

DOCTORAL THESIS

Knowledge engineering for mental-health risk assessment and decision support

Abu Ahmed

If you have discovered material in AURA which is unlawful e.g. breaches copyright, (either yours or that of a third party) or any other law, including but not limited to those relating to patent, trademark, confidentiality, data protection, obscenity, defamation, libel, then please read our takedown policy at <http://www1.aston.ac.uk/research/aura/aura-take-down-policy/> and contact the service immediately eprints@aston.ac.uk.



Knowledge Engineering for Mental-health Risk Assessment and Decision Support

ABU AHMED

Doctor of Philosophy

February 2011

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no quotation from the thesis and no information derived from it may be published without proper acknowledgement.

ASTON UNIVERSITY

Knowledge Engineering for Mental-health Risk Assessment and Decision Support

ABU AHMED

Doctor of Philosophy, 2011

Abstract

Mental-health risk assessment practice in the UK is mainly paper-based, with little standardisation in the tools that are used across the Services. The tools that are available tend to rely on minimal sets of items and unsophisticated scoring methods to identify at-risk individuals. This means the reasoning by which an outcome has been determined remains uncertain. Consequently, there is little provision for: including the patient as an active party in the assessment process, identifying underlying causes of risk, and effecting shared decision-making.

This thesis develops a tool-chain for the formulation and deployment of a computerised clinical decision support system for mental-health risk assessment. The resultant tool, GRiST, will be based on consensual domain expert knowledge that will be validated as part of the research, and will incorporate a proven psychological model of classification for risk computation.

GRiST will have an ambitious remit of being a platform that can be used over the Internet, by both the clinician and the layperson, in multiple settings, and in the assessment of patients with varying demographics. Flexibility will therefore be a guiding principle in the development of the platform, to the extent that GRiST will present an assessment environment that is tailored to the circumstances in which it finds itself. XML and XSLT will be the key technologies that help deliver this flexibility.

Keywords: ontology, classification, XML, XSLT, CDSS, GRiST

In the name of Allah; the beneficent, the merciful

Acknowledgements

The path I have taken in producing this thesis has been illuminated for me by so many people. It is not possible for me to convey the full depths of my gratitude using words, as language itself seems to be too crude a tool. Nevertheless, I shall humbly ask that my ineloquence be overlooked as I try to express my sincere thanks to those who lit my way.

I owe the greatest debt to my supervisor, Dr Christopher Buckingham. This thesis would not have been possible were it not for his boundless patience, his enthusiasm and support, and his unwavering belief in me. He made me see sense where I could only see chaos and has steered me through this endeavour with completely selfless dedication.

I should like to express my immense gratitude to Hifzur Rahman, who brought me back from the brink when I had lost my way. He has provided me with constant support and encouragement throughout this journey; and this, I will not forget.

I should also like to express my gratitude to my oldest friend, Mamun Uddin. He has been my stalwart supporter; relentlessly pushing me forward, yet also showing me that life can still resume during weekends!

My sincere thanks to Faisal Malik and Iram Noreen, who have both shown me the true meaning of dedication. They have been my role models and have inspired me to never turn my back on my aspirations.

I thank Homerha Masood for being my compass and pointing me in the right direction when I was in limbo. I also thank my good friends, Mahnaz Rouf, Salma Jabeen, and Saila Hussain for their kind words and support over the years, as I do Naumana Rabbani and Akhlaq Khan. I am indebted to Drs Ben Ingram and Nabeil Maflahi for their insight and advice on aspects of PhD work, as I am to my friends and colleagues within the Knowledge Engineering Group: Drs Keith Priscott, Anthony Jones, Les Hazlewood and Odetúnjí Odejobí.

I should like to acknowledge the financial support provided by Aston University; particularly

through the award of a School of Engineering-funded PhD studentship. I also acknowledge grants from The National Institute for Health Research, the Burdett Trust, and Advantage West Midlands, which have helped fund this research.

Finally, I should like to thank all the mental-health practitioners and service users who have taken part in this project. They have selflessly donated their time to the cause of developing a mental-health Decision Support System, as have the numerous researchers and programmers, whom I also wish to thank.

Contents

Acknowledgements	4
Contents	6
List of Tables	13
List of Figures	15
List of Abbreviations	16
Declaration	20
1 Introduction	21
1.1 Motivation	21
1.1.1 Current exigencies in mental-health assessment	23
1.2 Thesis Aims and Contributions	23
1.3 Thesis Organisation	24
2 The Galatean Model of Classification	27
2.1 Introduction	27
2.2 Classification Theories	28
2.2.1 Exemplar theory	28
2.2.2 Prototype theory	29
2.2.3 Dual-process theories	29
2.3 The Galatean Model	29
2.3.1 The Galatean classification process	30
2.3.2 The hierarchical Galatean model and its classification process	30

2.4	Conclusions	35
3	Clinical Decision Support Systems	37
3.1	Introduction	37
3.2	The Scope of CDSSs	38
3.2.1	CDSS knowledge representation and architecture	38
3.2.2	CDSS domains	40
3.2.3	CDSS contexts and audiences	41
3.3	Strategic Challenges to CDSS Implementers	42
3.4	The Characteristics of a Good CDSS	43
3.4.1	Practices and features yielding positive outcomes	43
3.4.2	Practices and influences reducing CDSS efficacy	47
3.5	Conclusions	48
4	Choosing a Representation Format for Domain Knowledge	50
4.1	Introduction	50
4.2	The Ubiquity of XML-based Serialisation	51
4.3	Web Ontology Language	52
4.4	On the Appropriateness of OWL for Developing GRiST	53
4.4.1	The maturity of supporting tools	54
4.4.2	The OWL file format is complex	54
4.4.3	Learning curve of OWL ontology creation tools	55
4.4.4	The dangers of feature overload and restrictions	55
4.5	Using XML and XSLT to Produce Flexible Knowledge Representations	56
4.6	XSLT Primer	57
4.6.1	XSLT and web Browsers	58
4.6.2	Standalone XSLT processors	58
4.7	Conclusions	59
5	Validation of Mental-health Knowledge Elicited From Experts	60
5.1	Introduction	60
5.2	Interviewing of Experts & Mind Map Generation	61
5.2.1	Reviewing of mind maps	63
5.3	Generic Software and Web Infrastructure for Remote Activities	64

5.3.1	Tree annotation program	64
5.3.2	Web architecture supporting tree annotation	66
5.3.3	Architecture supporting reviewing of expert annotations	67
5.4	Rationalisation of Consensual Knowledge Structure	67
5.4.1	Notional tree pruning points and their ratification	68
5.5	Interim Conclusions	69
5.6	Increasing Tree Validation Flexibility Through Structured <i>comments</i>	70
5.6.1	Using keywords inside comments	71
5.7	Transforming the Tree via XSLT	72
5.7.1	Results of transforming the pruned tree	74
5.8	Engagement in KE Activities by Panel Members	75
5.8.1	Web task participation	76
5.8.2	Response to emails	77
5.8.3	Navigating the website	77
5.8.4	Engagement with tasks	77
5.8.5	Satisfaction with project participation	78
5.8.6	Lessons learned	79
5.9	Conclusions	79
6	The Structure Tree and its Enrichment	82
6.1	Introduction	82
6.2	Overview of the ST at End of Knowledge Refinement	83
6.3	Semantics and Organisation of Generic Nodes	84
6.3.1	Generic concepts	84
6.3.2	Generic datums	85
6.3.3	Direct risk children	86
6.3.4	Rules governing generic nodes and where they are fully defined	87
6.4	Question-related Paraphernalia and Data Types	88
6.4.1	The different types of question: <code>question</code> , <code>filter-q</code> attributes	88
6.4.2	Generating rapid screening questions: the <code>layer</code> attribute	89
6.4.3	Data types associated with questions	91
6.5	Representing Membership Grade Profiles	96
6.5.1	Collection of preliminary value-mg data	96

6.6	Flexible Assessments Based on User Expertise Level	98
6.6.1	Practitioner expertise quantised as levels	98
6.7	Conclusions	101
7	Representing Relative Influence	103
7.1	Introduction	103
7.2	The ST is Not Suitable for Recording RI Information	104
7.3	Generating the Relative Influence Tree	106
7.3.1	Priming the ST for transformation	106
7.3.2	RIT generation process	107
7.4	RI Elicitation	108
7.5	The Galatean Tree Hierarchy	110
7.6	Conclusions	110
8	Synchronising RITs and STs	112
8.1	Preamble	112
8.2	Galatean Tree Roles	112
8.3	The Knowledge Engineering Process is Iterative	113
8.3.1	GRiST and population diversity	113
8.3.2	Organic evolution through clinical usage	114
8.4	The Problem Definition	114
8.5	Architecture for Tracking a Specific RIT Node	116
8.5.1	Unique node identifiers	117
8.5.2	Node fingerprint audit trail	118
8.5.3	Interim conclusions	120
8.6	Robust Tracking of Nodes with One-to-many Mappings Between ST and RIT . .	121
8.6.1	Fingerprint remediation in the RIT	124
8.6.2	Analysis of revised <i>fingerprint tracking algorithm</i> efficacy	124
8.6.3	Conclusions and discussion on node tracking	129
8.7	RI Reappraisal	130
8.7.1	Changes that affect RIs and schemes for RI reappraisal	131
8.7.2	The application of heuristics in RI reappraisal	132
8.7.3	The mechanics of RI reappraisal	133

8.8	Conclusions	134
9	Deployment of a Preliminary Risk Assessment Solution	136
9.1	Introduction	136
9.2	Rationale for Specialised Assessment Trees	137
9.2.1	The need for a Client Assessment Tree	137
9.2.2	Optimising storage and bandwidth through separation of question data	138
9.2.3	Further optimisation through separation of answer data	140
9.3	Generating the Assessment Trees	140
9.3.1	Generating the CAT	141
9.3.2	Generating the QT	142
9.3.3	Generating the AT	144
9.4	Relating the CAT, QT and AT to the Galatean Tree Hierarchy	145
9.5	Deploying the Galatean Risk Screening Tool	147
9.5.1	Paper-based GRiST	147
9.5.2	HTML-based thin-client GRiST	149
9.5.3	Java-based fat-client GRiST	157
9.6	Reporting in GRiST	158
9.6.1	Client answers report	159
9.6.2	The GTH and specialised reports	160
9.7	Conclusions	161
10	Further Customisation of GRiST for Different Populations and Contexts	164
10.1	Introduction	164
10.2	The Need for Lateral Customisation	165
10.2.1	The organisational perspective	165
10.2.2	The clinical perspective	166
10.2.3	The patient perspective	167
10.3	Rationale for a Super Structure Tree	167
10.4	Super Structure Tree Syntax	168
10.4.1	The <code>populations</code> attribute	169
10.4.2	The enhanced <code>layer</code> and <code>order</code> attributes	169
10.4.3	The SST Enhanced <code>question</code> , <code>filter-q</code> , <code>label</code> and <code>help</code> attributes	170

10.4.4	Inverting value-mgs	170
10.4.5	Population-specific pruning of nodes	171
10.4.6	Adding additional nodes	172
10.4.7	Adding new populations to the SST	172
10.5	The SST and its Incorporation into the GTH	173
10.6	Fingerprint Reconciliation and the SST	174
10.7	Machinery for Generating and Organising STs and Derivative Trees	174
10.7.1	Computer-assisted tree management	175
10.7.2	GTH Tree Validation	175
10.7.3	Incorporating amended trees back into the GTH	178
10.8	Conclusions	179
11	Full Deployment within NHS Trusts: A Case Study	181
11.1	Introduction	181
11.2	Introducing Participating NHS Trusts	181
11.2.1	Holbrook NHS Foundation Trust	182
11.2.2	Cradlemere Partnership NHS Foundation Trust	182
11.3	Deployment Considerations Generic to Trusts	183
11.3.1	Information governance issues	183
11.3.2	Database issues	184
11.3.3	Interface integration issues	184
11.4	The Generic Trust Interface to GRiST	185
11.5	Flexibility Through API Features	188
11.6	Impact of Electronic GRiST Deployment in Partner Trusts	189
11.7	GRiST Usage Outside Trusts and in the Wider Community	192
11.8	Conclusions	193
12	Conclusions and Future Work	195
12.1	Introduction	195
12.2	Review of Thesis Objectives	195
12.3	A Flexible Toolchain for Mental-health Assessment and Decision Support	196
12.4	Benefits of GRiST's Approach to KE	199
12.5	Contribution to CDSS Best-practice Theory	201

12.5.1 Strategic Challenges	201
12.5.2 GRiST and the “ten commandments” for effective CDSSs	202
12.5.3 The “eleventh commandment” for effective CDSSs	205
12.6 Future Work	205
12.6.1 Incorporation of the Galatean model into the server	205
12.6.2 Augmenting of the <i>populations</i> framework	206
12.6.3 Bilateral mappings with other risk assessment tools	207
12.6.4 Application of the GRiST toolchain to new domains	208
12.6.5 Automated mappings from the SST file format to OWL DL	208
12.7 Epilogue	210
References	211
References	211
Appendices	232
A Tree Manipulation Keywords Used Within Structured Comments	233
B Stages Involved in Enacting Tree Changes Using XSLT	236
C Algorithm for expanding “multiple-tick” nodes	240
D Unified Reconciliation Algorithm	242
E Rules and Algorithms for Generating QTs	244
E.1 Rules Governing QT Attributes	244
E.2 Algorithm for Generating a Level 0 QT	246
E.3 Algorithm for Generating QTs at Level 1 and Above	246

List of Tables

8.1	All hypothetical node instantiation configurations in the RIT.	123
8.2	Relocation operations that can be performed on the ST, and their effect on node Fingerprints.	126
11.1	A selection of enhancement requests from HTML GRiST users.	191
12.1	A comparison of GRiST's knowledge acquisition approach with that of Protégé. .	200

List of Figures

2.1	A part of the <i>intention to commit suicide</i> subconcept of the <i>suicide</i> risk structure.	31
2.2	Calculation and propagation of membership grades.	33
2.3	Propagation of membership grades when classifying a patient in the <i>intention to commit suicide</i> concept.	34
2.4	Flow of membership grades through the <i>intention</i> hierarchy.	35
5.1	Mind map coding template developed through content analysis of an interview.	62
5.2	Part of the fully-expanded <i>pattern of episodes</i> concept within <i>suicide</i> risk of the combined map.	63
5.3	Generic Flash validation tool showing knowledge tree with pruning information.	65
5.4	An example of an annotated XML file transformed and viewed in a web-browser.	68
5.5	Fully annotated knowledge structure before and after XSLT transformation.	74
5.6	Sequence of knowledge representations and transformations leading to the ST.	75
6.1	Schematic representation of the Structure Tree.	83
6.2	Schematic representation of the <i>generic nodes</i> pseudo-risk.	87
6.3	Flowchart showing how <i>rapid screening</i> questions are identified and displayed.	91
6.4	Validating the form of answer scales to use for data gathering.	92
6.5	An example eleven-point scale based upon a <code>values="scale"</code> ST node attribute.	93
6.6	An example date control based upon the ST date attributes.	94
6.7	The value-mg profile elicitation tool being used to elicit a profile for a <code>scale</code> node.	97
6.8	Part of the XML for holding information on <i>suicidal ideation</i> in the refined ST.	100
7.1	The RI elicitation tool.	109
7.2	Going from the original ST to the beginning of the Galatean Tree Hierarchy.	111

8.1	Automated and manual actions that are involved in tree generation/updating. . .	114
9.1	Part of a Level 0 CAT XML representing <i>suicidal ideation</i>	142
9.2	Part of a Level 1 CAT XML representing <i>suicidal ideation</i>	142
9.3	The composition of the QT.	143
9.4	Part of a Level 0 QT XML representing <i>suicidal ideation</i>	143
9.5	Part of a Level 1 QT XML representing <i>suicidal ideation</i>	144
9.6	Part of an example AT with answers related to some <i>suicidal ideation</i> questions.	144
9.7	The Galatean Tree Hierarchy extended to incorporate CATs and QTs.	146
9.8	An excerpt from the paper-based version of GRiST.	148
9.9	Back-end processes involved in generating the HTML version of GRiST and reports.	151
9.10	GRiST assessment management interface.	152
9.11	A screenshot of the HTML tool being used to conduct a repeat patient assessment.	153
9.12	A hypothetical U-shaped value-mg profile represented within a scale control. . .	154
9.13	A data validation run being performed as part of the assessment save process. . .	155
9.14	A screenshot of the Java tool being used to conduct a new assessment of a patient.	158
9.15	An example report generated from a hypothetical patient assessment.	159
10.1	The Galatean Tree Hierarchy augmented with the prepending of the Super Structure Tree (SST).	173
10.2	Automatically generating GTH trees from an uploaded SST.	176
10.3	An example error report associated with a generated GTH tree.	177
10.4	Uploading a revised RIT.	178
11.1	Interaction between the NHS Trust and the GRiST server.	187
11.2	Feedback that was received for HTML GRiST via online forms.	190

List of Abbreviations

ACL – Access Control List

AIR – Adobe Integrated Runtime

AJAX – Asynchronous Javascript and XML

API – Application Programming Interface

ASCII – American Standard Code for Information Interchange

AT – Answer Tree

CAT – Client Answer Tree

CB-SCID 1 – Computer-based Structured Clinical Interview for DSM axis 1

CDSS – Clinical Decision Support System

CSIP – Care Services Improvement Programme

CSS – Cascading Style Sheets

DBMS – Database Management System

DLL – Dynamic Link Library

DSM – Diagnostic and Statistical Manual of Mental Disorders

DSS – Decision Support System

ePEX – e-Protechnic Exeter

EXSLT – Extensions to Extensible Stylesheet Language Transformations

g – Generic

gd – Generic Distinct

GP – General Practitioner

GRiST – Galatean Risk Screening Tool

GTH – Galatean Tree Hierarchy

GUI – Graphical User Interface

HCR-20 – Historical, Clinical, and Risk Management – 20

HoNOS – Health of the Nation Outcome Scales

HTML – Hypertext Markup Language

HTTPS – Hypertext Transfer Protocol Secure

id – Identifier

ID – Identifier

IE – Internet Explorer

iPM – i.Patient Manager

ISP – Internet Service Provider

IT – Information Technology

KE – Knowledge Engineering

LAMP – Linux Apache MySQL PHP

LISP – List Processing

MD5 – Message-Digest algorithm 5

MG – Membership Grade

MySQL – My Structured Query Language

NHS – National Health Service

- NPfIT** – National Programme for IT
- OWL** – Web Ontology Language
- OWL DL** – Web Ontology Language Description Logic
- PAS** – Patient Administration System
- PC** – Personal Computer
- PDF** – Portable Document Format
- PHP** – PHP Hypertext Preprocessor
- PhD** – Doctor of Philosophy
- Pii** – Personally identifiable information
- QT** – Question Tree
- RACER** – Renamed Abox and Concept Expression Reasoner
- RDBMS** – Relational Database Management System
- RDF** – Resource Description Framework
- RDFS** – Resource Description Framework Schema
- RFL-OA** – Reasons for Living – Older Adults
- RIT** – Relative Influence Tree
- RI** – Relative Influence
- SCID 1** – Structured Clinical Interview for DSM axis 1
- SID** – Session ID
- SNOMED** – Systematized Nomenclature of Medicine
- SST** – Super Structure Tree
- ST** – Structure Tree
- SVG** – Scalable Vector Graphics

UI – User Interface

UK – United Kingdom

UMLS – Unified Medical Language System

W3C – World Wide Web Consortium

XHTML – Extensible Hypertext Markup Language

XID – Xyleme Identifier

XML – Extensible Markup Language

XMPP – Extensible Messaging and Presence Protocol

XOR – Exclusive OR

XP – Experience

XPath – XML Path Language

XSL-FO – Extensible Stylesheet Language Formatting Objects

XSLT – Extensible Stylesheet Language Transformations

Declaration

This thesis describes work carried out by the author between January 2005 and October 2010 in the Knowledge Engineering Group at Aston University under the supervision of Dr Christopher Buckingham.

This thesis has been composed entirely by the author and has not, nor any similar dissertation, been submitted in any previous application for a degree.

Portions of the work described in this thesis have been published in the peer-reviewed journal article: Buckingham, Ahmed, and Adams (2007).

1

Introduction

1.1 Motivation

Early detection of mental-health problems and associated risk has the potential to avert significant human suffering and to reduce health service and societal costs. Official estimates indicate a high and increasing disease burden associated with mental-health problems, and hence, significant potential to reap benefits by improving early detection and treatment.

The World Health Organisation predicts mental illness will account for 15% of the total world disease burden by 2020 (Whiteford, 2003). The Psychiatric Morbidity Survey shows approximately 20% of people are suffering from a neurotic (16.4%), personality (4.4%) or psychotic disorder (0.5%) at any point in time (Singleton, Bumpstead, Lee, & Meltzer, 2003), and that not all of them get the treatment they need (Singleton & Lewis, 2003). Furthermore, the National Service Framework for Mental Health states that 1 in 6 people have some form of mental illness, depression being the most common (NHS Executive, 1999).

