification and decision making is possible. These things are not simple and straightforward. An intelligent decision support system is required for providing consistent knowledge maintenance, knowledge engineering process and evolving expertise.

The XML is a quite simple language and it is difficult to design or develop knowledge with its correct semantics by using this representation structure. For example, the nodes in the XML trees have some attributes, which help in knowledge description, manipulation and management. By analyzing the attributes of the XML trees of the knowledge domains of the Galatean model of classification, it has revealed that some attributes (such as '*value-mg*', '*question*', '*filter-q*' and '*population*') cannot possess correct functionalities as an attribute, because they have their own characteristics. For example, a question has some contents and those contents are applied to some population. Some attributes should be dropped, because they are completely dependent on the syntactic structure of their associated nodes or contexts. The '*level-question*', '*level-code*', '*multiple-tick*' are temporary attributes that are completely based on syntactic structure of their associated nodes or contexts. The manipulation and alteration in the divergent XML trees, beside their complex attributes through a paper based specification, is a difficult task for the developers as the approach is applied to more complex domains such as mental health and logistics.

An assessment tool having complex mutative series of tree structures that is used by a number of different end users increases the importance and demand of knowledge codification in machine-centric form. The OWL can be a best option for dealing these knowledge engineering issues. It can help in formalising the human represented expertise and skills in the machine-centric shape. It can aid in Galatean decision support system by facilitating in creating, structuring and implementing different explicit concepts, roles, constraints and restrictions, which are closer to human expertise. The

OWL based structures can also increase the understanding of machine, which can make predictions in a very short period of time with consistent performance at remote venues at any time. The OWL reasoner makes reasoning and classifications over OWL based concepts and it also supports in consistency checking of OWL-based knowledge. Therefore, OWL can be an excellent choice for machine's interpretations, reasoning and understandability of human skills, it can help the machine to disseminate human expertise and the SWRL rules can be employed for making predictions over mathematical data or for classification process.

The next chapter is a review of the Semantic Web Technologies. This chapter will describe the Semantic Web Technologies regarding the Galatean model of classification.

Chapter 4

Overview of Semantic Web Technologies

4.1 Introduction

Semantic Web technologies and frameworks are a set of technologies that enable the Web data to provide a formal description of concepts, terms, and relationships within a given knowledge domain or application. The applications like proteomics ontologies, semantic web services, or blogs and social networks are constantly increasing by companies like Google, Amazon, YouTube, Facebook, LinkedIn and others [70]. The benefit of using these technologies is that the Web is not only providing information everywhere, but also facilitating in retrieving meaningful information. This chapter presents a brief introduction of the Semantic Web layer architecture. The Web Ontology language (OWL), and Semantic Web Rule Language (SWRL) are two prominent layers of the semantic web architecture that are discussed in detail in this chapter. The notations of the OWL and SWRL present in Manchester syntax, which is user-friendly and more easy to understand [71, 72].

4.2 Semantic Web

The semantic web has emerged a prominent role in web besides the machine readability, understanding and reasoning. The web pages are just the web documents.

These web documents or web pages are used to store and retrieve the information. The problem is that the machine does not understand the meaning of the contents of those documents. The semantic web helps the machine to understand the meaning behind the web pages or documents. Semantic web technologies help to emerge the human represented knowledge in machine-centric form and also supports in automatic manipulation of the information on the web.

The architecture for the semantic web is crucial to its eventual realisation, graphical representation and correct modelling. The layered architecture proposed versions of Berners-Lee [73] are most well-known in web designing. Haytham Al-Feel et.al evaluated the layered architecture of semantic web in their paper [74]. They evaluated the four versions of Tim Berner-Lee[1], [2], [3], [4] and pointed out some weaknesses of those architectures. They presented a new architecture that corrects shortcomings in the previous architecture. The final version of semantic web layering cake is presented in Figure 4.1.

The URI/RI (Uniform Resource Identifiers / Internationalised Resource Identifier) is the first layer of the architecture that provides a baseline for rendering characters used in most of the world languages and supports to identify things or concepts or resources on the Web and in the repository. Extensible Markup Language (XML) is a second layer of the architecture. It is easy to use knowledge representation language for creating a simple knowledge-base. The next layer of the architecture is the Resource Description Framework (RDF). It is a framework for representing information about resources in a graph form. It is based on triples subject-predicate-object that form a graph of the data. All data in the semantic web use RDF as the primary representation language. The heart of all Semantic Web applications is the use of ontologies. A success factor of the semantic web is reusing and sharing of the ontological knowledge, and its ontological framework permits the data to be shared and reused across applications, organisations, institutions and community boundaries [75]. The most well known language for defining web ontologies is OWL.

[1] `http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html`
[2] `http://www.w3.org/2003/Talks/01-sweb-tbl/Overview.html`
[3] `http://www.w3.org/2005/Talks/0511-keynote-tbl/`
[4] `http://www.w3.org/2006/Talks/0718-aaai-tbl/Overview.html`

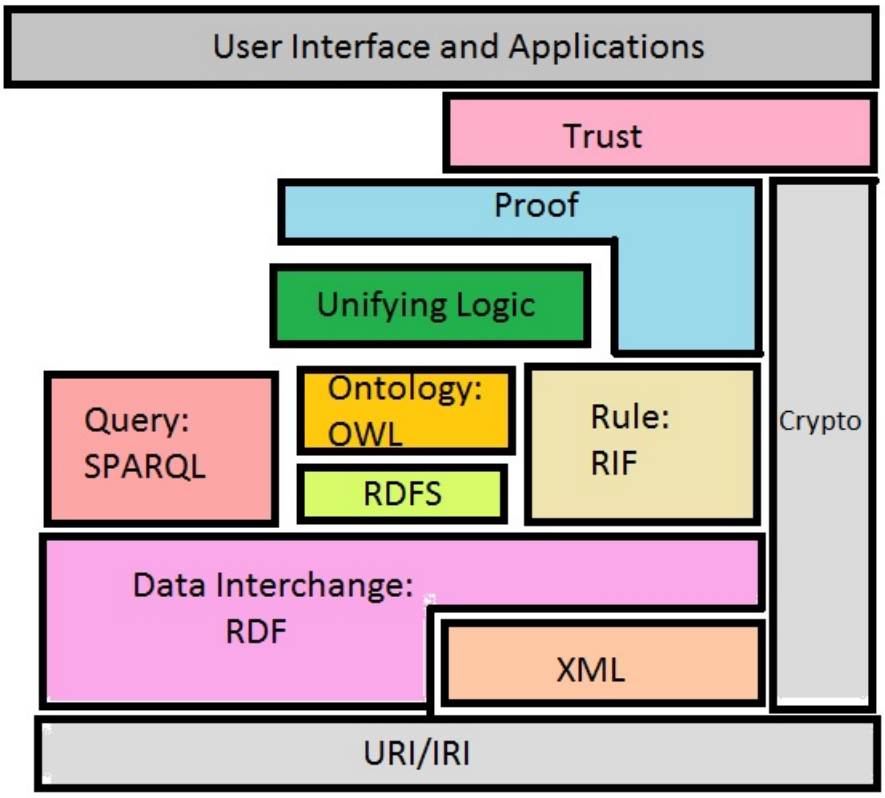

FIGURE 4.1: Semantic Web Layer Cake.

4.2.1 Web Ontology Language (OWL)

Web Ontology Language (OWL) [76, 77] is the latest standard in ontology languages, which was developed by members of the World Wide Web Consortium[5] and Description Logic community. It is a functional layer of semantic web layered architecture. OWL is a knowledge representation language that supports in defining explicit concepts, roles, constraints and restrictions, which are closer to human expertise. OWL assists in machine interpretability of web contents with additional vocabulary and formal semantics. OWL is based on RDF [78] and it is a rich language for defining terms. It is a more complex ontology language than RDF. It contains a repository of terms or symbols which are used to describe a specific domain and provides a mechanism for describing properties and their relation between different resources. It also supports in defining and declaring of machine processable descriptions of classes and properties and it can encrypt implicit rules constraining the structure of a piece of reality.

[5] http://www.w3.org/TR/owl2-overview/

OWL is currently becomes a most popular research topic. It attracts many artificial intelligence research communities, and these communities are interested to implement it in knowledge engineering, natural language processing and knowledge representation applications [79]. OWL enables these web applications to go beyond the identification of basic metadata elements to the interpretation of the semantics relating to the content of resources. It also facilitates the interoperability between heterogeneous systems involved in commonly interested domain applications, by providing a shared understanding or conceptualisation [80] of the problem domains. Jorge [81] has compared OWL with some other knowledge representation languages and concluded that most management facets can be translated into different tags in OWL and it is quite useful in the integration of diverse management domains.

There are many general benefits of using ontologies: (1) the consistency of the ontologies or knowledge-bases can be confirmed by using an OWL reasoner which makes OWL prominent to all other knowledge representation languages, (2) It can involve the machine in reasoning and understanding of human represented knowledge, (3) It can support in sharing of knowledge, (4) It can process the content on the basis of the meaning instead of its syntax, because it is a deliberate semantic structure [82], and (5) It can capture communal knowledge, which is not private to some individual or member, but accepted by a large group. OWL has some sub-languages. The details of these sub-languages is given below.

4.2.1.1 OWL sub-languages

OWL has three expressive sub-languages: (1) OWL Lite, (2) OWL DL and (3) OWL Full [83].

1. **OWL Full**

   OWL-Full is the most expressive OWL sub-language. It uses all OWL language primitives and it is fully compatible with RDF. It is used in situations where very high expressiveness is required than being able to guarantee the decidability or computational completeness of the language. It is therefore not possible to perform automated reasoning on OWL-Full ontologies.

2. **OWL DL**

   OWL DL is a sublanguage of OWL Full which is conceptually based on Description Logics. OWL-DL is less expressive than OWL Full. It is a decidable fragment of 'first order language'. It permits reasonably efficient reasoning support. It is possible to automatically compute the classification hierarchy and support in checking for inconsistencies in an ontology.

3. **OWL-Lite**

   OWL-Lite is a sub language of OWL DL. It is a simple language and easier to implement for the experts. It is used in situations where only a simple class hierarchy and simple constraints are needed.

An OWL ontology consists of classes, properties and individuals. A brief introduction of OWL individuals, properties, classes and restrictions will be presented next. It will help to understand the terminologies of web ontology language while describing and designing the structure of the Galatean model based on OWL.

4.2.1.2 OWL individuals

The real-world objects are called individuals or instances or objects. The individuals that are members of a given OWL class are called its class extension. A class in OWL is a classification of individuals into groups which share common characteristics. If an individual is a member of a class, it tells a machine reader that it falls under the semantic classification given by the OWL class. An object can belong to one or more classes. For example, the '*Cat*' instance belong to '*Animal*' and '*Mammal*' classes. OWL properties may be used to relate one individual to another.

4.2.1.3 OWL Properties

Properties are binary relations those link individuals. The OWL properties are independent to OWL classes, the existence and non-existence of OWL classes does not impact on their lives that make OWL distinctive to other computer languages. The OWL properties can be defined as sub and super properties as OWL classes. OWL has three

types of properties (1) Object Properties, (2) Data type Properties and (3) Annotation Properties.

1. **Object properties**

   The object properties are used to link an object to another object. These Properties can also be defined as the inverse. If a property links individual '*x*' to individual '*y*' then its inverse property will link individual '*y*' to individual '*x*'. For example, the '*isToppingOf*' is the inverse property of the '*hasTopping*' property [84]', both are object properties and these have identical characteristics. An object property can have various characteristics like functional, inverse functional, transitive, symmetric, asymmetric, reflexive and irreflexive. The details of these properties are given below.

   - **Functional properties**

     A functional property restricts at most one relationship with a given individual. For example, the '*hasGender*' is a functional property. If we say that '*Jhon hasGender Male*' ('Jhon' and '*Male*' are individuals) than the individual '*Jhon*' cannot make relationship to another individual (i.e. '*Female*') via using the '*hasGender*' property.

   - **Inverse functional properties**

     An inverse functional property keeps the characteristics of functional property, and its inverse property has also same characteristics.

   - **Transitive properties**

     A transitive property involves the transition of relationships between the fillers of the constraints. For example, '*Harry*' has sister '*Maria*' and '*Maria*' has sister '*Kat*', then we can deduce that '*Harry*' has sister '*Kat*' ('Harry', '*Maria*' and '*Kat*' are individuals). The relationships are defined between these individuals via using '*hasSister*' property. The characteristics of the '*hasSister*' property are defined transitive, then the OWL reasoner can infer that '*Harry*' has sister '*Kat*'.

   - **Symmetric properties**

     The symmetric properties make the relationships convenient, where the identical relationships are required for the individuals. For example, the '*hasFriend*' property has symmetric attribute. If the '*Sara*' has friend '*Tara*' then

it is inferred by using the OWL reasoner that '*Tara*' has friend '*Sara*'. In this way, the specified property (i.e. '*hasFriend*') is its own inverse property.

- **Asymmetric properties**

  The asymmetric or antisymmetric properties have opposite qualities, then the symmetric properties. For example, If the individual '*Robert*' is related to the individual '*David*' via the '*isChildOf*' property, then it can be inferred that '*David*' is not related to '*Robert*' via the '*isChildOf*' property. It is, however, reasonable to state that '*David*' could be related to another individual '*Bill*' via the '*isChildOf*' property. In other words, if '*Robert*' is a child of '*David*', then '*David*' cannot be a child of '*Robert*', but '*David*' can be a child of '*Bill*'.

- **Reflexive properties**

  The reflexive property must relate an individual to itself. For example, everybody has himself as a relative and '*hasRelative*' property can be defined as reflexive property.

- **Irreflexive properties**

  The irreflexive properties are opposite to reflexive properties. These are used in situations when both individuals in relation are not same or they must be distinctive. For example, the property '*isMotherOf*': an individual '*Alice*' can be related to individual '*Bob*' along the property '*isMotherOf*', but '*Alice*' cannot be '*isMotherOf*' herself.

2. **Data type properties**

   A data type property links an individual to an XML schema data type value or an RDF literal. OWL data type properties have only one characteristic (i.e. functional).

3. **Annotation properties**

   Annotation properties are used to define the metadata or description of the classes or individuals and properties.

4.2.1.4 OWL classes

The classes are the key building blocks of OWL and these are interpreted as sets of objects those represent the individuals in the domain of discourse [85]. The concepts of

the knowledge domains are defined in the form of OWL classes. The description of the OWL classes can be expressed through '*class descriptions*', which can be combined into '*class axioms*'. The details of these are given below.

1. **Class description**

   A class description describes an OWL class, either by a class name or by specifying the class extension of an unnamed anonymous class.

   (a) **Named classes**

   Named OWL classes are described in terms of their named. The '*Named*' classes are also called '*primitive*' classes. The '*Thing*' is a predefined '*primitive*' class and all the classes defined in the knowledge-base are the subclasses of it. It means any member of the subclass is also the member of the '*Thing*' class.

   (b) **Anonymous (unnamed) classes**

   An anonymous class is also called an '*unnamed*' class [83, 86]. When a restriction is added to a named class, it manifests itself as an anonymous superclass of the '*Named*' class. Anonymous (unnamed) classes are formed from logical descriptions. They contain the individuals those satisfy the logical description. Logical descriptions or expressions or classes are constructed from other classes using the boolean operators AND ($\sqcap$), OR ($\sqcup$) and NOT ($\neg$).

   i. **Enumeration**

   The '*enumeration*' is a kind of class description, which enables a class to be described by exhaustively enumerating its instances. For example, the '*Number*' class has following enumerated individuals (i.e. '*one*', '*two*' and '*three*').

```
Class: Number
    SubClassOf:
        hasNumbers only {one, two, three}
```

   ii. **Property restriction**

   Class expression formed by constraints on properties are called restrictions. All types of restrictions in an anonymous class can contain some

individuals. Restrictions act along properties, describing sets of individuals in terms of the types of relationships those individuals participate in. OWL restrictions are divided into two different categories (1) value restriction and (2) qualified cardinality restriction.

- **Value restriction**

  The '*value restriction*' along a property and a filler are used to state a set of individuals. These individuals represent all values or at least one relationship, along with a particular property. The '*value restriction*' is further divided into three different categories: (1) existential restriction, (2) universal restriction and (3) hasValue restriction.

  A. **Existential restriction**

  Existential restriction is actually an anonymous class and it is also called '*Some*' restriction. Existential restrictions are represented by the $\exists$ symbol. For example, existential restrictions describe the individuals those have at least (the existence of) one relationship to individuals those are members of some other specified class.

  ```
  hasTopping some PizzaTopping
  ```

  The above existential restriction describes the set of individuals that have at least one '*hasTopping*' relationship to an individual that is a member of the '*PizzaTopping*' class [87].

  B. **Universal restriction**

  The universal restriction is also called '*AllValuesFrom*' or '*only*' restriction. Universal restrictions are represented by the $\forall$ symbol. A universal restriction along the specified property has all values from the filler class.

  ```
  hasTopping only TomatoTopping
  ```

  The above universal restriction describes the set of individuals that only has '*hasTopping*' relationships to individuals of the '*TomatoTopping*' class.

  C. **hasValue restriction**

  A '*hasValue*' restriction describes the set of individuals that have at

least one relationship to another specific individual along a specified property. For example, following '*hasValue*' restriction describes the '*Canada*' (an individual) have at least one relationship along the '*hasCountryOfBirth*' property to a particular individual.

```
hasCountryOfBirth value Canada
```

The '*hasValue*' restrictions are represented by the ($\in$) symbol in description logic notation.

- **Qualified cardinality restriction**

  Cardinality restrictions describe sets of individuals in terms of the number (zero or more) of relationships that the individuals must participate in for a given property. There are three types of cardinality restrictions – (1) minimum, (2) maximum and (3) exactly.

  A. **Minimum cardinality restriction**

  Minimum cardinality restrictions describe the minimum number of relationships that an individual can participate in for a given property. For example, an interesting pizza required minimum 3 pizza toppings.

```
Class: InterestingPizza
    SubClassOf:
        hasTopping min 3 PizzaTopping
```

  The ($\geq$) symbol is used to represent the minimum cardinality restrictions which indicate the 'greater than or equal to' number (zero or more) of relationships via given property in description logic notation.

  B. **Maximum cardinality restriction**

  Maximum cardinality restrictions describe the maximum number of relationships that an individual can participate in for a given property. A maximum cardinality expression consists of four things (1) a nonnegative integer, (2) an object property expression and (3) a class expression and (4) cardinality restriction operator. For example, a vegetarian pizza can have a maximum of 5 pizza toppings.

```
Class: VegetarianPizza
```

```
SubClassOf:
    hasTopping max 5 PizzaTopping
```

The ($\leq$) symbol is used to represent the maximum cardinality restrictions which indicate the 'less than or equal to' number (zero or more) of relationships via a particular property in description logic notation.

C. **Cardinality restriction**

The cardinality restrictions describe the *exact* number of relationships that an individual must participate in for a given property. For example, a player must belong to exactly one team.

```
Class: Player
    SubClassOf:
        belongsTo exactly 1 Team
```

The ($\equiv$) symbol is used for representing an $exact$ cardinality restrictions in description logic notation.

iii. **Intersection, union and complement**

A. **Intersection**

An intersection is described by combining two or more classes using the AND operator ($\sqcap$). For example, the '*Vehicle*' and 'Car' are two classes and the '*Car*' is a subclass of the '*Vehicle*' class.

```
Vehicle and Car
```

The semantics of this intersection class means that the anonymous class that is described as a subclass of '*Vehicle*' and also a subclass of the '*Car*'.

B. **Union**

A union is created by combining two or more classes using the OR operator ($\sqcup$). For example, the '*Girl*' and '*Boy*' are two classes.

```
Girl or Boy
```

This describes an anonymous class that contains the individuals that belong to either the class '*Girl*' or the class '*Boy*' (or both).

C. **Complement**

This describes an anonymous class that contains the individuals that do not belong to the class extension of the class description. For

example, the '*NonVegetarianFood*' class is complement of the '*VegetarianFood*' class.

```
Class: NonVegetarianFood
    SubClassOf:
        not (VegetarianFood)
```

2. **OWL Axioms**

   Class descriptions shape the building blocks for defining classes through class axioms. Class axioms typically contain additional components that state necessary or necessary and sufficient conditions. OWL contains some language constructs for combining class descriptions into class axioms, their details are given next.

   (a) **Subclass axioms**

   These axioms represent '*necessary*' conditions. The necessary condition is a state of affairs that must prevail if another is to occur. For example, if '*Pizza*' is a necessary condition for the '*MeatyPizza*' class. It means without the '*Pizza*', '*MeatyPizza*' cannot exist, but it does not mean that the existence of the '*Pizza*' guarantees the existence of '*MeatyPizza*'.

```
Class: MeatyPizza
    SubClassOf: Pizza
```

   (b) **Equivalent class axioms**

   These axioms represent '*necessary and sufficient*' conditions. A '*complete*' and '*defined*' class is defined by employing the necessary and sufficient condition. A necessary and sufficient condition means that the former statement is true if and only if the latter is true. If an individual is a member of '*NamedClass*' then it must satisfy the conditions. If some individual satisfies the conditions, then the individual must be a member of '*NamedClass*'. For example, if an individual is a member of the class '*Pizza*' and it has at least one equipment that is a member of the class '*MeatTopping*' then these conditions are sufficient to determine that the individual must be a member of the class '*MeatyPizza*' (see below example).

```
Class: MeatyPizza
    EquivalentTo:
```

```
Pizza and (hasTopping some MeatToppingt)
```

(c) **Disjoint axioms**

Disjoint axioms represent additional '*necessary*' conditions. These necessary conditions are used for machine understanding that the listed concepts or classes are precisely different. OWL classes are assumed to overlap over by default. Disjoint axioms help to stop this overlapping. In case, two classes are defined disjoint classes, if a member belongs to one class, then that member cannot belong to another class. For example, the '*VegetableTopping*' and '*MeatTopping*' are completely different classes and these classes can be defined as disjoint classes. Moreover, any instance of the '*VegetableTopping*' class cannot belong to the '*MeatTopping*' class (see following example).

```
DisjointClasses:
    VegetableTopping, MeatTopping
```

(d) **Covering Axioms**

OWL has an open world assumption. This means that if we do not say anything explicitly, it does not mean that it is not true. For example, If a class (lets say a '*Spiciness*') is a super class of other classes ('*Hot*', '*Medium*' and '*Mild*'), it is not necessary for any individual of class '*Spiciness*' to be an individual of the subclasses. Such an individual could be simply loose between the subclasses. Unless it is defined in the ontology or knowledge-base that help the automated inference engine or reasoner for making inference that there can be no others, it is assumed that there can be. If such situation has to be avoided, we create an anonymous class which makes the subclasses (i.e. '*Hot*', '*Medium*' and '*Mild*') cover the full of class '*Spiciness*'.

```
Class: Spiciness
    EquivalentTo:
        (Hot or Medium or Mild)
Class: Hot
    SubClassOf: Spiciness
Class: Medium
    SubClassOf: Spiciness
```

```
Class: Mild
    SubClassOf: Spiciness
```

(e) **Closure Axioms**

A closure axiom in a '*Named*' class is consisted of a universal restriction that acts along a property to say that it can only be filled by the specified fillers. For example, an '*American*' pizza has only '*MozzarellaTopping*', '*PeperoniSausageTopping*' and '*TomatoTopping*' [84].

```
Class: American
    SubClassOf:
        hasTopping only (MozzarellaTopping or
        PeperoniSausageTopping or TomatoTopping)
```

3. **Inconsistent classes**

A class is deemed to be inconsistent if it cannot possibly have any instances [88]'. The inconsistent classes are detected by using an OWL reasoner. The OWL reasoners [89] are used for classification of the classes and checking the consistency of the knowledge-base. There can be a number of reasons for inconsistency, one reason can be subsumption and disjointness. For example, the '*Male*' and '*Female*' are two classes and these classes are defined as disjoint classes. If one creates the '*MixGender*' class and modelled it as a subclass of the '*Male*' and '*Female*' classes. It means that '*MixGender*' is a '*Male*' and a '*Female*'. More formally, all individuals that are members of the class '*MixGender*' are also (necessarily) members of the class '*Male*' and (necessarily) members of the class '*Female*'. This modelling leads to inconsistency of the '*MixGender*' class.

4.2.2 Semantic Web Rule Language (SWRL)

Semantic Web Rule Language (SWRL) is also another knowledge representation language that is used to define the rules. SWRL is compatible with OWL and it can be easily used by employing OWL classes and properties. OWL expresses the problem classification by using description logic and first order logic. First order logic does quantification over values, but not over functions and predicates. SWRL is an extension of OWL DL and OWL Lite and Unary/Binary Datalog RuleML sublanguages of the Rule

Markup Language [90]. It can express as a rule language. SWRL [6] presents semantic rules through that users can express certain logical relationships in a form suitable for machine processing. SWRL can be easily added into ontologies, and it can maintain intelligible semantics. It uses URIs to identify things that make it essentially compatible with RDF and OWL. The benefit of the Semantic Web Rule Language is that a rule is independent to the ontology. If some modification is required in the rule than it does not affect on the ontology structure.

An SWRL rule [50, 51] contains two parts, (1) antecedent part and (2) consequent part. The antecedent part of a rule is called body and the consequent part of a rule is called the head, and both body and head consist of conjunctions of some atoms. If all the atoms in the body are true, then the head must also be true. OWL classes and properties are used as functions or methods. The parameters of the method can be constants or variables or individuals. A question mark *?* is pre affixed to the variable name [50], that makes it distinguish from a constant of the SWRL rules. For example, If a player has won greater than or equal to 5 test matches, then he or she will be selected to play in next coming match.

```
Rule:
    Player(?p),  hasWon(?p, ?numberofMatchesPlayed),
    greaterThan(?numberofMatchesPlayed, 5)
     -> hasSelectedToPlay(?p, true)
```

The Protege, KAON2 and R2ML are some different editors, these infrastructures supports in managing of OWL-DL and SWRL syntax. The Pallet, Hoolet, Bossam, Fact++ and Racerpro are different OWL reasoners or rule engines, these help in processing of rules over SWRL-based syntax. These OWL reasoners are used to support in automatic reasoning over OWL-based syntax and SWRL rules. These reasoners are compatible with the editors, which support in editing, creating and development of OWL and SWRL syntax.

[6] http://www.w3.org/Submission/SWRL/

4.3 Conclusions

The domains of the Galatean model of classification are incorporated human represented knowledge. Humans are more flexible in the way knowledge is represented and communicated compared to machines, which have problems with interpreting ambiguity. The challenge is to see whether human expertise can be formally specified to remove ambiguity without losing the semantics of the human representation. The research questions are that (i) how OWL can validate the complex data structure of human psychology domains, (ii) how OWL can support in designing the paper manual specification in the machine centric form and (iii) how OWL can improve the knowledge-engineering process for decision support systems based on a psychological model.

In the next chapter, the general description of the Galatean model of classification based on OWL will be discussed. The concepts of different domains of the Galatean model of classification by using OWL classes, properties and restrictions will be illustrated.

Chapter 5

Specifications of galateas in OWL

5.1 Introduction

The effective description of the knowledge structure is a key factor in facilitating the semantic discovery of required knowledge. In order to overcome the limitations of syntactic representation of the knowledge, a flexible and expressive approach is required that describes the knowledge structure at a conceptual level. In this chapter, the specification of galateas in OWL is developed to describe the formal structures that will facilitate the use of logical reasoning over such description. The core purpose of this specification is to answer the following questions.

- What is a galatea?
- What are the essential characteristics of a knowledge structure?
- What are the features and data structure of the uncertain and other attributes?

The OWL Galatea model needs to capture and model the necessary knowledge relevant to the Galatean model of classification. This model is intended to provide a general framework for OWL Galatea model of classification that can be used as a base ontology to describe a range of its applications used in different fields of life. For this reason, the ontological framework is not bound to represent the concepts that are domain specific. The experts have suggested to creating ontology libraries, developing

widely usable base ontologies, specialising these concepts with respect to various relevant sub-domains and adding the domain specific extensions [91]. This thought guides to develop reusable ontologies. Therefore, the OWL Galatea specification is designed in this perspective, since it contains the base knowledge to represent conceptual framework of hierarchical structure, which can be used to create domain specific knowledge structure. The methodology adopted for the development of the Galatea specification is based on Semantic Web technologies.

5.2 Ontology development methodology

The term “methodology” indicates a procedure and process that is always utilised in the creation of a system. The IEEE [92] defines this terminology with these words “comprehensive integrated series of techniques or methods creating a general systems theory of how a class of thought-intensive work ought to be performed”. For the ontology development a number of techniques and methods are developed through the research community but there is no widely accepted methodology for ontology engineering [93]. In the following section, a brief overview of ontology design methodologies is going to be discussed.

5.2.1 Existing ontology engineering methodologies

This section presents a brief introduction on the methodologies available for the development of ontologies.

- **Uschold and King Method** [94] ideology is captured from the Enterprise Ontology[1] which is designed for enterprise modelling processes. This method proposes four activities for building an ontology: (1) identify the purpose of ontology, (2) ontology capture, (3) codification of knowledge and (4) integrating existing ontology and evaluation.

[1] http://www.aiai.ed.ac.uk/project/enterprise/enterprise/ontology.html

- **Cyc Method** [95] is the oldest method that has been mapped to many different ontologies. It is based on the experiences of building Cyc Ontology[2]. The limitation of this method is that it does not support the processes for requirements identification.

- **Toronto Virtual Enterprise (TOVE)** is also known as Gruninger and Fox methodology [96]. This approach is a first order logic approach that follows a stage based methodology where an informal framework of knowledge specification made at first and formalised at a later stage. The formalised concepts, roles, relations and axioms are used for the ontology development.

- **ONtologic Integration of Naive Sources (ONIONS)** strategy emphasises on the development of domain specific ontologies [97].

- **KACTUS** [98] approach is proposed by Bernaras et. al. This approach emphasise on the building of a particular application therefore, this techniques is known as an application-dependent strategy [99]. This method proposes three activities for the development phases of the ontology: (1) specification of the application, (2) preliminary design based on the top level ontological categories and (3) ontology refinement and structuring.

- **Methontology** [100] approach is also used for building ontologies. It has its roots on IEEE standard for the development of software life-cycle processes [97]. This methodology consist of four steps: (1) specification of the ontology, (2) conceptualisation of identifying concepts and building a conceptual model, (3) formalisation and implementation of conceptual model and (4) maintenance and corrections to the ontology when necessary.

- **On-To-Knowledge (OTK)** [96] is used to maintain ontology based knowledge management applications in the enterprise systems. This methodology was developed under the On-To-Knowledge project for the identification of goals that should be achieved by the knowledge management tools. The steps included in this strategy are: (1) kick-off for capturing the requirements and analysis of knowledge, (2) refinement of knowledge extraction and formalisation, (3) technology and user focused evaluation of the ontology and (4) application of the ontology for the intended use and maintenance.

[2] `http://www.cyc.com/`

- **SENSUS-based Method** [96] is a top-down approach for extracting domain specific ontologies at the large-scale. This methodology is used when ontology engineering process is linked to SENSUS ontology.[3]

5.2.1.1 Analysis of methodologies

In this section, a number of ontology engineering methodologies are discussed in detail. Each approach encapsulates some advantages and disadvantages, and has specific usage as far as the development of core ontologies is concerned. For example, Cyc and SENSUS methods are specifically intended to use with the core ontologies, therefore these approaches cannot be untilised for the development of a general purpose engineered ontology. The methodologies are also distinguished on the basis of the criteria of application dependence, which means that the strategy for building ontologies is dependent on the scope of its intended function. For instance, the Gruninger and Fox methodology, OTK, ONIONS and SENSUS-based approaches are application dependent. Therefore, these methodologies cannot be adopted for the development of a general purpose ontology.

The Uschold and King Method and Methontology are the application independent strategies. The Uschold and King methodologies are also called stage-based approaches where present stage completes the initial step and then the next stage of the development process start. On the other hand, the Methontology methods are evolving prototype based approaches. Jones et. al [101] presented his analysis that the stage based approaches are suitable when ontology engineering is required for a very specific purpose or application, and the evolving prototype approaches are better when the ontology is not targeted at a specific application or purpose beside that the requirements for ontology development are not very well defined or clear.

Galatean model of classification encapsulates a large number of knowledge domains. The horse bidding, psychodynamic, mental health risk assessment (i.e. GRiST) and logistics (i.e. ADVANCE) are the core applications of the said model (see details in

[3] `http://www.isi.edu/natural-language/projects/ONTOLOGIES.html`

Chapter 2). Therefore, the stage based approaches are not suitable for ontology engineering of the Galatean model due to their limited support for a single or specific application. Thus, evolution-based prototype approaches are better for the proposed model because such methodologies are not dependent on a specific application or purpose. Thus, Methontology is the most suitable approach for building the ontological framework for the Galatean model of classification and it can also provide support for the development of ontologies in its diverse range of knowledge domains.

5.2.2 Design Process

Methontology design process is selected for the development of OWL Galatea model after a comprehensive discussion on the present strategies presented in the aforementioned section. This method was developed in an artificial intelligence laboratory at the Polytechnic University by Fernandz in 1997 [102] to create ontologies from scratch and reuse existing ones. This methodology design process consists of some ontology development phases or stages such as: specification, conceptualisation, formalisation, implementation, evaluation and maintenance.

5.2.2.1 Specification

This phase is used to identify the purpose of developing the ontology. The main reason for the development of OWL Galatea ontology is to provide a general description framework for its decision domains, understanding about the psychological model capabilities and resources such as: uncertain attributes, transformation phases and classification mechanism. The purpose of this phase is to develop the conceptual model based on psychological theory to construct the knowledge in an informal way.

5.2.2.2 Conceptualisation

This phase of the design process deals with the conceptualisation of domain related concepts or terms. A semi-formal specification is designed based on the graphical representations that can be easily understood by the human experts involved in the ontology engineering process. The activities involve in this phase are: (1) identification

of the relevant terms and their semantics, (2) creation of concepts and their taxonomies in the ontology, (3) building the properties and their relationships, and (4) modelling axioms by analysing the knowledge domain. In the case of OWL Galatea specification, all the concepts, properties and relations are defined in the model are general. For instance, the concept ”Node” is further divided (such as concept node and datum node), the question, filter question (domain specific questions), value-mg list (domain specific lists) and population (can be a patient, transporter, horse bidder, etc). The question, filter question, value-mg list and population are attributes in the present XML series of knowledge trees (i.e. SST, ST, RIT, QT and CAT) (see details in Chapter 3), but these terms are analysed and thereby, aimed to be designed as entities. The Galatean model applications have hierarchical knowledge structures. Therefore, the relationships among the nodes are defined by using some explicitly defined properties.

5.2.2.3 Implementation or formalisation

This phase includes the formalisation of conceptual framework by using the concepts and properties decided in the final step. This step involves the development of a formal model by using ontology editors such as Protégé. The Web Ontology Language (OWL) Reasoner [103] was used with the Protégé editor to check the consistency of the ontology being developed.

5.2.2.4 Evaluation

This phase ensures that there is clarity and comprehensiveness in the designed ontology. The evaluation phase also guarantees the completeness of information.

5.2.2.5 Maintenance

Updating, modifying and correcting activities are involved in this phase. The OWL Galatea specification is based on Methontology, therefore it allows the additions, deletions and modifications in the conceptual framework. The said model was developed through an iterative way; therefore the final version of the model is published or arrived

at after a number of modifications.

In the next section, the OWL Galatean specification model is discussed in detail.

5.3 The OWL Galatea model

The galateas are core components of the Galatean hierarchy and these are actually decision classes. The objects of the decision classes are created during the classification process, where their uncertain attributes (i.e. MGs) are initialised. The knowledge passes to various transformation phases before this classification process. Various XML trees are utilised for maintaining the Galatean nodes, which have various uncertainties and question contents for various populations within a domain (see details in Chapter 3). The Galatean nodes in various XML knowledge trees use the various attributes for maintaining their data structure (see details in Chapter 3 and Section 3.3.3). The problem is that some knowledge cannot represent actual or correct semantics in the knowledge-base by using XML representation. For example, the '*population*' attribute cannot represent correct semantics. This concept (i.e. '*population*') has its own characteristics like: age, gender, marital status, etc.

Some attributes (i.e. '*generic*', '*generic-type*', '*generic-datum*') are used for maintaining repeating concepts or datum nodes. Decisions for initialising the uncertain attributes (i.e. '*RI*') of some nodes depends on the attribute(s) of concepts. If the concepts have the '*generic-type*' attribute then the contents (i.e. '*g*', '*gd*') of this attribute are evaluated and according to the result, the RIs of their sub nodes are initialised. The references of the associated concepts and datum nodes are stored in the '*generic*' and '*generic-datum*' attributes in the form of the path, where their complete definition is located (see details in Chapter 3). It means these attributes are just keeping syntax based information and similar knowledge (or attribute) is repeated again and again in various concepts and datum nodes.

The '*value-mg*' attribute keeps a set of pairs and each pair has a datum value and an associated MG. The user given data is compared with the datum values of the value-mg list and the MG is produced by comparing the object value with the graph of MGs and values, but the data is stored in this attribute in the form of a static linear list. Some algorithms and processes are used to manipulate the data of such lists with the support of a text-based document. This approach makes it difficult to know what type of data a value-mg list is actually holding. For exploring the data type of the value-mg list another attribute (i.e. '*values*') is used which indicates its data type (i.e. '*scale*', '*nominal*', '*real*', etc.) in the associated datum nodes (see details in Chapter 3). In fact, this attribute belongs to the value-mg list. It means this concept (i.e. value-mg) also cannot possess its correct semantics and relationships as an attribute. Moreover, it is difficult to check and confirm that the user given data (for initialising '*MGs*' of datum nodes) and expert given data (for initialising '*RIs*' of complete Galatean hierarchy) are within range by employing a specification document. The OWL Galatea model is designed in this context so that it can tackle such problems or issues by defining the correct semantics of the knowledge.

The OWL Galatea model is a generic model that is based on Galatean model of classification. The specifications of galatea in OWL is modelled to cover the conceptualisation of the Galatean model of classification, which uses to capture human expertise. The OWL Galatea model is developed for providing machine-centric judgments and representations for the knowledge domains of the Galatean model of classification without losing their psychological semantics. This model is designed in this perspective that it can help in making high level decisions on uncertain knowledge or information, and it can also support in consistency checking of the knowledge-bases by using the OWL reasoner. The OWL Galatea model consists of four main classes (1) Node, (2) Question, (3) Assessment-Type and (4) Value-Mg-Pair (see the graphical representations of the OWL Galatea model in Figure 5.1). The details of each class is going to discuss next by employing different illustrations.

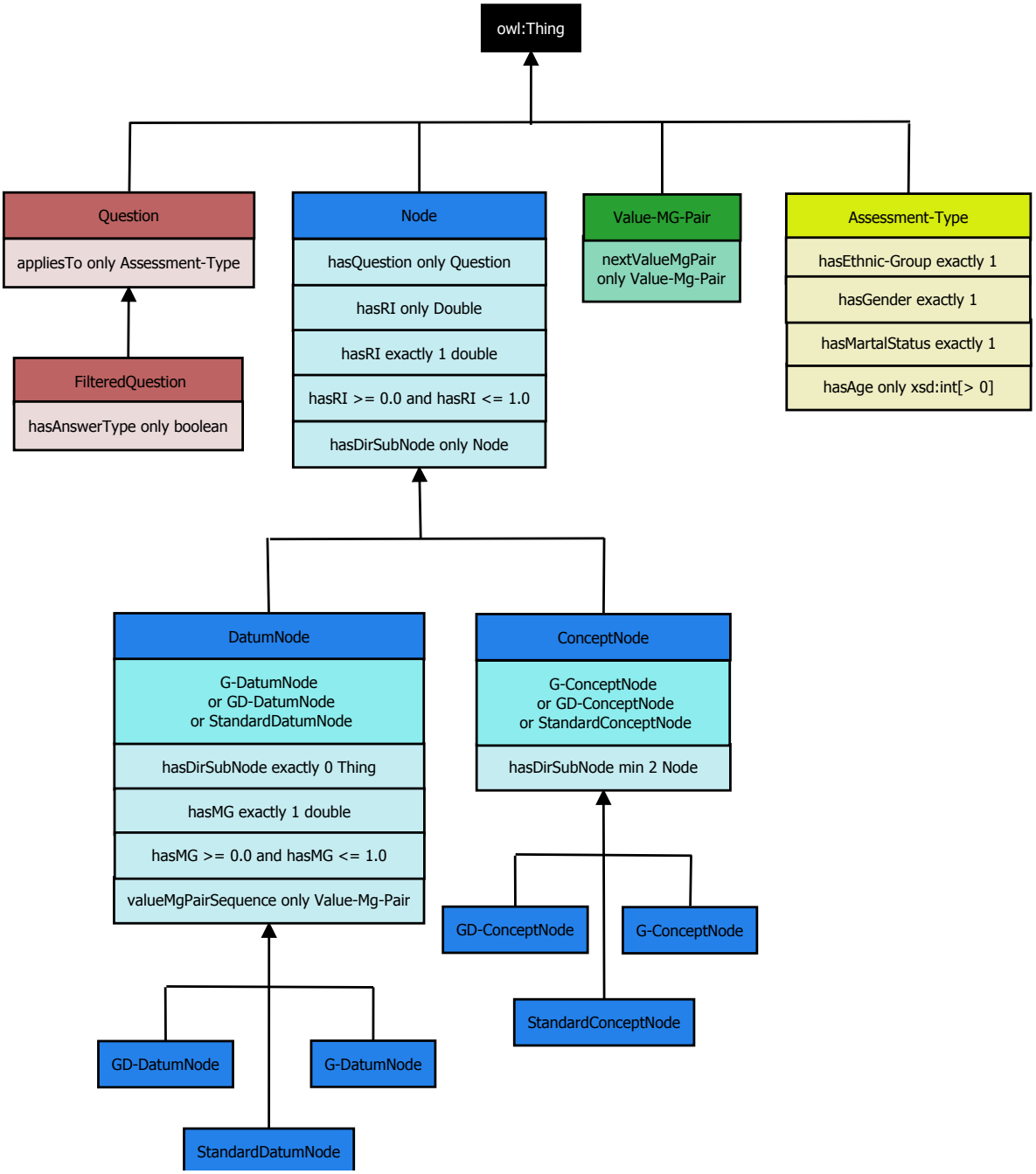

FIGURE 5.1: The architecture of the OWL Galatea model. The arrows of the edges are indicated is-a or sub and super class relationships. The Thing is a predefined class. The Node, Question, Assessment-Type and Value-Mg-Pair are core classes of the OWL Galatea model that are depicted in different background colours. The names of these classes are represented in specified dark colours and their constraints and relationships are represented in specified light colours. The Question class is depicted in red colour, and the Node class and its subclasses are represented in blue colour. The Value-Mg-Pair class is depicted in green colour and the Assessment-type class is represented in yellow colour.

5.3.1 Node

The '*Node*' class is an all-important class of the OWL Galatea model in that the concepts and datum nodes (see details in Chapter 3) of a domain constitute logical hierarchies by defining them as subclasses of it. They form their internal structure in the shape of a logical tree and keep the basic information related to them. The relationships are maintained between nodes by employing '*has-a*' relationship. For example, the '*eyes*' and '*eye-movements*' are two nodes, and a person's eyes have some movements through which his mental health situation or thinking can be judged. The '*eyes*' is a super node of the '*eye-movements*' node. The '*hasDirSubNode*' (OWL object property) is used for maintaining hierarchical tree structure. The following constraint is defined in the '*Node*' class.

```
Class: Node
    SubClassOf:
        hasDirSubNode only Node
```

The nodes (see details in Chapter 3) follow the above constraint by extending the '*Node*' class. The OWL Galatea model is designed for human psychological domains, which have complex mutative data structure. For example, '*A*', '*B*' and '*C*' are nodes and they are modelled as subclasses of the '*Node*' class. The '*A*' is a super node of '*B*' and '*C*' nodes. A node keeps initially some basic constraints in the source agent. The constraints and relationships are maintained through the universal restriction and an OWL closure axiom that act along with the '*hasDirSubNode*' property to say that it can only be filled by the specified union of fillers (i.e. '*B*' or '*C*') in the SST agent (see Figure 5.2).

This indicates that the '*A*' node has possible '*B*' and '*C*' sub nodes for reference.

```
Class: A
    SubClassOf:
        hasDirSubNode only (B or C),
        Node
```

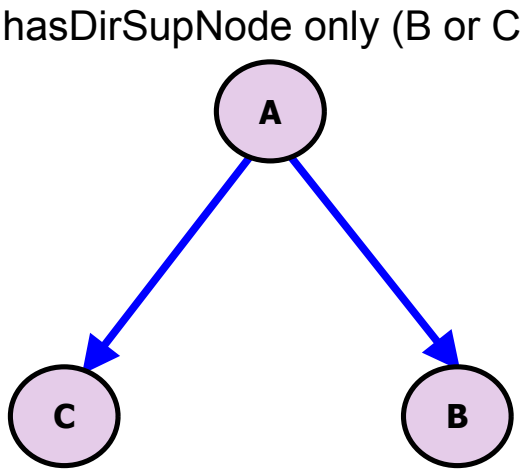

FIGURE 5.2: An example of a basic tree structure with universal restriction in the SST.

The reason for this restriction is that the SST retains knowledge for populations belong to all sectors of society within a domain. An ST agent keeps information for a particular population. A particular population can respond to particular information. There is a possibility to remove or eliminate the undesired nodes from the SST agent to the derived agent (i.e. ST). In this case, the existential restriction is utilised for the '*hasDirSubNode*' relationships in the super nodes as the discrete necessary condition for each sub node (see below example) in the SST knowledge-base, then the elimination process of the unwanted sub nodes will become difficult for the next derived agent (i.e. ST).

```
Class: A
    SubClassOf:
        hasDirSubNode some B,
        hasDirSubNode some C,
        Node
```

The existential restrictions are added only in the nodes are suitable for a specified population in the specified ST agent (see details in Chapter 7). The SST agent is imported and its nodes are extended for generating the ST agents. For example, the '*ChildA*' node is an extension of the '*A*' node of the SST in the ST. The '*ChildA*' node follows all constraints defined in the '*A*' node (see below example).

```
Class: ChildA
    SubClassOf:
        hasDirSubNode some B,
        hasDirSubNode some C,
        A
```

These constraints demonstrate the actual extension of the nodes and their constraints in the next state(s). However, this approach is not used throughout the thesis for demonstrating the extension of the knowledge. This is because, it may increase the complexity in depicting the knowledge in graphical and textual form. When the knowledge is transformed or extended then it will be assumed that the current state of the knowledge is the extension of the previous ones.

A node in OWL is a combination of a super class and one or more subclasses. For example, the '*C*' node consists of a super class (i.e. '*C*') and a subclass (i.e. '*c1*' and '*c2*') (see Figure 5.3).

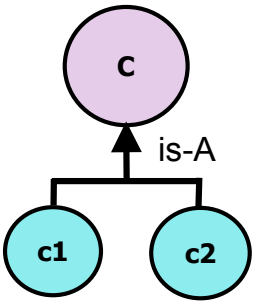

FIGURE 5.3: An example of an OWL node. The C is a super class of the c1 and c2.

The subclasses of a node are used to keep the reference of their direct super nodes or contexts and they also hold the specific information. General information is defined at the node (i.e. super class) level. Contexts related information is defined at subclass level. An instance of a node should inherit generic information from the super class and it should follow the specialise constraints of a subclass that is keeping an identical reference of a super node. The '*C*' node is a direct sub node of the '*A*' and '*X*' nodes. The '*c1*' class is kept the reference of the '*A*' super node and the '*c2*' class is kept the

reference of the '*X*' super node. The following necessary conditions are examples of them in the SST knowledge-base (see Figure 5.4).

```
Class: c1
    SubClassOf:
        hasDirSubNode only A,
        C

Class: c2
    SubClassOf:
        hasDirSubNode only X,
        C
```

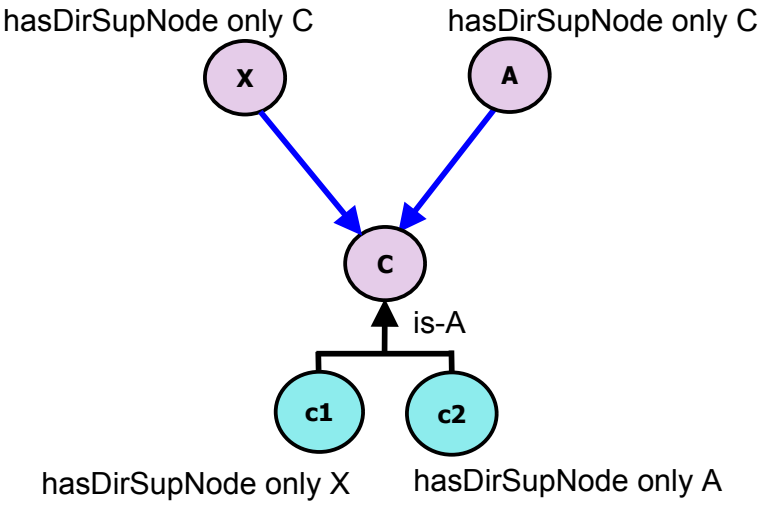

FIGURE 5.4: An example of a tree structure in the SST. The subclass or is-A relationship is depicted in blue colour edges and black colour edges are represented has-A relationship.

The other information related to a node can be inferred by the exploration of the constraints defined for a particular node. The nodes also make relationships with related question(s). Following constraint is defined in the '*Node*' class via using the '*hasQuestion*' property.

```
Class: Node
    SubClassOf:
```

```
        hasQuestion only Question
```

The eventual concept attributed to an individual cue depends on its RI (Relative Influence) in the knowledge-base. The RIs are weightings of the nodes those are given by the experts of the domains. The '*hasRI*' property is used to represent an uncertainty attribute of the Galatean model of classification. Each node must have exactly one RI. The data type of the constants is used in the '*hasRI*' relations must be '*double*'. The range and cardinality constraints of the '*hasRI*' attribute are also defined in the '*Node*' class.

```
Class: Node
    SubClassOf:
     (hasRI only double[>= 0.0]) and (hasRI only double[<= 1.0]),
      hasRI exactly 1
```

The concept and datum nodes are two different galateas. These are modelled as subclasses of the '*Node*' class, (1) '*ConceptNode*' and (2) '*DatumNode*'. Following definition is defined by using the covering axiom (see details in Chapter 4) in the Node class.

```
Class: Node
    EquivalentTo:
        ConceptNode or DatumNode
```

5.3.1.1 The Concept Node ('*ConceptNode*')

The '*ConceptNode*' is a subclass of the '*Node*' class. The rationale of the '*ConceptNode*' class is that it is a super node that has at least two direct sub nodes. Following constraints are defined in the '*ConceptNode*' class.

```
Class: ConceptNode
    SubClassOf:
        hasDirSubNode min 2 Node,
        Node
```

OWL Classes are assumed to overlap each other. An instance of the '*Node*' class cannot belong to the '*ConceptNode*' and '*DatumNode*' classes at the same time. Therefore, the '*ConceptNode*' class is defined as a disjoint class of the '*DatumNode*' class.

```
Class: ConceptNode
    DisjointClasses:
        DatumNode
```

The concept nodes (see details in Chapter 3) are further divided into three concepts, generic concepts, generic distinct concepts and standard concepts and these are modelled as subclasses of the '*ConceptNode*' class, (1) '*G-ConceptNode*', (2) '*GD-ConceptNode*' and (3) '*StandardConceptNode*'. Following definition is defined by using the covering axiom in the '*ConceptNode*' class.

```
Class: ConceptNode
    EquivalentTo:
        G-ConceptNode or GD-ConceptNode or StandardConceptNode
```

1. **Generic Concept Node ('*G-ConceptNode*')**

   The '*G-ConceptNode*' is a subclass of the '*ConceptNode*' class. It represents the generic concepts. The generic concepts are considered as repeated concepts and they are repeated in different contexts. The notion of a '*G-ConceptNode*' is that its sub nodes must have fixed '*RIs*' or weightings in different contexts wherever it occurs as an ancestor node (see details in Chapter 7 and in Section 7.3).

   The '*G-ConceptNode*', '*GD-ConceptNode*' and '*StandardConceptNode*' are completely different classes and these classes are defined as disjoint classes (see below constraint). Moreover, any instance of the '*G-ConceptNode*' class cannot belong to the '*GD-ConceptNode*' and '*StandardConceptNode*' classes.

   ```
   Class: G-ConceptNode
       DisjointClasses:
           GD-ConceptNode, StandardConceptNode
   ```

2. **Generic Distinct Concept Node (‘*GD-ConceptNode*’)**

   The ‘*GD-ConceptNode*’ is also a subclass of the ‘*ConceptNode*’ class. It represents the generic distinct concepts. The generic distinct concepts are also considered as repeated concepts and they replicate in different locations or contexts. The rationale of a ‘*GD-ConceptNode*’ is that the ‘*RIs*’ of its sub nodes can be varied in contexts to contexts. In this case, a ‘*G-ConceptNode*’ is a sub node of a ‘*GD-ConceptNode*’ than all the sub nodes under that ‘*G-ConceptNode*’ must have fixed ‘*RIs*’, but the ‘*RIs*’ of the intermediate ‘*G-ConceptNode*’ can be varied in different contexts.

3. **Standard Concept Node (‘*StandardConceptNode*’)**

   The ‘*StandardConceptNode*’ is also a subclass of the ‘*ConceptNode*’ class. It represents the nodes that do not repeat in different contexts or locations.

5.3.1.2 The Datum Node (‘*DatumNode*’)

A datum node (see details in Chapter 3) is a leaf node. The rationale of the datum node is that it must not have any sub node. This idea is modelled in the ‘*DatumNode*’ class by defining following constraints.

```
Class: DatumNode
    SubClassOf:
        hasDirSubNode exactly 0 Thing,
        Node
```

The Galatean model of classification represents uncertainty in terms of set membership grades. An associated ‘*MG*’ is measured from the selected or given datum value of an end user. The calculated value becomes the ‘*MG*’ of a particular datum node. The ‘*hasMG*’ property is used to represent the second uncertainty attribute of the Galatean model of classification. The range and cardinality constraints via using the ‘*hasMG*’ property are defined in the ‘*DatumNode*’ class.

```
Class: DatumNode
    SubClassOf:
```

```
(hasMG only double[>= 0.0]) and (hasMG only double[<= 1.0]),
hasMG exactly 1,
Node
```

A datum node keeps the value-mg list and each list maintains its sequence through some pairs. Each pair of the value-mg list is a combination of a datum value and an associated MG. The subclasses of the '*Value-Mg-Pair*' class are employed as elements or pairs of the list. The first pair of the value-mg list makes relationship to its associated datum node via the '*valueMgPairSequence*' property (see details in Section 5.3.4). Following constraints are defined for this purpose in the '*DatumNode*' class.

```
Class: DatumNode
    SubClassOf:
        valueMgPairSequence only Value-Mg-Pair,
        Node
```

Another attribute associated with the value-mg list is the data type of its datum values. These data types (see details in Chapter 3 and in Section 3.3.3.14) support in the determination of the datum values of the associated value-mg list. The '*hasValuesType*' property is inaugurated for this purpose. The constraint having the relationships of the '*hasValuesType*' property helps in decision making, while the '*MGs*' are calculated for the associated datum nodes. A user defined data type is associated to a particular datum node during the knowledge engineering of the knowledge-base of the assessment tool. Therefore, following constraint is defined in the '*DatumNode*' class.

```
Class: DatumNode
    SubClassOf:
        hasValuesType only string,
        Node
```

A particular datum node can add a further constraint regarding the '*hasValuesType*' property, in such case that datum node has user defined type of data (i.e. '*nominal*'). The following example is demonstrates such constraint in the '*gen-gender*' (a datum node belongs to mental health risk domain) node.

```
Class: gen-gender
    SubClassOf:
        hasValuesType some "nominal",
        DatumNode
```

The datum nodes are further divided into generic datum nodes, generic distinct datum nodes and standard datum nodes (see details in Chapter 3). These ideas are modelled as subclasses of the '*DatumNode*' class, (1) '*G-DatumNode*', (2) '*GD-DatumNode*' and (3) '*StandardDatumNode*'. Following definition is defined by employing covering axiom in the '*DatumNode*' class.

```
Class: DatumNode
    EquivalentTo:
        G-DatumNode or GD-DatumNode or StandardDatumNode
```

1. **Generic Datum Node ('*G-DatumNode*')**

   The '*G-DatumNode*' is a subclass of the '*DatumNode*' that represents the generic datum node. A generic datum node is repeated in different contexts. It has a single value-mg list in different contexts. A '*G-DatumNode*' must keep a fixed value-mg list in different contexts, as the sub nodes of a '*G-ConceptNode*' keep fixed '*RIs*' in different contexts.

   The '*G-DatumNode*', '*GD-DatumNode*' and '*StandardDatumNode*' are completely different classes and these classes are defined as disjoint classes (see below constraint). Moreover, any instance of the '*G-DatumNode*' class cannot belong to the '*GD-DatumNode*' and '*StandardDatumNode*' classes.

   ```
   Class: G-DatumNode
       DisjointClasses:
           StandardDatumNode, GD-DatumNode
   ```

2. **Generic Distinct Datum Node ('*GD-DatumNode*')**

   The '*GD-DatumNode*' is also a subclass of the '*DatumNode*'. It represents the generic distinct datum nodes, which are repeated in different contexts. The value-mg list of such datum nodes can be varied in context to context.

3. **Standard Datum Node (‘*StandardDatumNode*’)**

   The ‘*StandardDatumNode*’ is also a subclass of the ‘*DatumNode*’ class. It represents the datum nodes, that do not repeat in different contexts or locations.

5.3.2 Question

The ‘*Question*’ is also a primary class of the OWL Galatea model. The idea of this class is taken from the ‘*question*’ attribute (see details in Chapter 3) of the XML knowledge trees. The content is a basic asset of a question. The concepts and datum nodes have associations with the contents of the questions and the contents of the questions are applied to some particular populations. The ‘*Question*’ class is extended while the management and development of the knowledge-bases of the assessment tools and its instances make relationships with the contents by using ‘*hasContent*’ property (see details in Chapter 6). Such instances also make relationships to the particular instances of the ‘*Assessment-Type*’ class (see details in Section 5.3.3) by employing the ‘*appliesTo*’ property. Therefore, following generic constraints are defined in the ‘*Question*’ class.

```
Class: Question
    SubClassOf:
        appliesTo only Assessment-Type,
        hasContent only string
```

5.3.2.1 FilteredQuestion

The assessment tools belong to the Galatean model of classification have some filtered questions (see details in Chapter 3). The concept of the filtered questions are modelled as the ‘*FilteredQuestion*’ class, which is defined a subclass of the ‘*Question*’ class. At the beginning of the assessment only filtered questions are popped up. In this case, if an assessment type answers a question with ‘*YES*’ then all the sub concepts and datum nodes inside that concept (having a filtered question) will be expanded, otherwise one moves to the next sibling concept having a filtered question. For example, ‘self harm specific’ (‘*sh-specific*’) node has a filtered question’s content like: ‘Do you ever have

any thoughts about harming yourself?' in a mental health screening tool. In this case, the assessment type's given answer is '*YES*' then the sub nodes (e.g. '*self-harm-curr-sit-behav*' and '*sh-past-curr-ep*') of '*sh-specific*' will be expended (see a graphical representation of the example discussed here in Figure 5.5).

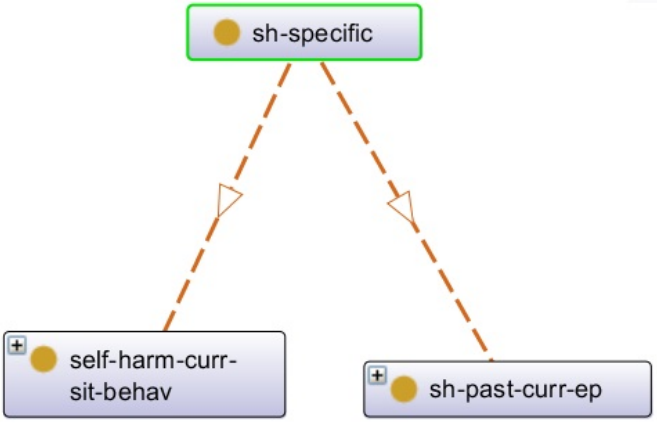

FIGURE 5.5: An example of the sh-specific concept that having a filtered question.

The answers type of the filtered question can only '*YES*' and '*No*'. The '*YES*' represents '*true*' and the '*No*' renders the '*false*'. Therefore, following constraint is defined in this class.

```
Class: FilteredQuestion
    SubClassOf:
        hasDatumValue only boolean,
        Question
```

5.3.3 Assessment-Type

The '*Assessment-Type*' class represents different types of the populations (see details in Chapter 3) of the society. This class is extended in the knowledge-bases of the assessment tools of the Galatean model of classification by defining the constraints and relationships for the knowledge of their specified end users (see details in Chapter 6). The generic constraints are defined in this class for example, an assessment type must have one ethnic group, marital status and gender type, and his or her age must be greater than zero.

```
Class: Assessment-Type
    SubClassOf:
        hasEthnic-Group exactly 1,
        hasGender exactly 1,
        hasMaritalStatus exactly 1,
        hasAge only xsd:int[>= 0]
```

The '*hasEthnic-Group*', '*hasGender*', '*hasMaritalStatus*' and '*hasAge*' are the sub properties of the '*hasDatumValue*' property. The '*hasDatumValue*' is a data type property, that is used in the value-mg lists (see details in Section 5.3.4) for keeping datum values in their pairs. This property is also used in making relationships with the assessment type's provided and selected data during one's assessment and classification of the knowledge hierarchy. The assessment type given data is then compared to the datum values of the pairs of the associated value-mg list and an MG is calculated or measured and attached to the associated datum nodes.

5.3.4 Value-Mg-Pair

The '*Value-Mg-Pair*' is also an important class of the OWL Galatea model. This class is modelled on the idea of the '*value-mg*' list attribute of the XML trees (see details in Chapter 3). The membership sets are associated only to datum parts of the galatea hierarchy (see details in Section 5.3.1.2). The datum components of the galatean tree are matched with their associated cues to produce a membership grade that represents the risks (in such case of GRiST tool) or decisions (in such case of ADVANCE tool) contribution of that particular cue value.

OWL support binary relationships naturally by employing a property to link an individual to another individual or value (literal). However, a pattern or format needs to adopt for creating the N-ary relationships, where an object or individual can make relationship with more than one objects or values. For example, the '*Christine*' (an individual) has a diagnosis relationship with another individual (i.e. '*_:Diagnosis_Relation_1*') as its value.

The '*_Diagnosis_Relation_1*' has a diagnosis value (i.e. '*Breast_Tumor_Christine*') and has a diagnosis probability (i.e. '*High*') (see Figure 5.6[4]).

FIGURE 5.6: A hypothetical example of a pattern of N-ary relationship.

The datum values and associated '*MGs*' are managed by adopting a pattern of N-ary relationship in the pairs. Each pair of the value-mg list adopts the identical N-ary relationships for keeping the references of the associated MGs and datum values. Therefore, following constraints are defined in the '*Value-Mg-Pair*' class.

```
Class: Value-Mg-Pair
    SubClassOf:
        hasMG only double,
        hasDatumValues only Literal
```

The pairs of a value-mg list are arranged to make a sequential list. The order or sequence or relation of the pairs or elements is designed via the '*nextValueMgPair*' property relationships between the subclasses of the '*Value-Mg-Pair*' class (see Figure 5.7).

The '*nextValueMgPair*' property (an object property and the domain and range of this property is the '*Value-Mg-Pair*' class) is used for this purpose. Following constraint is defined in the '*Value-Mg-Pair*' class.

[4] `http://www.w3.org/TR/swbp-n-aryRelations/`

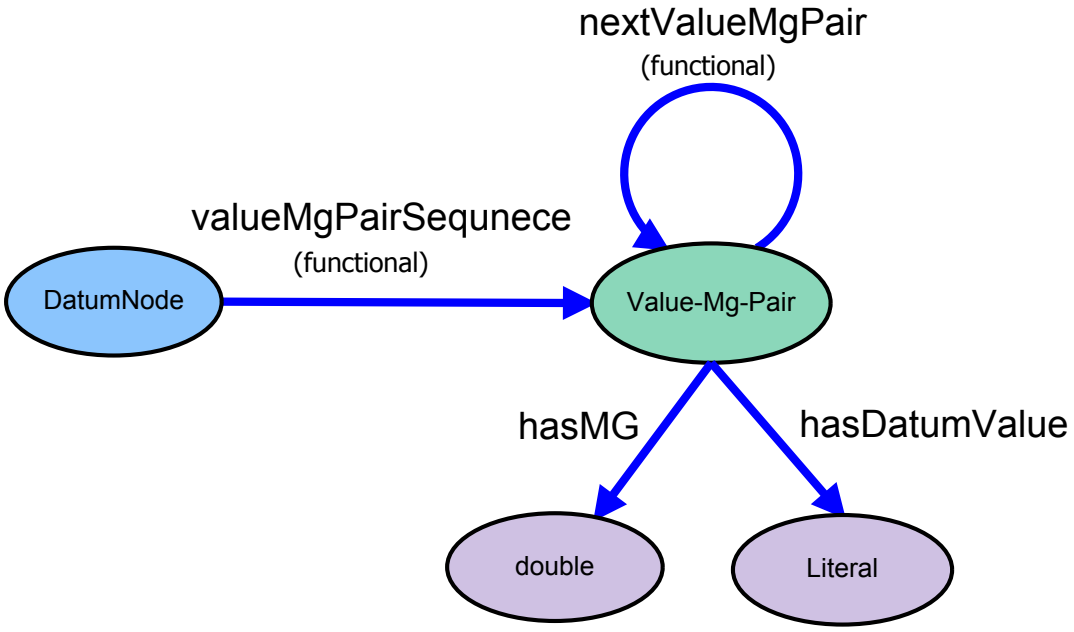

FIGURE 5.7: An abstract view of value-mg list.

```
Class: Value-Mg-Pair
    SubClassOf:
        nextValueMgPair only Value-Mg-Pair
```

Figure 5.8 is represented an OWL version of a value-mg list for the '*suic-fam-hist*' datum node in the mental health domain based on Galatean model of classification. Every pair has relation to the next pair, but the last or final pair has exactly 0 numbers of cardinalities to the '*Thing*' class (a predefined OWL class) in the list. It indicates that the final pair has no more pair or it is the end of the sequence or list.

Quantifying the galatea requires the experts to provide it with datum values that enable membership grades to be calculated. The associated '*MGs*' are calculated or measured through Semantic Web Rule Language (SWRL) from the assessment type given or selected data for a particular datum node in the CAT agent. The SWRL rule is consisted of some predicates and that can include OWL classes and properties. The '*suic-fam-hist(?N), hasDatumValue(?N, "Yes"), hasMG(?N, 1.0)*' are examples of OWL classes and properties. The arguments of the rule can be OWL individuals or data values, or variables (i.e. ?N or "Yes" or 1.0). All variables in SWRL, are treated as

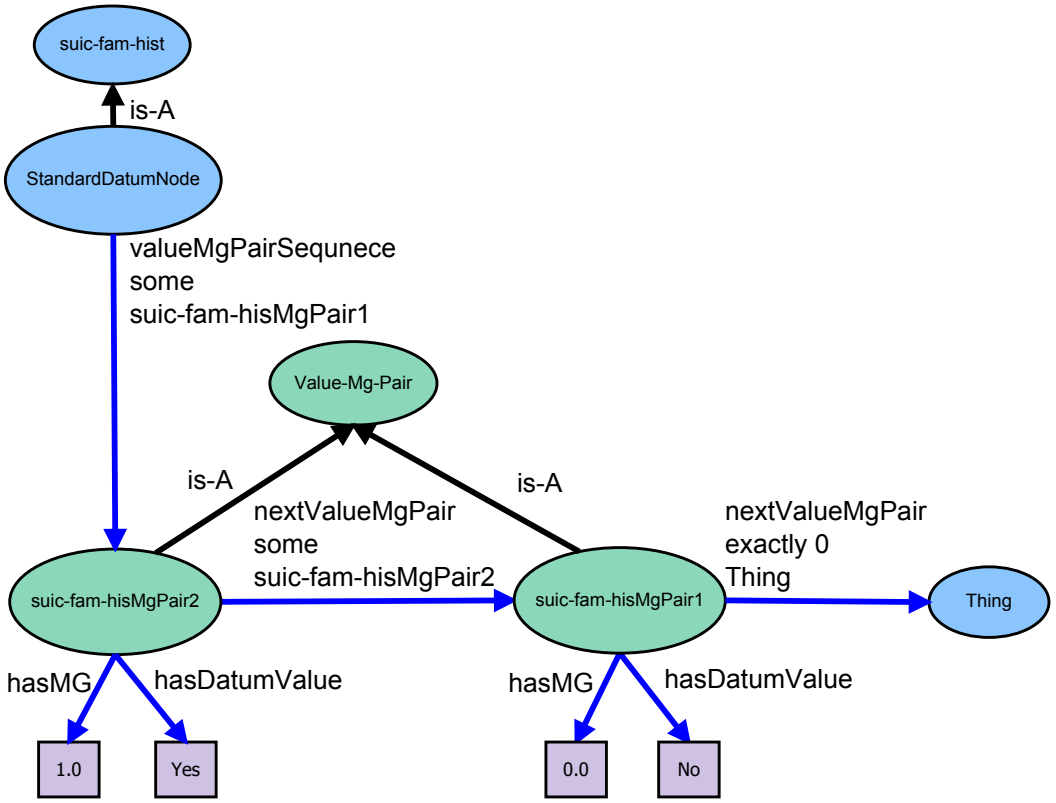

FIGURE 5.8: An example of value-mg list in the OWL version of SST ontology of GRiST tool.

universally quantified with their scope limited to a given rule. The -> symbol is used as an implication operator. The following rule is an example of the SWRL rule.

```
Rule:
    suic-fam-his(?N), hasDatumValue(?N, "Yes")
     -> hasMG(?N, 1.0)
```

The meaning of the above rule is that if any instance of the '*suic-fam-his*' class is initialised with '*Yes*' (datum value) than 1.0 '*MG*' is initialised by using the '*hasMG*' property. The datum values have different data types and associated '*MGs*' are calculated by determining them. Different SWRL rules are used for measuring the '*MGs*' and these are demonstrated in the next section.

5.3.4.1 The MGs calculations from the value-mg lists having numerical data types

By analyzing the data types (i.e. '*scale*', '*integer*', '*real*', '*date-year*', '*date-month*', '*date-week*' and '*date-day*') of the knowledge domains of the Galatean model of classification, it is uncovered that these data types actually represent numerical data. The "*scale*', '*integer*', '*real*', '*date-year*', '*date-month*', '*date-week*' and '*date-day*' data types represent that the associated value-mg lists have the data in the form of numbers. SWRL rules are defined for such value-mg lists in identical ways for associated datum nodes. For example, the $((0\ 0)(10\ 1))$ value-mg list having '*scale*' types of data. Figure 5.9 represents this value-mg list in a graphical form.

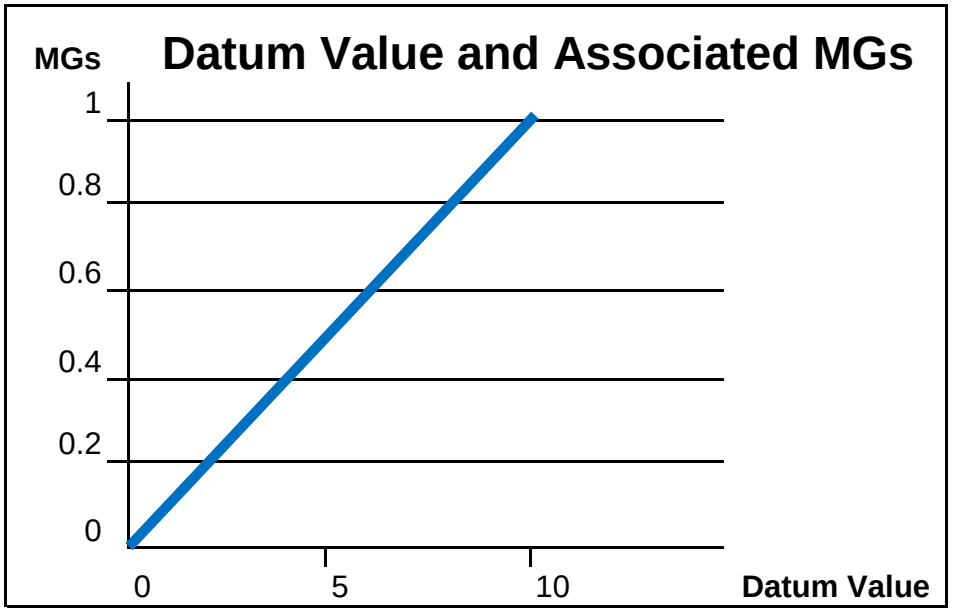

FIGURE 5.9: A graphical representation of a value-mg list having numerical data.

The '*MGs*' for the specified datum node are calculated through some mathematical formula. The range of the numeric data type (i.e. '*scale*') is from 0 to 10 and the range of the associated '*MGs*' is from 0 to 1 in the $((0\ 0)(10\ 1))$ value-mg list. Following mathematical equation is used for calculating the '*MG*'.

```
y = mx + b
```

The y variable is used in the equation to represent the '*MGs*', m variable represent the slope, x identifier is used for taking run-time data from the assessment type and b variable is the y intercept. The following two equations demonstrate the mathematical calculations of the $y = mx + b$ equation.

m = $\frac{y2-y1}{x2-x1}$

b = y - $\frac{y2-y1}{x2-x1}$ x

Following rule is utilised for the implementation of the above mathematical equations for the numeric types of data.

```
Rule:
    DatumNode(?N),
    valueMgPairSequence(?N, ?Npair1),
    nextValueMgPair(?Npair1, ?Npair2),
    hasDatumValue(?Npair1, ?y1),
    hasMG(?Npair1, ?x1),
    hasDatumValue(?Npair2, ?y2),
    hasMG(?Npair2, ?x2),
    hasDatumValue(?N, ?x),
    greaterThanOrEqual(?x, ?y1),
    lessThanOrEqual(?x, ?y2),
    subtract(?Xv, ?x2, ?x1),
    subtract(?Yv, ?y2, ?y1),
    divide(?m, ?Yv, ?Xv),
    multiply(?y, ?m, ?x2),
    subtract(?b, ?y2, ?y),
    multiply(?mx, ?m, ?x),
    add(?mg, ?mx, ?b) ->
    hasMG(?N, ?mg)
```

Explanation of each predicate or method or atom of the above rule is given below.

- `DatumNode(?N)`, the instance of the '*DatumNode*' class binds with the N variable.
- `valueMgPairSequence(?N, ?Npair1)`, it indicates that the instance of the '*DatumNode*' class has a relationship with the instance of the '*Value-Mg-Pair*' class. The instance of the '*Value-Mg-Pair*' class ties with the $Npair1$ variable.
- `nextValueMgPair(?Npair1, ?Npair2)`, this predicate shows the relationship between the two instances of the '*value-Mg-Pair*' class.
- `hasDatumValue(?Npair1, ?y1)`, this predicate indicates that the first pair has a datum value, which binds with the $y1$ variable.
- `hasMG(?Npair1, ?x1)`, this predicate indicates that the first pair has a MG, which ties with the $x1$ variable.
- `hasDatumValue(?Npair2, ?y2)`, this predicate indicates that the next pair has a datum value, which binds with the $y2$ variable.
- `hasMG(?Npair2, ?x2)`, this predicate indicates that the next pair has a MG, which ties with the $x2$ variable.
- `hasDatumValue(?N, ?x)`, the x variable holds the data given by the assessment type.
- `greaterThanOrEqual(?x, ?y1)`, the '*greaterThanOrEqual*' is a built-in method which compares the user given data to the datum value (i.e. 0) of the first pair of the value-mg list.
- `lessThanOrEqual(?x, ?y2)`, the '*lessThanOrEqual*' is a built-in method which compares the user given data to the datum value (i.e. 10) of the second pair of the value-mg list.
- `subtract(?Xv, ?x2, ?x1)`, it calculates the difference between two numbers or quantities (i.e. $x1$ and $x2$). The resultant value stores into the Xv variable (i.e. $10 - 0 = 10$).
- `subtract(?Yv, ?y2, ?y1)`, it calculates the difference between two numbers or quantities (i.e. $y1$ and $y2$). The resultant value saves into the Yv variable (i.e. $1 - 0 = 1$).

- `divide(?m, ?Yv, ?Xv)`, it calculates the slope by dividing the two quantities and stores resultant value into the m variable (i.e. $1 \ / \ 10 = 0.1$).
- `multiply(?y, ?m, ?x2)`, it calculates the y value by multiplying the m and $y2$ variables (i.e. $0.1 \ X \ 10 = 1$).
- `subtract(?b, ?y2, ?y)`, it subtracts the y into the $x2$ (i.e. $1 - 1 = 0$) and stores resultant value into the b.
- `multiply(?mx, ?m, ?x)`, it multiplies the slope (i.e. m) with the user given data (i.e. x) and stores resultant data into the mx (lets say the user given data is 8 than $(0.1 \ X \ 8 = 0.8)$).
- `add(?mg, ?mx, ?b)` -> it sums the values (i.e. $0.8 + 0 = 0.8$) of the b and mx variables and stores the resultant value into the mg variable.
- `hasMG(?N, ?mg)` finally, a calculated MG (i.e. 0.8) assigns to the instance of the specified datum node .

If a value-mg list has more than two pairs, like the $((0 \ 1)(10 \ 0.8)(18 \ 0))$ value-mg list, then it is necessary to calculate two different slopes for two different lines. Figure 5.10 depicts the data of this value-mg list in a graphical form. It has two different lines and each line has a different slope.

The assessment type given data is compared with the datum values of the pairs of this value-mg list. In such case, if a potential assessment type given data is 5 then the '*MG*' is calculated by a rule discussed in Section 5.3.4.1. In such case, if the assessment type's given data is 15 then the first rule is ignored because the 15 is not in the range of the first two pairs of the value-mg list. Another rule is required to represent the formal interpretation of this value-mg list for its third pair. The following rule is defined for this purpose.

```
Rule:
    DatumNode(?N),
    valueMgPairSequence(?N, ?Npair1),
    nextValueMgPair(?Npair1, ?Npair2),
    nextValueMgPair(?Npair2, ?Npair3),
```

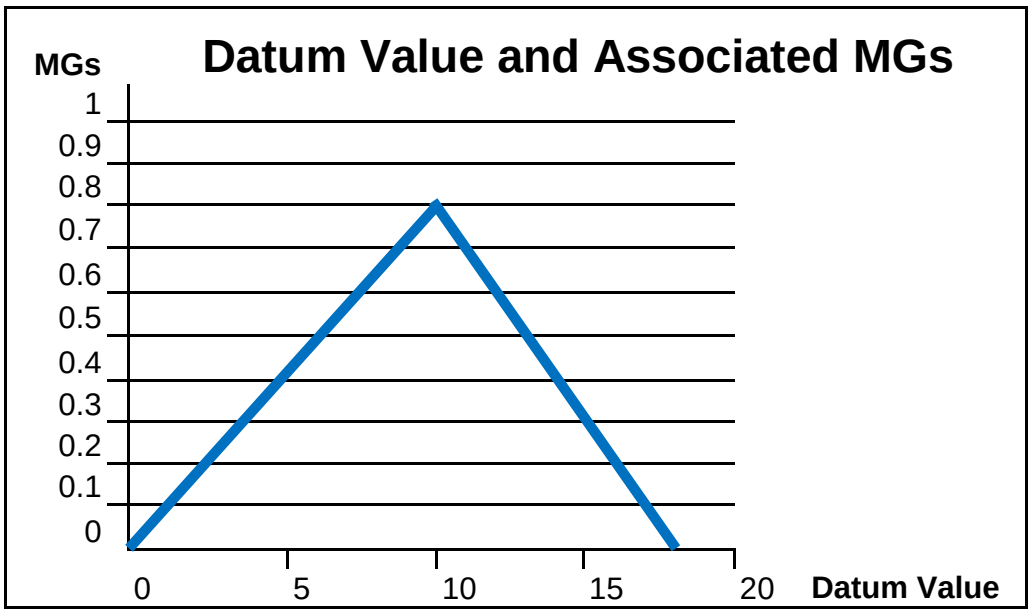

FIGURE 5.10: A graphical representation of a value-mg list having integer type datum values.

```
hasDatumValue(?Npair2, ?y1),
hasMG(?Npair2, ?x1),
hasDatumValue(?Npair3, ?y2),
hasMG(?Npair3, ?x2),
hasDatumValue(?N, ?x),
greaterThanOrEqual(?x, ?y1),
lessThanOrEqual(?x, ?y2),
subtract(?Yv, ?y2, ?y1),
subtract(?Xv, ?x2, ?x1),
divide(?m, ?Xv, ?Yv),
multiply(?y, ?m, ?y2),
subtract(?b, ?x2, ?y),
multiply(?mx, ?m, ?x),
add(?mg, ?mx, ?b) ->
hasMG(?N, ?mg)
```

The above rule has only one extra predicate (i.e. `nextValueMgPair(?Npair2, ?Npair3)`), which is a means to access the next or third pair of the value-mg list. The datum value

of the third pair of the value-mg list is then compared to the user given data. In such case, if a value-mg list has more than three pairs then other rule(s) can be defined on the same pattern by adding '*nextValueMgPair*' predicate(s).