Recent NHS policy (Department of Health, 2004c, 2004a, 2004b; Care Services Improvement Partnership, 2006) emphasises the need for early prevention of mental ill-health and for identify-

ing the associated risks of suicide, self-harm, harm to others and self-neglect. Reducing suicide rates has been a consistent NHS policy objective for a number of years (Department of Health, 2004b, 1999), and recent figures do in fact demonstrate some success amongst the general public (Department of Health, 2006), but not amongst service users recently discharged from hospital care (National Confidential Inquiry, 2006). Notwithstanding, the overall disease burden remains too high. Furthermore, NHS mental-health policy advocates: service users should receive choice in how they manage their own long-term health problems; community empowerment; and providing swifter access to specialist expertise closer to home, with the overall aim of mental-health promotion (Department of Health, 2004c, 2004a).

Formal risk screening and assessment is almost exclusively the domain of health and social care professionals who have undergone specialist mental-health training. These are complex processes because different mental illnesses and conditions carry with them different prognoses and risks to the patient and others, and assessing the likely future course of events involves consideration of a large number of factors. Risk screening and assessment is therefore accepted to be an uncertain business, even for experienced professionals, who can become desensitised to the level of risk presented by service users (National Confidential Inquiry, 2006).

At the heart of the issue is the fact that risk assessment is not a well understood process, either as a clinical activity or a probabilistic science. Although actuarial approaches are often preferred because they are evidence based (Bouch & Marshall, 2005), they may not incorporate the transient, dynamic, qualitative, and idiosyncratic cues that are crucial to clinical judgement (Holdsworth & Dodgson, 2003; Maden, 2001, 2003) or cue patterns (Skegg, 2005; Hanson, 2005; Castle, Duberstein, Meldrum, Conner, & Conwell, 2004).

As a result, there is little agreement over what cues should be recorded (Higgins, Watts, Bindman, Slade, & Thornicroft, 2005) or how they might be considered in combination (Monahan et al., 2000), and why a plethora of different (often paper-based) risk assessment tools are in use (Hawley et al., 2006). Indeed, since mental illness is characterised by uncertainty and unpredictability, some believe there is little scope to improve the decision-making of mental-health professionals (Maden, Scott, Burnett, Lewis, & Skapinakis, 2004), and are sceptical about the risk assessment tools they use (Doyle & Dolan, 2000). This view is not confined to commentators, but also skilled assessors such as doctors (Hawley, Gale, Sivakumaran, & Littlechild, 2010), who perhaps due to the black-box nature in which many assessment tools arrive at results, are left questioning their utility.

In response to the *status quo*, the Department of Health (National Risk Management Programme, 2007) has been seeking to develop a common approach to risk assessment to be followed by all mental-health professionals. Similarly, the NHS National Programme for Information Technology (NPfIT) is grappling with questions about what information to hold about mental-health service users, in the form of a standardised minimum data set.

1.1.1 Current exigencies in mental-health assessment

Having taken a bird's eye view of the state of mental-health assessment in the UK, the following exigencies can therefore be identified within the field, and serve as the motivation for this thesis.

- A need for a proven psychological model of classification that can be used to compute the severity of mental-health risks.
- A need for tool standardisation in the mental-health risk assessment arena. This should be based on a solid and comprehensive knowledge base, which resonates with clinicians' conceptualisation of their field, and which can utilise the psychological model for risk classification.
- A need for transparency in the way assessment outcomes are computed and communicated, giving assessors the opportunity to easily comprehend how a risk calculation has been arrived at.
- A need to take assessments out of the sole purview of mental-health experts and clinicians. Accessibility to appropriate assessment tools should be increased such that assessments can be conducted *confidently* by anyone, anywhere, including the service user themselves.

1.2 Thesis Aims and Contributions

In response to the identified exigencies, this thesis aims to advance the field of mental-health assessment within the UK by developing a principled toolchain for the development of risk assessment solutions. Built on this toolchain will be a comprehensive set of assessment tools for mental-health risk assessment. This system will address the deficiencies of extant assessment practices and tools by providing the following:

- The system would be able to provide risk calculation and decision support capabilities using a computer model of human classification decision making, namely the Galatean model

(Buckingham, 1992; Buckingham & Birtle, 1997; Buckingham, 2002a), to be introduced later.

- The system will both elicit and represent comprehensive domain expert knowledge in a way that is intuitive to domain experts, and which accords with psychological theories on how humans organise knowledge. In essence, it will conduct knowledge elicitation and representation in a way that is amenable to humans as opposed to one that is convenient to computers.
- The elicitation and representation process will be open to incorporating new requirements as would invariably arise when dealing with domain experts and an unfamiliar domain.
- The system will be capable of generating tools that take into account the technical capability of the user conducting the assessment, the user's clinical expertise, *and* the type of patient being assessed.
- The system will clearly communicate a breakdown of where in the assessment risk is being generated, in a way understandable to all parties.
- The system will be available to all through an ordinary web browser and internet connection.

In summary, this thesis will engineer a toolchain for mental-health risk assessment and decision support based on the Galatean model, using a knowledge representation format suited to humans. The platform will deliver the Galatean Risk Screening Tool (GRiST). Crucially, flexibility will be injected into each stage of this toolchain so that GRiST can fulfil its wide remit of being a tool for use by anyone, anywhere.

In a broader research context, the real-world deployment of the toolchain/GRiST will serve as a case study with which to evaluate some of the putative best-practice guidelines to decision support system success promulgated in the literature.

1.3 Thesis Organisation

The remainder of this thesis is organised as follows:

Chapter 2 — Discusses the hierarchical nature of knowledge representation in humans, and introduces the Galatean model of classification decision-making. This model will ultimately

serve as the engine that drives risk classification within the GRiST system.

Chapter 3 — Explores methods of creating clinical decision support systems in general, narrowing down on those that are oriented towards healthcare and mental-health. It examines some of the merits and shortcomings of existing approaches to the development of such systems, aiming to identify best practise. From these, a set of relevant best practice criteria are established. The real-world insight that will be gained from the present project will be used to test these criteria.

Chapter 4 — Considers XML-based representation schemes for the GRiST system’s knowledge structure.

Chapter 5 — Marks the beginning of system development—the starting point being a raw knowledge structure containing to-be-validated consensual domain knowledge. It acknowledges that domain experts are not also computer experts. Consequently, it describes how domain knowledge validation activities were constructed so as to make it easy for domain experts to contribute throughout the process. Furthermore, it describes the role of XSLT in adding additional needed flexibility to the process.

Chapter 6 — Reorganises the refined domain knowledge structure, known as the Structure Tree (ST), and eliminates redundancy. Furthermore, it adds constructs to the ST so that data relating to e.g., questions and answer formats can be instantiated. Further constructs are added in order to enable the ST knowledge to be represented in a more abstract format when needs dictate. This will ultimately enable GRiST to be shorter for more experienced users.

Chapter 7 — Expands the ST into a tree that is closer to the representation that will be used to drive assessments. The main benefit that ST expansion yields is the ability to incorporate uncertainty values (for use by the Galatean model). The expanded tree, known as the Relative Influence Tree (RIT) signals the evolving of GRiST’s knowledge structures as a cascading set of (generated) linked trees, which will ultimately drive the assessment tools. This hierarchy of generated trees is known as the Galatean Tree Hierarchy (GTH).

Chapter 8 — Describes a novel method for amending the ST without having to lose uncertainty data that may have already been manually added to an RIT generated from the ST prior

to the ST's modification. Essentially it is a method for carrying over uncertainty data from an old RIT to a newly generated RIT.

Chapter 9 — Expands the GTH (specifically the RIT) into trees to directly drive the assessment, maintain question data, and maintain answer data. A number of GRiST assessment tools built on these trees, each increasing in sophistication over the previous, are showcased.

Chapter 10 — A powerful mechanism for customising assessments according to the needs of various populations of users is incorporated into the GTH. This is achieved by specifying tree modifications for each population in a meta tree called the Super Structure Tree (SST). This is then transformable into ordinary population STs, which are subsequently used to generate the remaining GTH trees and drive assessments without any modifications to tools.

Chapter 11 — Demonstrates the deployment of GRiST (augmented with the *populations* mechanism) within two real-world settings—NHS Trusts—and evaluates its reception.

Chapter 12 — Summarises and evaluates the work in relation to the thesis objectives, and considers future directions.

2

The Galatean Model of Classification

2.1 Introduction

A fundamental aspect of cognition is the ability to make decisions based upon previous experiences. A subset of decision-making in general is the domain of classification. In classification tasks, the agent uses knowledge that has previously been acquired—expertise—in order to infer how to best describe a newly-presented stimulus. Classification decision-making has empirically been found to exhibit a plethora of idiosyncrasies, making this pervasive area of human cognition all the more intriguing. This has helped spawn a significant amount of research into the phenomenon, and as a result, two major psychological theories of classification have emerged: exemplar theory and prototype theory. In turn, a number of models based on these theories have been advanced. Although models have reached a certain level of sophistication, each model has its strengths and weaknesses, with no model being able to explain/emulate all the major nuances of classification decision-making.

This chapter examines the exemplar and prototype theories before exploring the Galatean Model of classification decision-making—a model based on the prototype framework. Model

features that make it suitable for use within a clinical decision support system for mental-health are highlighted as part of the model's explication.

2.2 Classification Theories

An important issue in cognitive psychology is the manner in which humans represent classes/categories in memory. There are two dominant theories of human classification that have both spawned numerous models with varying levels of explanatory power: exemplar theory and prototype theory. Both theories emerge out of a polarisation in the perspectives of the theorists who subscribe to the respective camp. Kruschke (2010) summarises these positions by terming those preoccupied with low-level structural representations (McClelland et al., 2010), e.g., connectionists, exemplar theorists etc., as *emergentists*. Those who emphasise high-level structured representations and probabilistic inference (Griffiths, Chater, Kemp, Perfors, & Tenenbaum, 2010), e.g., prototype theorists, are referred to by Kruschke as *representational pluralists*.

2.2.1 Exemplar theory

Exemplar theory, drawing from the domain of categorisation, posits that each category is represented by stored exemplars—memory traces of specific examples previously encountered, that have been categorised accordingly. A decision is made by comparing the *similarity* of the to-be-categorised item with the exemplars within each category; the category with the most similar exemplars being chosen (Medin & Schaffer, 1978; Nosofsky, 1986, 1992; Nosofsky, Kruschke, & McKinley, 1992). Similarity is measured and aggregated in various ways depending upon the model under consideration. Garner (1974) suggests the “city-block metric” as a suitable measure of raw psychological *distance*. Here, psychological space is represented as a grid, with exemplars being positionable on each corner of a grid cell. As opposed to a Euclidean distance being taken between two exemplars, the city-block distance is calculated by traversing the intervening cells via cell edges. The raw psychological distance can then be converted into similarity via a mapping function.

In the context of categorising cues in the domain of mental-health to ascertain whether e.g., a suicide risk is indicated, this would involve first examining presented cues. Cues may for example, be the number of previous suicide attempts, or a history of depression etc. The answers to these questions i.e., the cue values would then be compared to those for *all* previous patients

categorised as being in the suicide risk category. They would also be compared with values for all patients previously categorised within the non-suicide risk category. Once an aggregated similarity measure is arrived at, the model would be able to categorise the new patient.

2.2.2 Prototype theory

Prototype theory conceives an outcome as being an abstraction formed from all exemplars previously encountered (Posner & Keele, 1968; Estes, 1986; Smith & Minda, 1998). The prototype can be regarded as a central tendency or an average of encounters that have coalesced to define the category or outcome. This idea is consistent with the view that humans are “cognitive misers” (Fiske & Taylor, 1991), using heuristics and schemas to reduce the information processing and storage burden in arriving at day-to-day decisions.

Within the prototype framework, suicide risk would again be evaluated via a comparison with the suicide and non-suicide categories. However, unlike the case of exemplar models, there will only be *one* comparison to make for each cue in each category. This comparison will be with the prototypical category member, which maintains the average of the cue values from all previous cases falling in that category.

2.2.3 Dual-process theories

There is also a third view of classification that is gaining prominence, and which is liable to heavily influence classification models of the future. There has been recent evidence to suggest that neither theory can fully accommodate the nuances of classification learning and that a dual-process model may be more appropriate, e.g., Smith and Minda (2000) in the case of repeated exposure to individual exemplars, and Kruschke (2006), who advocates a combination of low-level representation in combination with bayesian inference. Currently, the most sophisticated models do not strictly fall into the remit of either theory but rather lean towards one while incorporating some elements of the other.

2.3 The Galatean Model

The Galatean model (GM) is a classification model grounded in prototype theory. In common with the central tenets of the theory, the model uses an abstract member to represent a class. In contrast to other models, which use the mean as the central tendency that defines the proto-

type, the GM elects to use a hypothetical ‘perfect’ member for its prototypical representation. The perfect member is thus the one whose constituent cues’ values yield the highest possible probability of membership—this member is termed by the GM as a *galatea*. Indeed, the GM takes its name from *Galatea*; the mythical Pygmalion’s perfect woman.

2.3.1 The Galatean classification process

The galatean model represents uncertainty in terms of set membership. If the outcome categories are to be considered as sets and the items being classified as potential members, the likelihood that an item is in any one set is given by the degree of membership. This amount is called the *membership grade* (Zadeh, 1965) which, like probabilities, may vary from 1, representing certainty that an object will be in a set, to 0, representing certainty that it will not be a member (Buckingham, Kearns, Brockie, Adams, & Nabney, 2004).

Prior to any classification, the GM will already have been set up with a category Galatea (e.g., suicide) instantiated with component galatean cues (e.g., number of past suicide attempts, history of depression etc). When a patient is presented to the GM for classification, each patient attribute (patient cue) is matched with the corresponding galatean cue. The galatean cue maintains a profile of cue values and corresponding membership grades. The membership grade of each patient cue is therefore ascertained by querying the corresponding galatean cue. The eventual risk attributed to an individual cue depends on its relative influence compared to the other cues it lies alongside within the galatea. As in the case of membership grade profiles, the relative influence of a given cue will be pre-assigned. Therefore, the total membership grade (MG) for the galatea in question is the sum of its component membership grades, weighted by each cue’s assigned relative influence (RI) value.

2.3.2 The hierarchical Galatean model and its classification process

The Galatean model as conceptualised thus far, assumes a single layer of cues, which relate directly to the category galatea. However, this type of relationship is only useful in modelling toy scenarios. Within a real-world domain, experts are aware of relationships between cues. As expertise increases, the organisation of cues becomes more efficient. Experts have been found to represent configurations of cues in an abstract form (Larkin, 1980; Eylon & Reif, 1984), grouping them together into higher level concepts. These higher level concepts can in turn be grouped in to yet still abstracted concepts. In this manner, experts’ conception of the domain

knowledge can be regarded as mainly being hierarchical in nature (Freyhof, Gruber, & Ziegler, 1992; Murphy & Lassaline, 1997). Pilot studies have confirmed this type of representation within the mental-health domain also (Buckingham & Chan, 2002).

Mental-health domain knowledge when conceptualised as a hierarchical structure can be envisaged as maintaining an abstract root called *mental-health risk*. The root node can be decomposed into into nodes representing each of the individual risk areas e.g., *suicide*, *self-harm*, *harm to others* etc. These risk areas can in turn be decomposed through successive iterations into constituent concepts. The final level of decomposition will be individual cues, which can be referred to as *datums*.

Figure 2.1 depicts a part of the hierarchical structure representing the *suicide* risk that was obtained from pilot studies. It can be seen that the *intention* to commit suicide was considered by experts to be a direct contributor to suicide risk. The diagram also shows that this concept was too abstract to be directly measurable by way of patient cues. Hence experts, decomposed *intention* into constituent components of *seriousness of intention* and details regarding the *plan or method* of suicide. *Seriousness* was regarded as being directly measurable, so was not decomposed further, thereby rendering it a datum component. The *plan or method* was regarded yet still as an abstract item i.e., a concept, and was therefore decomposed further into the datum components of *realism* of the plan and the *steps taken* to enact the plan.

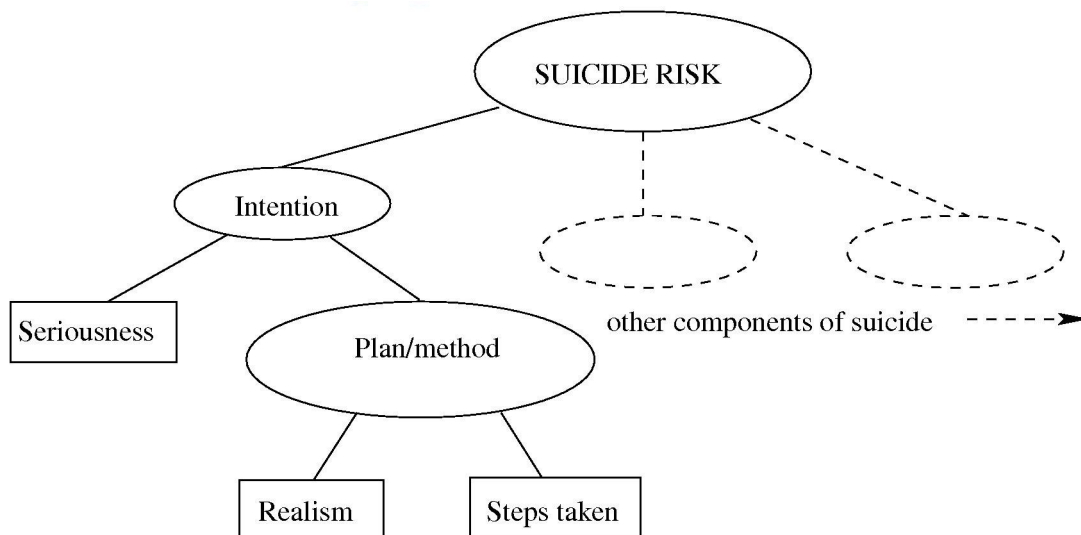


Figure 2.1: A part of the *intention to commit suicide* subconcept of the *suicide* risk structure; reproduced from Buckingham (2002a). Concepts are depicted as ovals, and datum cues as rectangles.

Hierarchical knowledge representations are in fact naturally accommodated by an extended

version of the GM that is to be used as GRiST's classification engine. Such structures are represented in terms of top level galateas, whose galatean cues comprise the immediate child concepts. These concepts can in turn be conceptualised as galateas in their own right. In essence, each concept within the hierarchical knowledge structure is considered to be a galatea. Datums at the lowest level of the hierarchy form the galatean cue components of their parent concepts.

Quantifying the hierarchical galatean structure requires experts to provide each datum (or leaf) component with values that enable membership grades (and thus risk) to be calculated. As in the case of the single layered galatean model, datum components will already have assigned to them a profile of answer values and corresponding MGs. The datum components of the galatean tree are matched with their associated patient cues to produce a membership grade that represents the risk contribution of that particular cue value. This is ascertained via a lookup of the profile of values and MGs i.e., the *value-mg* profile. The eventual risk attributed to an individual cue depends on its relative influence compared to the other sibling cues. Again, RI values will already have been pre-assigned for each component. Thus, the total risk attributable to the containing galatea is a sum of the RI-weighted MGs.

The MGs calculated for each galatea on a given level of the knowledge structure serve as the input MGs for the galatea on the parent level. The child galateas also have individual RIs specifying their weightings. Therefore, the parent galatea's MG (i.e., risk) can be calculated by summing the RI-weighted MGs of the child galateas (Figure 2.2). In this manner, the individual risk contribution of each galatea within the knowledge structure can be calculated, leading to an overall risk prediction for the top level risks, e.g., *suicide*.

Figure 2.3 presents a hypothetical and simplified assignment of membership grades and relative influences to the suicide *intention* substructure of GRiST, to give an illustration of how risks are quantified. The idea is that people focus on the perfect member of a class and are tuned in to the values that maximise membership. Experts are asked to provide the values of a cue that maximise the likelihood of the object being in the associated class and the values that minimise its likelihood. These values are easy to identify even though the real conditional probability would not be, and are respectively assigned MGs of 1 and 0.