5.3.4.2 The MGs calculations from the value-mg lists having nominal data types

The '*nominal*' data basically refers to categorically discrete data without any quantitative value (see details in Chapter 3). Such data type does not need any mathematical formulae. The user given data is directly compared to the datum values of the pairs of such value-mg lists. The $((Yes\ 1)(No\ 0))$ value-mg list keeps the nominal data (i.e. Yes and No). Following two rules are defined for this value-mg list.

```
Rule:
    DatumNode(?N),
    valueMgPairSequ
    valueMgPairSequence(?N, ?Npair1),
    hasDatumValue(?N, ?Udv),
    hasDatumValue(?Npair1, ?dv),
    hasMG(?Npair1, ?mg),
    equal(?dv, ?Udv) ->
    hasMG(?N, ?mg)
```

The above rule is used for the first pair of the value-mg list, whatever the nominal value a list has in its first pair (i.e. Yes or No). A built-in '*equal*' method is used to compare the assessment type given data with the stored datum value in the first pair of the value-mg list. In such case, if the user given data do not match to the stored datum value in the first pair then the following rule helps in initialisation of the MG of the datum node from the next pair of the specified value-mg list.

```
Rule:
    DatumNode(?N),
    valueMgPairSequence(?N, ?Npair1),
    nextValueMgPair(?NPair1, ?Npair2),
```

```
hasDatumValue(?N, ?Udv),
hasDatumValue(?Npair2, ?dv),
hasMG(?Npair2, ?mg),
equal(?dv, ?Udv) ->
hasMG(?N, ?mg)
```

The order of the value-mg list does not impact in deducing the MG, because both rules are defined generically. For example, if a datum node has the following $((No\ 1)(Yes\ 0))$ value-mg list and a potential user given data is Yes then $0\ MG$ is calculated. In this case, if the $((No\ 0)(Yes\ 1))$ value-mg list is associated to a datum node and the assessment type given data is Yes, then $1\ MG$ is calculated. In such case, if a value-mg list has more than two pairs like this: '*((DECREASING 0) (SAME 0.5) (INCREASING 1))*', and then more rules can be added with '*nextValueMgPair(?*p_i*, ?*p_{i+1}*)*' predicate(s).

5.4 Conclusions

The OWL Galatea model is designed for the Galatean model of classification by using OWL knowledge representation language. This model has four basic elements (i.e. '*Node*', '*Question*', '*Assessment-Type*' and '*Value-Mg-Pair*'). The '*Node*' is a core class of the OWL Galatea model. The concepts are the super nodes and the datum components are the leaf nodes of the Galatean hierarchy. These ideas are designed as subclasses (i.e. '*ConceptNode*' and '*DatumNode*') of the '*Node*' class. The relationship between super and sub nodes are defined by using '*hasDirSubNode*' and '*hasDirSupNode*' properties. The constraints defined in the '*Node*' and its subclasses (i.e. '*ConceptNode*' and '*DatumNode*') regarding these properties support retrieving and inferring the required information from the knowledge-base. The OWL reasoner can pick up the datum nodes, if the '*hasDirSubNode*' property constraints are modelled in them.

The knowledge domains of the Galatean model of classification have some repeating or replicating nodes. These can be managed into their appropriate categories by defining them as the subclasses of the classes (i.e. '*G-ConceptNode*', '*GD-ConceptNode*',

'*G-DatumNode*', '*GD-DatumNode*'). The non-repeating concepts and datum nodes can be managed as the subclasses of the classes (i.e. '*StandardConceptNode*', '*Standard-DatumNode*'). The generic constraints are defined in the '*Node*' and its subclasses (i.e. '*ConceptNode*', '*DatumNode*', etc) for creating relationships with other components or classes (i.e. '*Question*, '*Assessment-Type*' and '*Value-Mg-Pair*') by using different properties (i.e. '*hasQuestion*', '*valueMgPairSequence*', etc.) that help the concepts and datum nodes to share that knowledge. This approach or methodology also supports in minimising the replication of knowledge.

The '*Question*', '*Assessment-Type*' and '*Value-Mg-Pair*' classes are also primary components of the OWL Galatea model that are holding the concepts of the '*question*', '*population*' and '*value-mg*' attributes. The '*FilteredQuestion*' is a subclass of the '*Question*' class that keeps the concept of a '*filter-q*' attribute (see details in Chapter 3). A filtered or non-filtered question has some contents and those contents have association with the particular assessment types. The '*appliesTo*' property is utilised for defining the relationships between specified question(s) and the assessment type(s). The generic constraints are defined in the '*Assessment-Type*' class regarding the common or shared attributes of the assessment types. These constraints help the OWL reasoner in picking up the incorrect or inconsistent knowledge from the knowledge-base. For example, an assessment user inserts the current date in response of his date of birth information, which indicates that he is zero years old. The OWL reasoner can pick out such inconsistent information.

The '*MG*' and '*RI*' are two uncertainty variables of the Galatean model of classification. These concepts are managed in the OWL Galatea model by employing the '*hasMG*' and '*hasRI*' properties. Each node of the knowledge domain must have exactly one '*RI*'. Therefore, the constraints are defined regarding the range and cardinality of the '*hasRI*' property in the '*Node*' class. The OWL reasoner can test that the initialised '*RIs*' of each concept and datum node are within the given range and each Galatean node has exactly one '*RI*' value.

The value-mg list is a mandatory part of a datum component. The value-mg lists are

managed by employing the subclasses of the '*Value-Mg-Pair*' class as their elements. The data inside the value-mg lists is managed dynamically. Therefore, the first element makes relations to the specified datum component and also with the next element of the specified list via using the '*valueMgPairSequence*' and '*nextValueMgPair*' properties. The last element of the value-mg list not only holds the information of the datum values and MGs, but also keeps a cardinality constraint via using the '*nextValueMgPair*' property, which indicate the end of the list. The SWRL rules are employed for generating the '*MGs*' of the datum nodes from the user given data. The SWRL rules are utilised to implement the rationale of the data types (i.e. scale, nominal, integer, etc). The equal, greaterThanOrEqual and lessThanOrEqual built-in functions are employed in the SWRL rules for comparing the user provided data with the datum values stored in the pairs of the value-mg list. The OWL reasoner also supports in generating the MG for an associated datum node through the SWRL rule by comparing the potential assessment user given data with the graph of '*MGs*' and values. OWL represented constraints and restrictions are defined in the intelligent model (i.e. OWL Galatea model) in a way that these can support the extension of the OWL Galatea model in various decision domains of the Galatean model of classification.

In the next chapter, the extensibility of the OWL Galatea model in the knowledge domains (i.e. GRiST and ADVANCE) of the Galatean model of classification will be demonstrated. It will also highlight how OWL based specification can support in the reusing and sharing of common concepts, roles and constraints, how it can improve the knowledge engineering processes of the diverse knowledge domains of the Galatean model of classification and how it can reduce the replication of the knowledge.

Chapter 6

Extensibility of the OWL Galatea model

6.1 Introduction

The extensibility of the Galatean model is necessarily required, because there are various decision domains, which are based on the same classification theory. The challenge is how OWL based intelligent model (i.e. OWL Galatea model) can support the use of a large number of knowledge models and translation rules. This chapter demonstrates the extensibility of the OWL Galatea model to cover different manifestations of the galateas for different assessment types of the domains of the Galatean model of classification. This chapter presents how the OWL Galatea model helps in sharing and re-using common concepts, roles and constraints, and how it can support knowledge engineering processes of the diverse knowledge domains of the Galatean model of classification. The mental health (i.e. GRiST) and logistics (i.e. ADVANCE) are two distinct domains of the Galatean model of classification.

The mental health domain is closest to the level of psychological well being or the absence of a mental disorder, on the other hand logistics domain manages the flow of resources like material handling, production, packaging, inventory, fast delivery and small consignment sizes. The data structures of both domains are designed from the responses of a selected or focused group of experts. The concepts and successive

concepts are derived from their responses and they are currently managed in the XML knowledge trees. These XML knowledge trees keep the concepts and datum nodes on the basis of their syntax and locations, but the OWL Galatea model is quite self-sufficient and it does not depend on any pattern or sequence of the syntactic structure of them. Any alteration in the size of the knowledge-base of the domains (i.e. GRiST and ADVANCE) of the Galatean model of classification does not impact on the credibility of the OWL Galatea model. The comparison of both domains is presented by using appropriate illustrations from these domains.

The Super Structure Tree (SST) is manipulated as a root tree in these domains that holds knowledge for all assessment types within a domain. The techniques and methods are adopted for designing or generating the root knowledge-base, those are not only machine readable and understandable, but also represent the closeness to human expertise. The generation of the SST ontology is demonstrated by extending the OWL Galatea model and the SST.xml is employed for acquiring the existent knowledge of the nodes with their given knowledge structure. This discussion literally demonstrates the implementation of the OWL Galatea model in real-world human psychology domains.

6.2 Domains of the Galatean model of classification

The implementation of the OWL Galatea model is presented here through the case studies of GRiST and ADVANCE, which are based on the Galatean model of classification. GRiST belongs to the mental health domain, while ADVANCE affiliates to the logistics domain.

6.2.1 Mental health

Mental health problems can be described as '*depression*' [104] or '*anxiety*' [105] or '*mental disorder*' [106]. Mental disorder is still a main problem of our society, even after advancements in its treatments over the last fifty years [107]. Mental health care involves very complex structure [108], and barriers to mental health disorder need effective knowledge management through which correct predictions or decisions can be

possible. OWL provides supplementary vocabulary along with a formal semantics as compared to XML, RDF, and RDF Schema (RDF-S) and can facilitate in decision making and prediction tasks. The benefits of ontologies cannot be denied in the health sciences fields and especially in the mental health risk assessment field [109]. The Galatean Risk Safety Tool (GRiST) is a clinical decision support system and this tool stands under the umbrella of the Galatean model of classification. Here, the data structure of the GRiST tool is going to demonstrate how its concepts can absorb the constraints and attributes of the OWL Galatea model.

6.2.1.1 GRiST data structure

GRiST is a structured risk assessment tool [110], based on the Galatean model of classification. GRiST was designed to help clinicians assess risk of '*suicide*', '*self harm*', '*harm to others*', '*self neglect*', '*risk to dependents*' and '*vulnerability*' [111, 112]. GRiST was designed to accommodate different populations associated with each sector. The current version of GRiST tool's data structure consists of some XML trees i.e. SST(Super Structure Tree), ST (Structure Tree), RIT (Relative Influence Tree), CAT (Client Assessment Tree) and QT (Question Tree) [113].

6.2.2 Logistics

The term '*logistics*' is originated from the military [114]. The logistics is the act of planning, executing and coordinating complex projects or problems that need intensive human involvement for monitoring the movements of pallets [115]. They also highlighted the fact that the knowledge management and organisation of such complex projects is very difficult using manual tracking system or human involvements.

The long and slow moving supply chain system of many companies makes them notorious in providing poor services and also removes them from the list of the top class companies within identical domains. The slow and lengthy processing of the resources or pallet movements from one place to required destination [116] are due to the high volume of human-centric decisions. The human represented knowledge is the core or center of any decision support system. The use of incorrect and outdated information

leads to false and invalid decisions. For example, the resources are delivered to incorrect locations to reach their destination via long route that increase their cost, time and many other problems related to their delivery.

6.2.2.1 ADVANCE data structure

ADVANCE (Advanced Predictive-Analysis-Based Decision-Support Engine for Logistics) project is an EU 7th Framework Programme[1]. The ADVANCE framework is also constituted under the shade of the Galatean model of classification. It is used to improve transport network efficiency and organise all the information related to transport. The ADVANCE framework aims to manage information correctly and it is able to obtain crucial information related to the resources storing, obtaining, delivering and setting their delivery routes. Its knowledge-base system must be capable in making right decisions on right time.

The ADVANCE is designed to help transporters assess issues of reduce pallet overload, reduce spare capacity, and lorry or pallet unbalanced. The current version of the ADVANCE tool's data structure also consists of some XML knowledge trees i.e. SST(Super Structure Tree), ST (Structure Tree), RIT (Relative Influence Tree), CAT (Client Assessment Tree) and QT (Question Tree).

6.2.3 Commonalities and differences in the data structure of GRiST and ADVANCE tools

The GRiST and ADVANCE tools are two totally distinctive tools and both are used in extremely different domains of the society. One tool is employed to assess the mental disorder problems and the other deals with intensive varieties of decisions (e.g. path of packet optimisation, low cost, weight of pallet etc.) of the transporters. The concepts and datum nodes, and their contents of questions are also different from each other. Each assessment tool is applied or exercised on different varieties of the population. For example, the senior citizen (or '*older adults*') having '*learning disabilities*

[1] http://ec.europa.eu/digital-agenda/en/content/advanced-predictive-analysis-based-decision-support-engine-logistics

and mental health problems' are the assessment types of GRiST. On the other hand, the transporter (i.e. '*hub*') assesses the decisions through the ADVANCE on exchange of different vehicles or modes transportation for different palettes. The GRiST and ADVANCE tools have undoubted distinctive data, but there are still some common aspects in the two. Both tools are based on the Galatean model of classification. The data structures of these tools are currently managed in the form of XML trees and these trees are managed through some bespoke programs. The paper based specifications are used to manage these XML trees and check their knowledge consistency. The GRiST and ADVANCE tools have agitating and mutating data structures. An SST knowledge-base is managed for all assessment types of an assessment tool (GRiST or ADVANCE) and the ST is projected from the SST for a specified assessment type. The RIT agent is generated from the ST for initialising the RIs of the nodes. The CAT agent is transformed from the RIT and the QT agent is also derived from the RIT for keeping the questions for a specified assessment type (see Figure 6.1).

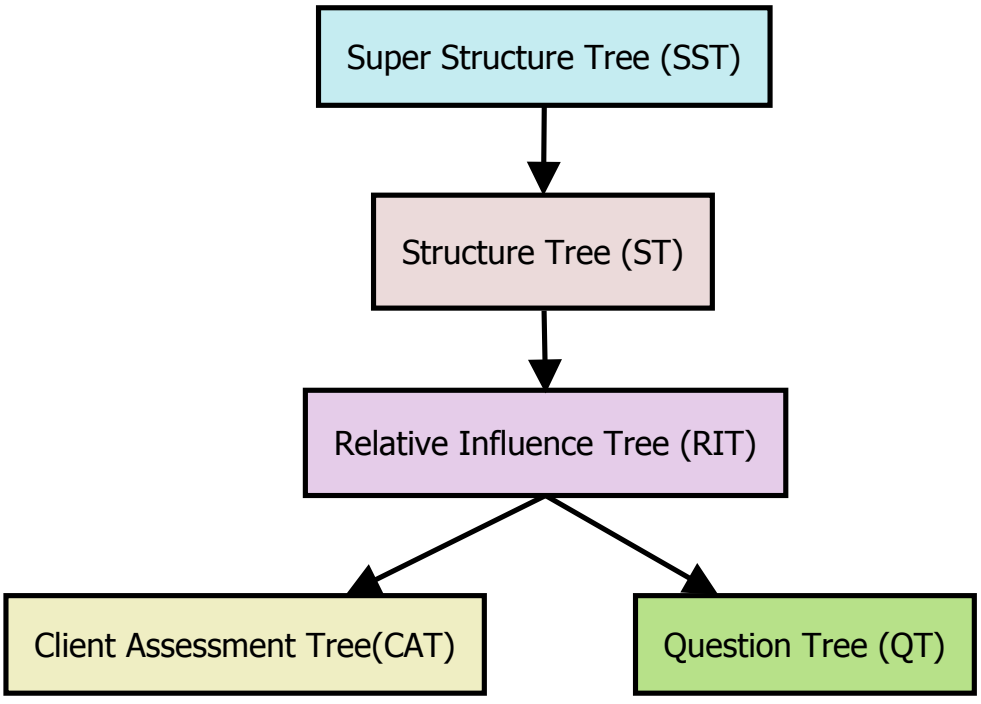

FIGURE 6.1: Common XML trees of GRiST and ADVANCE tools.

6.3 Generating the SST ontology for GRiST and ADVANCE by extending the OWL Galatea model

For generating SST ontology the OWL Galatea model is imported and the nodes of the SST.xml are iterated. The topmost node in the SST.xml hierarchical structure is a root node. The attributes affiliated to the root node (i.e. '*mental-health-risk*' in this case GRiST and '*lorry-pallet-balance*' in this case of ADVANCE) in the SST.xml are managed in semantically precise way and some extra constraints are also added for enhancing their intelligibility in the SST ontology. The following constraint is an example of it for the SST knowledge-base of ADVANCE.

```
Class: lorry-pallet-balance
    SubClassOf:
        hasDirSupNode exactly 0 Thing
```

The purpose of the constraint is to infer the root node without knowing it syntactic structure (see details in Chapter 7 in Section 7.3). The contents of the '*population*' attribute (see details in Chapter 3) attribute actually represent different populations or assessment types of these tools. The generic constraints and relationships are defined in the '*Assessment-Type*' class in the OWL Galatea model (see details in Chapter 5). It is going to demonstrate how to extend the '*Assessment-Type*' class in the knowledge-bases of both tools.

6.3.1 Extensibility of the '*Assessment-Type*' class and the translation of the '*population*' attribute

The contents of '*population*' attribute is declared as classes, and these classes are modelled as subclasses of the '*Assessment-Type*' class. The '*Assessment-Type*' class is extended according to different population of the assessment tool. Therefore, this class is going to demonstrate next how specific constraints are defined for the assessment types of GRiST and ADVANCE.

6.3.1.1 The assessment types of the GRiST

The '*older*', '*working-age*', '*child-adolescent*' and '*service-user*' are different assessment types of the GRiST tool (see Figure 6.2). Each assessment type has its individual or specific data constraints and these constraints are specifically defined in that class (i.e. '*older*', '*child-adolescent*',...). These assessment types inherit the constraints defined in their super class (i.e. '*Assessment-Type*') of the OWL Galatea model. Therefore, further constraints are added into the '*Assessment-type*' class in the SST knowledge-base of the GRiST tool. Following necessary and sufficient condition is defined by using covering axiom (see details in Chapter 4) in the '*Assessment-Type*' class.

```
Class: Assessment-Type
    EquivalentTo:
        older
         or working-age
         or child-adolescent
         or service-user
```

The above definition is impacted only in the knowledge structure of GRiST. Every tool of the domain of the Galatean model of classification modifies the '*Assessment-Type*' class by inserting a necessary and sufficient condition into it. The details of different assessment types of the GRiST tool are given below.

1. **service-user**

   The '*service-user*' class is defined as a subclass of the '*Assessment-Type*' class. The age limit for the '*service-user*' is not specified by the experts of the mental health risk domain. The '*older*', '*working-age*', '*child-adolescent*' and '*service-user*' are entirely different classes and these classes are defined as disjoint classes (see below constraint). Moreover, any instance of the '*service-user*' class cannot belong to the '*older*', '*working-age*' and '*child-adolescent*' classes.

   ```
   Class: service-user
       DisjointClasses:
           child-adolescent, older, working-age
   ```

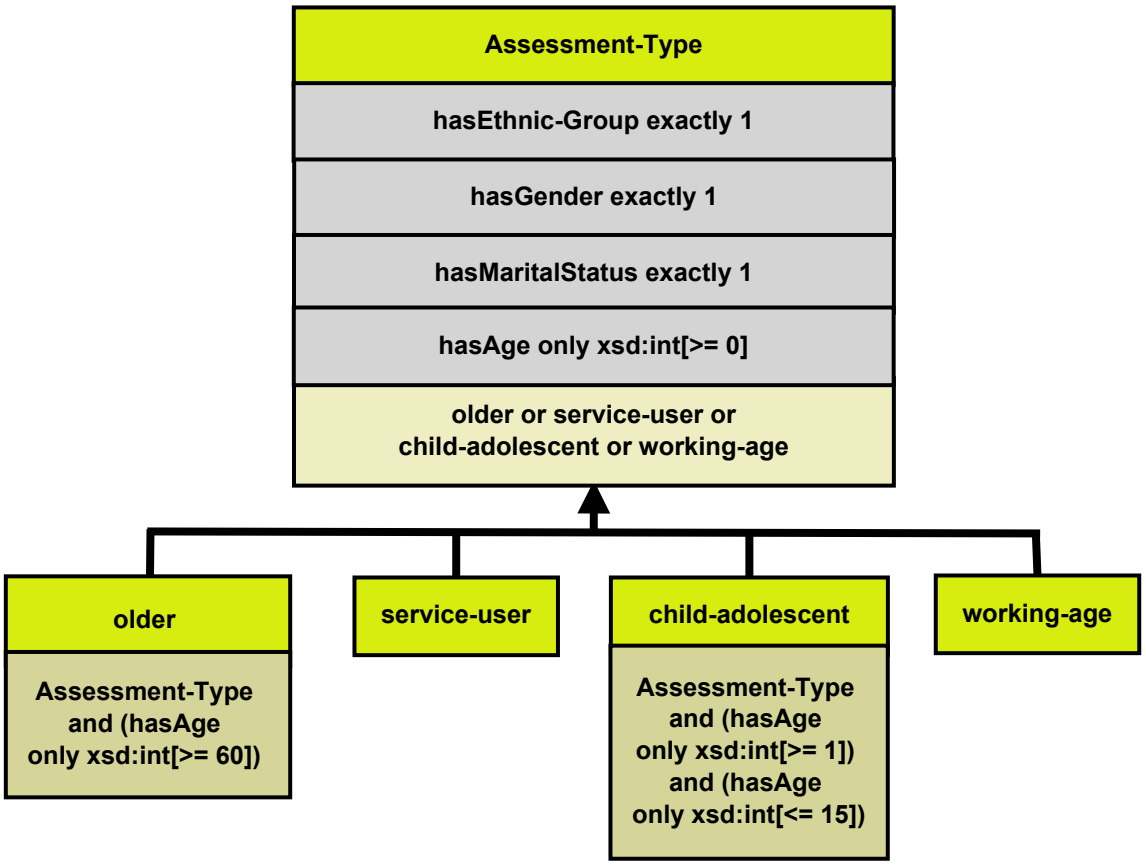

FIGURE 6.2: The demonstration of the extensibility of the '*Assessment-Type*' class in the SST ontology of the GRiST. The constraints of the OWL Galatea model are represented in the gray background colour and the constraints of the SST ontology of GRiST are depicted in the yellow background colour.

2. **older**

   The '*older*' class is also defined as a subclass of the '*Assessment-Type*' class. The minimum age limit for an '*older*' is specified greater than or equal to 60 years by the experts of GRiST.

```
Class: older

    EquivalentTo:

        Assessment-Type

         and (hasAge only xsd:int[>= 60])
```

   It means any instance of the '*Assessment-Type*' that has a value greater than or equal to 60 years via the '*hasAge*' property then that specified instance is inferred as the instance of the '*older*' class.

3. **child-adolescent**

   The '*child-adolescent*' class is also defined as a subclass of the '*Assessment-Type*' class. The age limit for a child-adolescent assessment type is defined by the experts of the GRiST tool between 1 to 15 years and it must be greater than zero. This constraint is defined in the form of following necessary and sufficient condition in the '*child-adolescent*' class.

```
Class: child-adolescent
    EquivalentTo:
        Assessment-Type
         and (hasAge only xsd:int[>= 1])
         and (hasAge only xsd:int[<= 15])
```

   It means any instance of the '*Assessment-Type*' that has a value between 1 to 15 years via using the '*hasAge*' property, that specified instance is inferred as the instance of the '*child-adolescent*' class. The '*child-adolescent*' class inherits the constraint (age value must be greater than zero) from its super class (i.e. Assessment-Type) of the OWL Galatea model.

4. **working-age**

   The working-age class is also defined as a subclass of the '*Assessment-Type*' class. The age limit for the '*working-age*' is not specified by the experts of the domain of discourse.

6.3.1.2 The assessment types of the ADVANCE

The '*ADVANCE*' tool has '*depot*', '*depot-close-hub*', '*depot-far-hub*' and '*hub*' assessment types, and these are modelled as subclasses of the '*Assessment-Type*' class (see Figure 6.3). Therefore, further constraints are added in the '*Assessment-type*' class in the SST knowledge-base of the ADVANCE tool. Following necessary and sufficient condition is defined by employing covering axiom in the '*Assessment-Type*' class.

```
Class: Assessment-Type
    EquivalentTo:
```

```
depot
 or depot-close-hub
 or depot-far-hub
 or hub
```

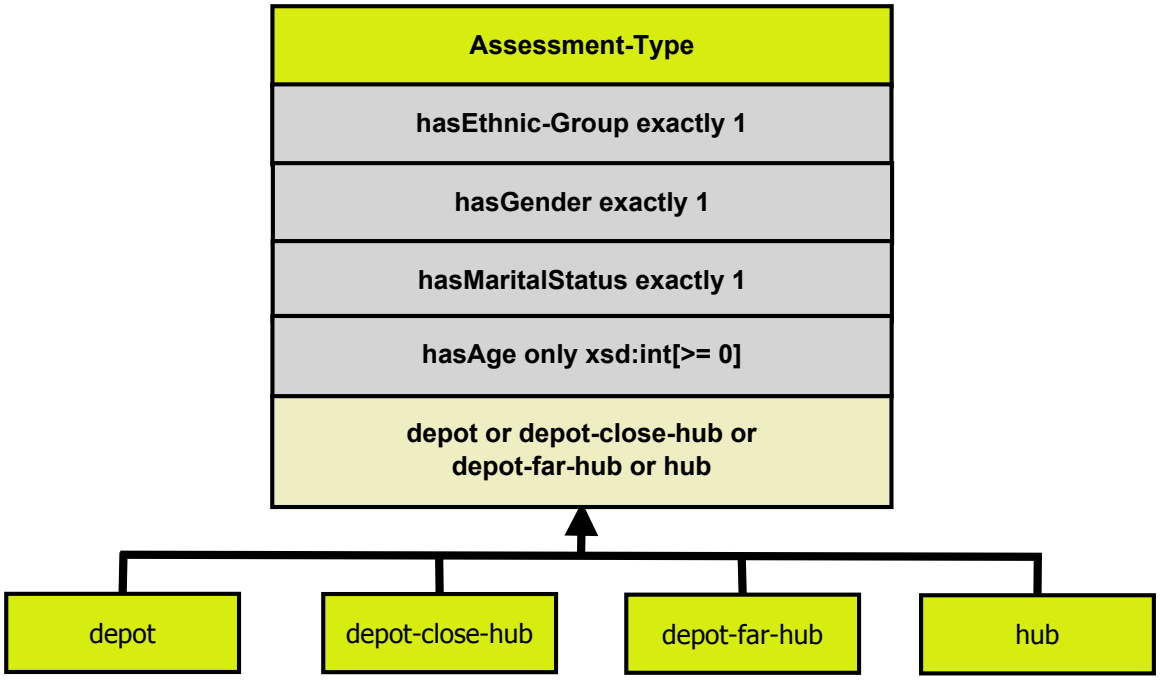

FIGURE 6.3: The demonstration of the extensibility of the Assessment-Type class in the SST ontology of the ADVANCE. The constraints of the OWL Galatea model are represented in the gray background colour and the constraints of the SST ontology of ADVANCE are depicted in the yellow background colour.

1. **depot**

   The '*depot*' class is defined as a subclass of the '*Assessment-Type*' class. The age limit for the '*depot*' is not specified by the experts of ADVANCE. The '*depot*', '*depot-close-hub*', '*depot-far-hub*' and '*hub*' are entirely different classes and these classes are defined as disjoint classes (see below constraint). Therefore, any instance of the '*depot*' class cannot belong to the '*depot-close-hub*', '*depot-far-hub*' and '*hub*' classes.

```
Class: depot
    DisjointClasses:
        depot-close-hub, depot-far-hub, hub
```

2. **depot-close-hub**

   The '*depot-close-hub*' class is also defined as a subclass of the '*Assessment-Type*' class.

3. **depot-far-hub**

   The '*depot-far-hub*' class is also defined as a subclass of the '*Assessment-Type*' class. The age limit for the '*depot-far-hub*' is not specified by the experts of the domain of discourse.

4. **hub**

   The '*hub*' class is also defined as a subclass of the '*Assessment-Type*' class. The age limit for the '*hub*' is also not specified by the experts of the domain of discourse.

6.3.2 Translation of the '*code*' attribute

The attributes of the SST.xml are translated into the SST ontology according to their correct semantics. Therefore, the contents of the '*code*' attribute (see details in Chapter 3) of the concepts and datum nodes of the SST.xml are employed as the node's name (i.e. OWL class name) in the SST ontology.

6.3.3 Translation of the '*label*' and '*help*' attributes

The '*hasLabel*' and '*hasHelp*' properties are used in SST.owl for providing the functionality of the '*label*' and '*help*' attributes (see details in Chapter 3) of the SST.xml. The following constraint is an example of it in the SST knowledge-base of ADVANCE that is defined in the '*reduce-deliv-pall*' class via employing the '*hasLabel*' property.

```
Class: reduce-deliv-pall
    SubClassOf:
        hasLabel some Labelreduce-deliv-pall
```

The '*Labelreduce-deliv-pall*' class has following constraints.

```
Class: Labelreduce-deliv-pall
    SubClassOf:
         hasContent value "reduce delivery pallets"
         appliesTo some depot-close-hub
```

Above information indicates that the contents mentioned in this class are applied to '*depot-close-hub*' assessment-type of ADVACE.

6.3.4 Extensibility of the subclasses of the '*Node*' class and the translation of the '*generic*', '*generic-type*' and '*generic-datum*' attributes

The SST.xml's nodes are modelled as OWL classes in the SST ontology. If an XML node has a '*generic-type*' attribute (see details in Chapter 3), then its value is tested. If the contents of the said attribute is '*g*' and the node is a concept node, then the node is defined as a subclass of the '*G-ConceptNode*' class otherwise it arranges as a subclass of the '*G-DatumNode*' class. If the value of the said attribute is '*gd*' and it is a concept node, then the node is defined as a subclass of the '*GD-ConceptNode*' class otherwise it is declared as a subclass of the '*GD-DatumNode*' class. In this case, the node has no such attribute (i.e. '*generic-type*'), and it has sub nodes, then it is codified as a subclass of the '*StandardConceptNode*' class. In the other case, it is categorised as a subclass of the '*StandardDatumNode*' class. If a repeating node has '*generic*' attribute (see details in Chapter 3), then just relationships are defined between the specified repeating node and its current super node via employing '*hasDirSubNode*' and '*hasDirSupNode*' properties and this attribute is not included into the SST ontology. The datum nodes that have '*generic-datum*' attribute (see details in Chapter 3), the contents of this attribute are used to find their actual definitions. The definition of the specified node must have the '*generic-type*' attribute and the contents ('*g*' or '*gd*') of this attribute indicates that the specified node should be defined as a subclass of the '*G-DatumNode*' or '*GD-DatumNode*'.

In this way, the generic section or context (see details in Chapter 3, and Sections 3.3.2.1, 3.3.3.4) is automatically excluded from the SST.owl because all the paths are translated correctly via employing '*hasDirSubNode*' and '*hasDirSupNode*' properties.

Previously, the generic section was used for merely hard-coded transformation of the RIT knowledge-base from the ST structure that is eventually removed or excluded from the RIT.xml. Therefore, the SST ontology is designed in this way that no further modification of the knowledge structure is needed in the RIT or any other structure. Each created concept node makes the relationship to its sub nodes via using the '*hasDirSubNode*' property. For example, the '*gen-meds-therpy*' and '*insight-resp*' nodes are the sub nodes of the '*generic*' node or context or section and has path information for the '*suic*' node in the GRiST tool. Hence, following constraints are defined in the '*suic*'.

```
Class: suic
    SubClassOf:
        hasDirSubNode only (gen-meds-therpy or insight-resp)
```

The sub nodes keep back reference to their direct super nodes into their subclasses. The '*Subgen-meds-therpy*' class is a subclass of the '*gen-meds-therpy*' class and the '*Subinsight-therpy*' is a subclass of the '*insight-therpy*' class. The reference is always created between the subclasses of the sub and super nodes and the '*Subsuic*' is a subclass of the '*suic*'. The following constraints are defined in these subclasses (i.e. '*Subgen-meds-therpy*' and Subinsight-therpy) for keeping the reference (as a necessary condition) of their super node (i.e. Subsuic) via employing the '*hasDirSupNode*' property.

```
Class: Subgen-meds-therpy
    SubClassOf:
        hasDirSupNode only Subsuic

Class: Subinsight-therpy
    SubClassOf:
        hasDirSupNode only Subsuic
```

Each subclass of a repeating node (i.e. '*G-ConceptNode*', '*GD-ConceptNode*', '*G-DatumNode*' and '*GD-DatumNode*') keeps a distinctive reference of the direct super node. For example, the '*bill-units-deliv-today*' (i.e. billing units matching delivery lorries

today) node is repeated in three different contexts (i.e. '*reduce-pall-overload*', '*reduce-spare-cap*' and '*lorry-pallet-unbalanced*'). Each of its subclass keep a unique reference from one of these contexts (see below example).

```
Class: Subbill-units-deliv-today1
    SubClassOf:
        hasDirSupNode only Subreduce-pall-overload

Class: Subbill-units-deliv-today2
    SubClassOf:
        hasDirSupNode only Subreduce-spare-cap

Class: Subbill-units-deliv-today3
    SubClassOf:
        hasDirSupNode only Sublorry-pallet-unbalanced
```

6.3.5 Extensibility of the '*Value-Mg-Pair*' class and the translation of the '*value-mg*' attribute

The contents of the '*value-mg*' attribute (see details in Chapter 3) are static in the SST.xml. The support of paper based specification is required for the explanation of the sequence and semantics of the '*MGs*' and datum values of a value-mg list. The contents of this attribute are managed by creating the subclasses of the '*Value-Mg-Pair*' class. For example, the '*suic-id-hi-risk*' has the $((0\ 0)(10\ 1))$ value-mg list in the GRiST tool. The '*suic-id-riskMg1*' and '*suic-id-riskMg2*' classes are modelled as the subclasses of the '*Value-Mg-Pair*' class for this purpose. The '*hasMG*' property is used to build the relationships with the MGs and the relationships with the datum values are maintained by using the '*hasDatumValues*' property. The sequence of a value-mg list is managed by employing the '*nextValueMgPair*' property and a subclass (i.e. '*suic-id-riskMg2*') of the '*Value-Mg-Pair*' class is used as a filler. The final element of the value-mg list represents the end of the list by inserting the zero number of cardinalities of the '*nextValueMgPair*' property to the '*Thing*' class. Following constraints are defined in the '*suic-id-riskMg1*' and '*suic-id-riskMg2*' classes.

```
Class: suic-id-riskMg1
    SubClassOf:
        hasDatumValue value 0,
        hasMG value 0,
        nextValueMgPair some suic-id-riskMg2,
        Value-Mg-Pair

Class: suic-id-riskMg2
    SubClassOf:
        hasDatumValue value 10,
        hasMG value 1,
        nextValueMgPair exactly 0 Thing,
        Value-Mg-Pair
```

The first pair of the value-mg list makes the relationship with the datum node via using the '*valueMgPairSequence*' property. The following constraints are defined in the '*suic-id-hi-risk*' class for making a relationship with its value-mg list.

```
Class: suic-id-hi-risk
    SubClassOf:
        valueMgPairSequence some suic-id-riskMg1
```

The types of the nodes are considered while making relationships with the specified value-mg list(s). A '*StandardDatumNode*' or '*G-DatumNode*' has always a single value-mg list and it preserves its relationship in the super class of the node. On the other hand a '*GD-DatumNode*' can have more than one value-mg lists. Each subclass of a '*GD-DatumNode*' (i.e. '*gen-gender*') holds a reference of a value-mg list and where it also retains a necessary condition for keeping the reference of its direct super node. In SST.xml, these lists along their specified paths or pointers are managed only in the generic context or location in the specified datum node (i.e. '*gen-gender*'), but in the RIT.xml, these specified datum nodes (i.e. '*gen-gender*') repeat with a single (context based) value-mg list in each context (which was mentioned in the path). For example, the '*g1*' (a subclass of the '*gen-gender*') class held the reference of its direct super node (i.e. '*gen-demog*') via the '*d1*' in the '*suic*' context (see Figure 6.4). The '*gen-gender*'

also has a value-mg list for the '*sh*' context. So, the '*g2*' (a subclass of the '*gen-gender*') class is kept the reference of the '*gen-demog*' via the '*d2*' towards the direction of the '*sh*' context. The '*d3*' and '*d4*' are kept the reference of the '*sn*' and '*hto*' contexts. In this way, each value-mg list is managed in the specified nodes by keeping the dynamic references of their direct super nodes, which benefits in querying and inferring such nodes from their particular contexts.

6.3.6 Translation of the '*values*' attribute

The '*hasValueTypes*' property is used to provide the functionality of the '*values*' attribute (see details in Chapter 3) for indicating the data types of the datum values of the value-mg list(s). The following constraint in the '*gen-gender*' class is an example of it.

```
Class: gen-gender
    SubClassOf:
        hasValueTypes value nominal
```

6.3.7 Translation of the '*prune-for*' attribute

The '*prune-for*' attribute (see details in Chapter 3) is used to discard or ignore some nodes that are prohibited for a specified assessment type and this attribute is managed in specific nodes which are identified by the experts of the specified domain. For example, the 'risk to dependents' (i.e. '*risk-dep*') is a concept in the GRiST tool and the child-adolescent is not a suitable population of this tool to provide his or her dependent information that are under risk (see Figure 6.5). The '*hasPruneFor*' property constraint is employed in the SST ontology for avoiding non-concerning concepts and datum nodes in a specified ST agent. The following constraint in the '*risk-dep*' class is an example of it.

```
Class: risk-dep
    SubClassOf:
        hasPruneFor some child-adolescent
```

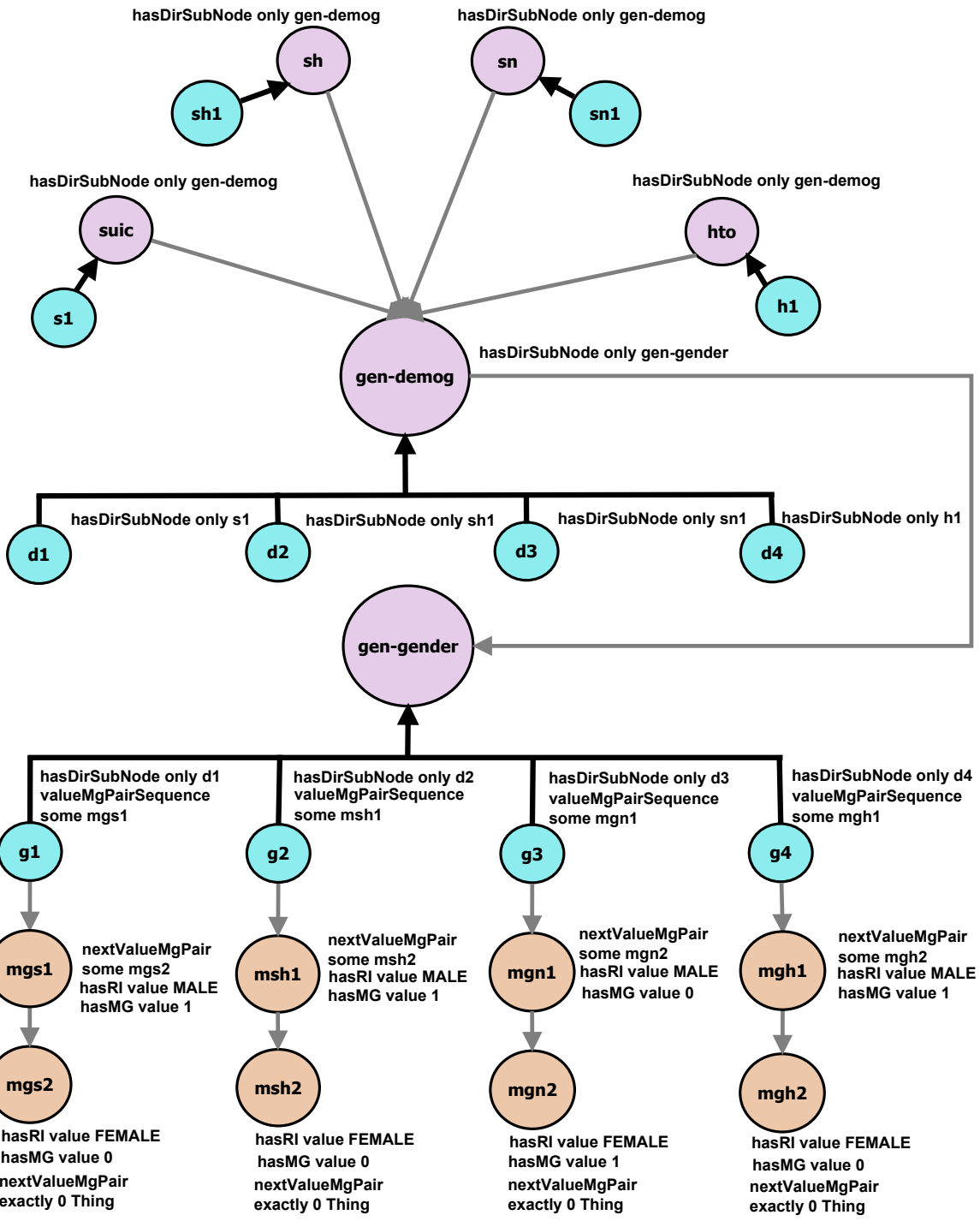

FIGURE 6.4: An example of a GD-DatumNode (i.e. gen-gender) with n number of value-mg lists in the SST.owl of the GRiST tool.

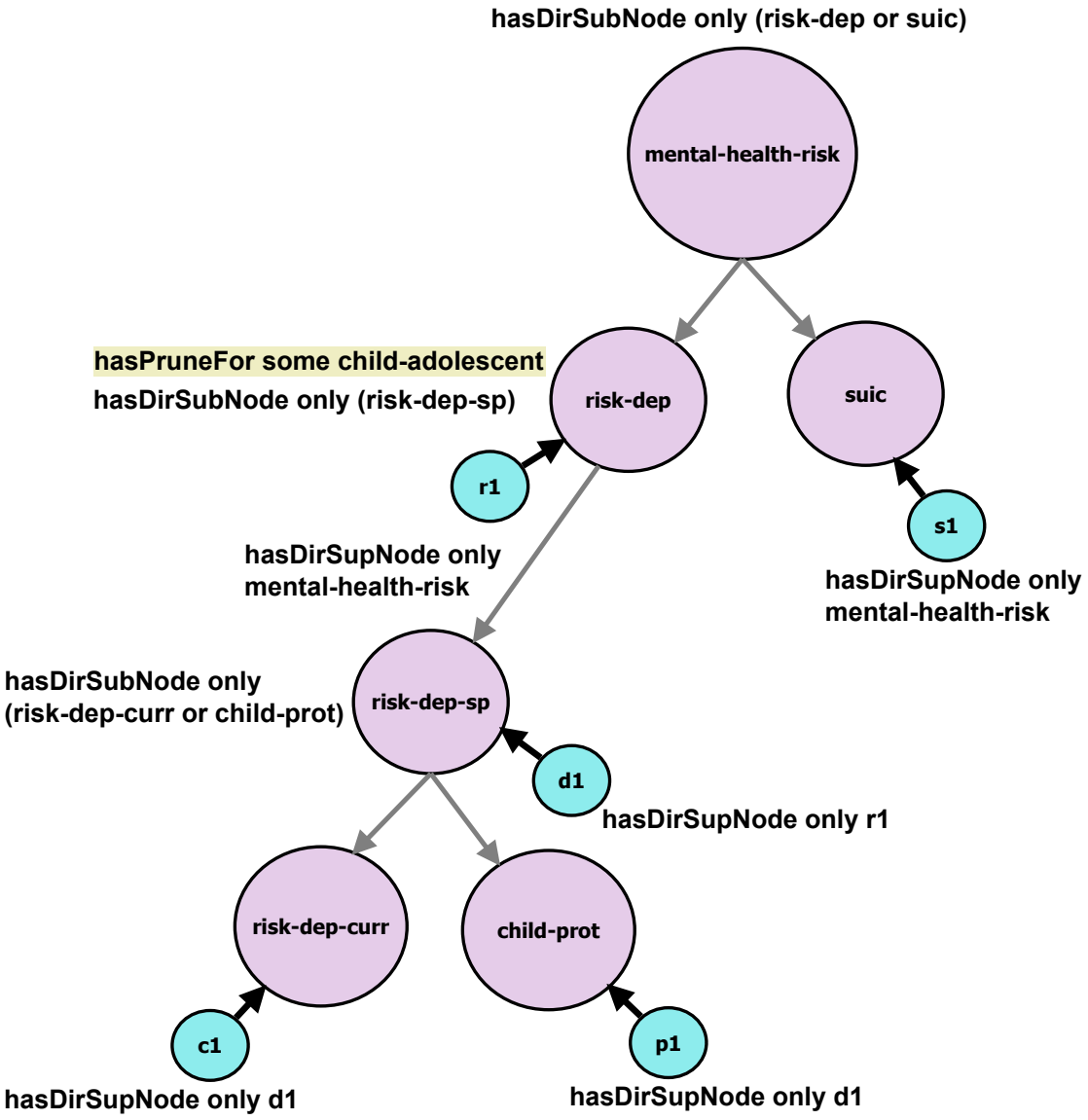

FIGURE 6.5: An example of a hasPruneFor constraint in the risk-dep concept in the SST.owl of the GRiST.

The benefits of using above constraint is that variant STs can be derived for variant assessment types through a simple query (see details in Chapter 7).

6.3.8 Extensibility of the '*Question*' and '*FiltredQuestion*' classes and the translation of the '*question*' and '*filter-q*' attributes

The contents of the '*question*' and '*filter-q*' attributes (see details in Chapter 3) are organised in the classes, which are modelled as the subclasses of the '*Question*' and '*FiltredQuestion*'. The relationships between these subclasses and their specified nodes are managed by employing the '*hasQuestion*' property. The following constraint is an example of the '*hasQuestion*' relationship with a filler of the '*Qusgen-gender*' (a subclass of the '*Question*' class) in the '*gen-gender*' class.

```
Class: gen-gender
    SubClassOf:
        hasQuestion some Qusgen-gender,
        GD-DatumNode
```

Each question (i.e. '*Qusgen-gender*') has some contents and those contents are applied to some assessment type(s). In this case, if the contents of a question are designed for more than one assessment types by the experts of the domain (i.e. GRiST or ADVANCE), then more than one subclass of the '*Assessment-Type*' class are listed along the '*appliesTo*' property as fillers into the specified subclass (i.e. '*Qusgen-gender*'). The following constraints are defined in the '*Qusgen-gender*' for this purpose.

```
Class: Qusgen-gender
    SubClassOf:
        hasContent value "What is you gender?",
        appliesTo some child-adolescent,
        appliesTo some older,
        appliesTo some service-user,
        appliesTo some working-age,
        Question
```

The GRiST and ADVANCE tools have two types of questions, (1) the generic questions and (2) the context-based questions. The generic and context based questions are

managed in the '*question*' attribute in the SST.xml, but in the RIT.xml the generic contents are only managed in the '*question*' attribute and the context based contents are managed under the '*level-question*' attribute (see details in Chapter 3). The rationale of both (i.e. '*level-question*' and '*question*') attributes are composed under one OWL property (i.e. '*hasQuestion*').

The relationships of the generic questions are maintained in the super classes of the concepts and datum nodes. A repeating node where ever it repeats, it keeps an identical questions' contents in different contexts for a specified assessment type. The '*gen-curnt-bhvrQus*' (a concept node in GRiST) has generic contents. The '*gen-curnt-bhvr*' is a repeating node, which has a relationship with the '*gen-curnt-bhvrQus*' class via the '*hasQuestion*' property. This relationship is maintained in the super class (i.e. '*gen-curnt-bhvr*') of the node. The subclasses (i.e. gen-crn1, gen-crn2, gen-crn3) of a node (i.e. '*gen-curnt-bhvr*') keep the identical contents in different contexts (see Figure 6.6).

The contents of the question of some repeating concepts and datum nodes (i.e. '*GD-ConceptNode*' and '*GD-DatumNode*') can vary in different contexts in a similar way as the '*RIs*' of the sub nodes of the '*GD-ConceptNode*' and value-mg lists of the '*GD-DatumNode*' can vary from context to context. These types of questions are called context based question. The relationships of the context based questions are created into the subclasses of the concepts and datum nodes via using the '*hasQuestion*' property. The '*gen-presentation*' is a node in the GRiST tool that has three different contents for three different contexts (see Figure 6.7).

For example, the '*gen-presentation*' node (in GRiST) has a question which has 'To what extent does the person's behavioral presentation during assessment match that of a young person who would give you maximum concern about suicide risk?' contents. These contents represent that this information is affiliated to only '*suicide*' (i.e. suic) context. This content is copied into the '*level-question*' attribute in the RIT.xml. The interim '*level-question*' attribute is removed, but its contents are copied back to the '*question*' attribute in CAT.xml. The '*gen-presentation*' node is managed in the SST ontology in three different contexts (i.e. '*sn*' (self neglect), '*suic*' (suicide) and '*sh*' (self harm)). Three subclasses (i.e. gen-p1, gen-p2 and gen-p3) are created for keeping

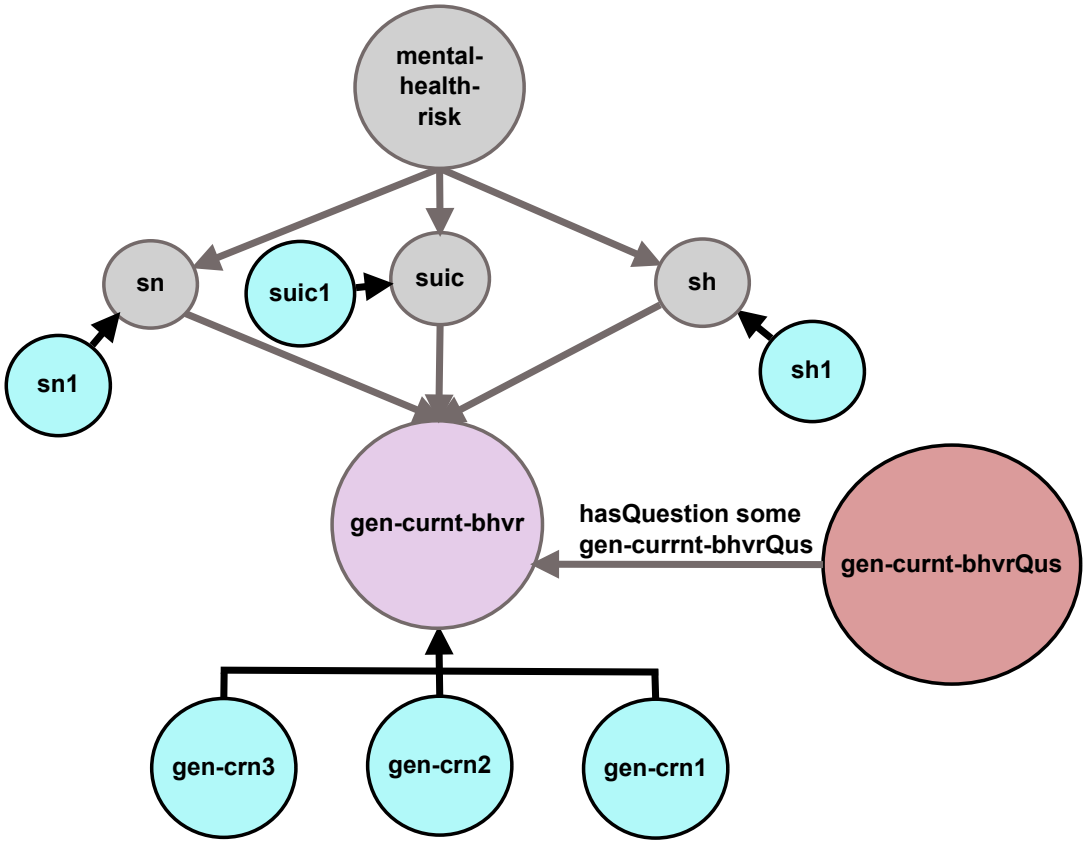

FIGURE 6.6: An example of the generic question of the concept of the GRiST tool.

the references of its different contexts. Each of its subclasses is kept a question, which have context-based contents.

6.3.9 Translation of the '*level*' attribute

The '*hasLevel*' property is used in the SST ontology for providing the functionality of the '*level*' attribute (see details in Chapter 3) of SST.xml. The following constraint is an example of it in the SST knowledge-base of GRiST.

```
Class: suic-patt-att
    SubClassOf:
        hasLevel value 1
```

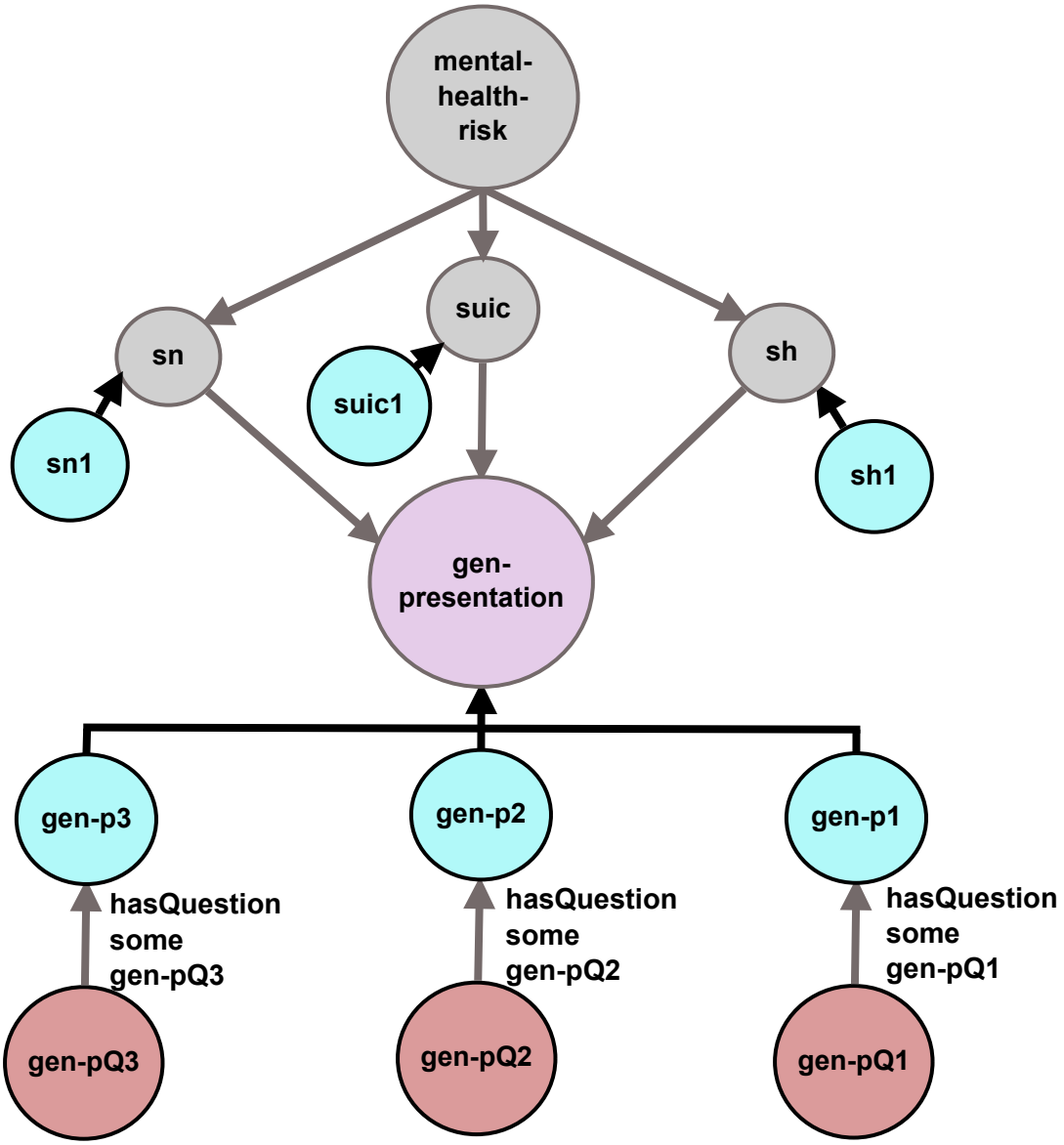

FIGURE 6.7: An example of the context based question of the concept of the GRiST tool.

6.3.10 Translation of the interim or temporary attributes

The “level-question”, “level-code” and “multiple-tick” attributes and their functionalities are not included into the SST ontology, because these attributes are completely dependent on the syntactic structure of the knowledge-base. For example, the ‘*hto-targets*’ node has the “multiple-tick” attribute (see details in Chapter 3 and Section 3.3.3.11). The contents of the said attribute are actually the sub node(s), and such contents are translated as sub nodes of the specified node which is translated as a generic distinct

concept from the datum node in the present RIT.xml. In contrast, the said node (i.e. '*hto-targets*') is defined as the subclass of the '*GD-ConceptNode*' and its contents are modelled as its sub nodes in the SST ontology. The relationships and constraints between these sub and super nodes are maintained via using the '*hasDirSubNode*' and '*hasDirSupNode*' properties in a similar way as other nodes of the SST ontology. The following constraint is an example of above discussion.

```
Class: hto-targets
    SubClassOf:
        hasDirSupNode only CARERS-FAMILY
        hasDirSupNode only SHARED-ACCOMNEIGHBOURS
        hasDirSupNode only FRIENDS-PEERS
        hasDirSupNode only HEALTH-WORKERS
        hasDirSupNode only AUTHORITY-FIGS
        hasDirSupNode only ETHNIC
```

The CARERS-FAMILY, SHARED-ACCOMNEIGHBOURS, FRIENDS-PEERS, HEALTH-WORKERS, AUTHORITY-FIGS, ETHNIC are the sub nodes of the '*hto-targets*' (see details in Chapter 3 and Section 3.3.3.11).

6.4 Conclusions

The worth of the OWL Galatea model is demonstrated by extending it through the knowledge structure of real world domains (i.e. GRiST and ADVANCE) of the Galatean model of classification. The GRiST and ADVANCE tools affiliate to two exceptional domains (i.e. mental health and logistics). The GRiST tool is operated for mental health screening purposes and the ADVANCE tool is employed for determining the decisions of the transporters. Comparing the knowledge structure of the GRST and ADVANCE tools revealed that both assessment tools have a complementary specification document, which is used for maintaining and manipulating of their XML knowledge trees (such as, SST, ST, RIT, CAT and QT). Both domains have repeated ('*G-ConceptNode*',

'*GD-ConceptNode*', '*G-DatumNode*' and '*GD-DatumNode*') and non-repeated ('*StandardConceptNode*' and '*StandardDatumNode*') concepts and datum nodes. The knowledge structure of the value-mg lists and questions are also identical in these tools. The OWL Galatea model supports overall knowledge engineering of the knowledge-bases of these assessment tools without exploring their syntax structure details. The nodes of the GRiST and ADVANCE tools inherit the concepts, properties and restrictions of the OWL Galatea model which help in reducing the replication of the knowledge. The OWL Galatea model supports making complex decisions in such complicated fields. The OWL Galatea decision support specification helps the machine to make automatic decisions on the basis of the semantics of the stored knowledge and to check the consistency of the knowledge-base by using the OWL reasoner.

This chapter renders the extensibility of the OWL Galatea model and the translation of the knowledge structure of the SST.xml of GRiST and ADVANCE for generating the SST ontology. Each class of the OWL Galatea model is extended and domain specific knowledge structures and their relationships are defined by using different OWL object and data properties. For instance, the '*Assessment-Type*' class extends and updates according to the knowledge structure of variant assessment types within each assessment tool. The '*child-adolescent*', '*older*', '*service-user*' and '*working-age*' are distinctive assessment types of the GRiST tool and the '*depot*', '*depot-close-hub*', '*depot-far-hub*' and '*hub*' are distinctive assessment types of ADVANCE. The '*hasEthnic-Group*', '*hasGender*', '*hasMaritalStatus*' and '*hasAge*' properties are employed to maintain their knowledge structures.

Different types of nodes are managed into their appropriate categories and the names of the nodes are formed in the SST ontology by extracting the contents of the '*code*' attribute of the SST.xml. The relationships between these nodes are managed by using '*hasDirSubNode*' and '*hasDirSupNode*' properties. The relationships and constraints between the question and the nodes of the SST knowledge-base are managed by using the '*hasQuestion*' property, these can support in inferring of them by calling simple queries (see details in Chapter 7).

The knowledge structure and sequence of the value-mg lists are managed and designed in a dynamic way such that the contents of the lists can be accessed without knowing the details of their encapsulated data. The current value-mg lists structure is needed to alter especially for '*gd*' datum nodes while transforming into the RIT.xml [117], but no modification is required in their knowledge structure while transforming from SST ontology to other knowledge state(s). The specialise concepts and constraints are arranged and organised individually by using some properties (such as '*hasPruneFor*, '*hasLevel*, '*hasLabel*',...) within the assessment tool (i.e. GRiST or ADVANCE). The attributes (such as '*multiple-tick*') which have syntactic dependency are not included into the SST ontology. The '*generic*' concept or context is also dropped, because this concept does not have any logical and semantic relations to the knowledge-base and it is just used for holding repeating nodes or knowledge in one location. Therefore, the relationships of its sub concepts and datum nodes are defined through the '*hasDirSubNode*' and '*hasDirSupNode*' properties to the individual concepts.

The knowledge domains of the Galatean model of classification have a progressive and mutative knowledge structure, which modifies into its next state(s). The SST ontology is managed in this perspective that can intelligently support further transformation or translation of its knowledge. In the next chapter, the transformation of different OWL agents (i.e. ST, RIT, CAT and QT) will be explored. These are derived from the SST ontology for a potential assessment type. It will also highlight how OWL based transformation processes can give advantage to a knowledge-base having a progressive knowledge structure.

Chapter 7

Knowledge-bases Transformations by using OWL Reasoner

7.1 Introduction

The knowledge-bases of the domains of the Galatean model of classification have progressive or mutative data structure. The knowledge management and maintenance of such knowledge-bases with a mutative data structure is difficult in various XML structures supported by a text-based specification for the human experts. In this chapter the derivation of different states of the knowledge is demonstrated with the help of suitable examples. Therefore, some formal transformation processes are utilised for the transformation of the knowledge into its different states. The OWL-approaches facilitate the minimising the replicating of knowledge and support the transforming of knowledge into its different states.

The key aim is to keep the consistency and stability of the knowledge-bases at top priority, while different transformation processes are executed. The transformation processes employ the OWL reasoner, which check or confirm the internal conflicts and consistency of the knowledge-bases [118], in its different states. The OWL classifier or reasoner [103, 119], classifies the subsumption relation, which leads to further transformations and also support in initialisation of uncertain attributes (e.g. '*RI*') of the Galatean model of classification. The OWL reasoner also minimises the need of

human expertise in classification and consistency checking of a huge amount of knowledge. The OWL based transformation processes discussed in this chapter are domain independent and also helpful in knowledge engineering of any domain based on the theory or concept of the classification.

The first state (i.e. SST ontology) of the knowledge-base keeps the knowledge for all population within a domain (see details in Chapter 6). The next state (i.e. ST) of the knowledge-base is derived from the SST ontology for a particular assessment type, and the irrelevant knowledge chunks are removed from it or ignored. This translation aids in further transformation, where the uncertain attribute (i.e. '*RI*') [120], is initialised in all nodes of the target ontology. This transformed state (i.e. RIT) with extra knowledge also helps in further dynamic transformations of the next states (QT and CAT). OWL API (an open source software component which facilitates in creating, manipulating and serializing ontologies) [87], is utilised for the transformations of these states or ontologies. This chapter demonstrates the transformations and derivations of different states of the knowledge-bases by employing different illustrations from GRiST and ADVANCE. The benefits of using OWL approaches, while in different intelligent transformations of the knowledge-base, are also discussed in detail at the end of this chapter.

7.2 Transformation of the ST ontology from the SST ontology

The SST ontology contains knowledge for all assessment types and the ST ontology is a short version of the SST ontology for a single assessment type within a domain. In such case, if an SST keeps the knowledge for five different assessment types then five different STs can be generated from it. The purpose of this transformation is to generate the ST ontology for a specified population. This transformation adds more constraints into the concepts and datum nodes, which are included in the ST ontology from the SST ontology. The OWL reasoner is utilised for this transformation and it is also used for consistency checking of the existing and new added constraints in the ST ontology. The following transformation process is employed for the derivation of the ST ontology.

7.2.1 The transformation process for the ST ontology from the SST ontology

This algorithm translates the ST ontology from the SST ontology for a specified assessment type. This process alters the relationships between the sub and super nodes, which are imported from the SST ontology. The purpose of this modification is to create the possibility to traverse the logical hierarchy of the ST ontology, and also add the nodes, which are concerning to a specified population by using the OWL reasoner in the ST ontology. The OWL universal ('*allValues*') restriction (see details in Chapter 4) is utilised for making the constraints and relationships between the sub and super nodes via using the '*hasDirSubNode*' and '*hasDirSupNode*' properties in the SST ontology (see details in Chapter 6). These existing constraints and relationships do not guarantee the existence of at least one relationship between the sub and super nodes. Further constraints and relationships are defined between sub and super nodes via employing these properties with existential restrictions (see details in Chapter 4) in the ST ontology through this process. The benefits of such constraints and relationships are that these not only aid in the traversing of logical hierarchy of this state (i.e. ST ontology), but also all logical hierarchies of the next states (i.e. RIT, QT and CAT). These constraints and relationships also support in removing the irrelevant information chunks for a specified population by using the OWL reasoner. This process is utilised for the transformation of the ST ontology.

1. The algorithm imports the SST.owl. All the concepts and datum nodes of the SST.owl are accessible for the ST ontology.

2. The concepts and datum nodes have the '*hasPruneFor*' (see details in Chapter 6 and Section 9.12) relationships and constraints for the specified assessment type, those nodes are not included in the ST ontology. For the sake of this a necessary condition is defined in the '*Node*' class by employing the '*hasPruneFor*' property and the specified assessment type class (a subclass of the '*Assessment-Type*') is used as a filler of the constraint, while execution of this process. For example, the ST ontology is generated for the '*child-adolescent*' of GRiST. Therefore, following restriction is defined in the '*Node*' class in the ST ontology.

```
Class: Node
```

```
SubClassOf:
  hasDirSubNode only
    (Node and
    (hasPruneFor exactly 0 child-adolescent))
```

Only the nodes that satisfy the above given necessary condition are included in the ST ontology from the SST ontology. The closure axiom and universal restriction (see details in Chapter 4) are exerted in the anonymous super class(es) of the concepts of the SST ontology for making the relationships with their sub nodes. For example, A, B, C, D, E and F nodes are part of the SST ontology. In SST, the direct sub nodes are listed as the union of fillers via the '*hasDirSubNode*' property in the universal restriction (see Figure 7.1). The following restriction is defined in the A class in the imported (i.e. SST) ontology.

```
Class: A
  SubClassOf:
   hasDirSubNode only (B or C or D)
```

The OWL reasoner is employed for the classification of the concepts and datum nodes, either having or not having the '*hasPruneFor*' relationships and constraints. The anonymous super classes having the existential restriction via the '*hasDirSubNode*' property are defined in the concepts for only concerning sub nodes, which do not have the '*hasPruneFor*' constraints with the specified assessment type (i.e. child-adolescent). Therefore, only A, B and C nodes are included in the ST ontology and the following anonymous super classes or necessary conditions are defined in the A class in the ST ontology.

```
Class: A
  SubClassOf:
   hasDirSubNode some B,
   hasDirSubNode some C
```

The necessary condition, by employing the existential restriction and '*hasDirSubNode*' property, is not defined in the D node in the specified ST ontology (see Figure 7.2). It is because the specified node (i.e. D) has kept the data that is not related

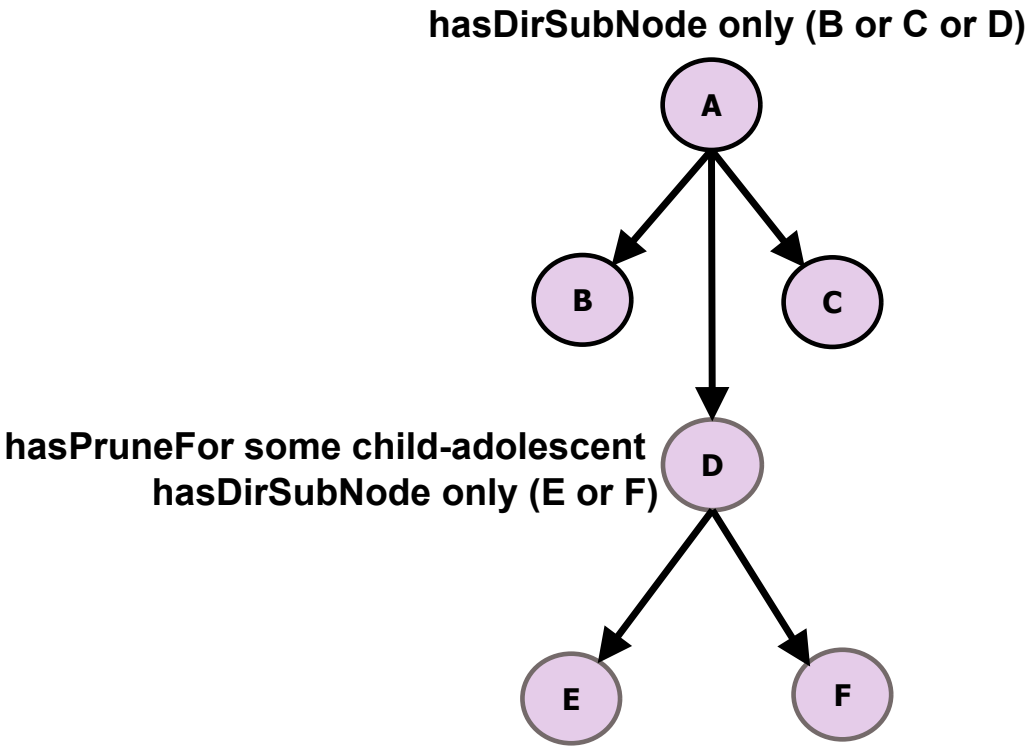

FIGURE 7.1: An example of the nodes have the hasPruneFor constraints related to a specified assessment type (i.e. child-adolescent) in the SST ontology. The nodes are represented in pink colour. The edges and necessary conditions are represented in black colour.

to the specified assessment type (i.e. child-adolescent), which means if a specified concept node has '*hasPruneFor some child-adolescent*' necessary condition, then that node with its sub nodes (i.e. E and F) are also not included in the specified ST ontology. The new constraint of the '*Node*' class, helps the OWL reasoner to pick up the unsatisfiable classes (i.e. '*D*') of the specified ontology.

The relationships from sub to super nodes are maintained via the '*hasDirSupNode*', which is an inverse property of the '*hasDirSubNode*' property. These relationships are formed in the subclass(es) of the sub nodes via employing the '*hasDirSupNode*' property with the universal restriction in the form of the necessary condition in the imported ontology (see details in Chapter 5 and Section 6.3.4). For example, $a1$ and $b1$ are subclasses of the A and B classes. The relationship from the B node to its super node (i.e. A) is maintained by using '*hasDirSupNode*' property (see Figure 7.3), and $b1$ class holds following anonymous super class for keeping the reference of the direct super node in the SST ontology.

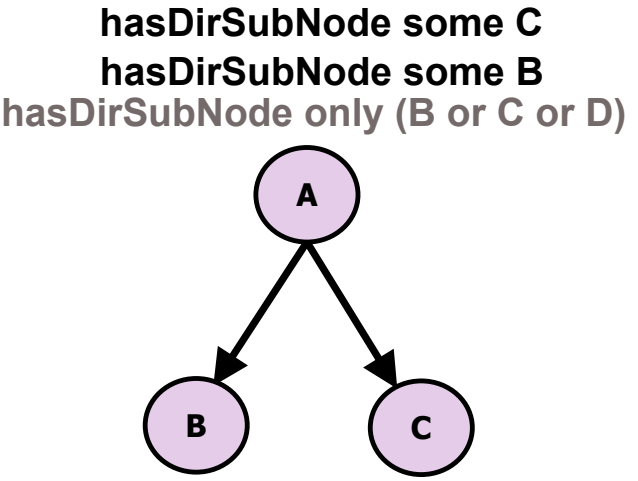

FIGURE 7.2: An Illustration of the concepts having $hasDirSubNode$ constraints in the ST ontology.

```
Class: b1

  SubClassOf:

   hasDirSupNode only a1
```

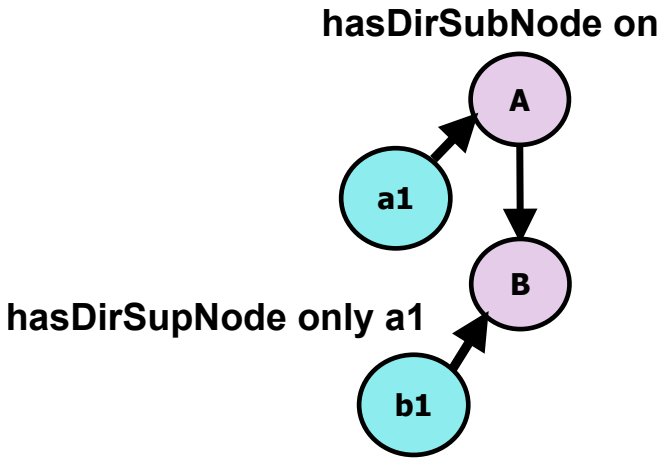

FIGURE 7.3: An example of the nodes have relationships from sub node to super node via using $hasDirSupNode$ property in the SST ontology.

A sub node can have more than one super nodes, and its subclass(es) keep the reference(s) of each context or super node. For example, the B node has two

super nodes (i.e. A and X), and it also has two subclasses (i.e. $b1$ and $b2$) and each subclass keeps a reference of a context (see Figure 7.4).

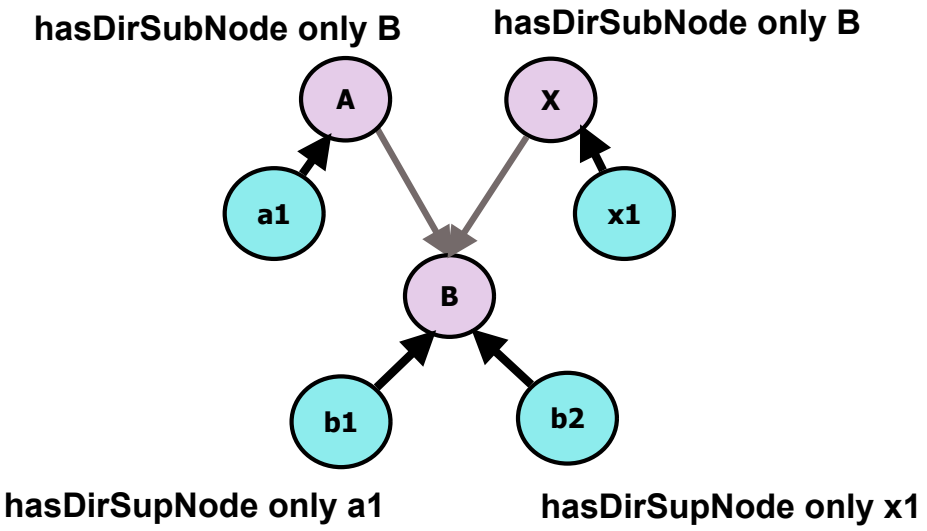

FIGURE 7.4: An example of a sub node (i.e. B), that have two super nodes (A and X) in the SST ontology. The edges between the sub and super nodes are represented in gray colour and the edges between sub and super classes are represented in black colour.

The issues with the existing constraints of the subclasses of the sub nodes in the SST ontology is that they do not confirm the existence of at least one relationship to their super nodes via using the '*hasDirSupNode*' property. Therefore, if an instance of the super class (i.e. B) of the sub node makes relationship to an instance of a super node (A or X) via the '*hasDirSupNode*' property, then it is difficult for the OWL reasoner to discriminate the type of the instance. It is because, the A and X nodes are not defined semantically different or disjoint from each other and the universal restriction can trivially satisfy an unsatisfiable filler. Therefore, the following extra necessary and sufficient condition is defined in the $b1$ class in the ST ontology (see Figure 7.5).

```
Class: b1
  EquivalentTo:
  B
   and (hasDirSupNode some a1)
   and (hasDirSupNode only a1)
```

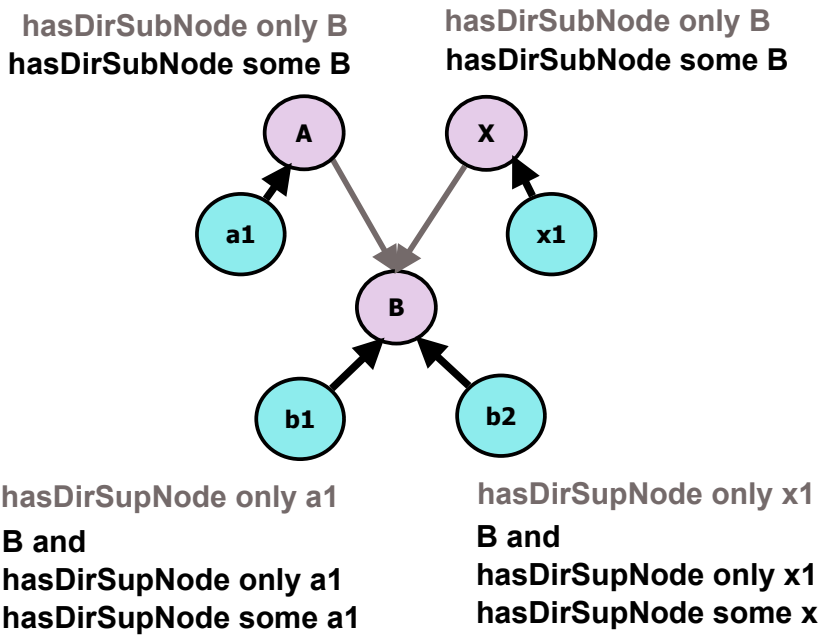

FIGURE 7.5: An example of a sub node (i.e. B), that have two super nodes (A and X) in the ST ontology. The constraints in the nodes of the SST ontology are represented in gray colour and the constraints in the nodes of the ST ontology are represented in black colour. The edges between the sub and super nodes are represented in gray colour and the edges between sub and super classes are represented in black colour.

The above given structure of the necessary and sufficient condition helps the OWL reasoner to identify the type of the member or instance which leads to correct or precise relationships between super and sub nodes. The benefits of the new constraint is that the ST ontology can only keep the concepts and datum nodes, that are related to a specified population. The significance of these extra constraints cannot be denied, while traversing the logical tree from bottom to top and classifying the decision classes in the last state (i.e. CAT) of the knowledge-base. All newly added constraints in the ST ontology are also helpful in fulfilling the requirements of the knowledge-base of an assessment tool, and support in inferring the required knowledge.