According to Figure 2.3, the intention concept has two subcomponents, *seriousness*, which is a datum component, and *plan/method*, which is a concept with its own subcomponents. The seriousness component measures the extent to which the patient is serious about committing suicide, and has values ranging from 0, meaning not serious at all, to 10 meaning completely

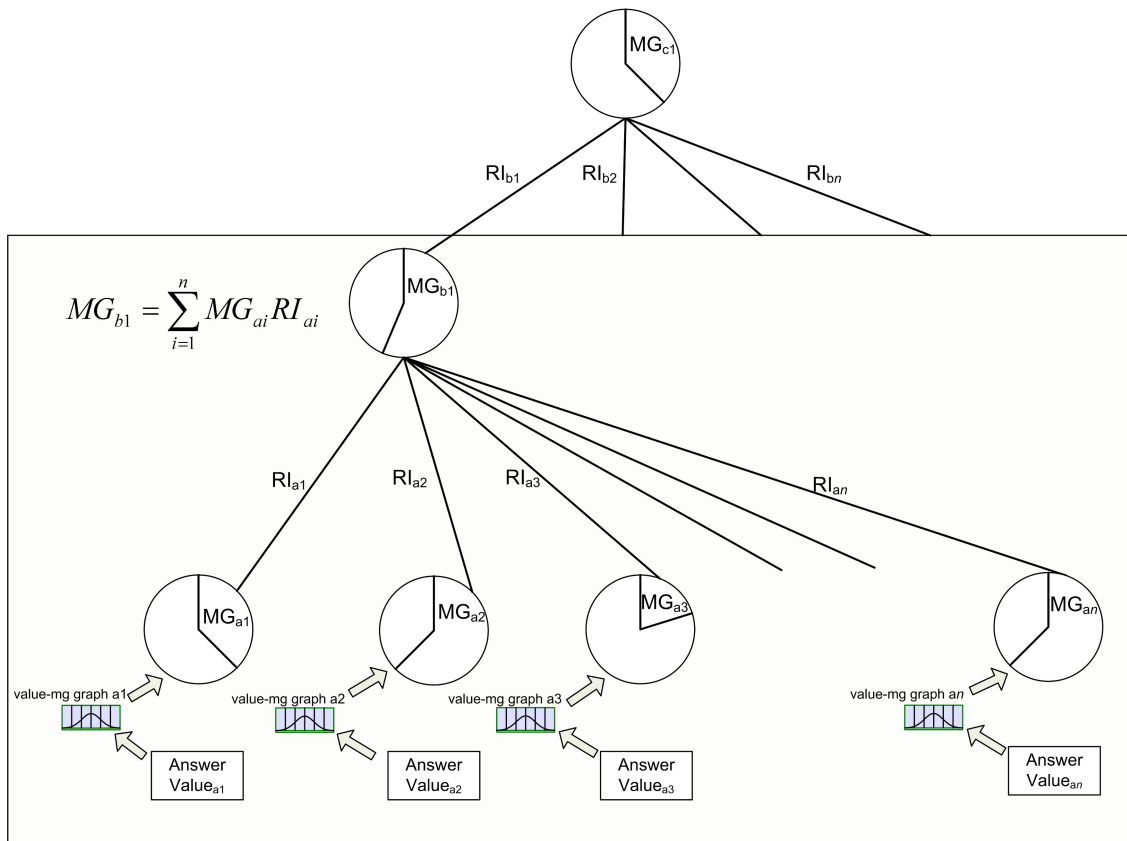


Figure 2.2: Calculation and propagation of membership grades. RI = relative influence; MG = membership grade. Membership grades for concepts on higher levels are calculated by summing the RI-weighted MGs of immediate children concepts.

serious. A value of 10 clearly provides the greatest risk and this is given the maximum membership grade of 1, with the opposite end of the scale having 0 membership in the high suicide-risk category. Any value inbetween has its membership grade determined using linear interpolation.

Next, weightings are provided for concept subcomponents in order to incorporate cue competition. The hypothetical domain expert has assigned a relative influence of 0.7 to the *seriousness* component and 0.3 to the *plan/method* one, indicating that the realism of the plan and the steps taken are less important than the underlying seriousness of the patient in carrying it out.

Using the above method, all components are quantified. To calculate a patient's contribution to *suicide* risk with respect to the *intention* subconcept, the patient's values are matched with the associated datum components as shown by Figure 2.3. For example, if the patient's plan is judged to have a realism value of 7 then it will generate a membership grade of 0.7 in the *realism* datum component. This is then multiplied by the relative influence for realism (0.6) to give its contribution to the *plan/method* subconcept. The patient's membership in all the

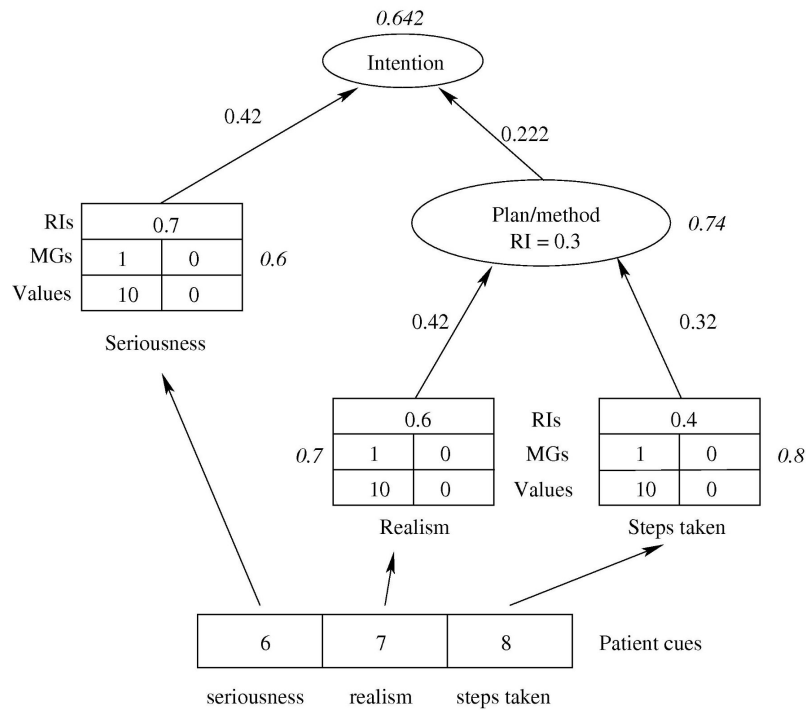


Figure 2.3: Propagation of membership grades when classifying a patient in the *intention to commit suicide* concept: RI = relative influence; MG = membership grade; reproduced from Buckingham (2002a).

components can be quantified by the same process to give a risk of 0.642 with respect to the *intention to commit suicide*. More details about the process and its rationale can be obtained from Buckingham (2002a) and Buckingham and Birtle (1997).

This description of the GM’s representation of expertise is purely hypothetical and does not demonstrate all the ways it can accommodate different aspects of expertise. The aim of the example is to provide a clear and intuitive explanation of how single, integrated risk judgements derive from separate patient cues. Figure 2.4 shows how it can do this for the suicide *intention* component. The values in bold outside the rectangular boxes are the patient’s cue values for the associated datum component. The values within the boxes are the membership grades corresponding to the patient cue values in the datum components; the values on the lines linking components to their parent concepts are the membership grades after multiplication by the components’ associated relative influence.

An important aspect of the Galatean model’s classification process is the ability to “look under the hood” and observe exactly how risk is accumulated. Clinicians are able to investigate where the risk originates, and are therefore in a better position to help the patient.

By manipulating the membership grade distributions of galatea leaf nodes and relative influ-

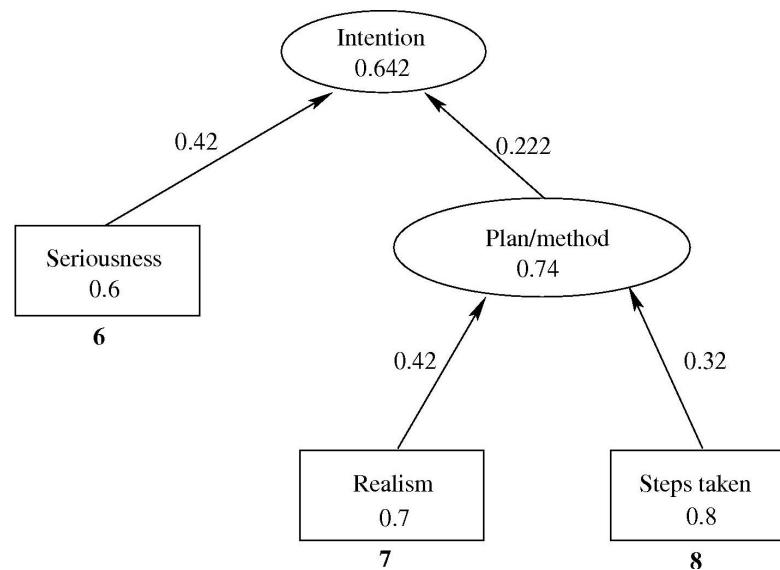


Figure 2.4: Flow of membership grades through the intention hierarchy: patient cue answer values are in bold.

ences of all nodes, experts can easily “tune” the classification process to provide class membership grades that correspond to their own estimates of risk for different patient values. Indeed, the above hypothetical example conceives cue values and MGs as having a linear relationship (specified by two data points). This may not always be the case, and galateas accommodate complex relationships by allowing for any arbitrary distribution of values and MGs to be recorded as a value-mg graph or profile (as depicted in Figure 2.2).

2.4 Conclusions

Classification decision-making is viewed by exemplar theorists as a comparison operation performed against all previously encountered class members. Prototype theorists on the other hand, adopt the stance that the comparison is performed against an abstracted representative of the class. Both theories have spawned several models of classification. This chapter has adopted the Galatean model as the base that will drive the development of a risk screening tool and decision support system for mental-health. The GM is a model based on prototype theory, but uses the hypothetical ‘perfect’ category member for its prototypical representation.

A number of qualities make the Galatean model well-suited for use as a classifier within mental-health:

- It is naturally amenable to representing hierarchical knowledge structures, which domain experts and clinicians will identify with. Such structures consist of central concepts, which are successively elaborated into constituent component concepts.
- Its method of calculation of risk for each galatea and percolation of that risk through the hierarchical structure allows clinicians to examine how and where risk is produced.
- The configurability of the quantification of uncertainty i.e., value-mg profiles and RI values, allows domain experts/clinicians to fine tune the calculation of risk so that it better accords with their conception of it.

Before embarking on the development of GRiST, it is instructive and edifying to examine some of the merits and shortcomings of existing approaches to the development of CDSSs. This is the focus of the next chapter.

3

Clinical Decision Support Systems

3.1 Introduction

The use of clinical decision support systems (CDSSs) has rapidly been increasing over the past decade. This has been fuelled by a combination of a need to increase the quality of health care, and in advances in computing that make it feasible to deploy complex clinical systems.

In preparation for the development of GRiST, it is important to review related literature in order that the lessons learned by others are not lost on the present project. The focus of this chapter is therefore one of ascertaining good practice in relation to the development and deployment of CDSSs. Once established, these core ideas can receive some real-world testing/validation via their consideration in the development and deployment of GRiST.

The chapter will initially review the scope and domains of extant CDSSs and how they compare with GRiST's wide remit. This will then be followed by an examination of design practices and system features that can inform the development of GRiST. The relevant theories testable using the GRiST project as a case study in their application will be highlighted. In later chapters, these will be revisited and evaluated in light of the insights and lessons gained

from the GRiST project.

3.2 The Scope of CDSSs

CDSSs can be defined as computer software designed to aid clinical decision-making. Specifically, they are systems where data about the individual patient are matched to a computerised clinical knowledge base, and where patient-specific assessments and/or recommendations are then presented to the clinician/patient for a decision (Sim et al., 2001; Garg et al., 2005).

3.2.1 CDSS knowledge representation and architecture

The inception of CDSSs dates back to the 1970s, with rule-based approaches to decision-making being the dominant paradigm. One of the earliest systems to gain prominence was MYCIN (Shortliffe, 1976), a proof of concept CDSS designed to identify infectious diseases and to recommend appropriate antibiotics. MYCIN encoded knowledge as a series of approximately 600 production rules, which were formulated via consultation with domain experts. These were statements that mapped patient cues to inferences that could be drawn from them. In essence rules stipulated pre-conditions that needed to be satisfied in order for the rule to ‘fire’.

By altering existing rules or adding new ones, developers were able to rapidly modify parts of MYCIN’s knowledge base. However, these could cause unintended side effects due to the way in which rules were chained together by the system to arrive at a specific decision (Musen, Shahar, & Shortliffe, 2001).

Rule-based systems are an example of knowledge representations that lean towards easy codification by machines, but which are difficult for experts to fully specify. Indeed, Fitts (1964) and Anderson (1982) argued that skill acquisition goes through a series of stages, the last of which is automaticity. More simply put, the greater the level of expertise, the harder for the expert to break down the process into a series of production rules.

Many CDSSs still use rules as the primary knowledge representational mechanism, and as such, are suitable in domains that do not require fuzzy logic (Peleg & Tu, 2006). Thus, rules are successful in deterministic decision-making scenarios that are required to generate e.g., alerts and reminders when certain conditions have been met (Peleg et al., 2001).

Numerous other CDSS architectures exist for knowledge representation and reasoning. For example, models based on Bayesian approaches to the classification problem are particularly suitable to domains where the decision-making outcome is uncertain; i.e., where the same set

of cues could indicate numerous outcomes. The Leeds Abdominal Pain System (Adams et al., 1986) demonstrated success at predicting the probabilities of seven possible causes for abdominal pain using Bayes theorem to process input data such as test results, symptoms etc. As a way of reducing the computational demands of the system, simplifying assumptions such as conditional independence of various disease findings were made (Musen et al., 2001).

Experimental scenarios such as the taxi-cab (Tversky & Kahneman, 1982) or the lawyer-engineer (Kahneman & Tversky, 1973) problems indicate that humans demonstrate poor judgment where conditional probabilities are involved. Gigerenzer (1994) proposes that humans have evolved to respond more to frequencies as opposed to conditional probabilities and percentages. Thus, when probability tasks are reformulated with information presented as frequency counts, performance is markedly improved (Cosmides & Tooby, 1996; Gigerenzer & Hoffrage, 1995). This suggests frequency counts may need to be considered in knowledge elicitation activities that will eventually drive bayesian probability models.

A number of studies involving decision-making where three or more outcomes are involved have found that humans make judgments that are not strictly consistent with the laws of probability (Fiedler & Armbruster, 1994; Koehler, 2000; Koehler, White, & Grondin, 2003). In such studies, when probability judgments for each of the possible outcomes were individually obtained from the person and then summed, the total for many of the cue patterns was found to exceed unity. Support theory (Rottenstreich & Tversky, 1997) elucidates how such a position is arrived at. Briefly, it postulates that the outcomes that are not currently under consideration are grouped together and their influence discounted (i.e. reduced) due to their being grouped. These phenomena represent a challenge to CDSSs based on strictly bayesian probability models, which may possibly affect the quality of classification decisions.

An emerging trend in knowledge representation for CDSSs is the use of ontologies. Ontologies are specifications that document the concepts that comprise a domain, together with their relations (Gruber, 1995). Once encoded in a formal language such as *frames* or *description logics*, ontologies are amenable to processing by inference engines. Thus, inferences can be made on supplied information by evaluation in the context of the concepts and relations held within the ontology/domain model. The results of inference can then be used to facilitative decision support.

The Protégé (Gennari et al., 2003) editor is gaining traction as the *de facto* tool for creating ontologies based on frames or OWL (an XML-based description logic). Protégé is well specified

and is constantly being extended by way of plugins developed by the open source community. However, the complexity of the OWL-DL language itself requires that Protégé be necessarily complex. This fact, combined with the computer science heritage of the nomenclature used within Protégé, results in a steep learning curve for domain experts (Timm & Gannod, 2005; Luciano & Stevens, 2008).

The hierarchical concept structure that will be used by the GM for classification can be considered to be an ontological representation of the mental-health domain. However, given that domain experts will be heavily involved throughout GRiST's development, OWL and Protégé will not be used to codify this knowledge. This is discussed further in Chapter 4.

3.2.2 CDSS domains

Although the operational definition of a CDSS given in Section 3.2 would appear rather narrow, there is considerable variety and scope in the CDSS programs that have been developed for use in the health sector. In a recent review of studies evaluating 100 CDSSs, Garg et al. (2005) broadly classified the systems into the following categories:

- Systems for disease management (40%)
- Systems for drug dosing and prescribing (29%)
- Reminder systems for prevention (21%)
- Systems for diagnosis (10%)

Therefore, the majority of extant systems are focussed towards managing existing conditions such as diabetes or respiratory illnesses (East et al., 1999), or for ensuring the safe prescribing of drugs, e.g., limiting interactions (Tamblyn et al., 2003). These systems typically involve the direct monitoring of patient attributes or the consulting of a knowledge base. Once pre-configured thresholds have been met or combinations of attributes present, these trigger actions within the system. Consequently, many of the current generation of CDSSs can be considered algorithmic or rule-based in nature. Indeed, 76% of systems in a review conducted by Berlin, Sorani, and Sim (2006) were rule-based, with only 3% using a probabilistic model for decision-making.

Using the above taxonomy, GRiST could be best classified as a system for diagnosis, given that it is to be used as a tool for assessing risks of suicide and harm. These issues would

be indicative of an underlying mental-health issue, although GRiST's remit is not to directly diagnose specific diseases. Of the ten diagnostic systems that were identified by Garg et al., only four were mental-health related (G. Lewis, Sharp, Bartholomew, & Pelosi, 1996; Cannon & Allen, 2000; Schriger, Gibbons, Langone, Lee, & Altshuler, 2001; Rollman et al., 2002), with none employing a probabilistic classification model having the sophistication of the GM, but rather, using e.g., scoring methods.

3.2.3 CDSS contexts and audiences

A more elaborate taxonomy for categorising CDSSs is developed by Berlin et al. (2006) and used to analyse 74 distinct CDSSs. The *context* axes of Berlin et al.'s taxonomy describe the setting, objectives and other contextual factors related to the system's use.

It emerges that the majority of systems are primarily developed for use within an outpatient setting (79%). Inpatient systems account for 19% of CDSS deployments.

Although CDSSs show considerable variety in their uses, two distinct classes of CDSS emerge when considering the target decision-maker. Berlin et al. found 62% of systems to be aimed primarily at the clinician as decision-maker, and 46% aimed at the patient. This indicates that there is currently little overlap between clinician-directed and patient-directed systems. Furthermore, out of all inpatient systems evaluated, none were found to support concurrent decision-making by both the clinician and the patient.

The stark lack of overlap between systems designed to be used by clinicians and those for patients is indicative of the inherent complexity within a specialised field such as medicine or pharmacy. The gulf between a physician's understanding of the field and its vernacular and that of the layperson makes developing a unified system challenging. A layperson would view a system designed for a clinician as incomprehensible, terse, difficult to navigate. Conversely, a clinician would view the patient-directed system as imprecise, inefficient, and containing superfluous verbiage. Consequently, system designers have tended to direct CDSSs to either audience but not both; making activities such as shared decision-making more difficult. This is further reinforced by Berlin et al.'s finding that a system was more likely to deliver point-of-care decision-making (i.e., during a shared clinician-patient encounter) if the decision-maker was a clinician than if they were a patient.

The polarity identified in systems with respect to target audience, and a bias towards the clinician is an area that GRiST hopes to improve upon. From the outset, GRiST will be a system

with the challenging brief of potentially being accessible to both clinicians and patients, with neither regarding the system as being inappropriate for them. Such a system would require a flexible architecture that allows for concepts to be polymorphic both in terms of their descriptions and in terms of the degree of their abstraction. Indeed, when such an architecture is in place, further avenues not considered within current studies can be addressed. For example, Berlin et al. consider all clinicians to be equal, irrespective of speciality and the degree of medical training. It may be that decision support can be improved by multiple versions of tools appropriate to the assessor's specific background.

The remainder of this chapter attempts to establish guiding principles for the development of GRiST. It attempts to identify strategic considerations, putative best practice wisdom, and potential pitfalls.

3.3 Strategic Challenges to CDSS Implementers

After consultation with expert CDSS developers, Sittig et al. (2008) identify ten areas of decision support where current implementers should focus their energies. These “grand challenges” offer insight into how decision support systems could be advanced through: the creation of new CDSS interventions; improvements in their effectiveness; better methodologies for the dissemination of existing knowledge and interventions. The top five specific challenges outlined in Sittig et al.'s paper are as follows:

- **Improvements in the human-computer interface** – Alerts and reminders disrupt work-flow, and are often “clicked-through” and ignored by clinicians (Weingart et al., 2003). New, unobtrusive methods of conveying this information and improving adherence need to be developed.
- **Dissemination of best practices in CDSS design, development and implementation** – There is a need for feedback mechanisms for assessing the performance of the CDSS, and comparisons across different implementations of the same CDSS tool. Standardised methodologies and metrics would also allow different systems to be compared.
- **Summarising of patient-level information** – As increasing amounts of the patient's data are transferred to/recorded using an electronic medium, there needs to be intelligent summarisation of data. This will help reduce information overload and bring pertinent matters to the attention of the clinician more quickly.