7.3 Transformation of the RIT ontology from the ST ontology

The rationale of the RIT ontology is that all of its concepts and datum components must keep the '*RIs*' or weightings (see details in Chapter 3 and Section 6.3.10). The

RIT ontology is inferred from the ST ontology. The hierarchical structure of the source ontology (i.e. ST) does not modify at this state and it is taken as such into the RIT ontology. On the other hand, the hierarchical structure of some datum nodes having the '*multiple-tick*' attribute (see details in Chapter 3 in Section 3.3.3.11) are translated as concepts in the RIT.xml. Such nodes are already translated and managed as concepts with their correct and precise semantics in the SST ontology (see details in Chapter 6). The distribution or initialisation of the RIs methodology is based on a mathematical theory that is designed by the experts of the Galatean model of classification. The following process presents how the RIs are distributed among the nodes of a knowledge domain.

7.3.1 The RIs distribution or initialisation mechanism

The RIs distribution mechanism is based on the mathematical theory presented by the experts of the Galatean model of classification [121]. The RIs are elicited according to the experts' opinion to weight each sub node of a concept with respect to its siblings. Equation 7.1 formalizes the process

$$RI_S = PD/(\sum_{i=1}^{n} RI_i) \tag{7.1}$$

where S is a sub node, RI is the relative influence generated at each sub node, i, of the concept, PD is the experts predefined data, from the concept node to the sub concept or datum node. The calculated number of sub nodes is divided on the PD. Eventually the relative influence are linked with all parts of the Galatea hierarchy or the logical RIT tree structure. The above discussed equation is designed in the process that transforms RIT ontology. The following process is used for the transformation of the RIT ontology from the ST ontology.

7.3.2 The transformation process for the RIT ontology from the ST ontology

This process initialises the '*RIs*' of the concepts and datum nodes in the RIT ontology. The '*RIs*' of the sub nodes depend on certain behavior or the type of their super nodes.

Some sub nodes keep constant or fix '*RIs*' in different contexts, and the '*RIs*' of some sub nodes can vary from context to context. This algorithm elaborates all types of concepts with their sub nodes while initialising the '*RIs*'.

1. The algorithm imports the ST.owl. All the concepts and datum nodes of the ST ontology are accessible for the RIT ontology.

2. The notion of the '*G-ConceptNode*' is that it repeats in different contexts and the '*RIs*' of its sub nodes are fixed or constant in different contexts (see details in Chapter 5). The subclasses of the '*G-ConceptNode*' class are iterated and each iterated class employs in the query for inferring its sub nodes. For example, the '*gen-hopeless*' is a subclass of the '*G-ConceptNode*', and it is employed as a parameter of the query from the iterated list. The following query is an example of it.

   ```
   hasSupNode some gen-hopeless
   ```

   The purpose of this query is to infer the sub concepts and datum nodes or branches inside of the '*gen-hopeless*'. The '*hasSupNode*' property (a transitive and super property of the '*hasDirSupNode*' property) is employed in this query to access the whole branch inside a '*G-ConceptNode*' (i.e. '*gen-hopeless*'). The '*gen-future-plan*' and '*gen-life-not-lvng*' are sub nodes of the '*gen-hopeless*' in the GRiST tool (see Figure 7.6). The '*gen-hopeless*' node is repeated in three different individual risks or concepts with all of its sub nodes in the given example. If the above query executes, then it lists the subclasses (i.e. '*gen-fplan1*', '*gen-fplan2*' and '*gen-fplan3*', '*gen-life1*', '*gen-life2*', '*gen-life3*'), which are defined for each context.

   It is compulsory for each sub node of the '*gen-hopeless*' to have a fixed '*RI*' in different contexts. It means all subclasses of the '*gen-future-plan*' class must have an identical '*RI*' in three different contexts (i.e. self-neglect ('*sn*'), vulnerability of service user ('*vuln*') and suicide ('*suic*')), because of the '*gen-hopeless*'. The '*gen-future-plan*' holds references of three different contexts into its subclasses (i.e. '*gen-fplan1*', '*gen-fplan2*' and '*gen-fplan3*'), and these all subclasses must have

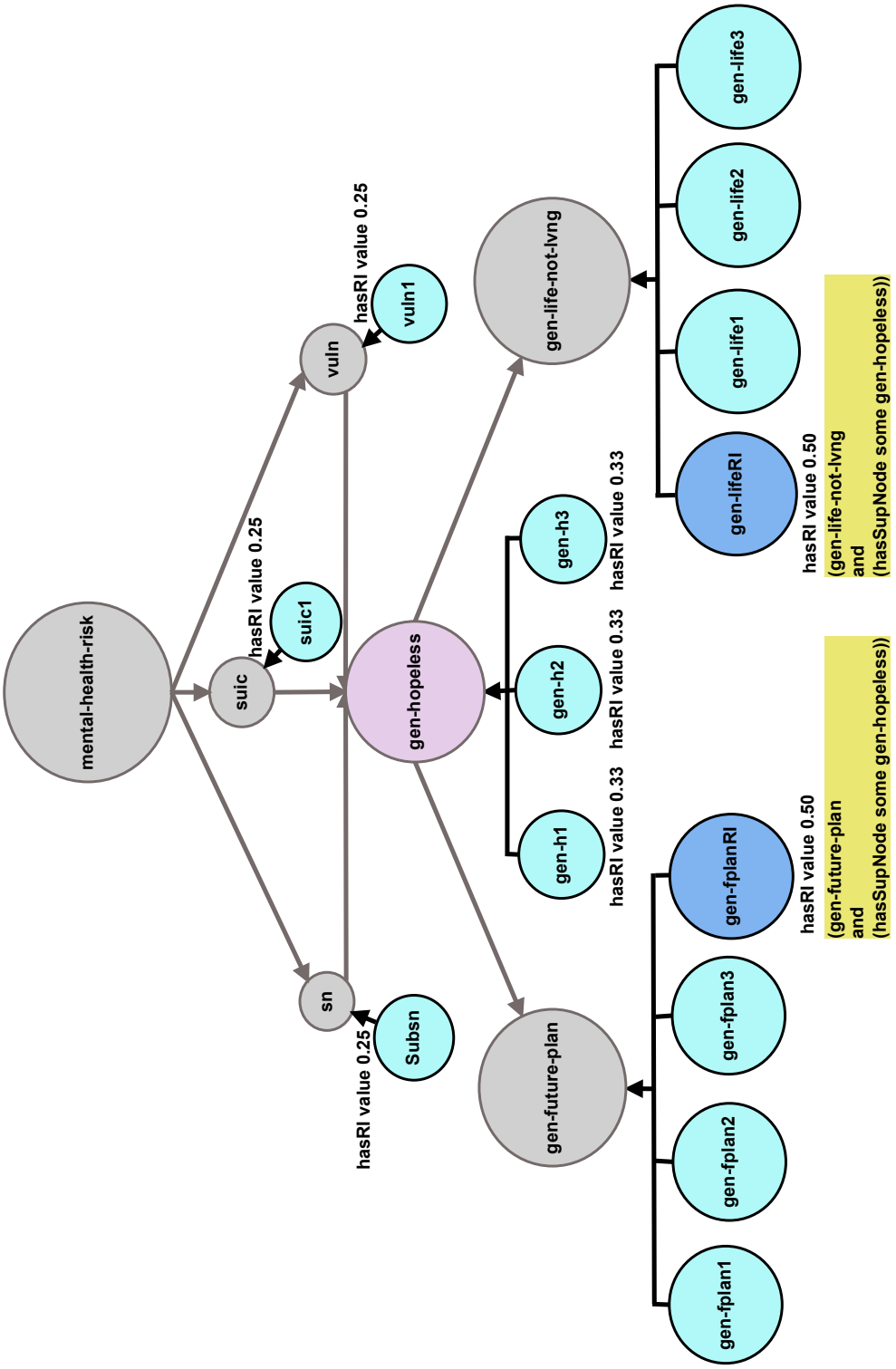

FIGURE 7.6: An example of the sub nodes of G-ConceptNode (i.e. gen-hopeless), that are represented in pink colour. The edges sub and super nodes are represented in gray colour. The classes created in the RIT ontology are represented in dark blue colour.

an identical '*RI*' for different contexts. The 0.50 is an '*RI*' value of the '*gen-future-plan*' (i.e. '*gen-fplan1*', '*gen-fplan2*' and '*gen-fplan3*') node. A new subclass (i.e. '*gen-future-planRI*') of the '*gen-future-plan*' class is created into the RIT ontology for this purpose, and this subclass makes the relationship with the '*gen-hopeless*' (i.e. '*G-ConceptNode*') ancestor node via the '*hasSupNode*' property and the '*RI*' value (i.e. 0.50) is also stored in it. So, the following constraints are defined in the '*gen-future-planRI*' class.

```
Class: gen-future-planRI
    SubClassOf:
        hasRI value 0.50
    EquivalentTo:
        gen-future-plan and
        (hasSupNode some gen-hopeless)
```

The definition of the '*gen-future-planRI*' class means any instance that has the type '*gen-future-plan*' class and has super node from the instances of the '*gen-hopeless*' class via the '*hasSupNode*' property is the instance of the '*gen-future-planRI*' class. For understanding the whole concept, it is necessary to explore the necessary and sufficient conditions of its (i.e. '*gen-future-planRI*') sibling classes (i.e. '*gen-fplan1*', '*gen-fplan2*' and '*gen-fplan3*') For example, the following definition is defined in the '*gen-fplan1*' class in the ST ontology.

```
Class: gen-fplan1
    EquivalentTo:
        gen-future-plan and
        ((hasDirSupNode only gen-hopeless1)
        and
        (hasDirSupNode some gen-hopeless1))
```

It means any instance of the '*gen-future-plan*' class that has direct super node from the members of the '*gen-hopeless1*' (a subclass of the '*gen-hopeless*' class) class via the '*hasDirSupNode*' property is a member of the '*gen-fplan1*' class. The members of the '*gen-fplan1*', '*gen-fplan2*' and '*gen-fplan3*' classes are inferred as the members of the '*gen-future-planRI*' class and these classes are

also inferred as the subclasses of the '*gen-future-planRI*' class by using the OWL reasoner. This concept has been exploited and the '*RI*' has been stored in the '*gen-future-planRI*' class and in this way the '*gen-fplan1*', '*gen-fplan2*' and '*gen-fplan3*' classes keep an identical '*RI*' in different contexts from their inferred super class (i.e. '*gen-future-planRI*'). The constraints of the '*hasRI*' property are already defined in the '*Node*' class in the OWL Galatea model, which force the concepts and datum nodes to keep exactly one '*RI*'(see details in Chapter 5 and Section 5.3.1).

3. The rationale of the '*GD-ConceptNode*' is that it repeats in different contexts and the '*RIs*' of its sub nodes can vary from context to context (see details in Chapter 5). The contextual references are maintained in their subclasses in the (S)ST ontologies (see details in Section 7.2). On the other hand, each sub node of the '*StandardConceptNode*' keeps specific individual context knowledge (see details in Chapter 5), and it has only one subclass, which holds a specific reference. So, it make sense, to store the context based '*RIs*' into their existing subclasses. Therefore, the '*RIs*' of the sub nodes of the '*GD-ConceptNode*' and '*StandardConceptNode*' are organised in their existing subclasses in an identical way, because no new pieces of knowledge are required to be created unlike the sub nodes of the '*G-ConceptNode*'. For example, the '*gen-currnt-bhvr*' node is an example of the '*GD-ConceptNode*' and the '*gen-app-diet*' and '*gen-sleep-diet*' are its sub nodes (see Figure 7.7). The '*RI*' of the '*gen-app-diet*' node is 0.33, which is stored into the '*gen-diet3*', while it has '*gen-depression*' ancestor node. The 0.14 '*RI*' is stored into the '*gen-diet2*' for the context of the '*suic*' ancestor. The 0.25 '*RI*' is saved into the '*gen-diet1*' for the context of the '*risk-dep*' ancestor.

   For initialising the '*RIs*' of the sub nodes of the '*GD-ConceptNode*' and '*Standard-ConceptNode*', a generic approach is used in this process for traversing the logical hierarchy from root node (see details in Chapter 3) to datum nodes. The root node is retrieved from the knowledge-base by determining the '*hasDirSupNode exactly 0 Thing*' necessary condition, which is defined in the root node of the SST ontology (see details in Chapter 6). F The following query is called for inferring the specified root node of the assessment tool (i.e. GRiST or ADVANCE).

```
hasDirSupNode exactly 0 Thing
```

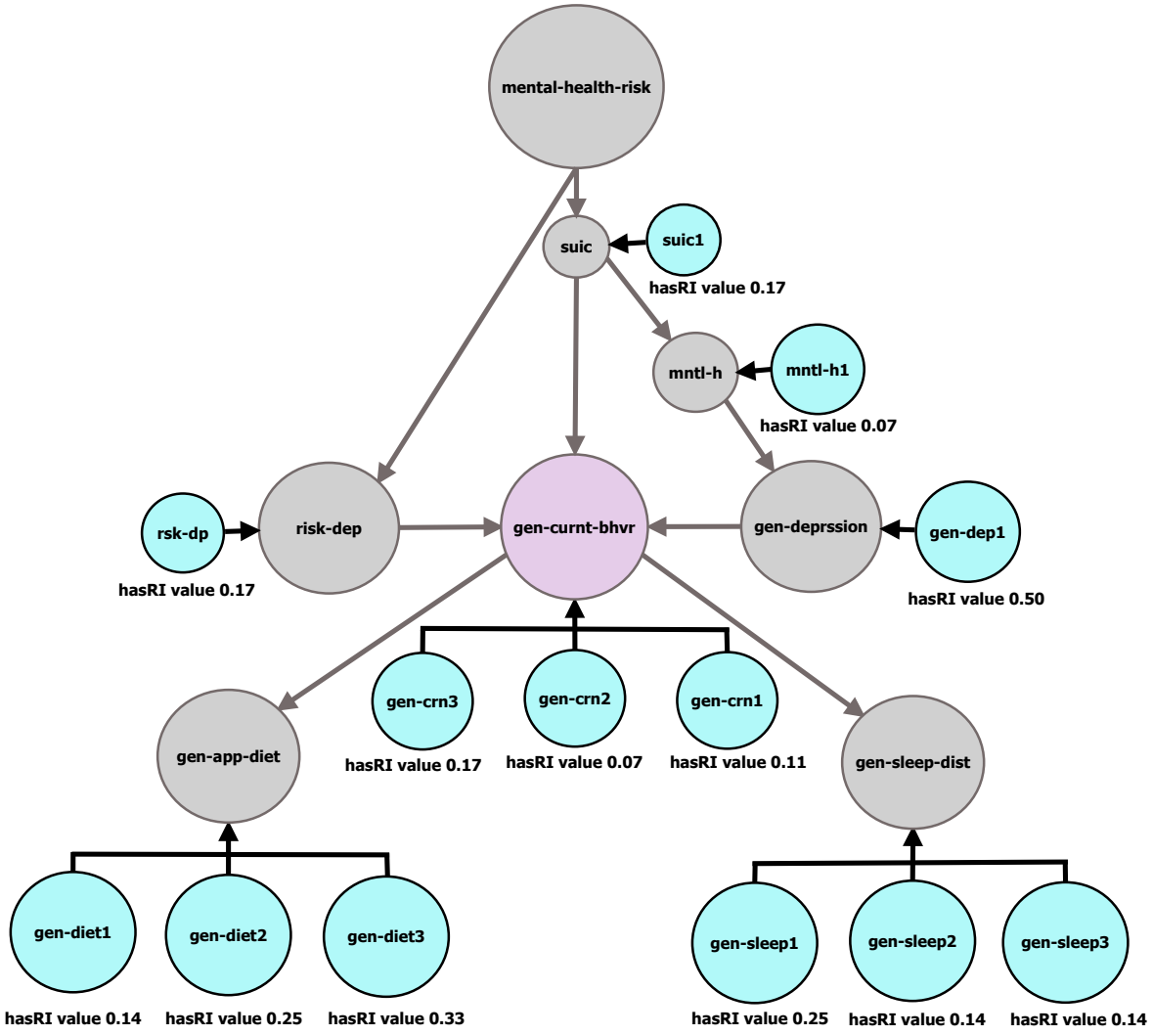

FIGURE 7.7: An example of the sub nodes of gen-curnt-bhvr (i.e. GD-ConceptNode) having context based RIs in the GRiST.

The above query infers the root node, which is utilised for iterating the hierarchical structure inside it. For example, if the RIT ontology is for the GRiST's assessment type, then the root node '*mental-health-risk*' is inferred from the above query and it (i.e. '*mental-health-risk*') is employed as a parameter of an inner query instead of the '*StandardConceptNode*' or '*GD-ConceptNode*'. The following query is an example of it.

```
hasDirSupNode some mental-health-risk
```

The '*hasDirSupNode*' property is used instead of '*hasSupNode*' property in the query. It is because, the concept node (i.e. '*GD-ConceptNode*' or '*StandardConceptNode*') does not impose on homogeneity of the uncertain attribute (i.e. '*RI*') in their sub branches or nodes. This query infers the direct sub nodes of the root node (i.e. '*mental-health-Risk*'). The grand sub concepts are called recursively reach to the datum nodes and this query is repeatedly called in each iteration by amending its parameter's value dynamically. The '*RIs*' of the concepts and datum nodes are initialised during recursive mechanism and this process is executed until the '*RIs*' of all concepts and datum nodes are initialised.

7.4 Transformation of the CAT from the RIT

The SST.xml, ST.xml and RIT.xml have more than one individual concepts or sections and one generic concept or section (see details in Chapter 3 and Section 3.3.2). The current approach for the management of the generic concept and its sub nodes are completely based on the syntactic structure of themselves and their context, but the OWL based approach facilitates in defining meaningful relationships. These relationships in the sub nodes of the '*generic*' node or context or section are defined in different associated individual concepts and this concept is truncated while the translation of the SST ontology from the SST.xml (see details in Chapter 6). This section actually facilitates initialising the uncertain attribute (i.e. '*RI*') of the sub nodes of the repeating concepts and later this section is removed from the last state (i.e. CAT.xml) in the XML trees [62], because the repeating concepts with their inside branches are initialised in their associated individual contexts in this tree. It means that this section has no logical association with the knowledge-base. The repeating concepts are just forced to keep their context based information in the '*level-code*' and '*level-question*' attributes (see details in Chapter 3), which support in generating the CAT.xml from the RIT.xml. Therefore, the '*level-code*' attribute is not defined in the RIT ontology. The rationale of the '*level-code*' attribute is achieved by calling some simple queries through the OWL reasoner. Therefore, the following simple algorithm is employed for generating the CAT ontology from the RIT ontology.

7.4.1 The transformation process for the CAT ontology from the RIT ontology

The CAT ontology is generated according to the experience level of the assessment type. Variant CAT(s) can be generated by determining the associated contents of the '*hasLevel*' attribute, which is associated with the concepts of the knowledge-base. The contents of this attribute indicates the experience level of the specified assessment type. In this case, if the '*hasLevel value 1*' necessary condition of the concepts are determined, then the CAT is used for the domain expert user. In this case, if the '*hasLevel value 0*' necessary condition of the concepts are deduced, then it is applied on the common people or those who have no basic knowledge of the domain.

The required nodes are inferred by calling some simple queries. This algorithm includes the '*GD-ConceptNode*' and its sub concepts and datum nodes into the CAT ontology, that either have or do not have a '*G-ConceptNode*' ancestor node, but it must have a necessary condition by employing the '*hasLevel*' property. This concept is actually the idea or notion of the '*level-code*' attribute. For example, the '*gen-currnt-bhvr*' (a '*GD-ConceptNode*') has the '*hasLevel value 1*' necessary condition in the GRiST tool (see Figure 7.8). The '*suic*' (a '*StandardConceptNode*') and '*gen-depression*' (a '*G-ConceptNode*') are the super nodes of the '*gen-currnt-bhvr*'. The following simple query executes for inferring the '*GD-ConceptNode*' (i.e. '*gen-currnt-bhvr*').

```
GD-ConceptNode and
(hasLevel value 1) and
(hasSupNode some G-ConceptNode or not (G-ConceptNode))
```

Each inferred '*GD-ConceptNode*' is passed as a parameter of an inner query for inferring its sub nodes or inside branches. The following query is executed for inferring the sub nodes of the '*gen-currnt-bhvr*'.

```
hasSupNode some gen-currnt-bhvr
```

The above query infers the sub nodes (i.e. '*gen-app-diet*') of the '*gen-currnt-bhvr*' and these inferred sub nodes (i.e. '*gen-app1*' and '*gen-app2*') with their super nodes (i.e. '*gen-currnt-bhvr*') are included into the CAT ontology.

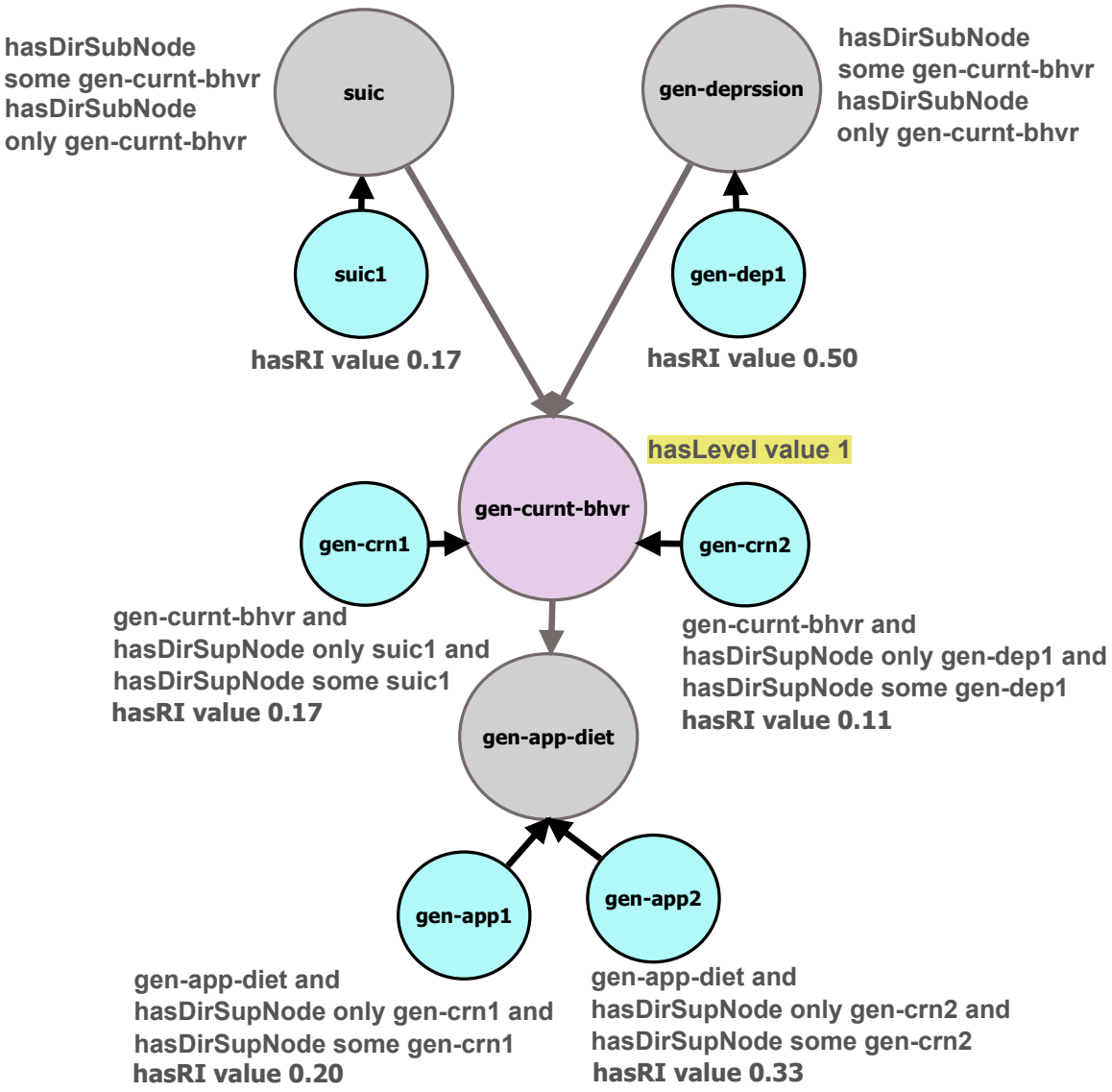

FIGURE 7.8: An example of a GD-ConceptNode (i.e. gen-currnt-bhvr) having the 'hasLevel value 1' constraint.

7.5 Transformation of the QT from the ST or RIT ontology

The Question Tree (QT) keeps the nodes, which have questions content related or associated to a specified assessment type. It is currently transformed from the RIT.xml tree by determining the contents of a temporary attribute (i.e. '*level-question*') (see details in Chapter 3). The contents of the context based questions are copied into the '*level-question*' attribute from the '*question*' attribute during the transformation of the RIT.xml from the ST.xml. The repeating concepts (i.e. '*g*' or '*gd*') (see details in Chapter

3) having the '*level-question*' are determined for generating the QT from the RIT.xml. This attribute and its contents are taken away in the similar way as the '*level-code*' attribute (see details in Chapter 3) during the transformation of the CAT.xml from the RIT.xml. The OWL based approach discourages the use of such temporary attribute(s) and provides a generic mechanism for making the relationships between concepts or a datum nodes with their associated questions. The context based questions make relationship with the subclasses of the concepts and datum nodes, and the generic questions make relationships to the super class(es) of the specified nodes (see details in Chapter 6 in Section 6.3.8).

7.5.1 The transformation of the QT ontology

The '*QT*' is generated by determining the necessary conditions of the concept nodes, which have the relationships and constraints of the '*hasLevel*' property. In such case, if the '*hasLevel value 0*' necessary condition of the concepts is determined then it means that the '*QT*' is employed only for naive or common assessment types or those users with no basic knowledge of the specified domain. In another case, the '*hasLevel value 1*' necessary condition of the concepts are deduced which means the '*QT*' is used for the experienced assessment type of the domain. In this algorithm, the constraints regarding the '*hasQuestion*' and '*hasLevel*' properties are determined to infer the concepts and datum nodes those having questions' content related to a specified assessment type. This process is demonstrated for an experienced assessment type, it means the concepts having the '*hasLevel value 1*' necessary conditions are included with their sub branches or nodes in the QT ontology.

The concepts and datum nodes having the '*hasLevel value 1*' necessary conditions and having the relationships with the fillers (i.e. the subclasses of the '*Question*' class) via the '*hasQuestion*' property are included in the QT ontology. The following query is used for inferring the repeating concepts (i.e. '*GD-ConceptNode*' and '*G-ConceptNode*'), which have the '*hasLevel*' and '*hasQuestion*' constraints.

```
((GD-ConceptNode or G-ConceptNode) and
  (hasQuestion some Question)) and
  (hasLevel value 1)
```

This query keeps the idea of the '*level-question*' attribute, which originated from the RIT.xml tree. The meaning of the above query is to infer the repeating concepts from the imported ontology. Each inferred concept employs an inner query for inferring its inside branches or sub nodes, which have question contents related to a specified population. For example, the '*gen-presentation*' (a repeating) node is employed in the following query for inferring its sub nodes for the '*child-adolescent*' user of GRiST.

```
hasSupNode some gen-presentation and
  hasQuestion some (appliesTo some child-adolescent)
```

In this case, if no repeating concept has the '*hasLevel*' and '*hasQuestion*' constraints in the knowledge-base, then the following query is called.

```
(hasLevel value 1) and
hasQuestion some FilteredQuestion
```

It means any concept that has the '*hasLevel value 1*' necessary condition and also has relationships with the classes having the '*FilteredQuestion*' type via the '*hasQuestion*' property is inferred. In such case, if the result of this query is also null then following query executes.

```
(hasLevel value 1) and
hasQuestion some Question
```

The flexibility of OWL based modelling is that a QT ontology can be inferred either from the ST ontology or from the RIT ontology. It is because the transformation of the QT ontology from the imported ontology (ST or RIT) is not conditional upon any temporary attribute (i.e. '*level-question*'). An intelligible DL query can infer the repeating concepts, which have relationships and constraints of the '*hasLevel*' and '*hasQuestion*'. In [111], two different algorithms are discussed for generating the QT.xml: (1) Level 0 QT and (2) Level 1 QT. Both types of the QT(s) can be generated by calling intelligible queries from an imported ontology by modifying the parameters of the calling query.

7.6 The benefits of OWL based approach

The OWL based approach provides canonical, unambiguous and machine processable syntax, which can be beneficial in many different ways to the knowledge-base of the domains of the Galatean model of classification. In this section, several benefits of using the OWL based approach will be discussed with relevant examples.

7.6.1 Checking the consistency of the knowledge base of the domains of the Galatean model of classification

The current knowledge base used by the psychological domains and their psychological theory foundation, i.e., the Galatean model of classification, is mainly based on paper specification complemented with some XML schemas. As a relatively new and complicated technology, the existing domain knowledge model may still contain errors. One of the advantages of logic based ontology languages, such as OWL, in particular OWL-DL or OWL-Lite, is that reasoners can be used to compute subsumption relationships between classes and to identify unsatisfiable (inconsistent) classes automatically. Therefore, with our OWL Galatea model, it is possible to find these errors and improve the quality of the Galetean model of classification and its domains. For example, the OWL Galatea model defines that each '*concept*' and '*datum*' node must have exactly one '*RI*'. These semantics are captured by the following OWL constraints.

```
Class: Node
    SubClassOf:
       hasRI exactly 1
```

For example, the '*s-spec*' (i.e. suicide specification) is a sub node of the '*suic*' (i.e. suicide) node. In this case, the node (i.e. '*s-spec*') has been defined with more than one RI, then the OWL reasoner can pick up the inconsistencies in the knowledge-base (see Figure 7.9).

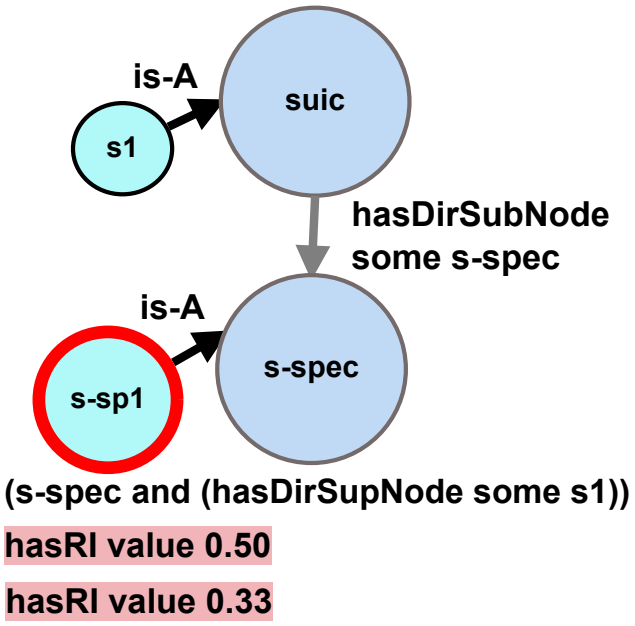

FIGURE 7.9: The s-spec node has two RIs (i.e. 0.50 and 0.33) and an OWL reasoner inferred the knowledge-base as inconsistent.

7.6.2 Automatic knowledge maintenance of complicated knowledge base

The OWL reasoner can automatically maintain the complicated knowledge of GRiST and ADVANCE, such as '*RI*' values. For example, the '*sn-hygiene*' is repeated in different contexts (see Figure 7.10) in GRiST. The '*sn-hyg1*' and '*sn-hyg2*' are the subclasses of the '*sn-hygiene*' class in the ST ontology. The '*sn-hyg3*' (a subclass of the '*sn-hygiene*') is created in the RIT ontology. The '*RI*' is only defined for the '*sn-hyg3*', which is a defined class. The definition of the '*sn-hyg3*' class, compel its sibling classes to become its subclasses. The OWL reasoners can infer all other subclasses (i.e. '*sn-hyg1*' and '*sn-hyg2*') of the '*sn-hygiene*' as the subclasses of the '*sn-hyg3*'. Therefore, the '*RI*' will be inherited and in this way the sub nodes (i.e. '*sn-hygiene*') of a '*G-ConceptNode*' (i.e. '*sn-aprance*') keep a constant '*RI*' in different contexts.

7.6.3 Automatical generation of the ST ontology from the SST ontology

A domain of the Galatean model of classification stores all the knowledge for all its users in a single knowledge base. For the different users and different stages of the

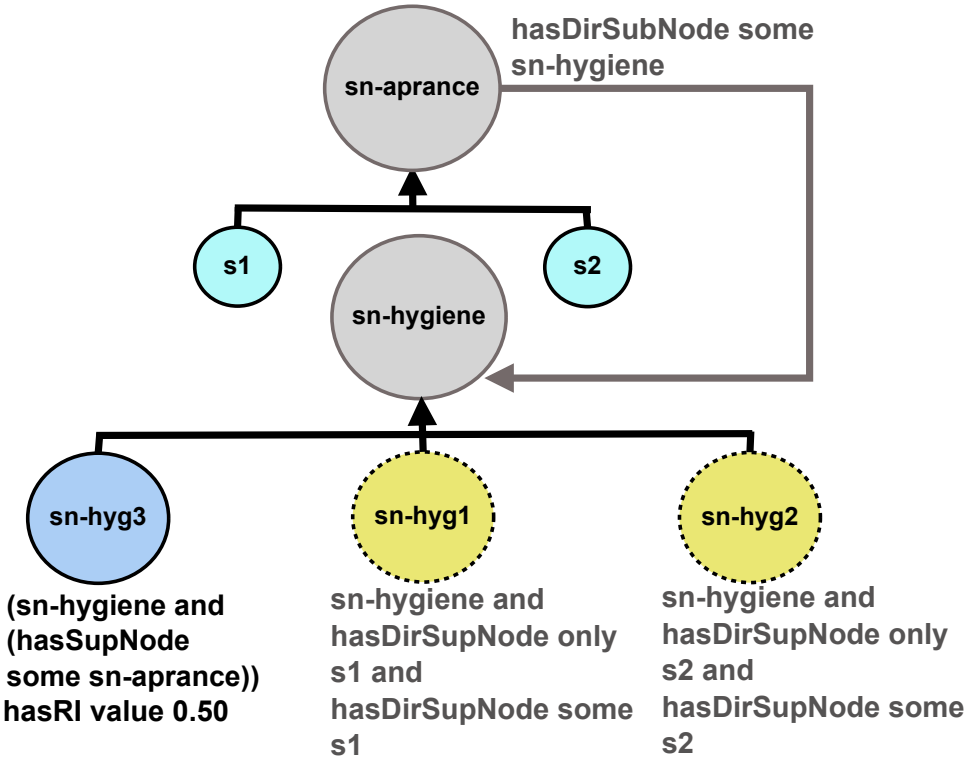

FIGURE 7.10: An example of the sub nodes of G-ConceptNode (i.e. sn-aprance) having RIs. The super classes are represented in gray color and subclasses are depicted in light blue color. The inferred subclasses are depicted in yellow color. The subclass of the sn-hygiene is depicted in blue color in the RIT ontology.

assessment, different aspects of the knowledge base are needed. Currently, GRiST and ADVANCE store their knowledge in separate XML files (such as, ST, RIT, CAT and QT) which are all derived from SST.xml [68, 117]. Maintaining all these XML files and keeping the knowledge consistency between these files are tedious tasks. Our OWL based approach allows us to automatically generate one ontology from another ontology. For instance, to generate the ST ontology for '*child-adolescent*', the users only need to import the SST ontology, and define the following extra restriction to the '*Node*' class in the ST ontology of GRiST's knowledge-base.

```
Class: Node
SubClassOf:
  hasDirSubNode only
    (Node and
    (hasPruneFor exactly 0 child-adolescent))
```

This restriction indicates that for every '*node*' in this ST ontology, it can only have sub nodes to those nodes which have not used this '*hasPruneFor some child-adolescent*' information. OWL reasoner can automatically generate the corresponding ST ontology for '*child-adolescent*' (Note: the '*hasPruneFor*' property kept the rationale of the '*prune-for*' attribute of SST.xml). For example, the '*mental-health-risk*', '*suic*', '*hto*', '*risk-dep*', '*child-prot*' and '*rsk-dp-cr*' nodes are part of the SST ontology (see Figure 7.11).

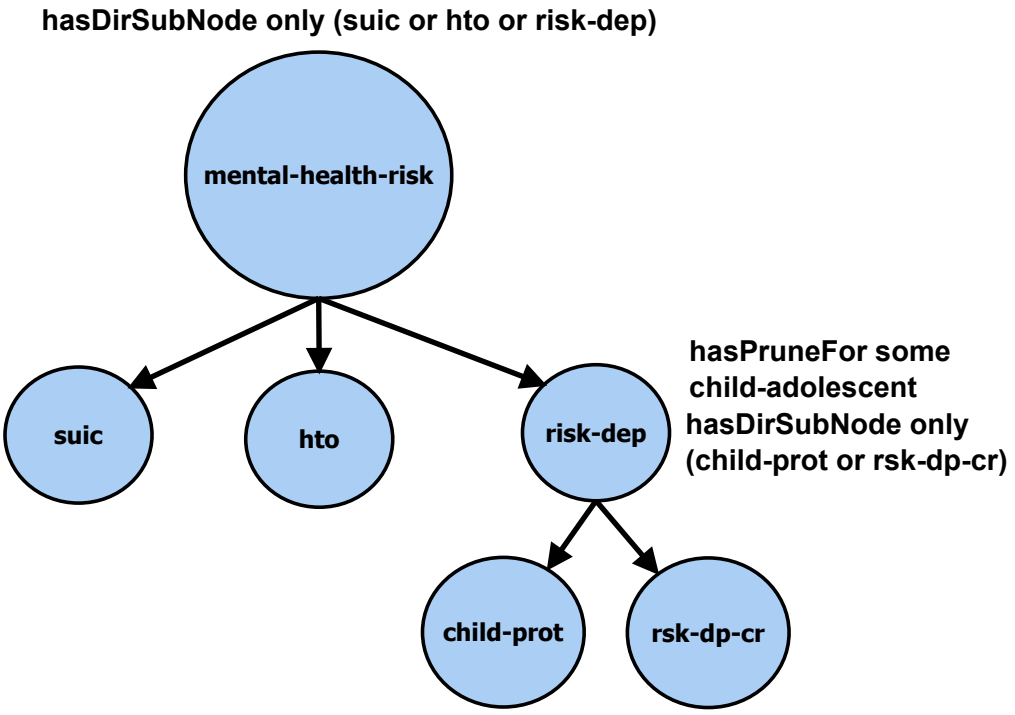

FIGURE 7.11: An example of a logical tree in the SST.OWL.

The relationships via employing '*hasDirSubNode*' and '*hasDirSupNode*' properties are only defined for the '*mental-health-risk*', '*suic*' and '*hto*' nodes in the ST ontology for '*child-adolescent*'. Such relationships are not defined for the '*risk-dep*' node in its super (i.e. '*mental-health-risk*') and sub nodes (i.e. '*child-prot*' and '*rsk-dp-cr*'), because this node has the '*hasPruneFor some child-adolescent*' (necessary condition) information (see Figure 7.12).

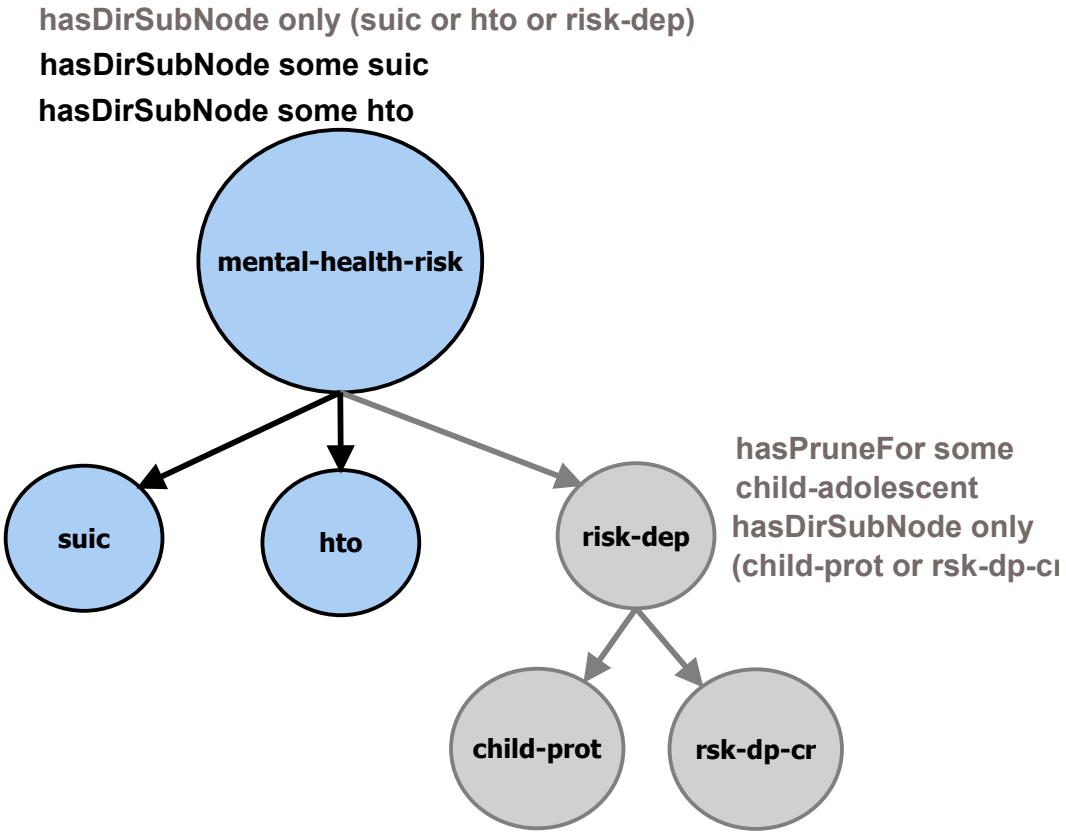

FIGURE 7.12: An example of a logical tree in the ST.OWL. The ignored nodes and edges are depicted in gray color.

7.6.4 Provision of generic and flexible knowledge-base

The flexibility and genericity are very demanding attributes of any knowledge-base. To ensure the flexibility and adaptability of the knowledge-base of the domains, which have theoretical or hypothetical background, is not a simple task. The OWL provides machine interpretable formal codes, which not only can be tested by employing the OWL reasoner, but also results in greater flexibility and elasticity. For example, the '*gen-depression*' ('*G-ConceptNode*') has '*ser-depression*', '*gen-presentation*' and '*gen-congruence*' sub nodes and it is repeated in two different contexts (i.e. '*suic*' and '*hto*') in GRiST (see details in Figure 7.13).

The '*ser-depression*' is repeated in two different contexts, due to its super node (i.e. '*gen-depression*'). Therefore, it must keep only one RI in both contexts of its super node. The '*gen-presentation*' and '*gen-congruence*' are repeated in '*suic*' context and

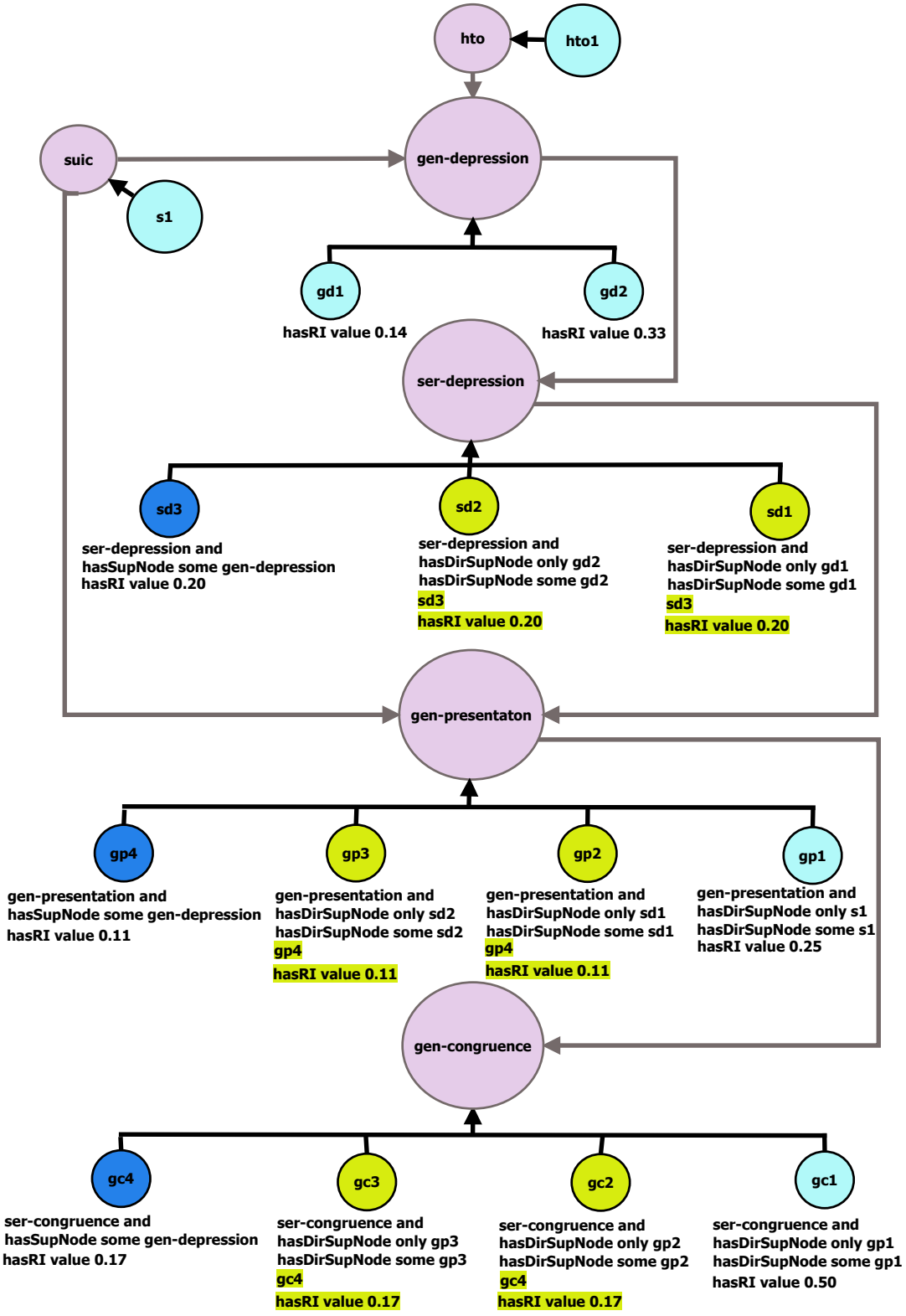

FIGURE 7.13: An example of flexibility of the constraints in the RIT ontology of the GRiST tool.

two contexts of their another ancestor node (i.e. '*gen-depression*'). Both of these sub nodes kept two different '*RIs*', one for the context of '*suic*', and other for the two different contexts of '*gen-depression*'. Their new subclass(es) hold a constant '*RI*' for two different contexts of '*gen-depression*'. The necessary and sufficient conditions of their new subclasses help in automatic, flexible and generic management and maintenance of the knowledge. Therefore, the deletions and insertions of the nodes in the ST ontology between the specified sub node (i.e. '*gen-congruence*') and the '*G-ConceptNode*' (i.e. '*gen-depression*') do not enforce modifying the definition of the subclass(es) (i.e. '*sd3*', '*gp4*' and '*gc4*'). In such case, if some new subclass(es) are created in the '*gen-congruence*' in the ST ontology, then those classes can automatically be inferred as the subclass(es) of the class(es) (i.e. '*gc4*') created in the RIT ontology.

7.6.5 Provision of domain independent transformations

The extension of the OWL Galatea model by employing the case studies of GRiST and ADVANCE (see details in Chapter 6), and their variant examples are demonstrated while in different transformation. Therefore, if the knowledge domains are based on the concept of classification theory, then their knowledge-base can adopt OWL based transformation processes, which are discussed in Sections 7.2 to 7.5. These intelligent transformation algorithms can be utilised in any knowledge domain without exploring its specific terminologies and phraseologies detail.

7.6.6 Provision of coherent and intelligible knowledge-base

The OWL based approach provides simple, reliable, coherent and intelligible knowledge-base for the knowledge domains of Galatean model of classification. It also supports different transformations of the knowledge-bases of such domains. The simplicity and intelligibility of the knowledge-base is increased as the knowledge-base becomes more mature and sophisticated. For example, the CAT and QT ontologies can be transformed by calling some simple queries (see details in Sections 7.4 and 7.5) that are the final states of the knowledge-base. A number of QTs can be inferred without exploring the syntax structure details of the concepts and datum components either from the ST or RIT ontology.

7.7 Conclusions

The OWL based approach provides a great number of advantages to the knowledge domains belonging to the psychological model (i.e. Galatean model of classification) and having mutative data structure. For instance, the OWL Galatea model can be imported in the root (SST) ontology of any domain based on the idea of classification theory. For deriving the next state of the knowledge-base, the previous state is imported, which helps in reducing the replication of knowledge. Each state of the knowledge-base can be tested, evaluated and used individually, and collectively if it has some prior states. The constraints in each state also support in the transformation of the next state without utilising the expertise of a human expert. The OWL based structures make it convenient to transform the complex progressive data structure of the domains of the Galatean model of classification into the formal structures, without losing their psychological validity and transparency.

The first state (SST) behaves as a universal set of knowledge for all assessment types within a domain. The next state(s) are reduced sets of knowledge, that are derived for a specified assessment type. The number of concepts and datum nodes are less than the imported ontologies, but these components have more constraints as compared to the first state. The second uncertainty attribute (i.e. '*RI*') of the Galatean model of classification is initialised in each concept and datum node in the second last state (i.e. RIT) of the knowledge-base. The '*RIs*' of the sub nodes of different concepts are initialised under some criteria and the OWL reasoner helps in deriving the knowledge pieces according to the given criteria or constraint. The consistency of the new and existing constraints of the concepts and datum nodes can be confirmed by using the OWL reasoner.

The CAT is a final state that actually presents to the specified assessment user for the assessment. The knowledge-bases of the assessment tools are managed from first state (i.e. SST) to third state (i.e. RIT) in a way that helps in further dynamic transformations. The CAT is only deduced by determining the genuine attributes of the concepts and datum nodes of the RIT ontology. The CATs and QTs can be derived by executing some ordinary queries according to the experience level of the assessment

user. The prior states (i.e. SST, ST and RIT) provide a greater flexibility and support in deducing the QT ontology either from the ST or from the RIT ontology.

Some advantages of using OWL-based techniques and approaches while variant transformations of the knowledge were discussed in the end of this chapter. The OWL based approaches support the generic knowledge management of the knowledge-base, where the undesired knowledge chunks can be discarded without removing them physically. The OWL based techniques also lead to the automatic knowledge maintenance of the complicated knowledge-bases of the domains which belong to human psychology. The OWL based approaches improve the knowledge engineering processes embedded in the decision support system that are required for its continuing evolution during use. The last but not least benefit is that the consistency of the knowledge-base can be checked or confirmed by using the OWL reasoner. It is concluded that OWL based constraints help in defining a generic and flexible knowledge-base.

The OWL has been used to manage the different transformations of the knowledge and it does not focus on only the hierarchical relationships, but the OWL based approaches also provide a number of benefits, which were discussed in detail in this chapter. The OWL based approaches also provide some other benefits, such as generating knowledge at run-time for a new use from the exiting knowledge. In the next chapter, the dynamic generation of the adaptive assessment questionnaire will be discussed in detail.

Chapter 8

Dynamically Generation of the Adaptive Assessment Questionnaire

8.1 Introduction

Human expertise is required for the maintenance and adaptation of the knowledge-bases having theoretical and hypothetical contexts. The OWL Galatea model (see details in Chapter 5), provides the similar expertise for the domains of the Galatean model of classification, which is based on hypothetical theory. Human cognitive skills become more mature with the passage of time or by learning from different experiences or exercises. The OWL based approach provides analogous structures such as: the previous states of the knowledge can be imported into the current state of the knowledge (see details in Chapter 7), and the machine behaves as a mature agent and make decisions without the guidance of human experts. The techniques used in the OWL Galatea model provides the opportunity to minimise the involvement of human experts in maintenance and adaptation of the knowledge-base without exploring the domain specific knowledge details.

The OWL Galatea model can be used in the knowledge-base of any domain, which is based on the idea or theory of classification (see details in Chapter 6). It also facilitates

the assessment tools of such domains in reducing the replication of knowledge and help their assessment users in making correct and precise judgments. For example, a user wants to screen out his or her general mental health issues or risks, therefore, the assessment tool should be capable of keeping the knowledge regarding general health issues of a patient, but it must keep some specific information on the basis of his or her personal attributes like: age, gender, marital status, ethnic group and some other attributes, which help in making more precise and accurate decisions. For the worthiness of the assessment tool, a flawless knowledge management of such attributes is necessary. The correct knowledge management of such attributes increases flexibility, dynamicity and robustness in the knowledge-base. In such case, if the knowledge of an attribute is ignored and not managed according to the directions and guidelines of the domains' expert, then a big portion of the knowledge-base is left to be managed or we can say the knowledge-base is incomplete.

The assessment user is the central entity of the assessment tool and the rest of knowledge revolves around it. For instance, the concept nodes keep the constraints of the '*hasLevel*' attribute, which indicates the experience level of the assessment user in a specified knowledge domain. The contents of the questions are also designed and applied according to the particular group of user. For example, the "*In your judgement, to what extent is the young person at risk of suicide?*" demonstrates that contents belong to a young or child-adolescent user of GRiST. The classification process in the final state of the knowledge-base is also triggered by the user given data, which is compared to datum values of the value-mg lists associated with particular datum nodes. In this chapter, the mandatory features or attributes of the assessment types are highlighted, these are mentioned in the paper manual of the knowledge domains of the Galatean model of classification, but the current pieces of software (i.e. XML trees) lacks such information. The way in which the management of these mandatory features can boost the strength and flexibility of the knowledge-bases will be demonstrated. The constraints and relationships are designed in the OWL Galatea model in this way that not only support the management of various hierarchical structures, but also help to extract information for new use and purpose at run-time from the existing knowledge. In this chapter, the main focus is on the dynamic generation of an adaptive assessment questionnaire without the involvement of human experts.

8.2 The assessment user and its attributes

The dynamic and flexible knowledge-bases keep more attraction than static ones. The flexibility and elasticity are the characteristics that increase the attraction of the assessment tool for its users. The assessment type is a central entity in the human psychology domains. The gender, marital status, ethnic group and age are essential demographic information [122] of an assessment type. The core knowledge or information of such mandatory attributes is ignored in the current knowledge hierarchies (i.e. SST, ST, RIT, QT and CAT xml trees). Such knowledge is only part of their paper manual. For making the flexible, dynamic and robust knowledge-base, the core attributes of the assessment types are managed, while the transformation of the SST ontology (see details in Chapter 6) by adopting OWL based approaches. The generic constraints regarding these attributes (i.e. '*hasAge*', '*hasEthinc-Group*', '*hasGender*' and '*hasMaritalStatus*') are defined in the '*Assessment-Type*' class in the OWL Galatea model (see details in Chapter 5). The specific constraints and relationships of these features are managed during the knowledge engineering of the knowledge-base (see details in Chapter 6) within the domains.

8.3 Generation of an adaptive assessment questionnaire according to the age of the assessment user

The required knowledge can be inferred and retrieved by employing different properties or characteristics of the assessment types. Here, it is going to demonstrate how an adaptive assessment questionnaire can be retrieved from the knowledge-base by passing different attributes as parameters of the DL-queries (see figure 8.1). The process is started by presenting the whole knowledge-base or SST ontology to an assessment type for assessment.

The domains of the Galatean model of classification have huge and complex data. For example, the GRiST tool maintains a large member of questions associated with Galatean nodes. Depending on the assessment type (patient's type), different Galatean node are accessed. Even for the same Galatean node, for different types of population, different questions might be asked. The effectiveness of the assessment tool largely

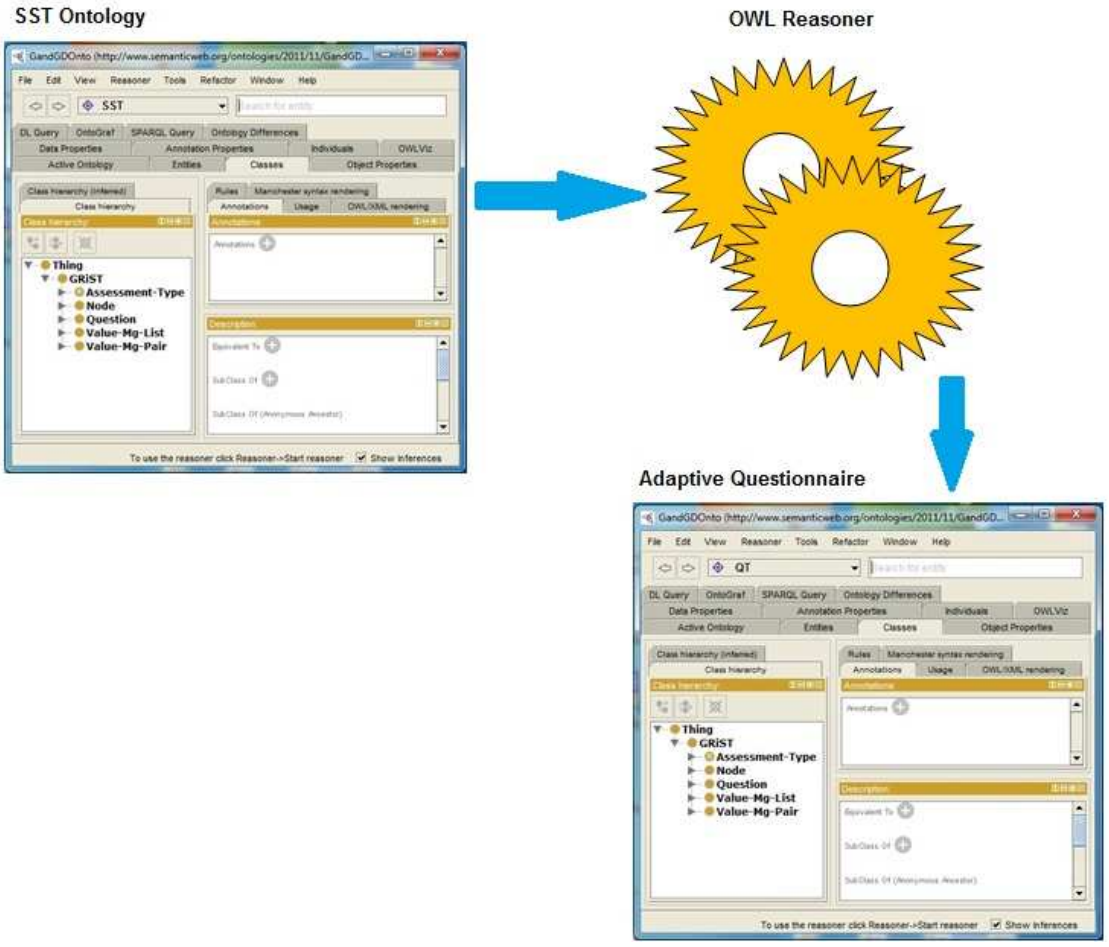

FIGURE 8.1: The generation of an an adaptive assessment questionnaire from the SST ontology.

relies on the right questions to be asked to assessment type (i.e. patient). The answers of these questions provide good indicators whether a particular clue is presented in the classification process. A challenge was how to design such a questionnaire which is the most suitable to be able to capture critical individual information accurately. OWL allows us to represent the semantics of each assessment population, which brings the possibility to dynamically generate a personalised questionnaire based on patient profile. The knowledge of the assessment types of GRiST is defined in a flexible and dynamic way. In this case, a patient is a '*child-adolescent*' and he also provides his or her date of birth information. The information system of GRiST generates an instance of the '*Assessment-Type*' class and initialises it with the given value by employing '*hasAge*' property. The information system by using an OWL reasoner infer the type of the object. In this case, if the user given data is 65 then the reasoner will infer the type of the object is '*older*'. In this case, if the given data is '*09*' then the object is inferred as

an object or instance of the '*child-adolescent*' class, because of its necessary and sufficient condition.

```
Class: child-adolescent
    EquivalentTo:
        Assessment-Type
         and (hasAge only xsd:int[>= 1])
         and (hasAge only xsd:int[<= 15])
```

The '*child-adolescent*' class is employed in the DL query for inferring the concepts and datum nodes those having the questions' contents related to it (i.e. child-adolescent) and does not miss out any information.

```
hasQuestion only (appliesTo some child-adolescent)
```

The flexibility of the knowledge-bases is demonstrated by utilising the '*hasAge*' attribute of the assessment types.

8.4 Benefits of adaptation

The adaptive behaviour which embeds with the assessment questionnaire expressed many benefits such as the provision of run-time adaptive information and minimised the number of progressive states of the complex knowledge.

8.4.1 Provide run-time adaptation

It is quite attractive to create knowledge at run-time for a new use or purpose from existing knowledge or facts. The OWL-approaches support in extraordinary ways that the adaptive knowledge is inferred at run-time by determining the attributes of the assessment user. Any user of the decision domain can provide his or her age as the seed for getting related assessment questionnaire. It can seamlessly adapt the structure and content of a questionnaire to the user interaction.

8.4.2 Reduce the uses of a series of complex knowledge trees

The approach used for generating adaptive questionnaire discourages the use of a series of knowledge trees for knowledge management of the decision domains of the Galatean model of classification having complex data structure. The idea of the Super Structure (ST) can be achieved when the knowledge is inferred through the queries discussed in Section 8.3. The relationships between and sub and super nodes can be defined by employing '*hasDirSubNode*' and '*hasDirSupNode*' properties as discussed in Chapter 7 and Section 7.2. The RIs of the whole Galatean hierarchy can also be initialised by using the '*hasRI*' property in the similar way as the constraints discussed in Chapter 7 in Section 7.3. The decision support systems developed in XML combine or integrate the question and classification tree for making decisions [62]. The proposed system does not need to integrate these trees, the question tree keeps all the knowledge required for the classification and decision making process. Some queries can be called for extracting the information according to the experience level of the assessment user. Finally, the logical tree structure is ready for decision making and classification process. In this way, the complex states of the knowledge are reduced from five to two states (i.e. SST and CAT).

8.5 Conclusions

OWL has potential to provide knowledge structures, which help in extracting new knowledge from existing ones. An adaptive assessment questionnaire is a pretty basic application of OWL representation, but the depth of its encapsulation of the knowledge semantics provides great flexibility for multiple purposes in the knowledge engineering and application domains. OWL provides additional functionality that is not automatically included with the previous decision support systems that are based on only XML representation. The correct and precise relationships of the knowledge help in inferring the required knowledge in multiple ways. If the domain experts describe the question contents association with other attributes (i.e. '*hasGender*', '*hasMaritalStatus*' and '*hasEthnic-Group*') of assessment user, then it provides more flexibility and adaptability in the knowledge derivation. For example, if the contents of the questions are designed by considering the gender differences of the assessment user, than affiliated concepts

and datum nodes can be inferred from the knowledge-base accordingly. It is concluded that the correct management of the attributes of the assessment type can enhance the flexibility and elasticity of the knowledge-base and support in reducing the number of complex mutative states.

In the next chapter, the evaluation on the findings of the research is presented with suitable real life case-studies. The chapter will also present the comparison of OWL derived XML files with the present XML files.

Chapter 9

Evaluation

9.1 Introduction

This chapter presents the evaluation of the core aims and objectives of this study such as: simplify the replication of knowledge, design machine interpretable specification, instantiate the uncertain variables, manage consistent and logically correct structures, represent the success of the OWL-based specification and support the adaptive generation of knowledge structures for new purpose and context. An empirical evaluation of the XML structures derived through the OWL-based specification is presented and the results obtained are discussed, and the conclusions drawn. The GRiST is an application of the Galatean model of classification. The experts of the GRiST want to keep the knowledge structures in the XML format but for the accuracy and reliability of these XML structures, a machine interpretable specification is essential and required. It is difficult for the developers to meet users' requirements and make alterations in a XML knowledge-base through the existing programs or processes by merely consulting and updating the paper manual. In this chapter, the present knowledge structures developed through the traditional programs or processes are compared with the knowledge structures derived through the machine reasoning and interpretation. Also, some case studies of the GRiST besides the comparison of present XML structures (i.e. SST, ST, RIT, QT and CAT) with the XML structures generated through the OWL-based specifications are discussed in detail. Finally, in the last section, a brief overview of the advantages of our proposed approach is presented.

9.2 Case-studies

Two real world case studies of the GRiST tool are going to be presented here. The purpose of these examples is to demonstrate the essential and significant benefits of the OWL-based specifications designed through this study.

9.2.1 Knowledge structure modification according to user's requirements

The experts of the GRiST have aimed to facilitate maximum requirements of the user(s) or patient(s). The prime challenge was to add flexibility in the knowledge management process so the knowledge-base can easily be adapted. The patient may access mental health services of many different types and thus in multiple ways. These variations are supported by the OWL-based approach, which can maintain the required variations in the GRiST knowledge structures. A user may want to modify the structure and alter it according to his or her personal or individual requirements and necessities. For example, a patient may present a risk of harm to others and especially to his or her dependents. The knowledge structures of the GRiST are basically managed in the mind map (see details in Section 3.2 and Chapter 3) files which are expressed in the XML formats. Figure 9.1 depicts the associated hierarchical knowledge structure in the mind map file, where the "Risk-to-dependents" node is a sibling node of the "harm-to-other" node.

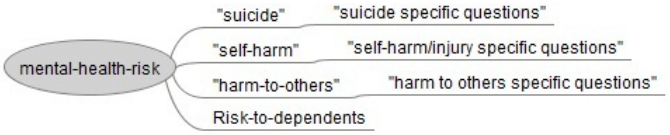

FIGURE 9.1: Initial version of SST mind map file.

A user altered the knowledge structure and made the "Risk-to-dependents" node as a sub node of its sibling node (i.e. "harm-to-other") in the mind map file (see Figure 9.2).

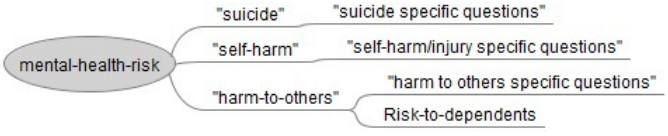

FIGURE 9.2: Modified version of SST mind map file.

These modifications create issues for the experts because the knowledge or nodes may lose essential information or attributes (such as help, label, question, etc.) while the hierarchical structure is being modified. Therefore, it becomes difficult to manage and manipulate the modified knowledge structure any further (i.e. ST, RIT, QT and CAT) through a simple language (i.e. XML). In contrast, the '*hasDirSubNode*' and '*hasDirSupNode*' OWL properties are defined for creating the knowledge structures (see Figure 9.3) and some other properties (such as hasHelp, hasLabel, hasQuestion, etc.) are used exclusively for the knowledge management purpose.

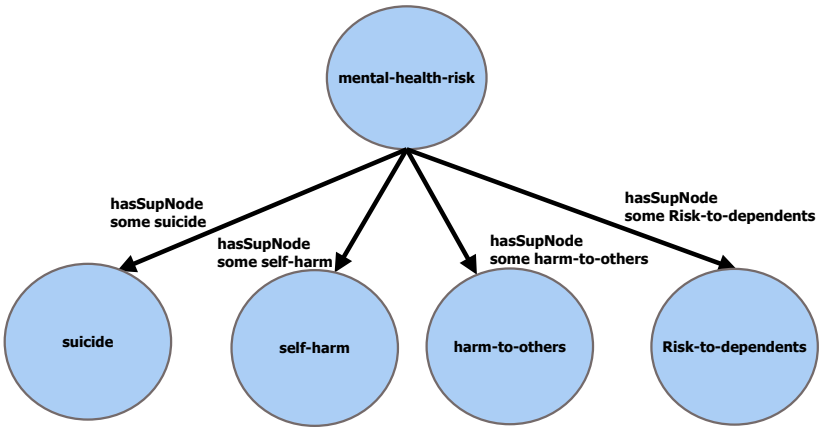

FIGURE 9.3: Initial version of OWL file.

Through our proposed approach, it is possible to drop the constraints or relationships of the '*hasDirSubNode*' and '*hasDirSupNode*' properties, which are created for the development of knowledge structures. In this manner, it becomes easy to re-build the new structures of the knowledge-base by using said properties according to the modifications given by a potential user (see Figure 9.4).

The benefits of the OWL-based approach are that the developers or knowledge engineers do not need to redefine the core knowledge or constraints of the nodes. In contrast, it is difficult to alter the hierarchical structure of the nodes directly in the XML format without undermining some other constraints. For example, the obvious place to change structures is in the SST where the full structure of nodes is defined in one place. However, paths to the single full definition are required from all other locations of the node. If any of these locations are changed, so must be their path, which is a non-trivial task to manage for knowledge engineers, let alone domain experts.

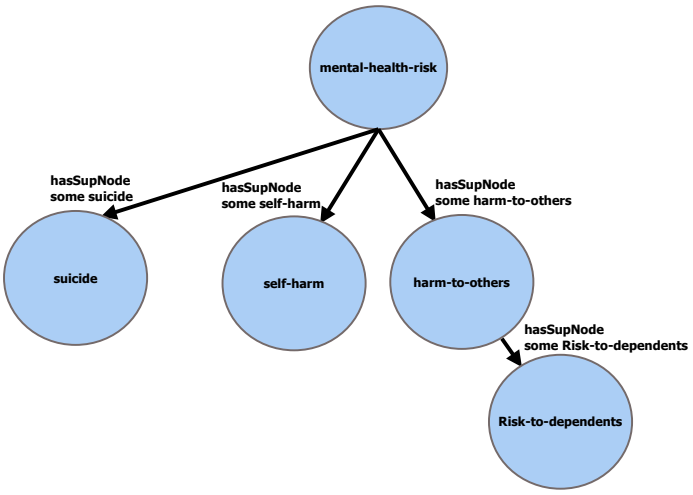

FIGURE 9.4: Modified version of OWL file.

9.2.2 Knowledge management for the forensic service

The GRiST is a multi-deployed tool that is being used in various organisations such as National Health Service (NHS), private hospitals, welfare organisations, charities, and for the case of the people who are in lock ups (the experts gave name a template-forensic service [123]). The current issues with the GRiST arise mainly when it is applied to child-adolescent, older, service-user and working-age users or patients having the template-forensic service, in these cases the wrong approach is to keep separate XML files which have huge bulk of similar knowledge with very little variations, one for existing patients and another for the template-forensic service. Therefore, the knowledge engineers have decided that the SST.xml should keep the knowledge for all patients and employ template-forensic as a service, however this workaround creates challenges for the transformation process of the ST structure with the XML representation. This is due to the reason that to find and add the knowledge in the ST structure that is unsuitable for a population (let's say working-age) but that is suitable for the template-forensic service through the existing computer programs or processes at the moment is impossible. Hence, most of the knowledge structures are maintained manually in the XML format. Previously, there was a complex or difficult or manual search procedure to find relevant information. The OWL allows us to automatically extract the required information.

Certain knowledge chunks are associated with the template-forensic service and at the same time some knowledge pieces are pruned for (or unsuitable to) a specified population (let's say working-age) in various contexts of the core knowledge (i.e. SST structure). The template-forensic service information has been added into the label, help and question attributes in the specified concepts and datum nodes. These attributes related information is managed in a way that helps to extract information easily and accurately in the SST.owl (see details in Chapter 6 and Section 6.3.3). On the other hand, it was quite difficult to extract the knowledge related to the template-forensic service along with the working-age population by using the present existing approach. The tool's developers manage the customised knowledge tree (i.e. ST for the working-age population) manually at the moment. In contrast, our designed OWL-based approach makes it convenient and easy to manage both the service and the population related information by using a simple DL-Query.

```
(hasLabel only (appliesTo some template-forensic) or
hasHelp only (appliesTo some template-forensic) or
hasQuestion only (appliesTo some template-forensic)) and
(hasPruneFor  some working-age)
```

The above query extracts all the nodes that have association with the template-forensic service and even are pruned for the working-age population and such nodes are added into the ST knowledge structure. The nodes that are purely pruned for the said population and do not have any information related to the template-forensic service are eliminated into the customised structure (i.e. ST).

9.3 Comparison of the present XML structures with the XML structures generated through the OWL-based specifications

This section presents the comparison of the present XML structures with the XML structures generated through the OWL-based specifications. XML Notepad 2007[1] is used for

[1] https://xmlnotepad.codeplex.com/

this purpose. This tool highlights the unusual or mismatched contents of both XML files in different background colours. For instance, the added contents are represented in yellow background colour, the removed contents are depicted in red background colour, the changed contents are portrayed in green background colour, the contents moved from and moved to are showed in blue background colour and the ignored or matched contents are displayed with no background colour.

9.3.1 SST structure

This tree structure keeps whole knowledge and the rest of the structures keep its customised views for the specified population(s). Chapter 6 and Section 6.3 presents details on the translation or development of the SST ontology from the SST.xml.

9.3.1.1 Path translation

The generic nodes keep definition only in one context and hold path information where it repeats in associated context(s) of the present SST.xml and ST.xml structures (see details in Chapter 3, and Sections 3.3.2.1, 3.3.3.3, 3.3.3.4). For example, the “gen-sh-cuts” is the sub node of the “suic” (i.e. suicide) node and it has the “generic-datum” attribute (see details in Figure 9.5). The said sub node has path information in an individual risk or context (i.e. “suic”) and its definitions is given at some other context(s) (i.e. “sh” (self-harm)). The contents of the said attribute helps to translate the sub nodes (i.e. “gen-sh-cuts”) with their complete definition in their associated contexts (i.e. “suic”) in the RIT.xml.

The paths are translated in the form of correct relationships between sub and super nodes in the SST.owl file (see details in Chapter 6 and Section 6.3). The functionalities of the the “generic-datum” attribute is managed by employing the “hasDirSubNode” and the “hasDirSupNode” properties. The “gen-sh-cuts” node is defined as the sub node of both nodes or contexts (i.e. “suic” and “sh”) via employing the “hasDirSubNode” property in the SST.owl file. Hence, no syntax based information is stored or maintained in the knowledge-base and the replicated knowledge is simplified with its correct semantics. The OWL reasoner extracts the relevant knowledge via the “hasDirSubNode” property (see details in Chapter 6 and Section 6.3.4), while the knowledge-base

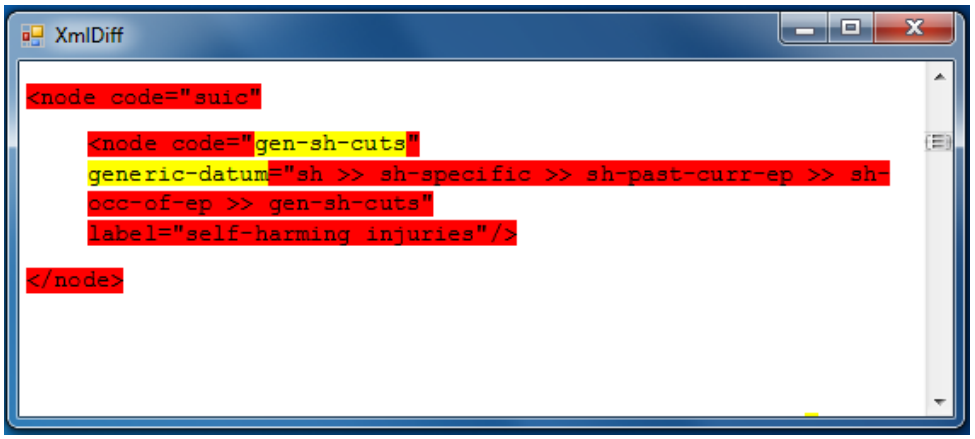

FIGURE 9.5: The present SST.xml file has a path for representing sub and super node relationships. Note: the highlighted text is depicted in yellow background colour.

is translated back to SST.xml from SST.owl. The text highlighted in yellow background colour in Figure 9.6 shows the “gen-sh-cuts” node in the “suic” context with its complete definition in the SST structure generated through the OWL-based specification.

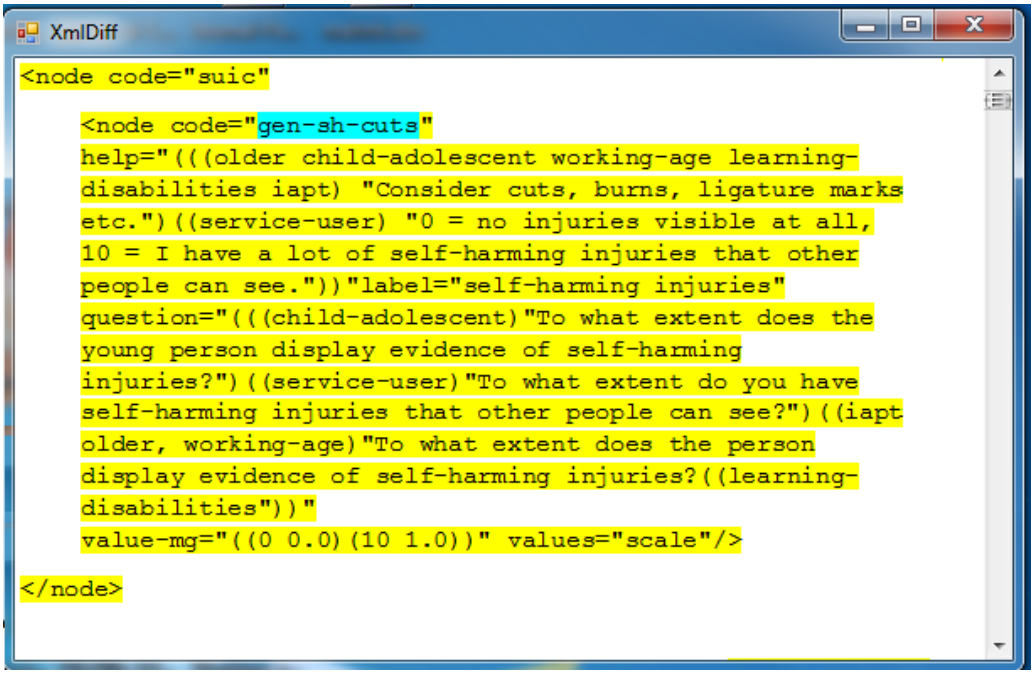

FIGURE 9.6: The SST.xml file generated through OWL-based specification facilitates the logical correct representation of the knowledge in the form of sub and super node relationships which does not need to be modified in its next state or structure. Note: the highlighted text is depicted in blue background colour.

9.3.1.2 Minimise the syntax dependency of the knowledge

The datum nodes that have the “multiple-tick” attribute (see details in Chapter 3 and Section 3.3.3.11) are generic or repeating nodes and these are located only in one location or context of the SST and ST trees. The said attribute holds the information related to the sub nodes of the specified node(s) (see Figure 9.7).

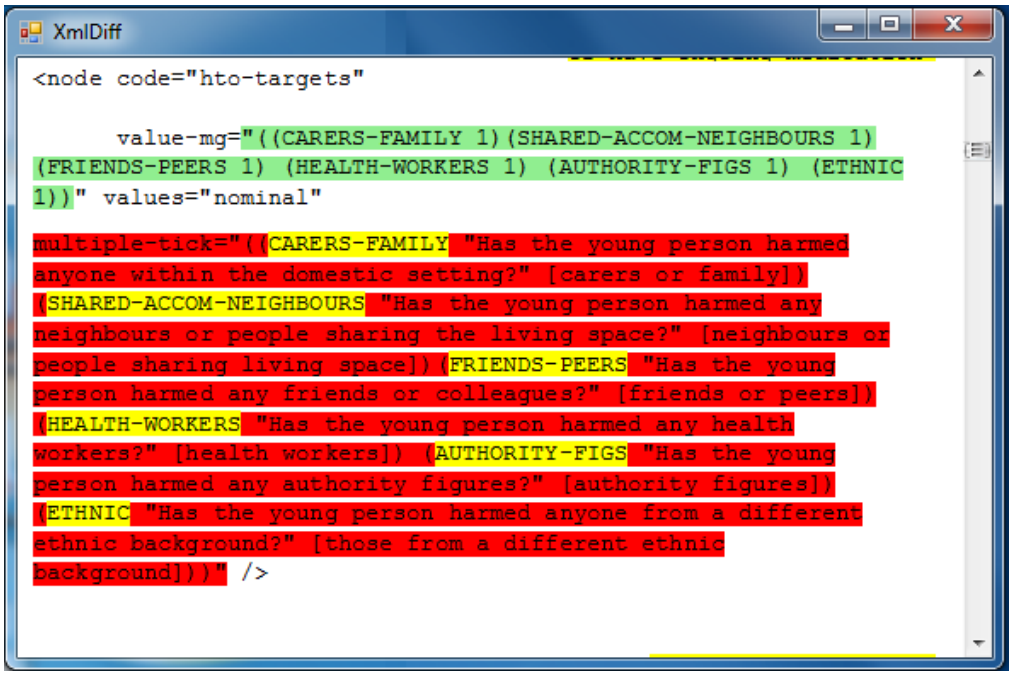

FIGURE 9.7: The “hto-target” node have the “multiple-tick” attribute in the SST.xml file. Note: the highlighted text is depicted in yellow background colour.