- **Prioritising and filtering of recommendations to the user** – There should be automatic prioritisation of recommendations after consideration of relevant factors. The driving inputs may be as diverse as clinician’s past performance through to the patient’s insurance coverage level. Improved filtering and prioritisation will help eliminate “alert fatigue” through reduced nuisance interruptions to the clinician’s work-flow.
- **Creation of architectures for sharing executable CDSS modules and services** – A set of standards-based interfaces to the CDSS could enable Patient Administration or Electronic Health Record systems to subscribe to a CDSS in a ‘plug-and-play’ fashion. This will lower the barrier to CDSS adoption and increase the proliferation of CDSSs, as currently, very few such health record systems have any sort of embedded CDSS (Simon et al., 2007).

The GRiST system will establish the foundations for addressing some of the above challenges. Particular strengths of GRiST will be the unobtrusive highlighting of risk, and the creation of a light-weight interface and API with which systems will be able to connect to the CDSS.

The next section explores some of the characteristics of existing systems that have been deemed good prognosticators of success.

3.4 The Characteristics of a Good CDSS

The many methods of representing knowledge and performing decision-making have spawned yet more numerous and innovative CDSSs of varying degrees of intricacy. Nevertheless, the complexity of a CDSS does not always guarantee its successful deployment and acceptance by end-users. A number of studies have been conducted to evaluate real-world CDSS deployments, e.g., Sim et al. (2001); Ruland and Bakken (2002); Bates et al. (2003); Kawamoto, Houlihan, Balas, and Lobach (2005). These have resulted in the identification of numerous practices and features considered to contribute to a more successful system. This section aims to highlight some of the findings in the literature, and draws from them a set of recommendations that can be tested with the deployment of the GRiST CDSS.

3.4.1 Practices and features yielding positive outcomes

In a thorough review of studies investigating CDSS success and failure, Kawamoto et al. (2005) were able to evaluate 15 putative recommendations of features considered important determi-

nants of clinical success. These were investigated via randomised controlled trial studies of CDSSs. The result of this work was the identification of four factors that were able to independently predict clinical effectiveness of a system:

- Automatic provision of decision support as part of clinician workflow — decision support is a natural consequence of the clinician’s activities, and does not require extra work by the clinician.
- Provision of decision support at the time and location of decision-making — immediacy in decision support leading to the clinician actively considering the outputs of the system when formulating e.g., the care plan for the patient.
- Provision of a recommendation rather than just an assessment — the clinician receiving an ‘opinion’ as opposed to raw data, which requires additional cognitive processing in order to derive the opinion.
- Computer-based generation of decision support — the clinician is not required to e.g., perform manual scoring or calculations in order to arrive at decision support.

Kawamoto et al. note that a common theme emerges out of the above statistically significant findings. All four features, when implemented, make it easier for practitioners to use the CDSS. It implies that a good system must minimise the effort required of a clinician when trying to solicit and act on decision support. This should therefore be a guiding precept in the development of GRiST.

Other important factors frequently cited by system implementers, although not found to be statistically significant in the study, include:

- Justification of decision support via provision of reasoning — clinicians may not always be content to blindly follow a recommendation, and may wish to query the reasoning behind the decision as a means of verification. Such a feature can help foster trust in the system, thereby improving user-acceptance levels.
- Local user involvement in the development process — development in collaboration and consultation with the people that will ultimately use the system ensures that the system does not deviate from user expectations. Furthermore, it promotes ownership of the system by the users, again helping to increase user acceptance and uptake.

- Provision of decision support results to patients as well as providers — Decision support should be accessible to patients (in addition to clinicians) so that it helps them to better understand issues related to their own health. Implied within this recommendation is the need for the patients to be able to understand the information that is presented to them, without alienating the practitioner. This is an issue that has been sidelined by many systems by virtue of their being tailored to the clinician or the patient, but not both.

GRiST will attempt to implement these guidelines by heavily involving a wide array of clinicians in knowledge elicitation activities and during incremental development of the system (as suggested by Sim et al., 2001; Iliffe et al., 2002). The system architecture will support assessment by a variety of user-types, including clinicians and patients. Furthermore, the GM, by virtue of the way in which it calculates and propagates risk, will be capable of demonstrating exactly how risk has been accrued.

The findings of Garg et al. (2005) and Wright et al. (2009) augment those of Kawamoto et al. (2005), citing numerous studies where the CDSS was deemed inefficient, requiring more time and effort from the user over paper-based systems, e.g., G. Lewis et al. (1996); Fitzmaurice et al. (2000); Weir et al. (2003). This again suggests a need to reduce the amount of work required of the clinician. Although, it can also be argued that if the CDSS provides enhanced functionality as compared to the paper-based system, then the extra effort might be considered a necessary price for that functionality.

Integration with the existing Patient Administration System (PAS) is a common method by which CDSSs have improved their efficiency e.g., Iliffe et al. (2002). Invariably, information already present within the PAS will form part of the data that will be required by the CDSS. By linking the two systems together, this data does not need to be manually re-keyed. Where GRiST will be used within an NHS Trust-type environment, efforts will be made to create a framework by which relevant patient information can be transferred from the PAS to GRiST. This framework will also help meet the ‘plug-and-play’ challenge set by Sittig et al. (2008), described in Section 3.3.

Finally, Peleg and Tu (2006) discuss the importance of designing and implementing a system that can be maintained and extended. Where knowledge representation and organisation is concerned, this implies that the structures used to encode domain knowledge should be amenable to the incorporation of new and updated knowledge, reflecting e.g., advances in the field. This has been argued as a particularly difficult activity with ontological knowledge structures (Brewster

& O'Hara, 2007). The GRiST system will address this issue head on by building mechanisms directly within the toolchain to facilitate easy updating of the knowledge structure, whilst minimising the need for remedial work.

Ten points for consideration in the development and deployment of a CDSS

An insightful paper by Bates et al. (2003) reports on the first-hand experiences of the authors in deploying decision support applications in the areas of drug ordering, laboratory tests, and radiology procedures. The efficacy of these endeavours has ranged from complete failure to significant levels of success. This has placed the authors in a unique position to provide insight into what works and what does not. Consequently, they advocate a set of ten “commandments” which implementers should follow in order to achieve effective clinical decision support systems. Bates et al.’s commandments are summarised as follows:

1. **Speed is everything** – The goal in terms of infrastructure response should be in the sub-second time-frame. A slow system correlates with marked dissatisfaction from users.
2. **Anticipate needs and deliver in real-time** – It is not enough to make information available for the clinician electronically: the system should be able to make intelligent associations between pieces of information and emphasise pertinent items. This reduces processing time for the clinicians, and increases the system’s usefulness.
3. **Fit into the user’s work-flow** – The recommendations and outputs of the CDSS should be integrated with the clinician’s practice. Maviglia et al. (2003) report that adherence to guidelines issued by the system was significantly increased when the guideline was actively issued at the time of its relevance.
4. **Little things can make a big difference** – Usability is a very important consideration, and can hugely improve a system. It should be made easy for the clinician to “do the right thing”. An example cited would be to use drop-down options instead of free-text where the response is restricted to a set of values. Dexter et al. (2001) report that altering screen flow so that clinicians find it harder to ignore important reminders yields positive outcomes.
5. **Recognise that clinicians will strongly resist stopping** – If clinicians are given suggestions by the CDSS not to carry out an (e.g., poorly effective) action, they will override and persist, unless provided with an alternative course of action.

6. **Changing direction is easier than stopping** – Changing e.g., default values for choices to more sensible ones can result in a change in the clinician’s behaviour regarding those aspects. This can result in cost savings, e.g., with respect to the over-prescribing of medicines.
7. **Simple interventions work best** – Overly long guidelines or items that require too much by way of input produce a barrier effect in clinicians. This reduces their motivation to use the system.
8. **Additional information should be asked for only when needed** - Making requests for information places burdens on the clinician, with the clinician’s possibly having to look-up the information. Each additional required piece of information thus has a cost in terms of overall clinician engagement with the system. The system should try to make provision for situations where required information is not entered.
9. **Monitor impact, get feedback, and respond** – It is important to monitor the effect of interventions, and then be prepared to alter the CDSS based on the feedback received, tuning the system so that it is maximally useful.
10. **Manage and maintain the knowledge-based system** – The system should aim to keep up with advances in the (medical) field. Additionally, tracking and analysing frequencies of alerts etc., can give insight into how the system is being used and identify any problems.

Bates et al. (2003)’s commandments could be considered as representing a general set of best-practice theories for generic CDSS implementers. Although not all of them will be equally applicable to GRiST, the development and deployment of GRiST can help to test many of these in the field of mental-health risk assessment. Therefore, GRiST will serve as a case-study in evaluating the relevance and importance of these guidelines.

3.4.2 Practices and influences reducing CDSS efficacy

In an interesting case study evaluating the deployment of a CDSS for depression, Trivedi et al. (2009) examine some of the barriers that were encountered throughout the process, and the subsequent lessons learned.

The poor level of computer literacy among clinicians was identified as a factor that reduced the effectiveness of the system. Clinicians often had very basic computing skills, which impacted

on the dexterity with which they were able to use the system. Consequently, there was variability in the degree of adherence to the system. Trivedi et al. suggest the provision of enhanced training on the CDSS so that clinicians achieve the necessary competence.

The problem of poor computer literacy is also a finding of the current GRiST research project. However, with the GRiST system, this issue extends through to the knowledge elicitation phases as well. To mitigate these issues, elicitation infrastructure and technology will need to be designed with simplicity in mind. As will be demonstrated throughout the thesis, the choice of technologies such as XML and XSLT (discussed further in Chapter 4) will help overcome some of these challenges.

The need to allow clinicians flexibility and autonomy in the use of the algorithm, as opposed to imposing a rigid process is another important finding from Trivedi et al.'s (2009) study. Ultimately, the clinician is in charge of the assessment and should be able to conduct the assessment as desired, with the ability to e.g., omit items and override recommendations. This sentiment is echoed in a study conducted by Bergman and Fors (2008), which compared the paper-based SCID 1 diagnostic tool for psychiatry with the computer-based CB-SCID 1. The computer-based tool took a very serialised approach to the conducting of the patient assessment. This made it technically difficult for clinicians to 'go back' to revise parts of the assessment or to undertake the assessment in a more randomised global manner.

In light of the above observations, GRiST should not enforce a strict ordering for the answering of the main assessment questions. Furthermore, a search facility should be made available to allow clinicians to locate and jump to specific areas of the question set.

3.5 Conclusions

This chapter has explored the numerous architectures that are commonly employed in the construction of CDSSs. Historically, rule-based systems have dominated. Although, probabilistic bayesian-based systems have also gained prominence. More recently, ontological systems, which try to accurately represent domain knowledge in terms of concept graphs and the relationships between them, are emerging. GRiST could be classed as an ontologically driven system, with the Galatean model also incorporating some probabilistic elements.

There are as of yet very few CDSSs in the area of mental-health, and many do not employ sophisticated reasoning to arrive at decisions. There also appears to be a dividing line between systems that are designed specifically for exclusive use by the clinician and those for use by

the patient. This has an adverse impact on the ability for shared decision-making and on the patient's general awareness about their condition. This is a gulf that the GRiST toolchain hopes to narrow.

A number of factors important to promoting a successful CDSS deployment have been identified in the literature. These can be summarised as the need to reduce the amount of extra work and effort required of the clinician in order for them to obtain decision support. Other factors include; good integration with existing systems and processes, the ability for the system to demonstrate how reasoning is arrived at, recognising that the clinician is in charge and thus removing artificial barriers on how the assessment is to be conducted, factoring in poor user IT skills into system training. Finally, the system should be organic, adaptable to user feedback, and maintainable to ensure longevity.

The next chapter will consider the knowledge representation format that will be used by GRiST, in light of the flexibility that is required of the system and the skill-set of the mental-health domain experts.

4

Choosing a Representation Format for Domain Knowledge

4.1 Introduction

The act of modelling domain knowledge has been referred to by some as more of an art than a science or engineering discipline (Fernández-López & Gómez-Pérez, 2002). What this implies is that there is no recipe that a knowledge engineer can follow in order to arrive at an optimal representation of an unfamiliar domain. There may be more than one way a given domain may be modelled, and each will have its advantages and disadvantages, which will only reveal themselves as the representation begins to crystallise.

The ultimate aim of the GRiST project is to build a toolchain for mental-health risk assessment. A necessary starting point in this endeavour is the construction of a model of the domain knowledge. Therefore, in addition to the ontology representational issues that are faced by the knowledge engineer, further challenges unique to CDSS development also need be considered.

This chapter considers the pressures a widely inclusive project such as GRiST brings to

knowledge elicitation and tool development. It questions the appropriateness of using an ontology modelling language such as OWL in light of these pressures, and argues that it is more appropriate to use a simplistic, yet evolving XML representation, together with XSLT to inject flexibility into the development process.

4.2 The Ubiquity of XML-based Serialisation

The past decade has witnessed a paradigm shift in file format serialisation methodologies. Where arcane binary file formats originally dominated, this is being replaced with human-readable structured text-based formats such as XML. Examples include the latest file formats for Microsoft Office software¹, and the SVG image specification² implemented in the newest iteration of the most popular web browsers. This shift has been initiated due to the convergence of numerous factors:

- A push by governments for open standards and file formats. This is to ensure that all citizens can have access to the public data produced by the state, and benefit from greater options as to how they use that data (Cerri & Fuggetta, 2007). A further issue is one of ensuring archive data can remain accessible to future generations long after the programs used to create that data have become obsolete.
- Increases in computer memory and processing capability. Where once binary formats were a way of teasing every last amount of performance from the hardware, adequate performance is now possible with more human-friendly (and consequently, less machine friendly) file formats.
- The explosive growth of the World Wide Web and the plethora of services built for the dissemination, search and consumption of data by both humans and machines in ways not originally envisaged. This means there is now a much greater need for the portability of data across many different systems and hence greater interoperability.

The Semantic Web (Berners-Lee & Hendler, 2001) is a vision of the World Wide Web that aims to extend the utility of the knowledge contained within it by making it easier for machines to *understand* to an extent, what this knowledge means. This in turn will allow machines

¹The Microsoft Office file format is ratified as ISO/IEC 29500-1:2008

²<http://www.w3.org/TR/SVG/>

to make deductions and inferences when tasked with searches, giving humans access to better quality information. It will also allow more intelligent sharing and linking of disparate yet related islands of knowledge.

Some of the the Semantic Web vision is currently being realised by XML languages such as RDF and RDFS, which aim to describe and relate web resources to each other by way of meta-data, e.g., the Dublin Core initiative (Weibel, 1997). The more advanced, Web Ontology Language (OWL), also XML-based, aims to further these goals by providing sophisticated constructs by which web resources can be described and linked to each other in meaningful ways, facilitating machine reasoning. Although OWL is yet to gain traction within the web developer community, it is due to its expressiveness that it is finding acceptance in the knowledge representation and ontology creation communities. The next section provides a brief overview of OWL Features.

4.3 Web Ontology Language

OWL is an XML-based language for modelling knowledge domains (McGuinness & Harmelen, 2004). Being a Description Logic (Baader & Nutt, 2003), it has the ability to define classes, properties, instances of classes etc. Thus, in some ways it is analogous to an object oriented programming language and utilises similar terminology.

OWL builds on top of its antecedent, RDFS, yet retains the same syntax—the full version of OWL (OWL Full) being valid RDFS. However, additional primitives added to OWL Full render the language undecidable from the perspective of a reasoner (Motik, 2007)—the expressibility of the language is such that not all statements can theoretically be proved or disproved by a reasoner. Consequently, a more constrained version of the language, OWL DL, has been specified. This is a syntactic subset of OWL Full, and ensures decidability (McGuinness & Harmelen, 2004).

The OWL DL language, hereafter referred to as OWL, is the most popular version of the OWL family of languages. It has ontology development tool support, and there are numerous reasoners that can be used with ontologies developed in the language.

As stated earlier, in common with other Description Logics, OWL maintains a distinction between *classes*, *instances* and *properties*.

Class Axioms describe classes. An OWL class is a collection of objects, which are regarded as instances of that class. An instance can belong to none, one, or more classes. Thus, OWL classes can be regarded as sets as opposed to mutually exclusive categories (Knublauch, Tetlow, Wallace, & Oberle, 2006). OWL classes may be subclasses of multiple other classes, and all classes are a subclass of OWL's root class, `owl:Thing`.

Complex classes can be built up using *class expressions*. These are class specifications created from set operations, i.e., union, intersection, complement. Additionally, expressions can include cardinality, define the class via explicitly enumerating its members, and can specify disjointedness.

Instances are individual objects. These can belong to classes, and can have properties assigned to them.

Property Axioms can be used to construct characteristics that may be applied to an object, i.e., define properties. OWL distinguishes between two types of property specifications: *datatype properties* and *object properties*. These link objects to data values and other objects, respectively. Furthermore, properties can be specified with domains and ranges of operation, can have cardinalities, can be related to other properties in multiple ways, can be specified as being transitive etc. An example of an object property is *cookedBy*, which may belong to a *meal* class and have a range encompassing the *person* class.

4.4 On the Appropriateness of OWL for Developing GRiST

OWL is an established language that is widely being used by the ontology development community. However, the goals of ontology development and CDSS development are not necessarily the same, and each area has its own set of needs, exigencies and challenges. Therefore, OWL should not be blindly employed in developing GRiST simply because it is appropriate for use within the ontology development field.

Given the manner and circumstances in which GRiST would be developed, a number of issues were identified with OWL and related paraphernalia that led to its rejection in this project. These concerns are now detailed.

4.4.1 The maturity of supporting tools

At the time of inception of the GRiST project, OWL was a relatively new specification. Consequently tool support e.g., Protégé and associated plugins were at a less advanced stage of maturity than at present. This raised concerns about the viability of the early tools in a project that would involve domain experts in a distributed manner, notably in the underdeveloped areas of collaborative ontology editing and auditing (Leopold, Coalter, & Lee, 2009). This situation has improved somewhat as tool development efforts have intensified.

4.4.2 The OWL file format is complex

OWL is a large specification, offering a vast array of constructs that can be used to model a domain. It is a general-purpose ontology creation language, imbued with the power to model a wide range of domains. This complexity raises a number of issues incongruent with the aspirations of the GRiST project.

Portability of underlying XML — OWL ontologies represent a graph structure. However, in order for OWL ontologies to be serialised as XML, they have to be ‘flattened’ into a sequential form. Unlike typical XML file formats such as XHTML, which have an implicit order dictated by the tree-like nature of XML, OWL serialisations are not bound by such constraints. OWL classes, properties etc., can be defined in any order because each item is represented as a self-contained piece of information. This makes OWL difficult to parse using conventional XML parsers, and so, there is a reliance on specialised APIs that can read and write OWL. This in turn restricts to an extent, the portability of OWL, and increases the complexity of programs that can be used to process it.

Human readability/manual modification of OWL — The serialisation of OWL ontologies into flattened fragments of information mean that it is difficult if not impossible for a human to conceptualise and visualise the underlying ontology from reading an OWL source file. Further, this implies that it is prohibitively difficult to modify an OWL file by hand without the risk of introducing errors and possible side effects. In a real-world modelling/CDSS development scenario governed by financial and temporal constraints, there will invariably be situations that could quickly and cheaply be resolved by ‘tinkering’ with the underlying file format. Indeed, one of the reasons for the paradigm shift to XML was to facilitate human readability of file formats.

4.4.3 Learning curve of OWL ontology creation tools

The Protégé tool is a mature ontology creation environment that is used by OWL ontology developers. It is a GUI-based program that enables users to access the full gamut of OWL features without needing to be concerned with the underlying OWL syntax. Resultant ontologies can be visualised using the built-in viewing capabilities of the program. These capabilities make Protégé a powerful tool in the hands of experienced knowledge engineers.

The GRiST project aim was to develop a CDSS from scratch by involving domain experts throughout the process. Experts would therefore be encouraged to take a hands-on approach to knowledge elicitation and validation tasks. It is at this point that the power of Protégé (which is a necessary consequence of the complexity of the OWL language) becomes a handicap. Had Protégé been deployed to the panel of domain experts, there simply would not have been any engagement from them. Protégé, with its specialised computer science terminology, and myriad ways of specifying items, has a learning curve that is too steep to expect non-computer scientists to master. The problem is further accentuated by the fact that panel members were often clinicians with full-time jobs, and what they signed up to was to help with the development of a CDSS *pro-bono*, and not to learn esoteric software. In fact, many clinicians appeared to have a very poor degree of IT literacy.

4.4.4 The dangers of feature overload and restrictions

As mentioned earlier, OWL is a general-purpose ontology language and has been created with a wide feature set. This does also have some drawbacks. On the one hand, the expressivity of the language means that there is often more than one way to model an ontological or merelogical construct. On the other hand, there are restrictions on certain combinations of constructs that ensure decidability of the language. These two factors make it difficult to decide *a priori* the design decisions that should be made to optimally model an unfamiliar domain (Hoekstra, 2009) and build a CDSS upon that model. Choices and modelling conventions adopted at inception may mean the ontology is unrealisable later on, resulting in the worst case, a major rethink of the modelling approach. This was deemed an unacceptable risk to domain expert participation, given that clinicians were to be involved throughout the development process.

A further issue was that the GRiST CDSS would not require a reasoner, since it would be employing its own classification engine for the calculation of risk. A representational format subject to the rigours of sophisticated reasoning thus represented unneeded complexity.

4.5 Using XML and XSLT to Produce Flexible Knowledge Representations

To address the concerns about OWL that were outlined in Section 4.4, it was decided to instead use a simplified markup language to represent the domain knowledge. Naturally, this would be XML-based, but unlike OWL, would treat the inherent tree-like structure of XML as a strength to be maximally utilised. The Galatean model draws from the literature and takes a hierarchical viewpoint of knowledge organisation. Therefore, the raw structure of XML could be regarded as a good starting point in representing knowledge in a manner that is consistent with experts' conceptualisation of it.

The skeletal structure of the file format would represent concept nodes as XML elements. The decomposition of a concept would be directly mirrored in the file via the XML element's children and descendants. The initial tree structure representing the concept hierarchy could then be embellished and augmented by newly defined XML attributes against an element to record specific details, relationships, links to other nodes etc. In essence, all the information about the element would be stored *in situ* as attributes. This approach would yield several benefits.

- The file format would be plain XML that could be parsed, viewed, and processed using basic XML-aware tools and widgets. In fact, the exploitation of XML's inherent tree structure would mean that even the raw source would convey a meaningful view of the knowledge, and could be edited by the knowledge engineer. The domain expert could have simplified tools created for viewing and modifying the files. These tools would present the hierarchical knowledge and maintain only essential functionality, which would operate on or annotate the underlying XML file.
- XML attributes corresponding to new functionality could be added as the need arose. The specification of these attributes would also be mandated to be human friendly, e.g., friendly path names as opposed to XPath expressions. These would ensure that all the syntax would be relevant and thus minimise tree-bloat, and would be easy to understand.

The most important benefit that would be provided by the bespoke XML format however, is flexibility. When combined with the XSLT language (introduced later), the static XML representation of the domain knowledge can be modified and molded in ways not originally envisaged.

This capability is important to developing a CDSS from scratch for a number of reasons; the whole elicitation process is fraught with uncertainties, potential dead-ends, time pressures, shortages in labour and e.g., skillsets, unforeseen requirements and idiosyncrasies specific to working with domain experts. Therefore, if the representation format can be made to be malleable, these issues are better accommodated, or at least able to be worked around.

XML is becoming increasingly important for CDSSs and their development, but often with respect to enabling applications to communicate despite having different native knowledge representations (Dong, Du, Lai, & Wang, 2004; Dotsika, 2003; Tamineé & Dillmann, 2003). Some systems emphasise the role of XSLT in translating XML, but with the main aim still being interoperability (Jovanović & Gašević, 2005; Tomić, Jovanović, & Devedžić, 2006). A few recognise the importance of XML and XSLT in knowledge engineering (Boegl, Adlassnig, Hayashi, Rothenfluh, & Leitich, 2004) and knowledge maintenance (Del Fiol, Rocha, Bradshaw, Hulse, & Roemer, 2005), especially the ease with which they can help make the process more transparent for domain experts. The current approach specifically exploits the flexibility imparted to the knowledge elicitation process itself, allowing the requirements specification to be altered dynamically without having to e.g., reprogram the elicitation tools used.

The need for flexibility does not end at the knowledge elicitation and representation phases of research. The remit of GRiST itself is to develop a CDSS that is flexible with respect to settings of use, the types and skillsets of users and also the types of patients. Again, if the underlying knowledge representation and tool configuration format can be easily and rapidly re-specified to accommodate these factors, then this is certainly a capability worth exploiting.

To these ends, XSLT will feature prominently throughout the development of GRiST's toolchain; therefore it is instructive to give a brief overview of what XSLT is.

4.6 XSLT Primer

Extensible Stylesheet Language Transformations (XSLT) is a language defined by the W3C (Clark, 1999) to transform XML documents of a particular syntax to those of any other syntax, be they XML, HTML, or even plain text (Kay, 2001). An XSLT file consists of a series of rules (or templates in XSLT vernacular), which tell the processor exactly what to output when elements that satisfy certain criteria are encountered in the XML file. In this way, XML can be modified or even transformed into any other arbitrary markup language. In order to effect a successful XSLT transformation, three components are necessary:

- A source XML file.
- A valid XSLT stylesheet containing transformation instructions to be applied to the source XML file.
- An XSLT processor, which takes the XSLT stylesheet file and applies it to the XML file.

4.6.1 XSLT and web Browsers

Modern web browsers are capable of dynamically applying XSLT stylesheets to XML files and displaying the results of the transformation. This is due to the fact that they have embedded inside them XSLT Processor libraries that are invoked when needed. In order for an XML file downloaded by a web browser to have an XSLT stylesheet applied to it, it is a simple matter of referencing the XSLT file from within the XML file via a header:

```
<?xml-stylesheet type="text/xsl" href="example.xsl"?>
```

When a browser encounters a reference to an XSLT stylesheet, the default behaviour of rendering the XML file as is, is overridden, and the browser instead passes to its XSLT processor the XML file and the referenced XSLT file. The transformation instructions in the XSLT stylesheet are then applied to the XML file, and the output of this transformation is passed back to the browser to render on the screen.

4.6.2 Standalone XSLT processors

Although it is useful for web browsers to incorporate XSLT processors, there are however benefits in using a standalone processor. For example, one of the drawbacks of XSLT processors in web browsers is that they are required to download the source XML file in its entirety even if the transformation will result in minor output such as a summary. Additionally, whereas transformations within a web browser are performed with the output residing in browser memory, it may be desirable to store the result into a file instead.

There are numerous open source standalone XSLT processors, e.g., Apache Xalan, Libxslt, and Saxon. The process for applying a stylesheet is slightly different than is in the case for web browsers, because typically, these tools are invoked from the command-line, passing to them the name of the XML file, the name of the XSLT stylesheet file, and the name of the output file. XSLT processors can also be invoked from within server-side scripting languages such as PHP.

4.7 Conclusions

This chapter considered the issue of a representation format for GRiST's knowledge structure. Driven by interests of accessibility, human readability and portability, XML was identified as an important technology in the area of file formats. Therefore, any prospective serialisation should be in an XML-based language.

The Web Ontology Language (OWL) was identified as the most feature-rich and pervasive XML-based language in extant use in the field of ontology engineering. However, it was argued that the goals of real-world CDSS development are not the same as those of ontology development, and therefore, each field will encounter its own unique challenges. The main challenge in the GRiST project would be that of a desire to include domain experts throughout the development process, whilst dealing with the systemic problem of relatively poor IT skills. Additionally, there was the uncertainty that the myriad options presented by OWL might lead to modelling 'dead-ends' given that the domain was unfamiliar to the knowledge engineers. Any delays as a result would run the risk of inducing apathy among domain experts, and possibly, their withdrawal from the project.

In light of the concerns about using OWL to develop the GRiST toolchain, it was decided to use a much simpler, bespoke XML file format, whose expressive power would be gradually extended as needs dictated. The inherent tree structure of XML is consistent with the Galatean model's conceptualisation of knowledge representation, and is thus suited to being utilised by the model. It was therefore important to maintain this tree structure, and use XML attributes to fortify the evolving file format with new capabilities.

XSLT would play a major role in shaping and molding the underlying XML representation so that it could be made flexible enough to service changing requirements. Furthermore, XSLT would be used beyond knowledge modelling phases, and help tailor GRiST for use in the diverse environments in which it would eventually find itself.

The next chapter details how domain knowledge gathered from experts was validated, and the role of XML and XSLT in facilitating this process.

5

Validation of Mental-health Knowledge Elicited From Experts

5.1 Introduction

The previous chapter introduced XML and XSLT. It explored the rationale for using such technologies in knowledge elicitation for decision support systems. The present chapter builds on this further; describing how XML and XSLT-based technologies were harnessed to traditional qualitative research methods to carry out the deconstruction and validation of mental-health risk knowledge from domain experts.

The chapter begins by briefly introducing the methods that were employed to elicit mental-health domain knowledge from experts. It describes how interview data from each expert was codified as an XML-based hierarchical mind map (Buzan, 2003) and then amalgamated into a consensual mind map.

The above research (specifically; interviewing of experts and mind map generation), described in Section 5.2, constitutes non-PhD work, and was carried out by *other* members of the research

team. It is thus presented only as a prologue to the main body of work reported in this chapter—the development of validation methodologies for data gathered. The goals of these methodologies are:

- Enabling of domain experts, unfamiliar with knowledge engineering (KE) software and methods, to participate in KE and validation activities without the need for extensive training.
- Imparting flexibility on the KE process itself and molding of it, without the need for extensive rework and modifications to existing tools.
- Providing researchers with the ability to audit and analyse feedback from all experts, and to automate their enactment.

To this end, the chapter describes intuitive and flexible methods for both soliciting, and rapidly re-specifying domain knowledge in unforeseen ways during validation rounds. The methods developed provide a vocabulary for annotating elicited domain knowledge data with little or no change to existing software. Furthermore, the methods employ XSLT to formalise annotations into XML markup, and then to finally enact the markup ‘directives’ to arrive at a re-specified knowledge structure.

5.2 Interviewing of Experts & Mind Map Generation

In order to gain a comprehensive model of the domain, the research was designed to accommodate many experts, with the aim of obtaining consensus. Experts from a range of disciplines and backgrounds were recruited to provide multiple perspectives and experiences of risk assessment. Individual open-ended interviews were conducted with 46 experts and later transcribed. This enabled content analysis (Neuendorf, 2002; Halcomb & Davidson, 2006) to be performed on the interviews in order to identify the concepts associated with risk assessment and their constituent components. A mind map coding template for each interview transcript thus emerged and was expanded upon as content analysis of the transcript progressed. An example mind map is depicted in Figure 5.1.

Mind maps originated from the earlier concept maps (see Novak, 2003, for a recent overview) and both are useful techniques for eliciting and representing human cognition (Abernethy, Horne, Lillis, Malina, & Selto, 2005; Siau & Tan, 2005). Mind maps are used in this instance as visual

5.2. INTERVIEWING OF EXPERTS & MIND MAP GENERATION

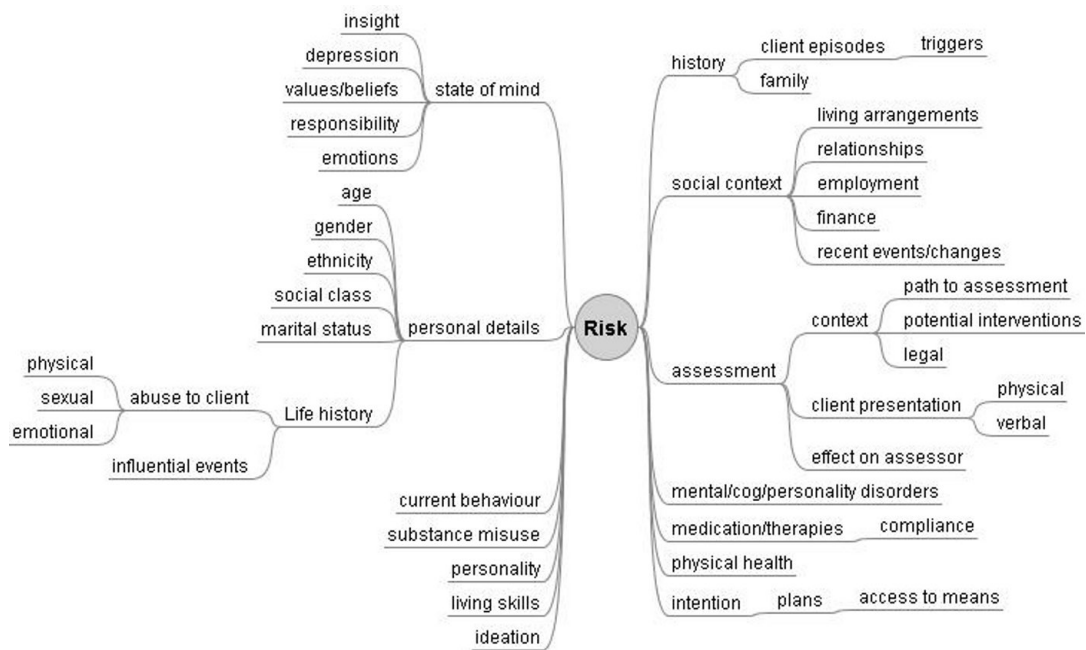


Figure 5.1: Mind map coding template developed through content analysis of an interview.

representations of the inherent structure within an expert’s knowledge, where the main idea is placed in the central “node” of the map. The focal node in Figure 5.1 is “*risk*”, which can be used to represent a particular high-level mental-health risk that has been identified, e.g., *suicide*. Mental-health risks have a number of subconcepts relating to them, which are the immediate branches of the central node of Figure 5.1, such as *history*, *social context*, *assessment*, etc. As the mind map spreads out from the central idea, the concepts become increasingly specific and detailed.

The structure inherent within a mind map of the expert’s knowledge accords with experts’ internal organisation of domain knowledge (Freyhof et al., 1992; Murphy & Lassaline, 1997). It also meshes with the Galatean Model of classification’s hierarchical vista of domain knowledge (Buckingham, 2002b), and thus was a logical choice for coding the data. Furthermore, such a representation is easily translatable into a relatively simple XML format; the knowledge engineering benefits of which, have been explored in Section 4.5. The open source mind mapping tool, Freemind¹, was used to create the mind maps. Freemind’s native file format is a very simple XML-based language, meaning that the output files could almost directly be utilised in further phases of the research.

In order to ensure the provenance and quality of any knowledge that will form part of

¹<http://freemind.sourceforge.net/>

the decision support system, it is important to maintain a full audit trail during knowledge engineering. Consistent with this aim, as each expert’s mind map was being developed, each node was enhanced with the transcript line number(s) which informed it. This meant that each mind map node could be traced back to the primary data that led to its genesis.

The next stage of analysis integrated each individual mind map into a combined map. When a node on an individual map matched one on the emerging combined map, or was added to the combined map for the first time if not already present, the expert’s identification number was placed after the node name, as shown by Figure 5.2. For example, 12 experts mentioned *most recent episode*, the top right subconcept in Figure 5.2. This process allowed each node on the combined map to be traced back to the mind maps of all experts who mentioned it, and from there to the relevant interview transcript lines, thus maintaining the audit trail.

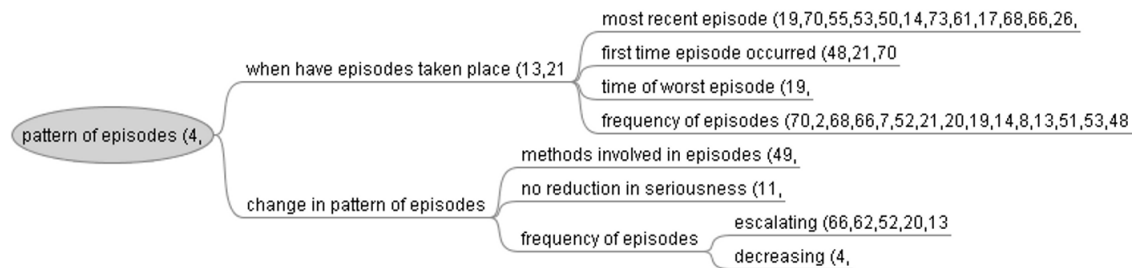


Figure 5.2: Part of the fully-expanded *pattern of episodes* concept within *suicide* risk of the combined map. Numbers after node names represent the identification numbers of different experts who mentioned the node.

5.2.1 Reviewing of mind maps

Good practice for qualitative data analysis includes participants reviewing analysis of their own interviews, and also the integrated results (Mays & Pope, 2000). This was achieved by distributing the results via the project website² and also by post. No experts queried the accuracy of the mind map representing their own interview’s analysis and only minor comments were made about the integrated knowledge structure (i.e., the combined mind map), none raising concerns about its overall legitimacy.

²www.galassify.org/grist

5.3 Generic Software and Web Infrastructure for Remote Activities

Sections 1.2 and 4.4.3 described a fundamental goal of the project as being able to involve non knowledge engineering specialists in knowledge engineering activities. Additional to this, geographical location should no longer be a barrier to participation in an Internet age. This phase of the research marks the point where web-based tools needed to be considered. These would give experts the freedom to work on validation rounds under conditions suited to them. This is also the point where methods for organising the flow of information from experts needed to be instituted. This section describes the tools and methods that were borne out of this. These are then presented against the backdrop of the validation activities that actually took place during this phase of the project.

5.3.1 Tree annotation program

A generic program was specified that would allow easy viewing and navigation of the current knowledge structure, and enable experts to perform validation work.³ It was developed in Adobe Flash⁴, making heavy use of the built-in *treeview* component that could be programmed to display XML files (Jacobson & Jacobson, 2002). Figure 5.3 presents a screenshot of the generic tool, in this case configured to assist with tree pruning, a KE activity discussed further in section 5.4.

From project inception, it was recognised that clinicians are not knowledge engineers; nor are they computer scientists. Furthermore, they were volunteers to the project, donating their time from an otherwise busy day-job. Therefore, as argued in Section 4.4.3, expecting geographically disparate clinicians to learn and use a fully featured Ontology editing application such as Protégé (Gennari et al., 2003) was not an option. It was thus a deliberate design decision to limit the complexity of the Flash program. The program offered only the functionality required to do the validation activity, thus reducing the learning curve for the user. Essentially, the program allowed clinicians to search for, add, delete and rename nodes. It also allowed clinicians to comment or annotate each node.

From a KE process point of view, the Flash validation tool was to serve as a display tool

³The specification for the program was developed as part of PhD work, but implementation was carried out by other members of the research team.

⁴<http://www.adobe.com/products/flash/>

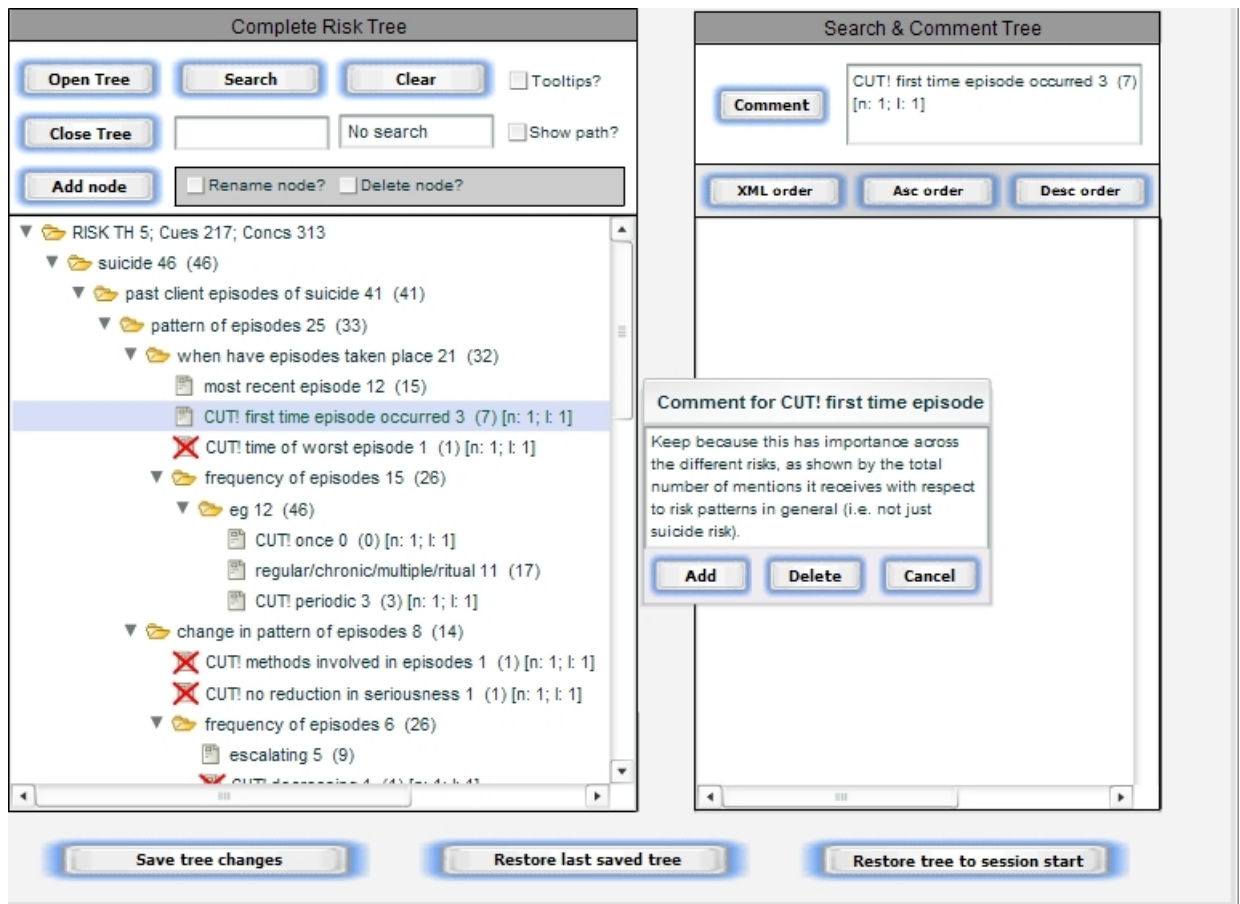


Figure 5.3: Generic Flash validation tool showing knowledge tree with pruning information. The *delete*, *add*, *rename*, and *comment* buttons allow the tool to annotate the underlying XML document, with a comment shown being added to the highlighted node.

and annotator for the XML-based knowledge structure. That is, it would only record comments and changes the expert wished to make, as opposed to physically making structural changes to the file. This had four important benefits:

- Dynamical display of structural changes could disorient the clinician due to e.g., overzealous tree manipulation coupled with the tree being worked on over multiple sittings.
- The XML would always be the same structure, thus allowing knowledge engineers to review feedback more systematically, and consider the input within the context in which it was made. This would help knowledge engineers retain overall control of the activity and facilitate auditing of changes.
- The XML could be transformed using a language suited to the task, e.g., XSLT, without worrying about coding corresponding UI and associated functionality within the Flash

tool. This would save on development effort and time.