The presence of the said attribute in the specified node indicates that this node must transform from datum node to concept in the RIT structure (see details in Chapter 3). The sub nodes of the specified node are extracted from the contents of the said attribute and their uniform RIs are supposed to initialise within various contexts of the present RIT.xml. In contrast, a generic format or law for creating the sub and super node relationships in the SST.owl is followed (see details in Chapter 6 and Section 6.3.10). The specified nodes do not enforce to modify their type in the RIT ontology because these are already defined as concepts in the SST ontology. Therefore, this approach facilitates in minimising the formation of the syntax-based information in the knowledge-base. Other benefit of this strategy is that the developer does not need to

re-create a separate customised RIT knowledge structure. A comprehensive and accurate knowledge structure of the SST ontology is maintained that supports its subsequent knowledge states (i.e. ST, RIT, QT and CAT) which keep only further or required information in the form of OWL constraints (see details in Chapter 7). Therefore, if any alteration is made in the OWL Galatea specification or SST ontology then that changes directly adapt to the next or derived structures (i.e. ST, RIT, QT and CAT). Figure 9.8 depicts that the specified node does not have such an attribute, but it keeps sub nodes in a similar way as these are the part of the present RIT.xml structure.

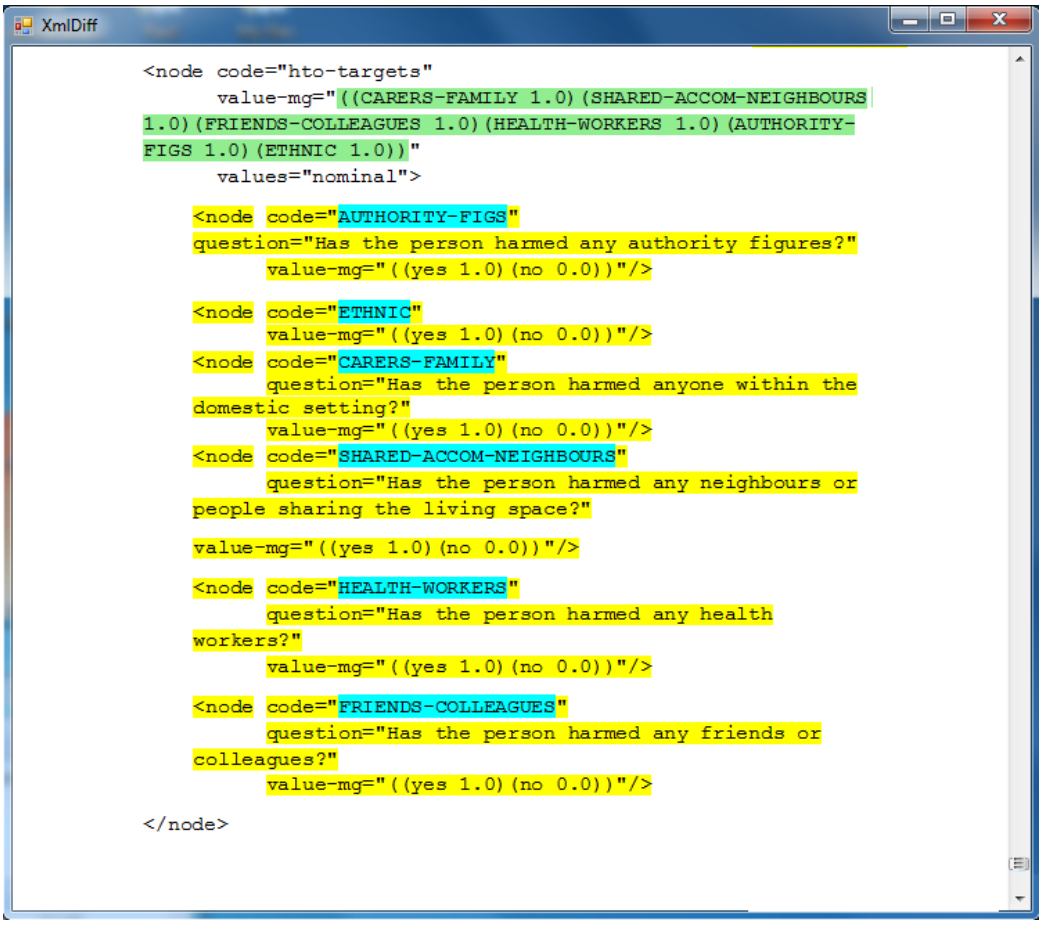

FIGURE 9.8: The “hto-target” node and its sub nodes without using the “multiple-tick” attribute in the SST.xml file derived from the SST.owl. Note: the highlighted text is depicted in blue background colour.

9.3.1.3 Determination of the nodes types

The “generic-type” attribute (see details in Chapter 3) is used for discriminating the type of the node in the SST.xml but this issue has been solved by subsumption relations

defined in the SST.owl file (see details in Chapter 6 and Section 6.3). For example, the "hto-targets" is defined as the subclass of the GD-DatumNode class. Therefore, when a node is inferred then its type is auto inferred while translating it into a XML format.

9.3.1.4 Generic section

A separate (i.e. generic) section is managed in the present SST.xml and ST.xml structures (see details in Chapter 3, and Sections 3.3.2.1, 3.3.3.3, 3.3.3.4). For instance, the "gen-representation" is a generic or repeating node that has path reference in the individual risk(s) (see details in Chapter 3 and Section 3.3.3.10) and this node has complete definition in the generic section of the SST.xml and ST.xml structures.

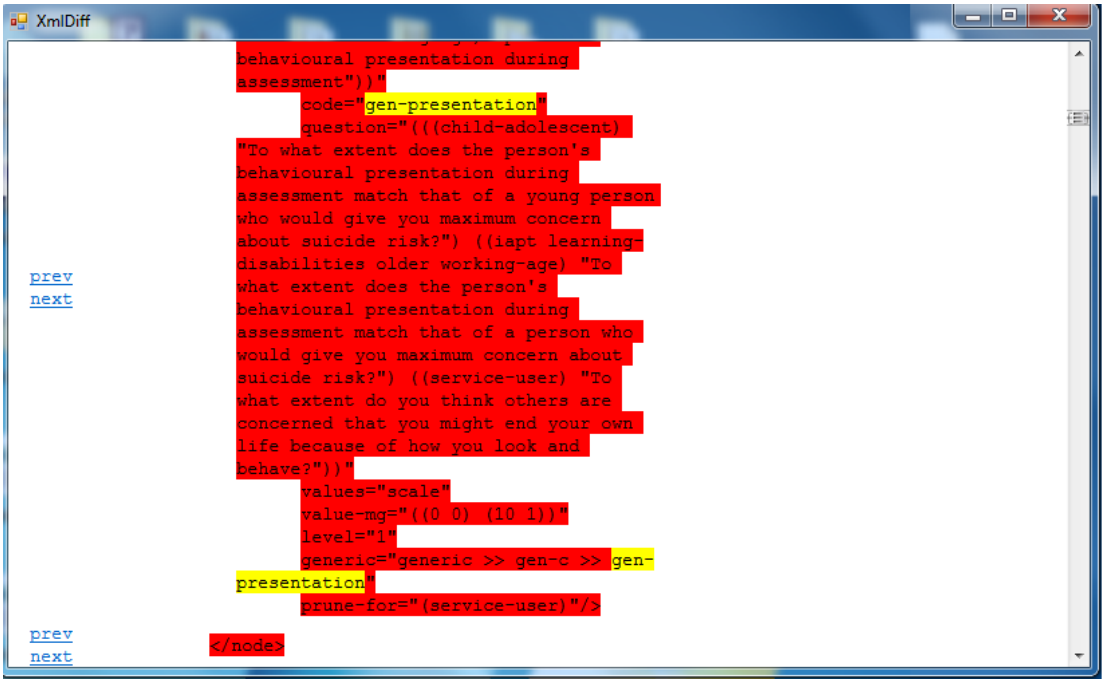

FIGURE 9.9: A repeating or generic node (i.e. "gen-presentation") has a path in an individual risk. Note: the highlighted text is depicted in yellow background colour.

The generic section actually holds repeating or generic nodes. This section is completely removed from the present RIT.xml structure and its sub nodes are restructured as the sub nodes of the individual risk(s) and the complete definition is taken from the path (see Figure 9.9) given in the "generic" attribute (see details in Chapter 3, and Sections 3.3.2.1, 3.3.3.3, 3.3.3.4). Furthermore, certain interim attributes such as the "level-question" and "level-code" (see Figure 9.10) attributes are used for restructuring in the present RIT.xml (see details in Chapter 3), but such nodes are linked to their

associated individual node(s) in the SST.owl file (see details in Chapter 6 and Section 6.3.4).

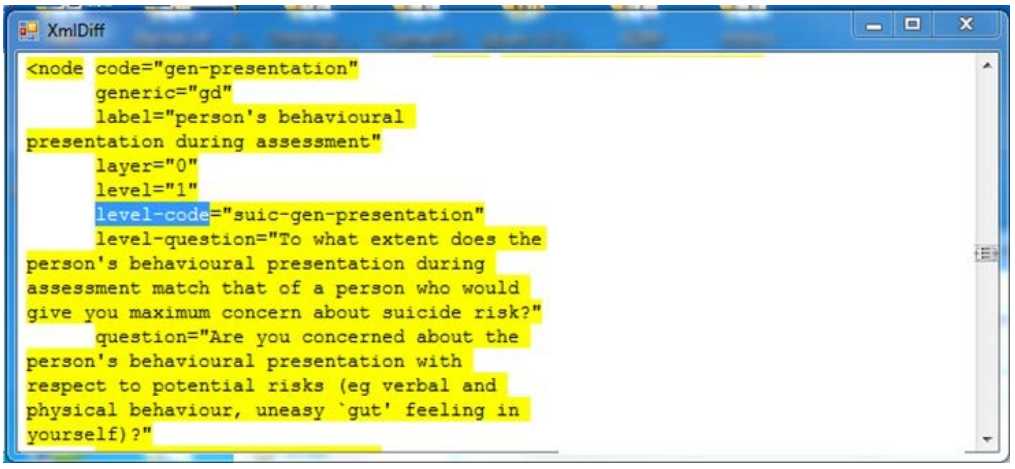

FIGURE 9.10: A repeating or generic node (i.e. “gen-presentation”) has “level-code” attribute. Note: the highlighted text is depicted in blue background colour.

So, when the translation of the SST.xml is derived from the SST ontology then all repeating or generic nodes are properly linked to their associated individual risks or contexts via using “hasDirSubNode” and “hasDirSubNode” properties. Such generic nodes keep their complete definition in all contexts of the SST.xml translated from the SST.owl. For example, the “suic” is super node of the “gen-presentation” which is a generic node and repeats in some other contexts (see details in Chapter 3 and Section 3.3.2.1). Figure 9.11 depicts that the “gen-presentation” keeps its complete definition in the “suic” context.

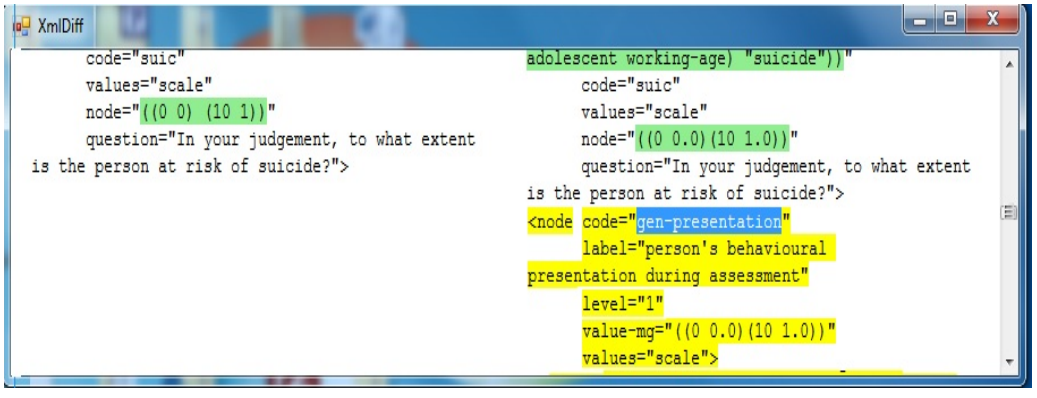

FIGURE 9.11: A repeating or generic node (i.e. “gen-presentation”) is inside an individual risk (i.e. “suic”) with its complete definition in the SST.xml generated from SST.owl. Note: the highlighted text is depicted in blue background colour.

This strategy makes the knowledge-base stable and consistent. Hence, the developers do not need to create certain run-time attributes such as the “multiple-tick”, “level-question” and “level-code” attributes (see details in Chapter 3) for the transformation of knowledge into its next or incremented state.

9.3.2 ST structure

The ST hierarchical structure is a customised view of the SST tree for a specified user. The process for extracting the ST.owl from the SST.owl is comprehensively presented in Chapter 7 and Section 7.2.

9.3.2.1 Unsuitable knowledge chunks

Present XML-based knowledge structure keeps some unsuitable or prune for (see details in Chapter 3 and Section 3.3.3.7) knowledge pieces for a specified user. For instance, the “gen-cog-think-mem”, “ld-probs-indep-liv” and certain other nodes have prune for information for the working-age user but these nodes are still a part of the present ST.xml (see Figure 9.12). These considerations produce many issues in the knowledge-base and leads to ambiguities.

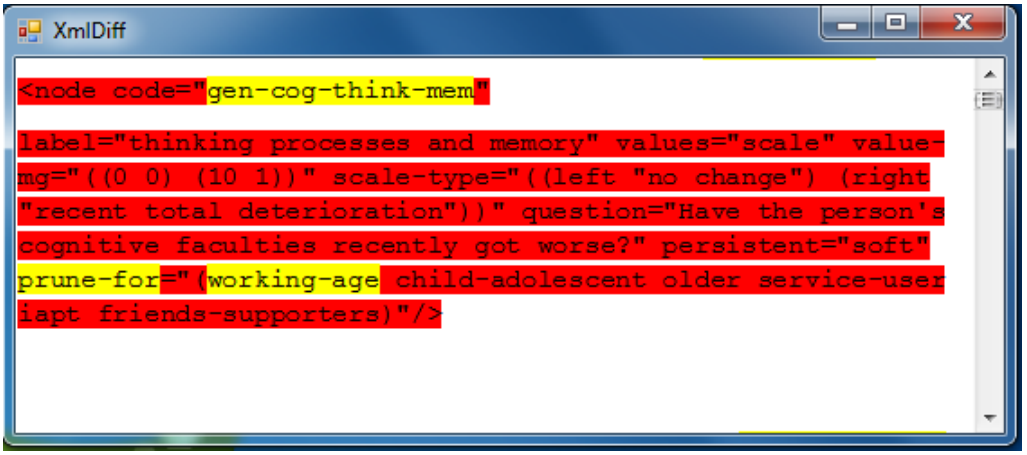

FIGURE 9.12: The “gen-cog-think-mem” is a unsuitable node for the working-age user but still it is a part of the present ST.xml file. Note: the highlighted text is depicted in yellow background colour.

9.3.2.2 Syntax independent knowledge management

Present techniques and approaches used for the development of ST structure create many issues and also abruptly remove the essential knowledge chunks. For example, the “gen-pass-aggress”, “hto-forced-by-others” and “gen-detached” nodes are pruned for the working-age population but their questions’ contents are associated with the template-forensic service. In a similar way, the “sh-change-mth” node is removed due to mere judgments of the “prune-for” attribute associated with the working-age through such programs instead this node has the “help” and “question” information related with the forensic-template service. Figure 9.13 depicts the “sh-change-mth” node which keeps the contents or information related to said service beside it is unsuitable for working-age user.

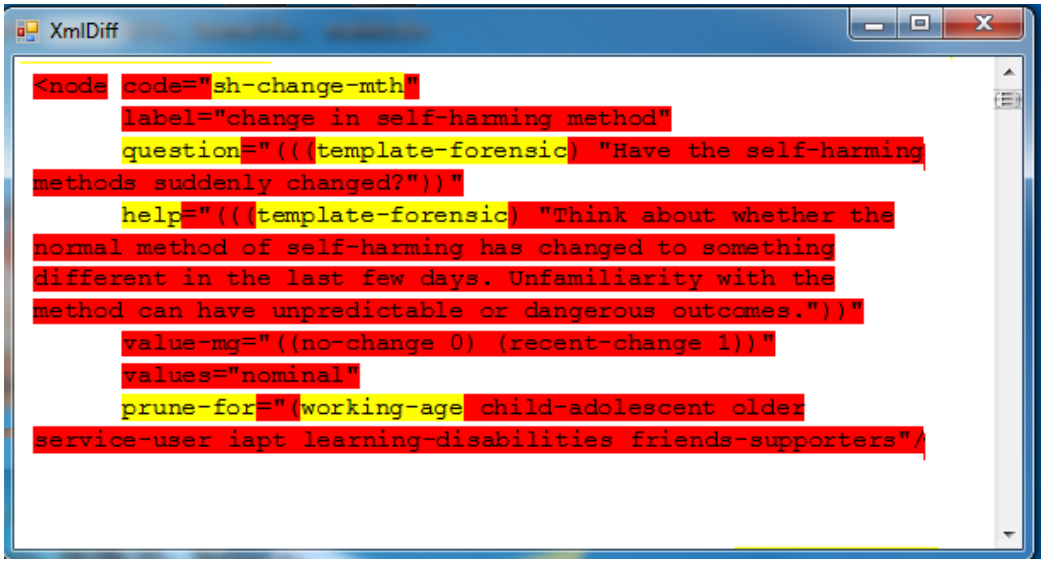

FIGURE 9.13: The “sh-change-mth” node is removed from the present ST.XML due to partially judgments of its core contents. Note: the highlighted text is depicted in yellow background colour.

These issues are resolved successfully through the adoption of the OWL-based specifications. Present ST structure lost or removed such information but the same structure generated through ST.owl keeps such important knowledge (see Figure 9.14).

A more interesting example is found while the comparison of the present XML trees with the new XML structures is generated through the machine-centric specification. There are three nodes with the “gen-home-type” code (see details in Chapter 3 and

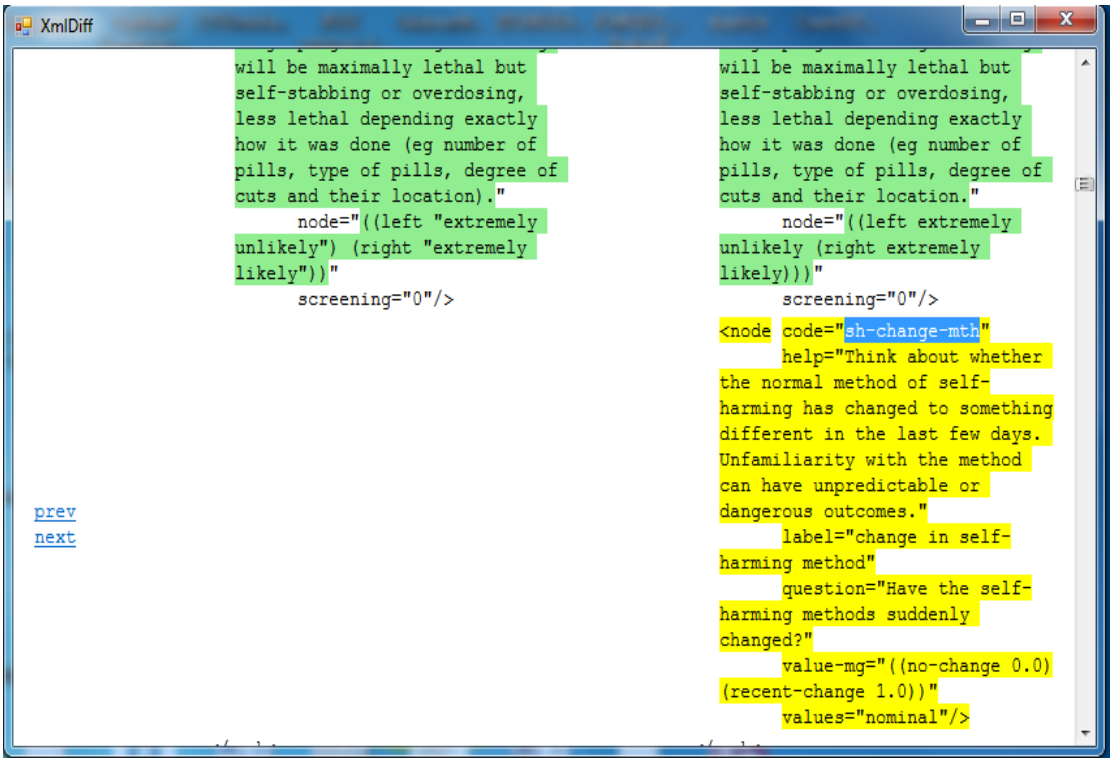

FIGURE 9.14: The "sh-change-mth" node is added in the ST.XML which is derived from ST.owl. Note: the highlighted text is depicted in blue background colour.

Section 3.3.3.1) or name. Two nodes have pruned for information for the working-age population and one out of them has question contents related to the template-forensic service. The third node keeps information that is unsuitable for the forensic-template service. The existing XML file includes third node instead of the node having question contents related to the forensic-template service. Such types of issues create many ambiguities in the knowledge-base and enforce on the need of machine interpretable and understandable translated structures. On the other hand OWL-based generated structures are more reliable and have demonstrated less issue (see details in Chapter 7 and Section 7.6).

9.3.3 RIT structure

The purpose of this tree is that all the concepts and datum nodes must have their uncertain attribute or RIs. The Chapter 7 and Section 7.3 presents the details of the RIT.owl transformation process from the ST.owl.

9.3.3.1 Initialisation of uncertain attribute

A big difference between the present RIT.xml structure, and the RIT.xml tree generated through the OWL-based specification is that the uncertain attributes or RIs (see details in Chapter 3) is initialised in each node of the knowledge-base (see Figure 9.15). The RIs information is only mentioned in the XML specification document but such information is not managed in the present RIT.xml. The distribution or initialisation of the RIs methodology is based on a mathematical theory that is designed by the experts of the Galatean model of classification. The translation of the said methodology implemented in the RIT.owl is designed in the transformation process of the RIT ontology from the ST knowledge-base (see in Chapter 7 and Section 7.3).

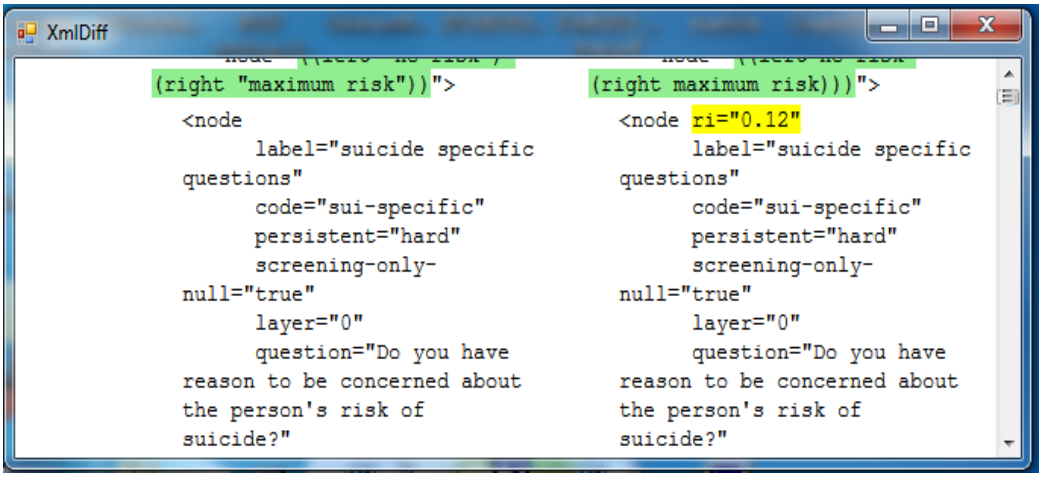

FIGURE 9.15: New RIT XML having RIs.

9.3.4 CAT and QT structure

The CAT is a classification tree that is actually presented to the end user and the QT is a questionnaire tree which is employed for extracting the question associated to a specified population (see details in Chapter 3). The derivation of the CAT and QT ontologies from the RIT.owl file is described in detail in Chapter 7, and Sections 7.4 and 7.5.

9.3.4.1 Elimination of Interim variables

The “level-question” and “level-code” attributes (see details in Chapter 3) are used for derivation of the present QT and CAT xml trees from the RIT.xml. For example, the “gen-presentation” node is a generic or repeating node that has “level-question” and “level-code” attributes (see Figure 9.16).

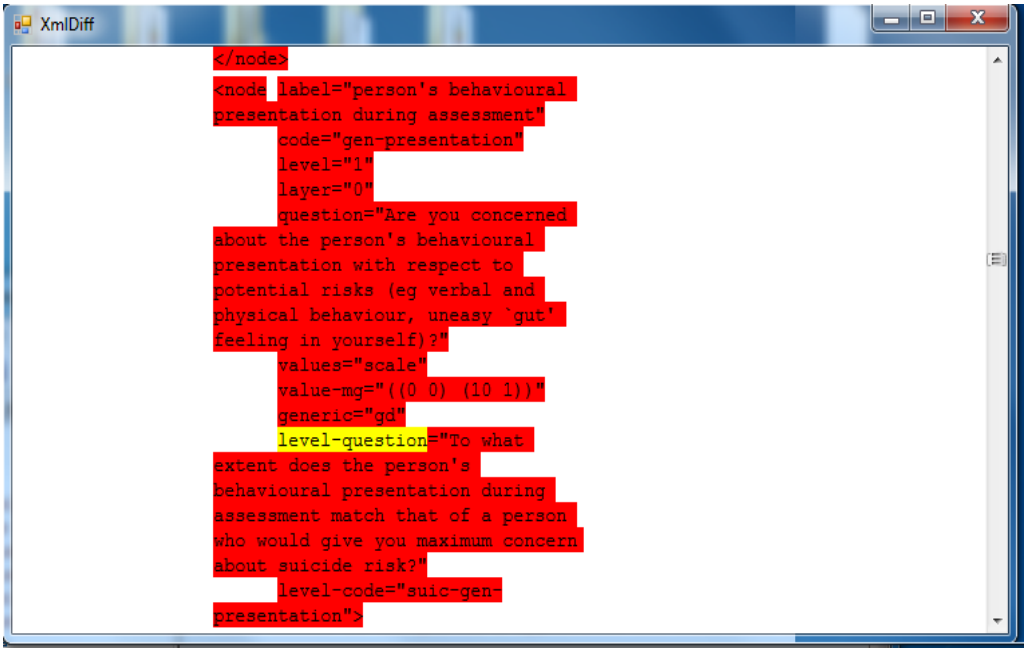

FIGURE 9.16: The “gen-presentation” node has “level-question=To what extent does the person’s behavioural presentation during assessment match that of a person who would give you maximum concern about suicide risk?” attribute and contents in the present RIT.xml. Note: the highlighted text is depicted in yellow background colour.

The contents of the said attributes are designed in the present RIT.xml through some criteria (see details in Chapter 3). The functionalities of these attributes are achieved through some ordinary DL-queries (see details in Chapter 7 and Sections 7.4 and 7.5) which only needed the core properties e.g. “hasLevel” and “hasQuestion” of the knowledge-base. In this way, the OWL-based specification supports the generation of CAT and QT structures without involving any run-time attributes.

The evaluations and results on the comparison of the present XML structures with the XML structures generated through the OWL-based specification are presented in the conclusions section.

9.4 Conclusions

The added value contributions of this study are given here:

- Our contribution is that we provide the idea how to fulfill the user requirements that were difficult to meet through the XML representation. A real world scenario has been discussed in Section 9.2.1.

- The case study discussed in Section 9.2.2 is a real world problem which we have solved by using the OWL-based specifications. The results of the OWL generated SST.xml, ST.xml, RIT.xml, CAT.xml and QT.xml are compared with the present XML based structures. It is concluded that the structures generated through our designed approach are more accurate, consistent and relevant (see details in Chapter 7 and Section 7.6). For example, “gen-cog-think-mem”, “gen-chronic-disease”, “heart-disease”, “dysphasia”, “high-blood-pressure”, “diabetes”, “kidney-disease”, “asthma”, “cerebral-palsy”, “arthritis”, “lung-disease”, “tinnitus” and others nodes should not be included into the customised ST structure because these nodes or their datum nodes have unsuitable information for the “working-age” population and do not have any information related to the template-forensic service.

- The present RIT.xml structure is discussed in detail in the paper manual but the RIs for the nodes are not initialised in the actual knowledge-base due to the complexity of the knowledge. It is difficult to maintain correct RIs in different contexts. It has been demonstrated that how OWL represented constraints help to extract and initialise relevant information from the existing knowledge (see details in Section 7.3 and Chapter 7). The distribution or initialisation of the RIs in the whole RIT structure is achieved through a process based on theory or idea developed by the Galatean experts. Although, the OWL reasoner validate the RIs cardinality relationships to the sub nodes of repeating and non-repeating super nodes, but it is not possible to validate and check the accuracy and correctness of the initilised RIs. For example, the “suic”, “sh” and “hto” are the sub nodes of the “mental-health-risk” node. Let’s say if the experts given data is 1.0 which must be equally divided among the sub nodes of the “mental-health-risk” node. So, the RIs for each sub node should be 0.33. If someone tries to initialise 0.50 RI for the

"suic" node then there is not a formal way to identify such mistake in ontological knowledge. It is because the SWRL does not have much expressivity to count the number of sub nodes and then validate that the experts given data is equally divided in all sub nodes (see a more relevant example in Chapter 10 and Section 10.2.1).

- Through the present XML based approach, the developers copy the contents from the previous structure (let's say SST) and paste the specified knowledge in the next structure (let's say ST). This process is carried on until they get the final classification assessment tree. A landmark difference in our approach is that the knowledge is created once only in the SST.owl structure and the rest of knowledge is filtered through the constraints and extracted by simple DL-queries. In this way, any modifications in the core knowledge (SST) can be accessible or viewable in the classification tree because each file imports the previous one (see details Chapter 7).

- There are also some issues in our adopted approach for example, when we try to generate the ST.xml from the ST.owl. In this case, the process requires some hard coded processes. For example, the "gen-gender" is a generic datum node that having different MGs in different contexts. The definition of this node is provided in the present SST.xml and ST.xml structures in one place and value-mg lists are given with label paths. For example, the ([suic $>>$ gen-demog $>>$ gen-gender] ((MALE 1) (FEMALE 0))) ([sh $>>$ gen-demog $>>$ gen-gender] ((MALE 0) (FEMALE 1))) value-mg lists are required to populate in suicide (i.e suic) and self-harm (i.e. sh) contexts in the CAT.xml. The SST.owl is designed in this way that the paths are logically translated in the form of OWL constraints. New relationships between sub and super nodes are developed if the relationships do not already exist among them. Therefore, some hard coded programs are required while translating ST.xml from ST.owl. Here, the OWL reasoner cannot be helpful when the syntax of the knowledge is required to extract and manage.

In the next chapter, the summary of the thesis will be presented. The chapter will also explore the findings of this study and provides the guidelines for future work.

Chapter 10

Conclusions and Future Research

10.1 Summary and Conclusions

Decision support by machines has traditionally been achieved through either a representation of human skills or by using pure mathematical algorithms. Pure mathematical algorithms may be too inflexible to provide correct decisions in the knowledge domains having a psychological background. Similarly, it is difficult for human experts to elicit decisions from the huge amounts of raw data. This study helps to integrate techniques of data-driven and knowledge-driven decision support systems within a single system. However, the focus is on how the machines can formalise and manipulate the human representation of expertise rather than on machine learning algorithms. This study is based on a system that represents human expertise in a psychological format.

The Galatean model is a psychological model that is used to capture human expertise. This model presents its decisions in the form of classification: the assignment of objects to decision classes, in effect. The Galatean model of classification is applied in multiple decision domains. This model was initially applied in the horse racing domain and later in the psychodynamic psychotherapy domain. The knowledge-bases of these domains consisted of only one decision class and only one type of user (i.e. a gambler for horse racing and an assessor for psychotherapy). A simple knowledge representation language such as XML is adequate to the manage the knowledge represented by a single class type and just one type of object being classified, but when this model

is applied in some complex knowledge domains such as mental health risk assessment (GRiST) and logistics (ADVANCE), which have various users, experts, contexts and circumstances, then a series of XML trees are required to provide the functionalities. It is quite difficult to manage the variants of class and object representations with the support of a paper manual. XML has inherent limitations that cannot provide reasoning and deducing facilities. A knowledge representation is required that represents the knowledge structures, which are closer to human perception and understanding, and provides support in knowledge engineering processes. This research works on the idea of how a machine specification such as Web Ontology Language (OWL) can encapsulate the paper-based specifications required for interpreting and manipulating XML. It explored the advantages of automating XML processing, including ensuring that all structures are correctly specified and consistent.

Chapter 2 presented a brief overview of the Intelligent Decision Support System (IDSS). This chapter in particular, encapsulated the decision making for the domains of the psychological models and more specifically the Galatean model of classification and its applications. The limitations of the present approach or decision making mechanism used for the development of knowledge structures are discussed in detail. The current applications of the Galatean model are the Web-based decision support system which exhibits many syntactic dependency issues in the knowledge structures. In recent years, the demands and popularity of Semantic Web technologies has increased due to the Web pages are searched and maintained on the basis of their semantics rather than on syntactic information which is difficult to modify and manage.

XML, RDF, OWL and SWRL are different standards of Semantic Web architecture. XML is a simple language that does not provide machine interpretable contents and also unable to facilitate reasoning and deducing tasks. The RDF is a meta-language that represents the knowledge in the form of triplet (i.e. subject, predicate and object) but this representation is less expressive than OWL. The data structure of the Galatean model embeds within the human-centric knowledge besides requiring certain arithmetic computations to make accurate predictions, thus it is not possible to construct the descriptive mathematical algorithms or rules over the RDF represented knowledge. Therefore, the OWL is exclusively suitable to our problem because the SWRL rules can

facilitate arithmetic computations besides formal conditions in the form of DL-rules. In the end of the chapter, the requirements of a good decision support system based on Semantic Web technologies have been discussed in detail.

Chapter 3 opened with a brief introduction of the Galatean model of classification. This model presents a unique classification theory which is basically a process for classifying the objects into their related classes. The classification process is triggered through the users providing data, which is used in the calculation of the Membership Grades (MGs) for the associated datum components. The Relative Influences (RIs) are the weightings of the nodes that are given by the experts of the knowledge domains and each node of the Galatean hierarchy must have an RI value. MGs are used at the leaf nodes to generate support for the root classes. For example, the calculated outcomes of the MGs and RIs of the datum node(s) or sub concept(s) are initialised as the MGs of their direct concept(s) and finally a membership grade is assigned to their root classes.

A simple classification assessment tree in XML represents complete knowledge of the decision domains having a single decision category or class, but a chain of logical knowledge trees is required for the decision domains having more than one decision category or class, and multiple users, experts, circumstances and contexts. The knowledge domains like GRiST and ADVANCE have a complex data structure. These assessment domains (i.e. GRiST and ADVANCE) have some progressive trees. The Super Structure Tree (SST) manages the complete knowledge of a decision domain in hierarchical form for various users. The Structure Tree (ST) manages the hierarchical knowledge structure for a particular user. The Relative Influence Tree (RIT) maintains the uncertainty variables (i.e. RI) of each Galatean node. The Question Tree (QT) keeps specific question contents for a specific assessment user. The Classification Assessment Tree (CAT) is a classification tree, where the classification process is actually carried out. At present, a paper manual and some bespoke programs are consulted and utilised for the manipulation of these XML trees.

The responses of the domain experts are managed in the form of nodes in these XML trees. The nodes are divided into two main categories, the super and leaf nodes. A

super node is called a concept node and a leaf node is denoted as a datum node. The concepts and datum nodes are further categorised into repeating and non-repeating components. The presence of repeating concepts either '*g*' or '*gd*' effect on the uncertain knowledge (e.g. '*RIs*') of their sub nodes. For example, the sub nodes of the '*g*' nodes have fixed '*RIs*' and the sub nodes of the '*gd*' nodes have varying '*RIs*'. Each node (either repeating or non-repeating) must have exactly one RI. The value-mg lists of some repeating datum nodes can vary from context to context and some repeating datum nodes keep fixed or constant value-mg lists in all contexts.

The XML nodes have various attributes (e.g. '*code*', '*label*', '*generic-type*', '*prune-for*', '*values*', '*level-code*', '*level-question*', '*multiple-tick*' etc.). Some of these attributes present the description of knowledge and some cause the modifying of the knowledge-base. For example, the '*label*' attribute provides the complete description of an associated node, the '*multiple-tick*' attribute is used to transform the datum nodes to the concepts from the SST.xml to ST.xml and the '*prune-for*' attribute is used to remove unwanted nodes from the knowledge-base for a specified assessment type. It is quite difficult to manage such knowledge-bases for various populations, experts, contexts and circumstances in various knowledge trees with the support of a specification document.

A machine-centric complementary specification is necessarily required that can support decision making without the assistance of human expert(s). The Web Ontology language (OWL) provides the formal structures without losing the psychological validity or transparency of the knowledge and consistency of such knowledge structures can also be confirmed by using the OWL reasoner. Therefore, OWL is the best choice for representing human expertise into machine representations. The '*MGs*' of the datum nodes are measured through some mathematical functions, which compare the users provided data to the graph of '*MGs*' and data given in the value-mg lists. OWL does not support quantification of data over predicates and properties. Therefore, the Web Semantic Rule Language (SWRL) is employed for defining some explicit rules, which can be employed over OWL represented knowledge structures for calculating the mathematical data. Therefore, the SWRL rules can be defined for calculating the '*MGs*' for the datum nodes by using this representation. Chapter 5 provided the rationale and evidence for integrating OWL and SWRL represented knowledge within the proposed

semantic model.

Chapter 4 reviewed different Semantic Web Technologies such as OWL and SWRL. OWL is a well-known machine-centric language. OWL shows a better representation of human-centric models by utilising its classes, properties and restrictions. It contains a repository of terms or symbols which are used to describe a specific knowledge domain and provides a mechanism for describing properties and their relation between properties and different resources. OWL represented knowledge can be further modelled through semantic rules for deriving the calculations and computations for the pure mathematical data. OWL does not support quantification of data over predicates or properties. SWRL is the extension of the Web Ontology language and it provides the facility to quantify the data over predicates. Therefore, the SWRL language is used for defining rules on ontologicaly represented concepts or classes and predicates or properties. These rules can be utilised for measuring the MGs to the concepts and datum nodes. Several editors such as Protege, KAON2 and R2ML are available for editing and creating ontologies and rules, and the OWL reasoner can be employed for retrieving required knowledge and confirming the internal consistency and stability of the knowledge-base.

Chapter 5 presented a complete description of the OWL Galatea model. The '*Node*', '*Question*', '*Assessment-Type*' and '*Value-Mg-Pair*' are the major components of this model. The concepts and datum nodes of a knowledge domain constitute logical hierarchies by defining them as the subclasses of the '*G-ConceptNode*', '*GD-ConceptNode*', '*StandardConceptNode*', '*G-DatumNode*', '*GD-DatumNode*' and '*StandardDatumNode*', which are the subclasses of the '*ConceptNode*' and '*DatumNode*' classes. The '*ConceptNode*' and '*DatumNode*' are the subclasses of the '*Node*' class in the OWL Galatea model. The constraints in the subclasses (i.e. '*ConceptNode*' and '*DatumNode*') of the '*Node*' class are defined for ensuring that all types of concepts and datum nodes either repeating and non-repeating must have a consistent knowledge. For example, each concept has at least two direct sub nodes and the data range of the '*MGs*' for the datum nodes must be greater than or equal to zero and less than or equal to one. The '*hasDirSubNode*'and '*hasDirSupNode*' properties are defined for making relationships

between the sub and super nodes.