- The *comments* functionality could be used to drive manipulation of the XML file in completely unanticipated ways. ‘Structured’ comments would in effect allow the tool to be used to specify new transformational rules/tree information, which could later be formalised and enacted using XSLT. This is discussed further in Sections 5.6 and 5.7.

5.3.2 Web architecture supporting tree annotation

Each expert participating in a validation exercise was given their own login to the project website.⁵ The Flash tool was available once logged in, and was driven by the Knowledge Hierarchy XML file (the yet to be ratified consensual knowledge structure). Each expert was provided with an individualised *copy* of the Knowledge Hierarchy during a validation exercise. The benefit of this approach was that conflicting changes and views between experts could be captured, because each expert started from the same initial position. Additionally, more accurate and easier auditing of participation could be performed (see Section 5.3.3), with the knowledge engineers remaining in ultimate control of the structure.

Another important benefit of individualised XML files over having a shared XML file and using for example, namespaces (Bray, Tobin, Thompson, Hollander, & Layman, 2009) to separate each expert’s input, was that it sidestepped synchronisation issues. The scope of the project was such that multiple experts could decide to conduct a KE activity at the same time from their web browsers. If each expert were to start the session with a copy of the shared XML file, a race condition (Stallings, 2009) would occur when saving the file—data would be lost due to newer saves overwriting older saves not necessarily from the same expert. There are of course numerous real-time messaging and updating protocols e.g., XMPP (Saint-Andre, 2005), Google Wave Federation Protocol (Ferraté, 2010), that could be implemented to enable concurrent editing. However, the overhead of developing such real-time updating systems i.e., employing more sophisticated logic on both the server and client to ensure the system behaviour were deterministic, was not deemed worth the benefits. The number of experts that would potentially be using the system was not high enough to justify a real-time shared workspace, yet was great enough to make synchronisation an issue.

⁵Experts were being recruited throughout the duration of the project, with nearly 100 experts by 2006. Not all experts took part in all validation activities.

5.3.3 Architecture supporting reviewing of expert annotations

Complementing the tree annotation program and architecture driving it were sophisticated reporting features to help review and aggregate expert input. XSLT was employed to effect a transformation on each user's XML tree file, creating a navigable HTML webpage within the web browser, which isolated any comments, deletes, additions etc., and their locations within the tree. Node paths were shown together with yellow highlighted comments. Deletes were in red strikethrough text, and renames were in green with the new label next to the old label (see Figure 5.4). This process could be carried out dynamically for any expert's tree, all from an ordinary computer with a web browser and internet connectivity. No specialised software was required on the browser-end because all modern browsers are capable of applying XSLT stylesheets (i.e., the transformation instructions) to a raw XML file supplied to them, thereby creating the HTML webpage.

Ensuring that all the trees returned by the panel members had the same structure made it easy for XSLT to display the comments for each one and to ensure that directives and comments applying to equivalent nodes could be collated. That is, the XSLT reporting features were able to aggregate and display feedback from all the experts' trees. This significantly expedited review activities.

5.4 Rationalisation of Consensual Knowledge Structure

The complete consensual knowledge structure contained 7,210 nodes. Although many of them were a result of nodes repeated across the root risks of *suicide*, *self-harm*, *self-neglect* etc. In all, there were 1439 unique nodes. Clinicians invariably prefer shorter assessment tools, and this is reflected in the size of many mental-health assessment questionnaires currently in use (Kroll et al., 2003; Watts et al., 2004; Roaldset, Hartvig, & Bjørkly, 2010). Feedback from the expert panel indicated 50 to 100 questions being a comfortable number. Such an unwieldy structure would therefore require heavy pruning and refinement before its use in risk assessment and decision support. This section details how the methodologies established in Section 5.3 were used to successfully rationalise this structure to a more manageable size, and validate the resulting tree.

5.4. RATIONALISATION OF CONSENSUAL KNOWLEDGE STRUCTURE

mental-health-risk --> suicide --> ~~past client episodes of suicide~~ [past and current episodes of suicide] --> client's current perspective on previous episodes of suicide --> ~~thoughts/feelings relating to suicide~~ [thoughts/feelings related to previous suicide attempts]
Thoughts and feelings at the time of suicide are really internal triggers, which pair with the external triggers question. so maybe have a single concept called "triggers of previous suicide attempts" with external and internal (emotional) ones being the two subdata. Then we can ask a question about whether the triggers are still prevailing.

mental-health-risk --> suicide --> past client episodes of suicide --> client's current perspective on previous episodes of suicide --> thoughts/feelings relating to suicide --> ~~after suicide attempts~~
difficult to ask this question without lots of subquestions. The information it picks up is only relevant to the extent that the client will try again, which will be picked up in the current intentions concept. Hence the reason for deleting it.

mental-health-risk --> suicide --> past client episodes of suicide --> client's current perspective on previous episodes of suicide --> thoughts/feelings relating to suicide --> ~~at time of suicide attempts~~
REMOVE cos now part of internal triggers

mental-health-risk --> suicide --> ~~past client episodes of suicide~~ [past and current episodes of suicide] --> client's current perspective on previous episodes of suicide --> ~~insight into lethality of suicide attempts~~ [insight into lethality of previous suicide attempts]

mental-health-risk --> suicide --> past client episodes of suicide --> client's current perspective on previous episodes of suicide --> ~~would client respond the same now to the same circumstances~~

mental-health-risk --> suicide --> past client episodes of suicide --> ~~triggers of past or current episodes of suicide~~
ADD: name [external triggers of suicide episodes]; add-help [suicide >> past client episodes of suicide >> external triggers of previous suicide episodes] ADD: name [internal (emotional) triggers of suicide episodes]; add-help [suicide >> past client episodes of suicide >> client's current perspective on previous episodes of suicide >> thoughts/feelings relating to suicide >> at time of suicide attempts]

mental-health-risk --> suicide --> ~~self-harm behaviour indicative of suicide~~
single question but with help to identify behaviours that might indicate suicidal motivations

Figure 5.4: An example of an annotated XML file transformed and viewed in a web-browser.

5.4.1 Notional tree pruning points and their ratification

A program was developed to provide size/importance metrics on the tree as a whole and on each node. Node statistics included; the number of expert ids mentioning the node in situ and at other locations, the number of nodes in the branch, the number of leaf nodes associated with the branch (which would ultimately be correlated with the number of questions for this branch). The program was provided with parameterised constraints for these statistics. Branches not meeting the constraints would be treated as candidates for pruning, and their labels prepended with the word "CUT".

A threshold of five expert "votes" against a node appeared to ensure that all the important

factors were retained and that the tree was of a manageable size. However the notional pruning points informed by the program required expert ratification to ensure no crucial information was being pruned out. The most efficient method of doing this was by way of a series of focus groups.⁶ Software was developed for clearly displaying the pruning information and recording the decisions of the clinicians, and is described in Section 5.3. Figure 5.3 depicts a screenshot of the tool that was developed to show pruning points within a representation of the tree. These are indicated by the word “CUT” prepended against the node label, followed by some voting statistics appended to the label. Experts were able to ratify the *cut* by clicking on *delete node*, which resulted in a visual indicator denoting its deleted status. The node wasn’t physically deleted from the tree, the rationale for which, was explained in Section 5.3.1. An ancillary benefit of this was that it allowed comments etc., to still be made against the node.

Expert input on the pruning validation activity was reviewed by the research team via the reporting features described in Section 5.3.3. A fresh copy of the XML tree was annotated using the annotation tool to incorporate all the expert participants’ views on nodes to prune. Server-side XSLT stylesheets were developed to enact the *delete* directives present in the tree, thereby reducing the structure and making it ready for the next phase of validation. These features were made available alongside the already developed KE tools described in Section 5.3.1. Therefore, they could be used through any browser, thereby not constraining the project to one geographical location.

5.5 Interim Conclusions

Thus far, the process for arriving at a consensual representation of mental-health risk knowledge has been described. Concomitantly, an evolving ‘toolkit’ of supporting technologies and methodologies for working on the *freemind*-generated knowledge structure has been elucidated. These revolve around a light-weight, minimal tree view/annotation program, generic in nature. Its purpose is to display the tree and serve to record *simple* changes (such as node *additions*, *deletions*, *renames*) and annotations by way of *comments*. Prior to a validation activity, XSLT or (any other appropriate language) is used to supplement the tree with information that may be helpful to that activity. This increases the utility of the generic annotation program without the need for re-coding it. During a validation activity, each expert’s tool receives an individualised copy of the tree. Post validation, XSLT is used extensively to audit tree changes suggested by

⁶Focus groups were conducted by other members of the research team.

5.6. INCREASING TREE VALIDATION FLEXIBILITY THROUGH STRUCTURED COMMENTS

experts, thereby allowing the knowledge engineer to use the tree annotation program (or any other method) to mark up a fresh copy of the XML file. Finally, XSLT can be used to enact any changes. In essence, domain expert-facing tools are simple, with all the processing and transformation logic remaining on the back-end.

The pruning activity is an example of the flexibility afforded by adopting the above pattern in KE for mental-health. Task-specific information was automatically coded directly within the tree, obviating modifications to the domain expert-facing program. As an illustration of the benefits of server-side enacting of recommendations, it was decided *post-hoc* that physical *delete* operations should not result in complete data loss. That is to say, deleting, in addition to deleting the branch, should as a by-product, keep all deleted node names as a `help` attribute within the parent of the deleted branch. This was so that a decision support system based on the tree may ultimately be able to utilise this information to provide contextual information to the user when answering the particular question. It would not make sense to formalise this (and other ad-hoc logic) within the tool because of two reasons:

- Validation activities would have to be stalled each time the tool was readied. Suspending the process to reprogram tools with new requirements is expensive and time consuming, resulting in loss of project momentum, even to the point of endangering successful completion.
- The nature of the KE process in an ambitious project such as this means that not all implementation ideas are deemed to be worth pursuing beyond the prototype stage. Thus, using modular, small and self-contained programs (i.e., stylesheets) in a language specifically designed to easily manipulate XML (i.e., XSLT), would be the logical *modus operandi*.

The next section describes how this design pattern is extended and applied to further validation rounds.

5.6 Increasing Tree Validation Flexibility Through Structured comments

The pruned knowledge hierarchy was reviewed in detail both by individual panel members over the web and by a series of focus groups. The web-based tasks were organised along the lines of Delphi consultations (Ayyub, 2001; Landeta, 2006). These elicit the independent views of each

5.6. INCREASING TREE VALIDATION FLEXIBILITY THROUGH STRUCTURED COMMENTS

participant, which are collated and sent back to the individuals, who are asked to review their input in the light of the collective sample view.

Specific questions to be addressed by the focus groups and by experts working individually over the web were:

1. Are the individual concepts correctly defined? In particular, are there any:
 - missing components;
 - components that should be removed;
 - components that should be renamed?
2. Are the concepts correctly situated within the hierarchy?
3. What is the lowest level of information that people without mental-health backgrounds could reasonably be expected to record?

The above activities therefore involved greater use of *all* the features of the generic tree annotation program, i.e., node addition, deletion, renaming, commenting. One of the issues that emerged from the initial pruning phase of validation was that panel members (i.e., the mental-health experts participating in the pruning task) were wanting to perform a richer set of actions on the tree than was facilitated by the Flash validation tool. However, it was also apparent (from their lack of usage of the tree modification functionality) that users did not like a complicated tool, and were happy with communicating any views via the commenting functionality. This lent more credence to the earlier design decision of not overcomplicating the tool with features. It also highlighted the potential for using the comments facility for recording the richer and sometimes unanticipated operations experts wished to have performed on the tree. This was a method that was particularly amenable to being used in the numerous focus groups, which were led by members of the research team. The next section will explain the use of keywords inside comments to indicate additional operations.

5.6.1 Using keywords inside comments

An expanded set of keywords for transforming the tree was drawn up and published. This enabled focus group leaders (and more computer-literate clinicians) to specify manipulations additional to the default *add*, *delete* and *rename* operations present in the tool. For example,

it was often decided that a particular level (concept node) of the tree was redundant and that the node's children should all be attached directly to the node's parent, with the node itself disappearing. The keyword for this was `DELETE LEVEL`. A node could be annotated with this directive via the original *comments* functionality. Unlike in the case of the default tree manipulations that were available, a keyword did not alter the structure of the underlying XML corresponding to that node. It merely resulted in the creation of a `comment` attribute which was populated with the keyword. In effect, the tool regarded keywords as standard comments; meaning no changes were required to the tool to incorporate the new functionality. A full set of keywords and their functions is presented in Appendix A.

Structured comments and keywords and the general use of marking up operations are intrinsically useful because they provide an audit trail of the evolution of the knowledge structure. They effectively self document within the knowledge structure, what was done in order to get from one round of validation to the next, and the reasoning behind it. This resolves a key difficulty in qualitative research: documenting exactly how the raw data leads to results and conclusions (Silverman, 2004).

The other major benefit of structured commenting and marking up of operations is that they can actually be used to automatically enact the evolution, as described in the next section.

5.7 Transforming the Tree via XSLT

Manually enacting the tree modifications embedded inside structured comments is a laborious and potentially error-prone process. It was therefore decided to automate these manipulations by formulating a series of modular XSLT stylesheets that would enact each of the directives.

The first stage involved using XSLT to parse out the specific operations e.g. *COPY* (and the corresponding location to copy from), *REMOVE*, *REORDER* etc., from the otherwise free-form `comments` attribute of the underlying XML file. XML attributes for these operations were then created in their own right against the node to produce a more structured XML file, marked up with directives to be followed. This is effectively the position the XML file would already have been in, were the generic Flash tool to have the 'keyword' features and associated UI directly baked into it.

An algorithm was formulated to decide on the logical precedence of the directives represented by these new attributes, i.e., the order in which they should be applied. For example, it would not make sense to rename nodes when there were other pending operations that still contained

paths to nodes incorporating the original node labels. A set of XSLT stylesheets were then developed to implement each class of operation, and thereby arrive at a fully revised XML tree at the end of the process.

It can be observed that whereas the transformations in the annotation review phase (Section 5.3.3) were performed solely within the web browser (leaving the original XML file unmodified), the transformations in the subsequent phases required any structural changes to be persistent. This therefore required using a dedicated XSLT processor (a program capable of applying XSLT stylesheets to XML files, and saving the resulting output). The open source *Libxslt*⁷ program was used to apply the stylesheets. It was chosen because of its *libre* nature, its ability to be invoked from within the webserver, and more specifically, because it implemented functionality specifications that were additional to the W3C XSLT 1.0 specification (Clark, 1999). These (community-developed) EXSLT specifications⁸ enabled actions such as dynamic evaluation of strings that contained XPath statements (Harold & Means, 2002; Tidwell, 2008). These were required to perform operations that involved copying a subtree from one location of the XML file to another location, based on the location path specified. Had an XSLT processor that was limited to the XSLT 1.0 specification been used, achieving node copy operations would have been possible, but very tedious.⁹

Another hurdle was the act of generating machine-friendly XPath statements from the more user-friendly path notation adopted in structured comments.¹⁰ These involved ensuring apostrophes etc., were properly escaped, because these have special meaning in XSLT. Therefore, public domain XSLT search and replace functions were used to create temporary attributes that contained escaped XPath versions of paths. These operations were performed as pre-processes and their reverse operations were performed as post-processes, all within the pipeline of transformations towards the final tree.

The different stages of transforming the knowledge tree with XSLT are provided in Appendix B.

⁷<http://www.xmlsoft.org/XSLT/>

⁸<http://www.exslt.org/>

⁹An XSLT processor stopping at XSLT 1.0 compliance *could* be made to achieve copying of subtrees based on paths supplied as strings by: providing it with a stylesheet that would process path strings in the XML, which would then in turn dynamically create a new XSL stylesheet that would replace those strings with equivalent XPath statements, which would then need to be applied to the XML to arrive at the transformed XML. A laborious process!

¹⁰The user-friendly path notation adopted for annotations was a list of node labels separated by a > sign e.g., `label > path > to > node`

5.7.1 Results of transforming the pruned tree

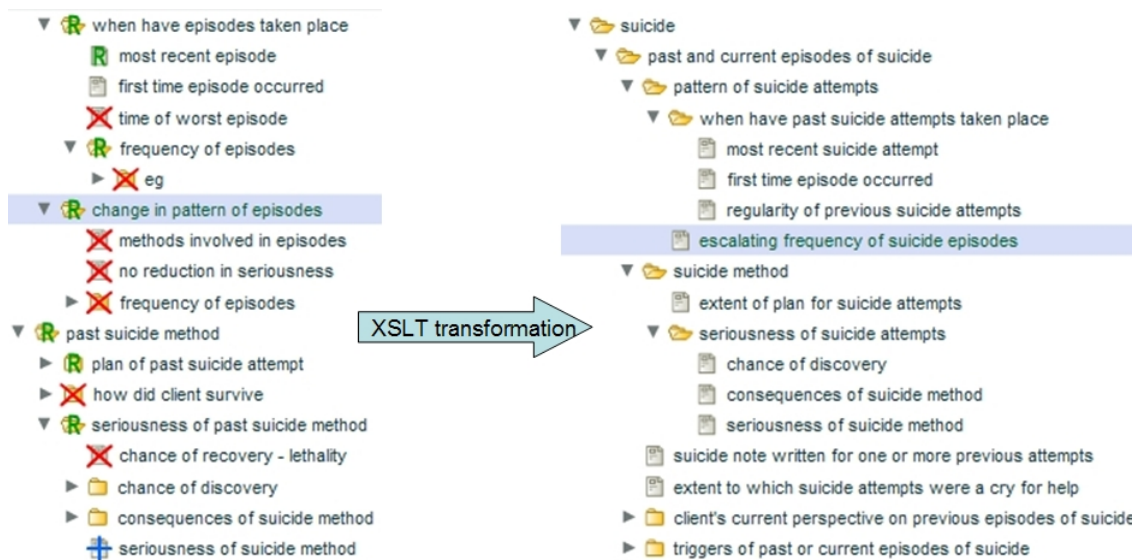


Figure 5.5: The fully marked up knowledge structure on the left shows how the *rename* (R), *delete* (X), and *add* (+) commands mark the nodes; the tree on the right shows how the attributes in the underlying XML are used by XSLT to produce a transformed tree, with the highlighted node illustrating the name change.

Figure 5.5 illustrates the relationship between the marked-up pruned tree and its transformation, as they appear in the Flash tree annotation program. It shows part of the *suicide* risk concerned with *past episodes*, with the *rename* function enabled for the *change in pattern of episodes* node. The following XML extract shows the underlying representation of the annotated node:

```
<node renamedLabel="escalating frequency of suicide episodes"
  label="change in pattern of episodes">
  <node delete="delete" label="methods involved in episodes"/>
  <node delete="delete" label="no reduction in seriousness"/>
  <node delete="delete" label="frequency of episodes">
    <node label="escalating"/>
    <node label="decreasing"/>
  </node>
</node>
```

The transformed tree was more refined with respect to the structure of its knowledge. However before this structure could be used within any potential clinical data gathering or risk screening tool, question text needed to be formulated. Notional questions and answer types were defined for leaf (*datum*) nodes inside dedicated attributes within the underlying XML file (refer to

Section 6.4). These were subjected to validation rounds similar to those already described in this chapter. Further details on this and on the validation rounds described thus far are provided in Buckingham et al. (2007).

The tree resulting from the above validation processes was named the Structure Tree (ST). It was represented as XML, with every leaf node being associated with a question that needed to be considered for it during a potential assessment. The ST represented the end point of the knowledge hierarchy development phase of the project, producing a tree with 394 nodes in total, of which there were 124 unique concepts and 228 unique leaves. Figure 5.6 summarises the evolution of the Structure Tree to this point.

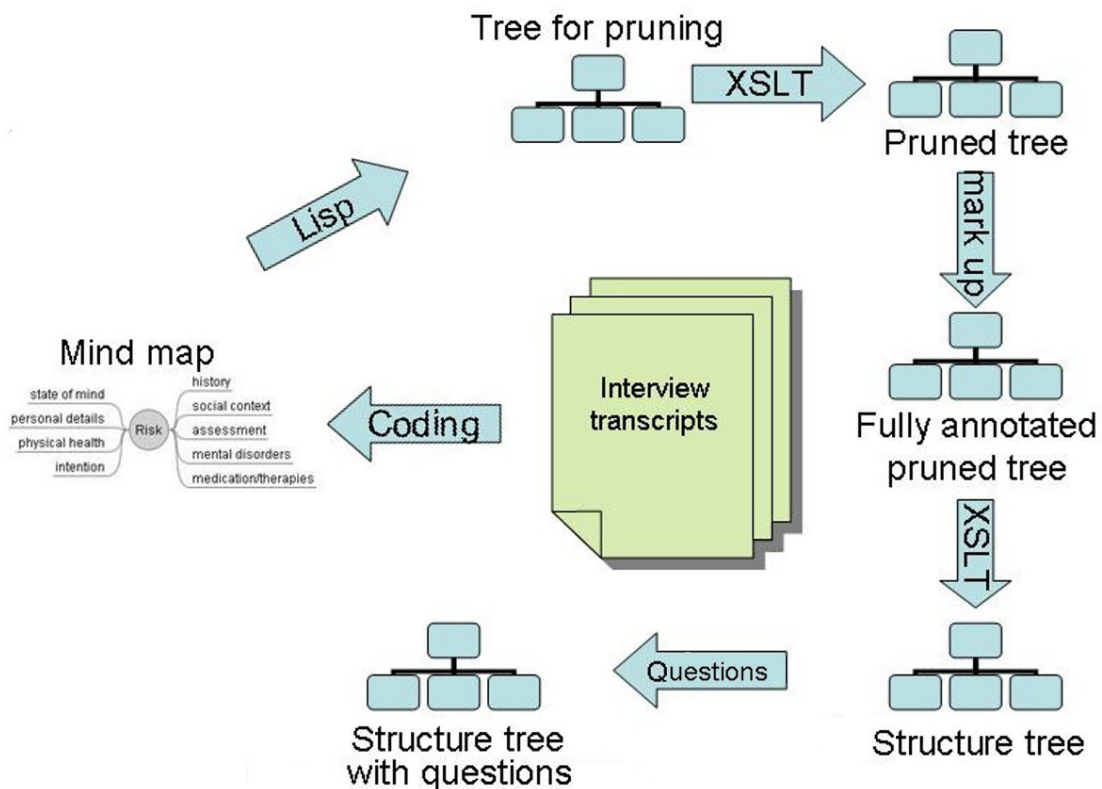


Figure 5.6: Sequence of knowledge representations and transformations leading to the genesis of the Structure Tree (ST).

5.8 Engagement in KE Activities by Panel Members

It is enlightening to pause at this juncture and reflect on the experts' experiences with the KE activities, and their level of engagement in arriving at the ST. In order to solicit this information,

5.8. ENGAGEMENT IN KE ACTIVITIES BY PANEL MEMBERS

during a hiatus in the dissemination of KE activities, a telephone survey was conducted.¹¹ To keep numbers manageable, only the then enlisted expert panel members that had participated in the initial interview phase were contacted (n=38). The survey would allow the research team to take stock and ascertain whether anything could be done differently to increase/improve participation.

The telephone survey was conducted using a structured interview schedule. Following initial orientation to the purpose of the call, experts were asked about their response to a letter they had recently received (informing them of the impending survey), project emails, their login history and future intentions about project participation. Those who had logged into the website were asked further questions about their motivation to do so, ease of site navigation, and ease of KE task completion. Interviews were therefore of varying lengths, and not all experts were asked all of the questions. The following sections will summarise the responses to the different areas investigated during the interview. Sample sizes will vary depending on the number of experts a particular question was applicable to.

5.8.1 Web task participation

The 38 experts surveyed had logged into the website between 0 - 25 times. The mean number of logins was 4, but the mode and median values were 8 and 6 respectively. When examining the most recent login date for each expert, it was found that some had logged in within the previous month, whilst others had not done so for over 2 years. Results over time can be summarised as follows:

- 3 experts logged in within the last month
- 5 experts logged in within the last 3 months
- 12 experts logged in within the last 6 months
- 17 experts logged in within the last 12 months
- 30 experts logged in within the last 24 months
- 8 experts had never logged in.

Taken together, these findings show that the majority of experts who had logged in had been reasonably active on the web, but that participation had tailed off over time, meaning that some re-invigoration of the expert panel was required.

¹¹The survey design was a joint effort within the research team, and analysis does not constitute PhD work.

5.8.2 Response to emails

Since the beginning of the project, the research team communicated with the mental health experts via a series of emails. Twenty experts reported that they had been receiving these emails, with another 9 reporting that they were not. It transpired that 5 of this number had changed their email address without informing the research team. Of the 17 experts who were asked if they read the emails, 10 reported that they did, 5 said they did not, and 2 could not recall. The reasons cited for not reading the emails sent by the research team mainly reflected barriers to participation existing on the experts' side. Reasons given were lack of time due to short staffing, holidays, not being 'good' with computers, problems with a laptop computer, and the emails being buried under a pile of work emails. Messages that needed to be heard by the research team however were that one expert reported needing more support in undertaking these intermittent tasks, while another expressed frustration about the emails just repeating web task instructions already sent out by letter, or else appearing on the website.

5.8.3 Navigating the website

The majority (n=13) of experts asked (n=25) reported that it was easy to navigate around the project website, with only 6 reporting it was not easy, and a further 6 reporting that they could not remember. Comments made in relation to this reflect some expert-side barriers, such as attempting to undertake the tasks outside the deadlines specified on the web, meaning that access was denied; and not giving sufficient time to really looking at the website. Some comments highlighted technical difficulties however, which were important for the research team to be cognisant of. Two experts found the task instructions unclear, and one reported having a 'battle' to get things to work. One expert in particular had difficulty with saving work, and confirming whether it was saved or not. The expert in question requested that there should be more reassuring 'pop up' messages, and reported having lost some of the work done in the past, which is very regrettable.

5.8.4 Engagement with tasks

Experts were asked whether they felt comfortable with being asked to undertake web-based tasks, and the majority of the 25 responding to this question (n=20) responded that they were. Those providing comments about this raised some key issues. Two commented on the large amount of work involved, and the need for a greater sense of achievement in proportion to the

amount of time they had put into the project. Others highlighted more general issues, such as feeling isolated, preferring to work on paper and needing successive explanations from the research team about the overall aims of the project.

Those who had not completed any web-based tasks were asked if they had attempted to do them, with only 2 responding positively. Reasons for not attempting the tasks were: too busy (n=3), leaving mental health work (n=1), confusion (n=1), not allowing sufficient time (n=1), technological problems (n=2) and not receiving email prompts (n=1).

When asked specifically about what would encourage experts to engage more in the web-based tasks, a number of very useful suggestions were put forward. The majority of comments were requesting more personal contact with the research team: focus groups (with provision of food—which was implemented), more meetings with more prior notice, and more site visits to experts in the workplace—to conduct tasks over lunch. Others requested more information, e.g., regular research summaries and more prompting by the team to participate.

5.8.5 Satisfaction with project participation

Twenty five provided responses to a question about overall satisfaction with project involvement. Twelve were either ‘satisfied’ or ‘very satisfied’ with only 2 reporting being ‘dissatisfied’. However there was a sizeable group in the middle reporting ‘neutral’ feelings. There is obviously room for improvement in these figures.

Again, explanatory comments were sought. One expert cited personal reasons for ‘losing track’ of the project, whilst others highlighted issues that were needed to be address by the research team; namely, not communicating the project aims sufficiently clearly, and using too much technical language. Others made suggestions about strategies that could be adopted to facilitate experts’ web participation. These included producing quarterly newsletters about the project, putting photos of the project team onto the website, more telephone calls (particularly to talk through completed tasks), and arranging more ‘refresher’ meetings. Some experts wanted more personal contact and a sense of a working relationship with the research team and other experts. At the other extreme, one expert felt that the team had communicated very effectively: even to the extent of over-communicating with experts!

It was encouraging that all of the experts (n=15) that were asked about whether they wished to undertake future project tasks, confirmed that they would. Only 2 experts out of the whole sample of 39 surveyed indicated that they wished to withdraw from the project.

5.8.6 Lessons learned

The telephone survey proved to be an extremely useful exercise. It was gratifying that the majority of experts found the website easy to navigate and felt comfortable with being asked to undertake web-based tasks. The majority also expressed satisfaction with project participation, and voted overwhelmingly to continue. However, the exercise also highlighted that more could be done to improve experts' experiences.

Some of the lessons that were learned indicated a greater need for the following:

- Ensuring the clarity of language used in any communication, particularly avoiding technical jargon.
- Reminding experts of project aims and objectives periodically.
- Providing regular updates about progress, thus avoiding any hiatuses in communication.
- Providing experts with greater positive feedback to mark and/or reward achievement in completing web-based tasks.
- Giving more notice about planned meetings.
- Adopting mixed methods of communication and interaction with experts.
- A readiness to 'pester' experts more about completing tasks.

The issues of *avoiding technical jargon*, and the research team not fully appreciating the *size of KE tasks* are both indicators of the gulf between knowledge engineers and domain experts. They highlight the fact that even though KE tasks had already been stripped of specialised terminology and parcelled into manageable units of work, this still fell short of suitability for domain experts. There was a greater need for simplification than was envisaged. This finding reinforces the motivations for developing simple KE applications and methods, and eschewing OWL and Protégé.

5.9 Conclusions

Developing a CDSS often necessarily involves the fusion of two separate domains: Computer Science, and the domain of expertise (in this case, mental-health). It is the role of the knowledge engineer to bridge the gulf between these two areas by providing tools and services necessary

to capture knowledge from the experts and represent it in a form amenable to both computers and the experts themselves. The knowledge engineer must recognise that experts should be involved in all phases of the work, thereby engendering a sense of ownership of the project within participants. This in turn feeds back into greater quality of input from experts, and increases the overall success of the project. The knowledge engineer must also recognise however, that a greater degree of involvement from experts sometimes requires their crossing over into the realm of computer science. This presents a barrier to participation, more so where experts are geographically separated from the knowledge engineer and indeed from each other.

Compounding to the problem of enabling experts is the reciprocal problem of knowledge engineers being immersed in the exploration of an unfamiliar domain of expertise. Changing requirements during knowledge engineering are therefore inevitable because the process is not amenable to a comprehensive and watertight specification in advance, even in well-understood domains (Meyer & Booker, 2001).

The aims of this chapter were to develop a KE methodology that appreciated the above difficulties inherent to a project such as GRiST. The design pattern that emerged out of this work can be summarised as:

1. Maintaining a simple front-end tool used to view and manipulate the hierarchy in a domain-expert friendly representation of the expertise.
2. Employing markup of the underlying (XML) file to represent new information, changes etc., on a copy of the tree that is specific to the user.
3. Augmenting KE features of the tool via structured comments where UI and markup have not yet been finalised, or are not desirable.
4. Performing all manipulations on the server.

The key enabler of this methodology was the choice of XSLT as a language to transform the underlying tree-based XML knowledge structure. By employing XSLT, the knowledge structure itself became malleable and adaptable to changing requirements, whilst keeping clinician-facing tools simple and relatively stable. Furthermore auditing of experts' decisions was ensured as a byproduct of the process.

This application of this methodology was presented against the backdrop of the tree validation rounds that needed to be undertaken. This led to the successful refinement and evolution of the knowledge structure, which ultimately became the Structure Tree (ST).

A survey was conducted with experts to examine their level of engagement with the tree validation activities. The results indicated that even though tasks could technically be carried out over the web, the best way to achieve objectives was to employ a mix of methods, e.g., focus groups, small inducements etc. The survey findings also supported the view that it was right to eschew complicated KE tools and terminology in favour of simple tools and staged KE activities.

The next chapter focuses on the composition of the ST, and how this was enhanced, enriched and evolved further.

6

The Structure Tree and its Enrichment

6.1 Introduction

The elicitation, refinement, and validation activities of Chapter 5 resulted in the birth of the Structure Tree (ST). This represents the sum of the knowledge that will ultimately be used to drive the risk screening tool(s) and the decision support system based on the Galatean model. The present chapter provides an overview of the organisation of the ST. It explains how structural redundancies inherent to the gathered knowledge necessitated the birth of new constructs and conventions to distill them. The aim of the chapter is to demonstrate the seamless process by which domain knowledge representation segues into the capability to gather data for risk assessment. Therefore, the chapter provides a detailed explication of the enrichment process that was carried out in order to metamorphose the ST from a pure knowledge structure to one more capable of driving a risk screening tool. Finally, it discusses how customisability was injected into the ST (and therefore, risk assessment) in order that clinician experience can be taken into account when generating screening tools.

6.2 Overview of the ST at End of Knowledge Refinement

The elicitation process of Chapter 5 resulted in a knowledge structure that identified five main areas of mental-health risk. These were, *suicide*, *self-harm*, *harm to others*, *self-neglect*, and *neglect of dependents*. These “top-level” risk nodes were represented in the ST as direct children of the root node, *mental-health-risk*. Each top-level risk was composed of further sub-concepts that broke it down into its constituent components. In this manner, the risks recursively decomposed into further *concepts* (i.e., nodes that have children) and finally, to individual *datum* or leaf nodes. Each datum node represented a piece of information that could be solicited by a potential data gathering or risk screening tool. Thus, each datum node had attached to it a question that could be asked to collect this information. This was stored against the datum node inside a `question` attribute. Indeed, within the ST’s XML-based file format, any supplemental information in relation to a node was stored as an attribute within the element representing the node. A schematic representation of the ST is presented in Figure 6.1.

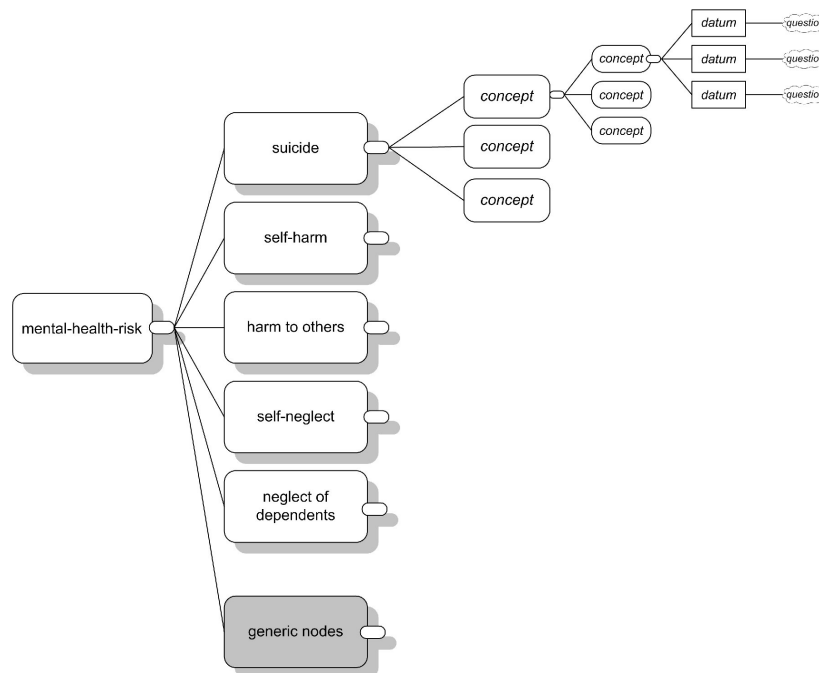


Figure 6.1: Schematic representation of the Structure Tree. Below the root node, there are five top-level root risks (coloured white) and one pseudo-risk (coloured grey) to store repeating nodes. Rectangular nodes are datums, with attached clouds indicating `question` attributes.

The process of coding interview transcripts and generating the initial risk hierarchy also identified the need for a distinction between repeating concepts and those that are specific to

particular risks. It was for this reason that there were initially such a large number of nodes.¹ For example, *depression* is a concept that repeats throughout the knowledge hierarchy across all top-level risks, and so is termed *generic*. Wherever such generic concepts appear in the tree, they necessarily have the same constituent components. Therefore, it was decided that all such repeating nodes, both concepts and datums, should be defined under a special *generic nodes* top-level pseudo-“risk” (see Figure 6.1). This would result in repeating nodes having their structures and questions located in one place. Everywhere else they occurred in the tree, a “stub” node would be placed containing a path to the generic definition of the node. This principle is similar to the design of relational databases, where redundancy of information is eliminated by normalisation to prevent inconsistencies and duplicate data entry (T. Connolly & Begg, 2010).

6.3 Semantics and Organisation of Generic Nodes

Section 6.2 established a need for identifying generic concepts and defining them in one central place. However, further specification of generic nodes was required in order for the tree to adequately serve any potential decision support system based on the Galatean model. These elaborations are described below.

6.3.1 Generic concepts

Recall from Section 2.3.1 that additional to a concept’s structural components, the Galatean model of classification requires Relative Influence (RI) values for these components. These provide weightings for each internal component relative to each other. When reconsidering generic concepts in light of RIs, it emerges that although *all* instantiations of a given generic concept share the same structure, *not all* generic concepts will have instantiations that will *additionally* have the same configuration of RIs across them. That is to say, some generic concepts will have RIs that vary with the location of that concept. This is true for *feelings/emotions*, for example; two sub-concepts of which are *anger* and *hopelessness*: the RIs for these subcomponents would not be the same for the risks of *suicide* and *harm to others*.

On the other hand, many generic concepts are not context dependent, and always have the same internal RI configuration wherever they occur. The *depression* concept is an example of this. *Depression* is a direct measure of a patient attribute, therefore its internal components

¹Initially, there were 7,210 nodes.

are stable and it can thus be treated as a “black box” when instantiated. *feelings/emotions* however, is a concept that is essentially a holdall for a collection of properties of the patient (*anger, mood swings, distress* etc.) but is not in itself a meaningful patient variable. Instead its meaning is derived from how its subcomponents collectively contribute to the root risk. Thus, when the root risk context changes, so does the collective contribution of the components of *feelings/emotions*. The distinction is a subtle but important one which would come into play when RIs were to be instantiated. Thus, in preparation for this, the following distinctions were introduced in the evolving ST:

g concepts – a homogenous generic concept (i.e., a generic concept with fixed internal RIs) was dubbed a *g* concept. This was indicated within the ST by a `generic-type="g"` attribute against the concept node at the location of its definition.

gd concepts – a heterogenous generic concept was dubbed a *gd* (meaning *generic distinct*) concept. This was indicated by a `generic-type="gd"` attribute against the concept node at the location of its definition.

Furthermore, the *generic nodes* pseudo top-level risk was refined to contain a *generic concepts* pseudo concept child (refer to Figure 6.2). Both *g* and *gd* concepts were now all placed inside the *generic concepts* node. Instantiation locations retained a stub node in lieu of the instantiation. The stub node contained a pointer to the generic definition, i.e., an attribute:

```
generic="generic nodes >> generic concepts >> path >> to >> generic  
>> concept "
```

The ST specification was also extended to allow the definition of nested generic concepts inside the *generic concepts* node if that more naturally reflected the structure of the concepts under consideration.

6.3.2 Generic datums

Most repeating datum nodes (i.e., leaves of the ST that correspond to directly assessed patient data) will only repeat within generic concepts. As a consequence, they will automatically be defined only once within the generic concept definition. However, there may be some that do not fall in this category, and will repeat outside generic concepts. As described in Section 2.3.2, datums will require a value-mg profile to be associated with them. This describes the degree to which a given answer value for the datum indicates membership or degree of activation of

that datum. Since an answer value corresponding to a repeating datum is not envisaged to be contingent on its location, it would imply that the datum could be classed as type *g* using the taxonomy introduced in Section 6.3.1. Therefore, having the datum physically repeating in the ST would introduce a redundancy with respect to its value-mg profile (and possibly its question text). To eliminate this potential redundancy, such generic datums were defined in a *generic datums* pseudo-concept, which was a sibling of the *generic concepts* pseudo-concept (refer to Figure 6.2). These adopted the same identifying syntax as was defined for generic concept definitions; namely:

```
generic-type="g"
```

attributes against the datum definition. For completeness,

```
generic-type="gd"
```

attributes were also specified. Stub nodes at generic datum instantiation locations contained a `generic-datum` attribute with a path pointing to the location of the datum's definition.