The relationships between the nodes and other components (i.e. '*Question*' and '*Value-Mg-Pair*') of the OWL Galatea model are developed by using the '*hasQuestion*' and '*valueMgPairSequence*' properties. The value-mg lists are managed in associated datum components by adopting OWL N-ary pattern, which helps in constructing the sequence of the pairs in the specified value-mg list. The '*Assessment-Type*' class is defined for holding the information of the assessment user or population. Various question contents are applied to various users within a domain. Therefore, the '*appliesTo*' property constraints and relationships are defined in the questions that indicate the relationships between contents and user(s). Such relationships support in inferring and retrieving the question contents related to a particular assessment user.

The '*hasMG*' and '*hasRI*' properties are defined for holding the uncertain data. The data range for the '*hasRI*' property constraints is defined in the '*Node*' class on the other hand the data range for the '*hasMG*' property constraints is defined in the '*DatumNode*' class. The OWL reasoner picks up the nodes having such data beyond the given range by interpreting constraints defined regarding these (i.e. '*hasMG*' and '*hasRI*') properties. The value-mg lists are managed by creating the subclasses of the '*Value-Mg-Pair*', and their sequences are managed by using different OWL properties (i.e. '*valueMgPairSequence*' and '*nextValueMgPair*'). Some semantic rules are defined for calculating MGs of the datum nodes. These properties (i.e. '*valueMgPairSequence*' and '*nextValueMgPair*') are used as the predicates or functions in the SWRL rules. The user provided data, and the MGs and datum values stored in the pairs of the value-mg list bind with the variables defined in these predicates. Some built-in functions (e.g. equal, greaterThanOrEqual, lessThanOrEqual, etc.) are employed for comparing the user provided data with the datum values stored in the pairs of the value-mg list. In this case, if the matched is true, then an MG is assigned to an associated datum node. The OWL reasoner interprets the SWRL rules and assigns the inferred MGs to the associated datum nodes. OWL and SWRL approaches have provided a machine formalism for the knowledge without losing its psychological validity or transparency and the validity of these formal structures can be confirmed or tested by using some OWL inference engines or reasoners. Their represented structures also facilitate end users to

understand the knowledge and help in the extension of this intelligent model (i.e. OWL Galatea model) in variant knowledge domains of the Galatean model of classification.

Chapter 6 elaborated on the extension of the OWL Galatea model through the knowledge domains (i.e. GRiST and ADVANCE) of the Galatean model of classification. GRiST is exercised and used in mental health risk screening sector, on the other hand ADVANCE is used in logistics domain. Both decision domains have distinctive assessment types or users, but there are many other commonalities in their data structures such as various hierarchical structures, contexts, experts, users, circumstances, questions, value-mg lists and '*RIs*'. Therefore, the knowledge engineering process for the root knowledge-base (SST ontology) for these decision domains is managed in identical ways. Moreover, this chapter presented the generation of the SST ontology by importing the OWL Galatea model from the SST.xml file, which is used as an input to the knowledge engineering process of the knowledge-base.

The SST ontology underpins the multiple knowledge representations in the complex knowledge domains. It keeps the knowledge for all populations within a knowledge domain. All the basic knowledge (i.e. '*hasLabel*', '*hasLevel*', '*hasPruneFor*', '*hasQuestion*', etc.) and their relationships and constraints are defined in this tree structure for all users of the domain. The specific knowledge constraints are defined regarding the users of the decision domains by the extension of the '*Assessment-Type*' class of the OWL Galatea model. The concepts and datum nodes are modelled by extending the '*G-ConceptNode*', '*GD-ConceptNode*', '*StandardConceptNode*', '*G-DatumNode*', '*GD-DatumNode*' and '*StandardDatumNode*' classes. The relationships between sub and super nodes are defined by using the '*hasDirSubNode*' and '*hasDirSupNode*' properties for forming the initial hierarchical structure.

The questions and value-mg lists are modelled by extending the '*Question*' and '*Value-Mg-Pair*' classes. The relationships between nodes and question contents are defined by using the '*hasQuestion*' property, and the relationships between contents and the

users are defined by using '*appliesTo*' property. The relationships between an associated datum node and the first pair of the value-mg list are managed by using the '*valueMgPairSequence*' property. The relationships for the next element(s) of the value-mg list are developed by using the '*nextValueMgPair*' property. The zero number of cardinality relationships are defined by using the '*nextValueMgPair*' property in the last element of each value-mg list that indicate the end of the list. The '*generic*' concept or context is dropped, because this concept have not any logical and semantic relations to the knowledge-base and it is just used for holding repeating nodes or knowledge in one location. Therefore, the relationships of its sub concepts and datum nodes are defined through the '*hasDirSubNode*' and '*hasDirSupNode*' properties to the individual concepts. The unnecessary attributes (e.g. '*multiple-tick*') are also dropped due to their dependency on syntax structures and contexts during the generation of this ontology.

The SST is an all-inclusive logical tree for all populations of a domain and other knowledge trees (i.e. ST, RIT, QT and CAT) which contain basic knowledge from this root ontology. Therefore, the constraints, restrictions and relationships in the root (SST) ontology are defined in a way that can support in knowledge transformation and evolution. For example, the relationships defined for maintaining hierarchical structures by using '*hasDirSubNode*' and '*hasDirSupNode*' properties can support adding further constraints for only a single user. The contextual information is also linked or connected by using these properties that help in initialising the uncertain data (e.g. '*RIs*'). The question contents and their relationships with nodes and users are developed by using these properties (i.e. '*hasQuestion*' and '*appliesTo*') that help in inferring specific knowledge.

Chapter 7 showed the transformations of different ontologies and different algorithms that manage these transformations. The SST is a root ontology or main repository. The derived ontologies used the fundamental knowledge including the existing constraints from the SST ontology for a specific assessment user. The ST is the specific class manifestation for a particular type of object to be classified as circumstance of classification. Further constraints are defined in the '*Node*' class in the ST ontology that help the OWL reasoner to derive knowledge or nodes that are related to a potential user. The existential restrictions are defined by employing the '*hasDirSubNode*'and

'*hasDirSupNode*' properties in each inferred node. The constraints and relationships of these properties support in traversing the Galatean hierarchies and their super properties (i.e. '*hasSubNode*'and '*hasSupNode*'), which are transitive properties help in retrieving and inferring the sub concepts and datum nodes or branches inside a concept node by using the OWL reasoner. The RIT ontology is derived by importing the ST ontology for initialising the '*RIs*' of the whole Galatean hierarchy for a specific user. The '*RIs*' of the sub nodes of different concepts are initialised under some criteria and the OWL reasoner supports in inferring the knowledge pieces according to the given criterion or constraint. For example, some new pieces of knowledge are created for storing the '*RIs*' in the nodes, which are sub nodes of some generic (i.e. '*G-ConceptNode*') concepts. Existing knowledge chunks are utilised for storing the '*RIs*' of the sub nodes of some generic (i.e. '*GD-ConceptNode*') and non-generic (i.e. '*StandardConceptNode*') concepts.

The QT ontology can be derived either from the ST or RIT ontology by calling some ordinary queries for keeping the questions related to a specified assessment user. In fact, the transformation of the knowledge completes on RIT state by employing the OWL based approach and further knowledge is extracted by calling some simple queries. Therefore, the QT and CAT is transformed by calling some simple queries from the RIT ontology according to the experience level of the assessment user. The CAT is the actual classification tree and everything else in the decision domain is about supporting variations of the CATs for different purposes. At the end of the chapter, some benefits of using the OWL based approach were also discussed by using some suitable examples from the applications of the Galatean model of classification. OWL has been used to manage the different transformations of the knowledge and it does not focus on only the tree relationships, but also the OWL based approach provides a number of benefits like: automatic knowledge maintenance, provision of flexible, coherent and intelligible knowledge-base, and the consistency of the knowledge-base can be confirmed by using the OWL reasoner.

Chapter 8 demonstrated the added functionality of OWL implementation by automating the generation of different data-gathering interfaces. It is quite attractive for the users of knowledge domains if adaptive knowledge management techniques are used.

The dynamic derivation of an adaptive assessment questionnaire was demonstrated in this chapter. The derivation of the adaptive assessment questionnaire has portrayed the versatility and flexibility of the OWL represented knowledge. The generation of an adaptive questionnaire has demonstrated how OWL provides additional functionality that is not automatically included with the previous decision support systems that are based on only XML representation. The knowledge related to an assessment user can be deduced or retrieved from the knowledge-base by determining the constraints defined for a particular assessment type in the knowledge-base. The constraints of the '*hasAge*' property in particular instance are used for deducing the type of the instance. The identified type (a subclass of the '*Assessment-Type*') is used to infer the concepts and datum nodes, which have related question's contents from the knowledge-base. The case of using age as the seed for creating alternative QTs is a pretty basic example. This is just a simple application of OWL, but the depth of its encapsulation of the knowledge semantics provides great flexibility in outputting XML trees for multiple purposes in the knowledge-engineering and application domains.

Chapter 9 expressed the evaluation of the XML structures (i.e. SST, ST, RIT, QT and CAT) derived through the OWL-based specification by comparing with the present XML structures. Two real time case studies were also presented in detail that demonstrates the undeniable benefits of OWL-based specification. One case study demonstrated how easy and convenient it is to represent the user requirements which are difficult to fulfill through the XML representation. The other case study was about knowledge management for the forensic service. It was difficult to manage, maintain and transform knowledge into different knowledge structures that belong to a specified population (let say working-age) and a service (for instance; template-forensic).

The XML Notepad 2007 is used for the comparison of existing XML structures with the XML structures derived through the OWL-based specification. This chapter presented the contrast of both the (present and new) XML structures with suitable examples of GRiST application. At the end of this chapter the core contributions gained through this study were also discussed in detail.

The conclusions are that:

- The intelligent model (i.e. OWL Galatea model) developed through this study completely encapsulates the written specification and underlying cognitive model. Some generic semantic rules are defined in the OWL Galatea model that can calculate the uncertain data (i.e. MGs). The OWL Galatea model is defined at quite an abstract level, which helps in its implementation in multiple knowledge domains having a cognitive background.

- Reliable and accurate transformation of the knowledge in different states (SST, ST, RIT, QT and CAT) has been a big challenge for the experts of the Galatean model of classification. This issue is tackled by creating the core knowledge in the main structure (i.e. SST.owl) and then its customised views are generated for different context(s) or circumstance(s) or user(s) or service(s). This process is achieved by importing the previous structure (let say SST) into the next state (let say ST). The benefits of our presented approach are that the conflicts between different structures or states of the knowledge-bases no longer exist and if any modifications occur in the core repository then these lead to its all subsequent states.

- The OWL-based approach reduced the number of transformation states from five to three, and if the approach is used through the techniques used for generating adaptive assessment questionnaire then only two states of the knowledge can maintain such complex knowledge.

- An adaptive assessment questionnaire is generated by using OWL approach which is flexible and robust compared to the previous decision support systems that are based only XML on representation.

It is concluded that the OWL and SWRL-based approaches provide support in many ways such as: validating the complex data structure of human psychology domains by using the OWL reasoner, maintaining the flexibility in the knowledge-base, designing the paper manual specification in a machine centric form, improving the knowledge-engineering process for decision support systems based on a psychological model and facilitating the application of the model to new knowledge domains.

10.2 Directions for future research

In this thesis, an intelligent method is developed as a test bed for formalising human expertise into machine formalism that supports automatic knowledge maintenance and helps in making correct decisions of knowledge domains having complex and uncertain data.

10.2.1 Machine support for classification process

Some processes and functions cannot be expressed properly due to the limitations of OWL and SWRL representations. It is because OWL is based on the description logic and first order logic, and certain concepts are difficult to design using this representation. For example, the first order logic has no strength to express the complete structures with an infinite domain such as natural numbers. Therefore, it is difficult to represent certain concepts by using OWL language and sometimes some extra concepts or relationships are needed for expressing certain knowledge. For instance, to express the formal relationships between uncle and nephew through OWL representation, just two classes or entities are not sufficient to model this concept.

Although, the SWRL is a bit more expressive than OWL, it is also difficult to express certain knowledge by using this representation. For example, the SWRL rules are defined for initialising the '*MGs*' of the datum nodes by comparing the user given data to the data stored in the pairs of the associated value-mg list. These rules are actually represented in implementations of the graph data of '*MGs*' and datum values that support the calculations of the '*MGs*' for associated datum nodes by using the '*hasMG*' property. The aim was that the '*MGs*' of the concepts are also initialised by using some generic SWRL rules, but the SWRL language has limitations and it does not support the propagation of the '*MGs*' from datum nodes to concepts. For example, a concept or super node (let's say '*X*') has three datum nodes (say '*d1*', '*d2*' and '*d3*'), and the '*i1*' is the instance of the '*d1*', '*i2*' belongs to '*d2*' class and the '*i3*' has the '*d3*' type.

Each instance keeps a resultant value (let's say i1=0.2, i2=0.3 and i3=0.1) of the product of the '*RIs*' and '*MGs*' through a SWRL rule. For example, the '*i1*' instance has 0.4 RI

and 0.5 MG and the product of these (i.e. '*RI*' and '*MG*') are 0.2 and other instances (i.e. '*i2*' and '*i3*') are kept as the product of the '*RIs*' and '*MGs*' in identical ways. The '*MG*' of the X node is expected to be 0.6 (i.e. $0.2 + 0.3 + 0.1 = 0.6$), but more than one '*MG*' is initialised for a concept node (i.e. '*X*') such as 0.5, 0.3 and 0.4 by using '*hasMG*' property that represents incorrect modelling of knowledge. A generic rule cannot be helpful in initialising the '*MGs*' of the concepts. The SWRL inference engine (an OWL reasoner) cannot distinguish the difference in the type of the instances that leads to the Cartesian product results, and it is difficult to cut or stop the binding of variables in more than one number of times in a SWRL rule. There is a need to employ some other formal methods or representations for initialising the '*MGs*' of the concepts.

A more expressive language is needed for the '*MGs*' propagation process from datum nodes to generate support for the root class autonomously and automatically. A major problem for reasoning in a more expressive language (based on higher order logic) is the unification, which is undecidable in the higher-order case [124]. Therefore, the OWL and SWRL representations are used for designing the psychological model of classification, because these provide decidable reasoning for a huge amount of certain knowledge. A more expressive language can complement the OWL and SWRL approaches used in the OWL Galatea model.

10.2.2 Integrate adaptive assessment questionnaire with an ontology-based adaptive Information Collection Systems (ICS)

The adaptive assessment questionnaire discussed in Chapter 8 can be integrated with an ontology-based adaptive Information Collection Systems (ICS) or method for designing complex medical questionnaires, which is presented by Matt-Mouley et al. [125]. Their presented ontology based method helps in the decision making for displaying questions by judging the user cues. Their presented approach and our intelligent method for developing adaptive assessment questionnaire can mutually benefit from each other. The adaptive assessment questionnaire can be employed for inferring the knowledge related to a particular age-group of a user, and the techniques used in ICS can help in displaying questions by determining the users' responses.

The OWL and SWRL approaches and techniques discussed in this thesis can become a guide for formalising psychological represented structures and their developers can evolve knowledge in multiple ways for facilitating their assessment users, and fulfilling the requirements of human psychological experts.

Bibliography

[1] Christopher D. Buckingham, Ann Adams, and Chris Mace: *Psychological cue use and implications for a clinical decision support system*. Journal of Mental Health, 17(3):299–314, 2008.

[2] Christopher D Buckingham, Abu Ahmed, and Ann Adams: *Designing multiple user perspectives and functionality for clinical decision support systems*. In *Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on*, pages 211–218, September 2013.

[3] Carlo J. Evangelisti and G Goertzel: *The Architecture of a Decision Support System*. International Journal of Bio-Medical Computing, 17(1), 1985.

[4] D.J. Power: *Decision Support Systems: Concepts and Resources for Managers*. Quorum Books, 2002.

[5] James F Courtney: *Decision making and knowledge management in inquiring organizations: toward a new decision-making paradigm for decision support system*. Decision Support Systems, 31(1):17–38, 2001.

[6] Daniel J Power: *Understanding Data-Driven Decision Support Systems*. Information Systems Management, 25(2):149–154, 2008.

[7] Rita L Atkinson, Richard C. Atkinson, Edward E. Smith, Daryl J. Bem, and Susan Nolen-Hoeksema: *Hilgard's Introduction to Psychology*. Harcourt Brace College Publishers Philadelphia PA, 1996.

[8] Claudia A. Perry: *Knowledge bases in medicine: a review*. JOURNAL of Bulletin Medical Library Association, 3(78):271–282, 1990.

[9] Mihail Popescu and Gerald Arthur: *OntoQuest: A Physician Decision Support System based on Ontological Queries of the Hospital Database*. AMIA Annual Symposium Proceedings, pages 639–643, 2006.

[10] Severin Lemaignan, Ali Siadat, Jean Yves Dantan, and Anatoli Semenenko: *MASON: A proposal for an ontology of manufacturing domain*. In *Proceedings of the IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications*, pages 195–200. IEEE, 2006.

[11] Amin Saremi, Mostafa Esmaeili, Jaafar Habibi, and Arman Ghaffari: *O2DSS: A Framework for Ontology-based Decision Support Systems in Pervasive Computing Environment*. In *Second Asia International Conference on Modelling Simulation*, pages 41–45. IEEE, 2008.

[12] Naomi Wrighton and Christopher D. Buckingham: *Defining an OWL Ontology That Could Be Used to Integrate Mental-Health Risk Information within a Decision Support System*. In *International Conference on eHealth, Telemedicine, and Social Medicine*, pages 245–250, 2009.

[13] Hans W. Gottinger and Peter Weimann: *Intelligent decision support systems*. Decision Support Systems, 8(4):317–332, 1992.

[14] Stephanie Guerlain, Donald E. Brown, and Christina Mastrangelo: *Intelligent decision support systems*. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 3, pages 1934–1938. IEEE, 2000.

[15] Woolery LK and Grzymala Busse J.: *Machine Learning for an Expert System to Predict Preterm Birth Risk*. Journal of the American Medical Informatics Association, 1(6):439–446, 1994.

[16] G Octo Barnett, James J Cimino, Jon A Hupp, and Edward P Hoffer: *DXplain-An Evolving Diagnostic Decision-Support System*. JAMA: The journal of the American Medical Association, 258(1):67–74, 1987.

[17] Randolph A Miller: *Medical diagnostic Decision Support Systems–Past, Present and Future: A Threaded Bibliography and Brief Commentary*. Journal of the American Medical Informatics Association, 1(1):8–27, 1994.

[18] Mary E Johnston, Karl B Langton, R Brian Haynes, and Alix Mathieu: *Effects of Computer-based Clinical Decision Support Systems on Clinician Performance and Patient Outcome: A Critical Appraisal of Research*. Annals of internal medicine, 120(2):135–142, 1994.

[19] Dereck L Hunt, R Brian Haynes, Steven E Hanna, and Kristina Smith: *Effects of computer-based clinical decision support systems on physician performance and patient outcomes*. JAMA: The journal of the American Medical Association, 280(15):1339–1346, 1998.

[20] Hokey Min and Sean B Eom: *An Integrated Decision Support System for Global Logistics*. International Journal of Physical Distribution & Logistics Management, 24(1):29–39, 1994.

[21] Avgoustinos Filippoupolitis and Erol Gelenbe: *A Decision Support System for Disaster Management in Buildings*. In *Proceedings of the 2009 Summer Computer Simulation Conference*, pages 141–147, 2009.

[22] Motoi Suwa, A Carlisle Scott, and Edward H Shortliffe: *An Approach to Verifying Completeness and Consistency in a Rule-Based Expert System*. AI Magazine, 3(4):16–21, 1982.

[23] John R Anderson, Dan Bothell, Christian Lebiere, and Michael Matessa: *An Integrated Theory of List Memory*. Journal of Memory and Language, 38(4):341–380, 1998.

[24] John E Laird: *Extending the Soar Cognitive Architecture*. Frontiers in Artificial Intelligence and Applications, 171:224–235, 2008.

[25] John Laird: *The Soar Cognitive Architecture*. Massachusetts Institute of Technology (MIT) Press, 2012.

[26] C. D. Buckingham, G. Kearns, S. Brockie, A. E. Adams, and I. T. Nabney: *Developing a Computer Decision Support System for Mental Health Risk Screening and Assessment*. Current perspectives in healthcare computing, pages 189–195, 2004.

[27] Christopher Buckingham: *A prototype model of classification based on conditional probabilities*. Birmingham University: Unpublished doctoral thesis, 1991.

[28] Christopher D. Buckingham and Janice Birtle: *Representing the assessment process for psychodynamic psychotherapy within a computerized model of human classification*. The British Journal of Medical Psychology, 70:1–16, 1997.

[29] Laura Vail, Ann Adams, Eleanor Gilbert, Alice Nettleingham, and Christopher D Buckingham: *Investigating mental health risk assessment in primary care and the potential role of a structured decision support tool, GRiST*. Mental Health in Family Medicine, 9:57–67, 2012.

[30] Rudi Studer, V. Richard Benjamins, and Dieter Fensel: *Knowledge Engineering: Principles and Methods*. Data & Knowledge Engineering, 25(1):161–197, 1998.

[31] Daniel J. Power: *Decision support systems: A historical overview*. In *Handbook on Decision Support Systems*, pages 121–140. Springer, 2008.

[32] Hemant K. Bhargava, Daniel J. Power, and Daewon Sun: *Progress in web-based decision support technologies*. Decision Support Systems, 43(4):1083–1095, 2007.

[33] Ozan Cakir and Mustafa S. Canbolat: *A web-based decision support system for multi-criteria inventory classification using fuzzy ahp methodology*. Expert Systems with Applications, 35(3):1367–1378, 2008.

[34] Vassilis Kostoglou, Nikolaos Ploskas, Michael Vassilakopoulos, and Vaia Ekaterini Tsantopoulou: *ANALYSIS AND DESIGN OF A WEB-BASED DECISION SUPPORT SYSTEM FOR CHOOSING HIGHER EDUCATION STUDIES*. Yugoslav Journal of Operations Research, 24(3):399–414, 2014.

[35] Allan Leck Jensen, Peter S. Boll, Iver Thysen, and B.K. Pathak: *Pl @nteInfo – a web-based system for personalised decision support in crop management*. Computers and Electronics in Agriculture, 25:271–293, 2000.

[36] Hermant K Bhargava and Clay Tettelbach: *A web-based decision support system for waste disposal and recycling*. Computers, Environment and Urban Systems, 21(1):47–65, 1997.

[37] G. Banias, Ch. Achillas, Ch. Vlachokostas, N. Moussiopoulos, and I. Papaioannou: *A web-based Decision Support System for the optimal management of construction and demolition waste*. Waste Management, 31:2497–2502, 2011.

[38] Ian Horrocks, Bijan Parsia, Peter Patel-Schneider, and James Hendler: *Semantic web architecture: Stack or two towers?* In *Principles and Practice of Semantic Web Reasoning*, pages 37–41. Springer, 2005.

[39] Xiaobing Wu: *Knowledge Representation and Inductive Learning with XML*. In *Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 491–494. IEEE Computer Society, 2004.

[40] Paolo Bouquet, Luciano Serafini, and Heiko Stoermer: *Introducing Context into RDF Knowledge Bases*. In *Semantic Web Application and Perspectives(SWAP)*, volume 5, pages 14–16, 2005.

[41] Jeen Broekstra, Michel Klein, Stefan Decker, Dieter Fensel, Frank van Harmelen, and Ian Horrocks: *Enabling knowledge representation on the Web by extending RDF Schema*. Computer Networks, 39(5):609–634, 2002.

[42] Ian Horrocks: *DAML+OIL: A Description Logic for the Semantic Web*. IEEE Data Engineering Bulletin, 25(1):4–9, 2002.

[43] Grigoris Antoniou and Frank Van Harmelen: *Web ontology language: Owl*. In *Handbook on ontologies*, pages 67–92. Springer, 2004.

[44] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider (editors): *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[45] Andreas De Vries: *XML framework for concept description and knowledge representation*. Logic in Computer Science; ACM-class: I.7.2; E.2; H.1.1; G.2.3; arXiv:cs.AI/0404030 Ver. 1, 2004.

[46] JICHANG DONG, HELEN S. DU, K. K. LAI, and SHOUYANG WANG: *XML-BASED DECISION SUPPORT SYSTEMS: CASE STUDY FOR PORTFOLIO SELECTION*. International Journal of Information Technology & Decision Making, 3(4):651–662, 2004.

[47] J.Z. Pan and Ian Horrocks: *RDFS (FA): Connecting RDF(S) and OWL DL*. Journal IEEE Transactions on Knowledge and Data Engineering, 2(19):192–206, February 2007.

[48] Brian McBride: *The resource description framework (RDF) and its vocabulary description language RDFS*. In *Handbook on ontologies*, pages 51–65. Springer, 2004.

[49] Grigoris Antoniou, Enrico Franconi, and Frank Van Harmelen: *Introduction to Semantic Web Ontology Languages*. In *Reasoning Web*, pages 1–21. Springer, 2005.

[50] Martin O' Connor, Holger Knublauch, Samson Tu, Benjamin Grosof, Mike Dean, William Grosso, and Mark Musen: *Supporting Rule System Interoperability on the Semantic Web with SWRL*. In *ISWC'05 Proceedings of the 4th international conference on The Semantic Web*, pages 974–986, November 2005.

[51] Christine Golbreich, Olivier Dameron, Olivier Bierlaire, and Bernard Gibaud: *What Reasoning Support for Ontology and Rules? The Brain Anatomy Case Study*. In *Proceedings of the OWLED 2005 Workshop on OWL: Experiences and Directions, Galway, Ireland*, volume 188, pages 11–12, November 2005.

[52] Lars Ludwig and David O'Sullivan: *Deploying Decision Support Systems Using Semantic Web Technologies*. International Journal of Decision Support System Technology (IJDSST), 2(1):49–59, 2010.

[53] ASAD MASOOD KHATTAK, RABIA BATOOL, ZEESHAN PERVEZ, ADIL MEHMOOD KHAN AND SUNGYOUNG LEE: *Ontology evolution and challenges*. JOURNAL OF INFORMATION SCIENCE AND ENGINEERING, 29:851–871, 2013.

[54] LULE AHMEDI and EDMOND JAJAGA: *Normalization of relations and ontologies*. In *Recent Researches in ARTIFICIAL INTELLIGENCE, KNOWLEDGE ENGINEERING and DATA BASES*, pages 419–424, 2010.

[55] Valéry DONFACK GUEFACK, Valérie BERTAUD GOUNOT, Régis DUVAUFERRIER, Annabel BOURDE John MORELLI, and Jérémy LASBLEIZ: *Ontology Driven Decision Support Systems for Medical Diagnosis*. Quality of Life Through Quality of Information: Proceedings of MIE2012, 180:103–108, 2012.

[56] Kamran Farooq, Amir Hussain, Stephen Leslie, Chris Eckl, Calum MacRae, and Warner Slack: *An ontology driven and bayesian network based cardiovascular*

decision support framework. In *Advances in Brain Inspired Cognitive Systems*, pages 31–41. Springer, 2012.

[57] Chryssa H. Thermolia, Ekaterini S. Bei, and Euripides G. M. Petrakis: *Prediction of the evolution of bipolar depression using semantic web technologies*. In *Information, Intelligence, Systems and Applications, IISA 2014*, pages 391–396. IEEE, 2014.

[58] Kyoung Yun Kim and Hyungjeong Yang: *THE ROLE OF MEREOTOPOLGY AND SWRL RULES TO REPRESENT JOINT TOPOLOGY INFORMATION FOR DESIGN COLLABORATION*. pages 1–9, 2008.

[59] Li Cong-dong, Li Jun, Liao Can, Xie Tian, and Chen Zhenghui: *The Workflow Selection Reasoning Framework of Collaborative Manufacturing System Based on SWRL*. In *Conference Anthology, IEEE*, pages 1–7. IEEE, 2013.

[60] Hendro Wicaksono, Sven Rogalski, and Jivka Ovtcharova: *Ontology Driven Approach for Intelligent Energy Management in Discrete Manufacturing*. In *4th International Conference on Knowledge Engineering and Ontology Development (KEOD 2012)*, pages 108–114, 2012.

[61] Joyce Mitchell: *e-Study Guide For Introduction to Health Care*. Academic Internet Publishers, 2013.

[62] Christopher D. Buckingham and Abu Ahmed: *Operational specification of the GRiST Decision Support System Version 2.0 RC 4*, 2010.

[63] Donald T Campbell: *The Informant in Quantitative Research*. American Journal of sociology, 60(4):339–342, 1955.

[64] Ulla Hallgren Graneheim and Berit Lundman: *Qualitative content analysis in nursing research: concepts, procedures and measures to achieve trustworthiness*. Nurse education today, 24(2):105–112, 2004.

[65] Martin Davies: *Concept mapping, mind mapping and argument mapping: what are the differences and do they matter?* Higher Education, 62(3):279–301, 2011.

[66] Tony Buzan and Barry Buzan: *The Mind Map Book*. BBC Books, London, 3rd edition, 2006.

[67] Christopher Buckingham, Ahmed A, and Adams AE.: *Using XML and XSLT for flexible elicitation of mental-health risk knowledge*. Medical Informatics and the Internet in Medicine, 32(1):65–81, 2007.

[68] Christopher Buckingham: *Operational specification of the XML structures driving the ADVANCE cognitive model of decision making, GALMOD,Version 2*, November 2013.

[69] Marco Rospocher and Luciano Serafini: *Ontology-centric decision support*. In *International Workshop on Semantic Technologies meet Recommender Systems & Big Data (SeRSy 2012), co-located with ISWC2012, CEUR Workshop Proceedings (CEUR-WS.org)*, volume 919, pages 61–72, 2012.

[70] Dieter Domingue, John; Fensel and James A. eds Hendler: *Handbook of Semantic Web Technologies*. Springer, 2011.

[71] Boris Motik, Peter F. Patel-Schneider, and Ian Horrocks: *OWL 1.1 Web Ontology Language – Structural Specification and Functional-Style Syntax*. Technical report, The University of Manchester, May 2007.

[72] Matthew Horridge, Nick Drummond, John Goodwin, Alan Rector, Robert Stevens, and Hai Wang: *The Manchester OWL Syntax*. In *OWLED2006 Second Workshop on OWL Experiences and Directions*, Athens, GA, USA, 2006.

[73] James Hendler Tim Berners-Lee and Ora Lassila: *The Semantic Web*. Scientific American, 284(5):29–37, 2001.

[74] Haytham Al-Feel, M.A.Koutb, and Hoda Suoror: *Toward An Agreement on Semantic Web Architecture*. World Academy of Science, Engineering and Technology, 25(131):806–810, 2009.

[75] Elena Simperl: *Reusing ontologies on the Semantic Web: A feasibility study*. Data and Knowledge Engineering, 68(10):905–925, 2009.

[76] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler: *OWL 2: The Next Step for OWL*. Web Semantics, 6(4):309–322, 2008.

[77] Li Ding, Pranam Kolari, Zhongli Ding, and Sasikanth Avancha: *Using Ontologies in the Semantic Web: A Survey*. Springer, October 2005.

[78] Eric Miller: *An Introduction to the Resource Description Framework*. Bulletin of the American Society for Information Science and Technology, 25(1):15–19, 1998.

[79] LiLi, Yun Yang, and Baolin Wu: *Implementation of agent-based Ontology Mapping and Integration*. In *IEEE International Conference on e-Business Engineering*, pages 208–215, 2006.

[80] Wang X, Gorlitsky R, and Almeida JS.: *From XML to RDF: how semantic web technologies will change the design of 'omic' standards*. Nature Biotechnology, 23(9):1099–1103, 2005.

[81] Jorge E. Lopez de Vergara, Victor A. Villagra, and Julio Berrocal: *Application of the Web Ontology Language to define management information specifications*. In *Communications Magazine, IEEE*, volume 42, pages 20–23, July 2004.

[82] Nicola Guarino: *Formal Ontology and Information Systems*. In *Proceedings of the 1st International Conference*, pages 3–15. IOS Press, 1998.

[83] Matthew Horridge, Holger Knubaluch, Alan Rector, Robert Stevens, and Chris Wore: *A Practical Guide To Building OWL Ontologies Using The Protege-OWL Plugin and CO-ODE Tools Edition 1.0*. Technical report, 2004.

[84] Matthew Horridge, Holger Knubaluch, Alan Rector, Robert Stevens, and Chris Wore: *A Practical Guide To Building OWL Ontologies Using Protege 4 and CO-ODE Tools Edition 1.2*. Technical report, 2009.

[85] Matthew Horridge, Holger Knubaluch, Alan Rector, Robert Stevens, and Chris Wore: *A Practical Guide To Building OWL Ontologies Using Protege 4 and CO-ODE Tools Edition 1.3*. Technical report, 2011.

[86] Matthew Horridge, Holger Knubaluch, Alan Rector, Robert Stevens, and Chris Wore: *A Practical Guide To Building OWL Ontologies Using Protege 4 and CO-ODE Tools Edition 1.1*. Technical report, 2007.

[87] Matthew Horridge and Sean Bechhofer: *The OWL API: A Java API for OWL ontologies*. Journal of Semantic Web, 2(1):11–21, January 2011.

[88] Alan Rector, Matthew Horridge, Robert Stevens, Hai Wang, and Chris Wroe: *OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors and*

Common Patterns. In *Proceedings of the European Conference on Knowledge Acquistion, Northampton, England, 2004, Lecture Notes on Computer Science LNAI3257, Springer-Verlag*, volume 3257, pages 63–81, 2004.

[89] Neha Dalwadi, Bhaumik Nagar, and Ashwin Makwana: *SEMANTIC WEB AND COMPARATIVE ANALYSIS OF INFERENCE ENGINES*. International Journal of Computer Science and Information Technologies (IJCSIT), 3(3):3843–3847, 2012.

[90] Domenico Redavid, Luigi Iannone, and Terry Payne: *OWL-S Atomic services composition with SWRL rules*. In *17th International Symposium on Methodologies for Intelligent Systems (ISMIS'08), York University, Toronto, Canada*, volume 4994, pages 20–23, May 2008.

[91] A. TH. S CHREIBER G. VAN H EIJST and B. J. WIELINGA: *Using Explicit Ontologies in KBS Development*. International Journal of Human-Computer Studies, 46(2-3):183–292, 1997.

[92] IEEE Computer Society: *IEEE Standard Glossary of Software Engineering Terminology* . 1990.

[93] Asuncion Gomez Perez and V. Richard Benjamins: *Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods*. In *IJCAI and the Scandinavian AI Societies. CEUR Workshop Proceedings*, 1999.

[94] Michael Uschold and Martin King: *Towards a methodology for building ontologies*. In *In Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995.

[95] Stephen L. Reed and Douglas B. Lenat: *Mapping Ontologies into Cyc*. In *AAAI 2002 Conference Workshop on Ontologies For The Semantic Web*, 2002.

[96] Oscar Corcho, Mariano Fernandez-Lopez, and Asuncion Gomez-Perez: *Methodologies, Tools and Languages for Building Ontologies: Where is Their Meeting Point?* Data Knowledge Engineering, 41–64(1), 2003.

[97] Aldo Gangemi Geri Steve and Domenico M. Pisanelli: *Integrating medical terminologies with onions methodology*. In *Information Modelling and Knowledge Bases VIII (IOS)*. Press, 1997.

[98] Guus Schreiber Bob, Bob Wielinga, and Wouter Jansweijer: *The kactus view on the 'o' word*. In *In IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing*, pages 159–168, 1995.

[99] M Fernández López: *Overview Of Methodologies For Building Ontologies*. In *Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5)*, pages 4–13, 1999.

[100] Oscar Corcho, Mariano Fernez-L, Asunci-Pz, and Angel L: *Law and the semantic web*. chapter Building Legal Ontologies with METHONTOLOGY and WebODE, pages 142–157. 2005.

[101] Dean Jones, Trevor Bench-Capon, and Pepijn Visser: *Methodologies for ontology development*. 1998.

[102] Mariano Fernez L, Asunci-Pz, Juan Pazos Sierra, and Alejandro Pazos Sierra: *Building a chemical ontology using methontology and the ontology design environment*. IEEE intelligent Systems, 14(1):37–46, 1999.

[103] Sunitha Abburu: *A Survey on Ontology Reasoners and Comparison*. International Journal of Computer Applications, 57(17):33–39, 2012.

[104] U.S. DEPARTMENT OF HEALTH and HUMAN SERVICES: *Women and Depression: Discovering Hope*. U.S. DEPARTMENT OF HEALTH and HUMAN SERVICES National Institutes of Health, 2009.

[105] U.S. DEPARTMENT OF HEALTH and HUMAN SERVICES: *Anxiety Disorders*. U.S. DEPARTMENT OF HEALTH AND HUMAN SERVICES National Institutes of Health, 2009.

[106] WHO: *Prevention of Mental Disorders: EFFECTIVE INTERVENTIONS AND POLICY OPTIONS*. Technical report, World Health Organization Geneva, 2004.

[107] Mental Health Surgeon General: *Department of Health and Human Services*. Technical report, Washington, DC, 1999.

[108] Karina Gibert, Carlos GarcAlonso, and Luis Salvador-Carulla: *Integrating clinicians, knowledge and data: expert-based cooperative analysis in healthcare decision support*. Health Research Policy and Systems, 28(08):01–16, 2010.

[109] David Coyle and Gavin Doherty: *Towards Ontologies for Technology in Mental Health Interventions*. In *IEEE First International Workshop on Ontologies in Interactive Systems*, pages 18–26, 2008.

[110] Ann Adams and Christopher Buckingham: *GRiST Training Manual*, 2012.

[111] Abu Ahmed and Christopher Buckingham: *Operational specification of the GRiST Decision Support System Version 2.0 BETA 5*, 2009.

[112] Abu Ahmed and Christopher Buckingham: *Data-gathering tool (DGT) requirements- Version 2.0 RC 2*, 2010.

[113] Naomi Wrighton and Christopher D. Buckingham: *Defining an OWL ontology that could be used to integrate mental-health risk information within a decision support system*. In *International Conference on eHealth, Telemedicine, and Social Medicine*, pages 245–250, 2009.

[114] Yung yu TSENG, Wen Long YUE, and Michael A P TAYLOR: *THE ROLE OF TRANSPORTATION IN LOGISTICS CHAIN*. In *Proceedings of the Eastern Asia Society for Transportation Studies*, volume 5, pages 1657–1672, 2005.

[115] Nurizati Syakirin Rosli, Ahmad Yusairi Bani Hashim, and Zamberi Jamaludin: *Tracking of Pallets in Manufacturing Environment Using RFID System*. American Journal of Industrial Engineering, 1(2):36–40, 2013.

[116] Edward A Morash and Steven R Clinton: *The role of transportation capabilities in international supply chain management*. Transportation Journal, 36(3):5–17, 1997.

[117] Christopher D. Buckingham and Abu Ahmed: *Data-gathering tool (DGT) requirements Version 2.0 BETA 3*, 2009.

[118] Kathrin Dentler, Ronald Cornet, Annette ten Teije, and Nicolette de Keizer: *Comparison of Reasoners for Large Ontologies in the OWL 2 EL Profile*. Semantic Web, 2(2):71–87, 2011, ISSN 1570-0844.

[119] Johannes Bauer, Ulrike Sattler, and Bijan Parsia: *Explaining by Example: Model Exploration for Ontology Comprehension*. In *Proceedings of the DL Home 22nd International Workshop on Description Logics*, pages 337–349, November 2009.

[120] S. E. Hegazy and C. D. Buckingham: *A Method for Automatically Eliciting node Weights in a Hierarchical Knowledge Based Structure for Reasoning with Uncertainty*. International Journal On Advances in Software, 2:76–85, 2009.

[121] Christopher Buckingham and Ann Adams: *A decision support system for mental-health risk screening and assessment*, 2007.

[122] Shih Yung Chou and John Pearson: *A Demographic Study of Information Technology Professionals' Organizational Citizenship Behavior*. Journal of Management Research, 3(2):1–15, 2011.

[123] Christopher Buckingham: *Operational specification of the GRiST Decision Support System -Version 4.1*, 2015.

[124] Christian Prehofer: *Solving Higher-Order Equations: From Logic to Programming*. Birkher, 1995.

[125] Matt Mouley Bouamrane, Alan Rector, and Martin Hurrell: *Gathering Precise Patient Medical History with an Ontology-Driven Adaptive Questionnaire*. In *21st IEEE International Symposium on Computer-Based Medical Systems*, pages 539–541, 2008.