6.3.3 Direct risk children

A number of generic concepts and datums were identified as being common to all of the five top-level risk strands depicted in Figure 6.1. The native positions of these generic concepts were as direct children of the top-level risk nodes. It was decided not to refactor these concepts using the conventions established in Sections 6.3.1 and 6.3.2. This was because their numerosity meant that the number of stub nodes that would be required to service them post-relocation would in itself be adding to ST bloat. Instead it was decided to create an additional sibling to *generic concepts* and *generic datums* in which to directly house this class of nodes. Thus, the nodes were placed in a *direct risk children* pseudo-concept within the *generic nodes* pseudo-risk. No stub nodes were retained within each of the risk areas for *direct risk children*. Rather, a convention was established that generic nodes contained within the *direct risk children* pseudo-concept were to be instantiated in each of the top-level risks as direct children. This convention obviated the need to specify stub nodes and paths, thereby eliminating both redundancy and tree bloat. Figure 6.2 depicts the now augmented structure of the *generic nodes* pseudo-risk first introduced in Figure 6.1.

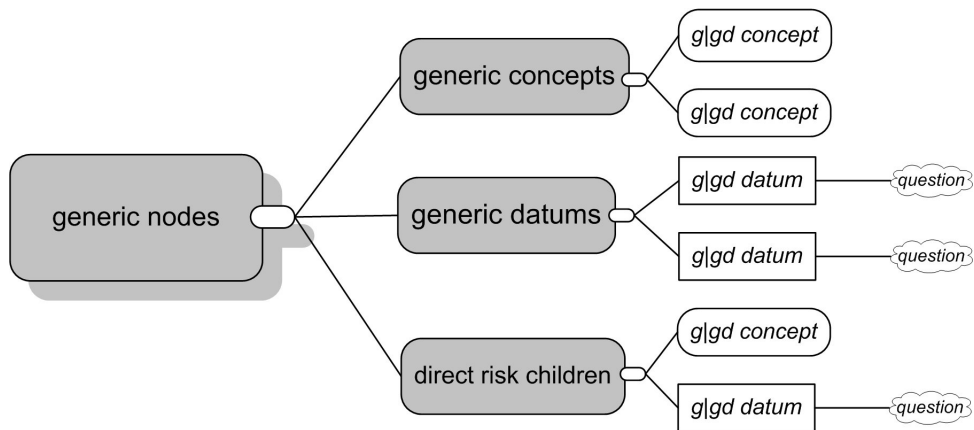


Figure 6.2: Schematic representation of the *generic nodes* pseudo-risk. It contains three pseudo-concept children, which store generic concept and datum definitions. These are referenced from the rest of the ST via paths contained in stub nodes.

6.3.4 Rules governing generic nodes and where they are fully defined

A number of rules were established to clarify exactly where generic concepts were to be defined and the implications of more complicated scenarios involving nested generic concepts, taking into consideration the semantics of *g* and *gd* concepts. This would ensure that the tree remained viable (semantically) for use in a risk screening tool. Furthermore, formalising of such rules would help with automated validation of the correctness of the reorganised ST. The rules are as follows:

1. Generic nodes have the same identifier wherever they occur in the hierarchy.
2. All generic nodes have their full structure defined in one place only.
3. *g* concepts are never *defined* within a *gd* concept. However, see the next point.
4. If a *g* concept is *contained* within a *gd* concept and the *g* concept is not itself fully defined as a *direct risk child*, then the full definition of the *g* concept is within the *generic concepts* section.
5. A *g* concept's components will remain homogenous even where that *g* concept is contained in a *gd* concept.
6. *g* concepts can be fully defined within other *g* concepts even if the *g* subconcept is independently referred to from elsewhere outside the containing *g* concept.

7. *gd* concepts can be fully defined within other *gd* concepts even if the *gd* subconcept is independently referred to from elsewhere outside the containing *gd* concept.
8. A *gd* concept can be fully defined within a *g* concept even if the *gd* subconcept is independently referred to from elsewhere outside the containing *g* concept.
9. A *gd* concept fully defined within a *g* concept will be restricted to being homogenous in locations where it has been transitively referenced via a reference to the containing *g* concept.

The above rules, although not explicitly mentioning generic datums (for reasons of brevity), apply to them equally.

6.4 Question-related Paraphernalia and Data Types

Section 5.7.1 briefly detailed that question text and associated data types were finalised in consultation with experts. The present section elaborates on the results of this process and outlines how these, and other related considerations, were accommodated within the ST specification. These ST enhancements would help enable the tree to directly specify risk assessment tools, thereby bridging the gap between ontology and assessment.

Not all of the constructs outlined in this section were conceived of during this epoch of the research. That is to say, some items were identified as being useful during later research phases, and so, retrospectively incorporated within the ST, as it was the logical location for their placement.

6.4.1 The different types of question: `question`, `filter-q` attributes

Each leaf node of the ST corresponds to an item of patient data for which, a question can be asked. Hence the term *datum* has been applied to them. Figure 6.1 shows that each datum node stores its associated question within a `question` attribute.

Earlier feedback from the expert panel indicated that experts do not like to answer very large sets of questions during a patient assessment. This was mirrored in the work of e.g., Kroll et al. (2003), Watts et al. (2004) and Roaldset et al. (2010). A limitation of paper-based questionnaire tools is that all questions need to be presented whether they are applicable or not. The situation is improved somewhat by the use of ‘filter-questions’, which direct the agent to a different section in cases where it has been established the current section is not applicable.

The hierarchical nature of the ST and the notion that concepts break down into sub-concepts and ultimately, datums, meant it was easy to adapt the filter-question approach to the ST. Rather than having a risk-screening tool that presented all datum questions in one go, it was possible to designate certain (higher-level) concept nodes as filter nodes. These were signified by a `filter-q` attribute that contained a *yes/no* question to be answered by the expert. These nodes could act as gates within the tool, with the questions underneath only being revealed if the `filter-q` question were to be answered in the affirmative. This would considerably reduce the number of questions exposed to the assessor.

6.4.2 Generating rapid screening questions: the `layer` attribute

A common approach to risk data gathering is the administering of a short screening tool, where all the important questions are asked first (e.g., Watts et al., 2004; Patel, Harrison, & Bruce-Jones, 2009). This then allows the clinician to see at a glance whether further assessment using the full set of questions is necessary (Brooker & Fox, 2009). The hierarchical nature of the ST is a drawback in this respect since there is an implicit ordering in the nodes. The ordering may make sense for a one-stage monolithic tool, but there needs to be a ‘layered’ approach (Vickers, 1994) when more than one type of ordering and prioritisation (i.e., rapid screening tool and full tool) needs to be accommodated.

The flexibility provided by the extensibility of XML meant that layering could be relatively easily accommodated in the ST with the introduction of an additional attribute. Layering was implemented by introducing the

```
layer="n "
```

attribute, where $n \in \mathbb{N}_0$. The presence of the `layer` attribute would imply that any associated question (present in a `question` or `filter-q` attribute) was to be presented ‘up-front’ as a rapid screening question. The value of n would indicate the order of precedence in relation to other screening questions, zero being highest precedence.

Algorithm for the treatment of `layer` nodes

An algorithm was established as to how a potential risk assessment tool should use the `layer` attribute during generation/rendition of rapid screening questions. This is outlined below. A flowchart corresponding to the algorithm is presented in Figure 6.3.

1. If the **layer** attribute is against a datum node, there is no change to the question output; the only purpose of the **layer** attribute in this case is to prioritise that particular node for the rapid screening question display.
2. If the **layer** attribute is against a concept node and there is a **question** attribute against the concept, the question will relate to the **layer** attribute. The node is essentially treated the same as a **filter-q** node, with the treatment being as follows:
 - (a) the answer type is always *yes/no*;
 - (b) the underlying questions are only displayed if the answer is yes;
 - i. Process the underlying questions in the same way (i.e., repeat from Step 1);
 - ii. *Note:* layer nodes differ from filter nodes in that a *no* answer to a layer question does not render the subtree irrelevant. Instead, it can be taken as a statement that the assessor does not wish to see the subtree questions at present, even if some of them have already been answered.
 - (c) the answer can be *no* for the layer question even if some of the underlying tree questions have been answered, which is where it differs from filter nodes. The layer question is about whether the underlying questions need to be visible or not.
3. If the node is instead a filter node (i.e., has a **filter-q** attribute instead of a **question** attribute), then the node is treated *exactly* as a filter question, including using the same question text. The only role of the **layer** attribute in this case is that it ensures the filter question is displayed on occasions when it might otherwise not be (i.e., when the layer processing exposes it).
 - (a) This means the algorithm must check that the **filter-q** attribute is not present whenever a **layer** attribute is met;
 - (b) if it is, the node should be processed as a filter question node.
4. *Note* the same layer number can be associated with nodes nested within each other. It just means the parent layer node may be opened but the nested layer node may still not be opened, if the answer to it is not given or is no.

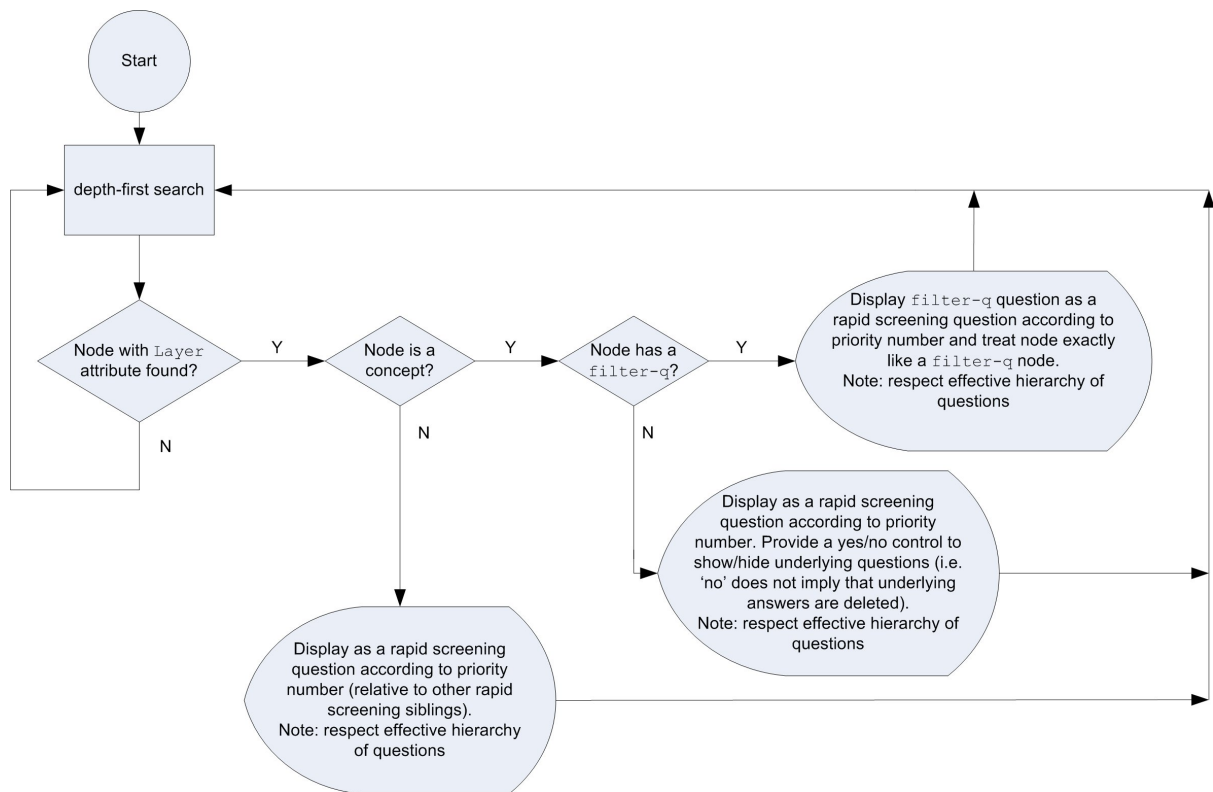


Figure 6.3: Flowchart showing how *rapid screening* questions are identified and displayed from within the full knowledge structure.

6.4.3 Data types associated with questions

Consideration of question text cannot be done in complete isolation from the data types and controls that would be used for data gathering. A parallel activity to question text formulation and validation was thus, work on data types.

The fundamental data type for data gathering: the `scale` attribute value

It was decided to use a scale data type for questions that involved the degree to which a person possesses a particular property or represents a particular risk. Validation activities² revealed preferences by experts for 3 point, 5 point and 11 point numerical scales with labelling. Figure 6.4 depicts a scale validation task that could be carried out online by experts, with the data being collected within a MySQL database.

The relative merits of numbers versus labels become more clear in the light of research indicating that the reliability of assessments increases with the number of graded points available

²Scale validation activities were directed by other members of the research team, with technical elements constituting PhD research.

6.4. QUESTION-RELATED PARAPHERNALIA AND DATA TYPES

Answer Scale Validation

There are a number of possible responses to questions, the most basic being just yes or no. However, research clearly indicates that binary answers are too crude for items that can be present to a greater or lesser degree. A scale of responses is required (e.g. yes, somewhat, no), which could have three, five, or even more different labels. We have opted for a five point labelled scale overlaid on numbers from 0 to 10. This gives the optimal discrimination but is not particularly popular with clinicians. On the other hand, the National Programme for IT requirements for recording national risk data will also use a 5 point scale (you can blame the GRiST Chief Investigator for this, who was on the committee). In short, nothing is ideal but we would like your views on the proposed scale for GRiST. Please view the example, respond accordingly, and submit the response with the button at the bottom of the page.

Questions that involve judgements about the degree to which a person possesses a particular property or represents a particular risk will be recorded as follows:

0 1 2 3 4 5 6 7 8 9 10 ?

very low low medium high very high D/K

Example: to what extent is the person expressing feelings of anger?

0 1 2 3 4 5 6 7 8 9 10 ?

very low low medium high very high D/K

Answer: 7, representing the lower end of the high range.

Please record your views on the answer scale below:

Agree with form of answer.

There should be no numbers (i.e. only categories).

There should be no shaded categories (i.e. only numbers).

Disagree completely with form of answer.

Please provide any comments you may have:

Figure 6.4: Validating the form of answer scales to use for data gathering.

for each item. The greatest increase is up to 7, whereupon it levels off; and after 11 little is to be gained with finer grading (Kline, 2000). So ideally, 11 graded points (i.e., a 0-10 scale) should be the scale of choice. However, this complicates the provision of semantic labels. It becomes increasingly difficult to produce meaningful terms for more than seven points and concomitantly more difficult for people to keep the item labels in mind when comparing their appropriateness for a specific rating. Indeed, Miller's (1956) classic paper reports the average number of items that can be comfortably held in working memory to be 7 ± 2 .

Taking above findings and literature into consideration, a compromise situation was arrived at. The default scale to be utilised within an assessment was an 11-point colour-coded numerical scale, with five semantic labels. This format was chosen because it is well understood: every point along the scale gains 10% and it contains 11 response points, which provides the maximum reliability for all practical purposes. Furthermore, it is not necessary to keep the numbers in

6.4. QUESTION-RELATED PARAPHERNALIA AND DATA TYPES

the assessor's mind: their ordinal and equal-spacing are fundamental concepts that everyone possesses. Finally, the scale is enhanced with a memory-friendly five-item labelling scheme.

All (datum) nodes deemed to require scale answers were given an additional attribute:

```
values="scale"
```

This could then be used by a potential risk assessment tool to render the appropriate control (shown in Figure 6.5).

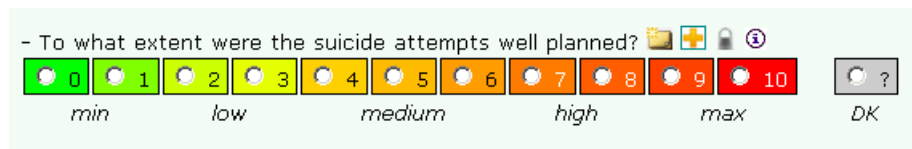


Figure 6.5: An example eleven-point scale based upon a `values="scale"` ST node attribute.

It was recognised that the default scale label strings (*min-max*) depicted in Figure 6.5 might not always agree with the phraseology of the associated question. Therefore, a supplemental attribute was specified, which could direct a potential risk assessment tool to render one of three alternative sets of labels:

```
scale-type="low|much|likely"
```

which were predefined as:

low: min; low; medium; high; max

much: not much; a little; medium; much; very much

likely: very unlikely; unlikely; possible; likely; very likely

Recording dates: the date-year, date-month, date-week, and date-day attribute values

Some questions required date information to be collected from the assessor. It was decided to collect all date information from the user in a consistent, yet flexible manner. An exact date was specified to be collected and stored in the form *DDMMYYYY*. It is often the case that an exact date is not available (and additionally, may not really make much difference to a risk calculation). To cater for inexact dates, further entry formats of *MMYYYY* and *YYYY* were specified for the date control.

6.4. QUESTION-RELATED PARAPHERNALIA AND DATA TYPES

For the purposes of associating with MGs and also for reporting of dates in a sensible manner, a number of different units of measurement for elapsed time were identified:

date-year - The unit of measurement used in value-mg definitions will be *years*, thus any elapsed time calculation should be in *years*. For the purposes of reports based on an assessment, elapsed time should be displayed in *years* where practicable.

date-month - Elapsed time should be calculated in *months*. Reporting should use *months* where practicable.

date-week - Elapsed time should be calculated in *weeks*. Reporting should use *weeks* where practicable.


date-day - Elapsed time should be calculated in *days*. Reporting should use *days* where practicable.

The fact that an ST node required a date control, and the units of measurement were both signified within the node via the following attribute:

```
values="date-year|date-month|date-week|date-day"
```

An example of the resulting date control is presented in Figure 6.6.

Where an inexact date was provided, the Galatean model would convert the date range effectively indicated by the inexact date into an exact date by taking the midpoint. For example, 041988 would imply a date between 1st April 1988 and 30th April 1988. Therefore, for calculation purposes, the date entered would be considered to be 15th April 1988. This conversion would only be for the purposes of classification, and would not result in the entered date being physically altered. Reports based on an assessment would consider the date data type, its granularity, and the magnitude when determining the best format in which to display elapsed time.



* Date of birth 📅 🛡️
(Please enter a date in the format ddmmyyyy, mmyyyy or just yyyy)

Figure 6.6: An example date control based upon the ST date attributes. Its rendition as a single textbox, whose contents are validated, ensures quick data entry.

Recording numbers: the `integer` and `real` attribute value types

Two data types for questions requiring numerical answers e.g., to record the number of past occurrences of an event, were established:

```
values="integer|real"
```

These were rendered by potential tools as text-boxes. An attribute to define upper and lower limits was not specified. This was because the value-mg profile that would be established for answers would effectively define value ranges that are relevant. Values falling outside these would thus not influence the risk calculation in an adverse manner.

Recording nominal categorical data: the `nominal` and `multiple-tick` attributes

A data type to handle XOR categorical answer values (i.e., pre-defined answer values to be selected via radio buttons) was defined as:

```
values="nominal"
```

In this circumstance, a risk assessment tool would look to the node's value-mg definition to ascertain the category labels to use to generate radio buttons.

The need for a data type to represent a node where multiple categorical answers could be selected (i.e., pre-defined answer values to be selected via check boxes) required an attribute additional to `nominal`. This involved defining a `multiple-tick` attribute, which detailed all the category labels and any associated question text; an example of which is presented below:

```
multiple-tick="(&quot;Has the person targeted any particular group of
people rather than complete strangers?&quot; (
(DOMESTIC &quot;Has the person harmed within a domestic setting?&quot; )
(FRIENDS-COLLEAGUES &quot;Has the person harmed friends/colleagues?&quot; )
(HEALTH-WORKERS &quot;Has the person harmed any health workers?&quot; )
(AUTHORITY-FIGS &quot;Has the person harmed any authority figures?&quot; )
))"
```

The `multiple-tick` attribute borrows from the LISP programming language (Seibel, 2005), the idea of using lists to embed a data structure inside an otherwise atomic attribute. Information that may otherwise needed to have been coded as child elements in an alternative XML scheme thus maintains the ability to be coded as node attributes using the present scheme. This yields

the benefit that the ST's underlying file structure maintains a close correspondence with the hierarchical knowledge structure it is coding. That is, each XML element corresponds with an element of the ontology, with no other XML element types being introduced within the tree—making processing and visualisation of the knowledge structure within (and without) tools very easy. A further advantage is that the list format within an XML attribute is a much more compact representation than a scheme using pure XML (*cf.* Crockford, 2006).

Consideration of the `multiple-tick` syntax against the relatively plain `values="nominal"` syntax that is used in the XOR case helps to illuminate a nuance between the two. Whereas the `multiple-tick` format is comprehensive enough to specify question text against each item, the XOR format is not. This is because it was apparent that radio button controls did not generally require individualised questions for each option: node labels collected from the node's `value-mg` specification would be enough given that only one nominal alternative could be chosen. `multiple-tick` nodes on the other hand, can have more than one answer, and could in theory be represented as a concept with children nodes corresponding to each item: each item could be answered individually, thus each item should be able to have its own question text.

6.5 Representing Membership Grade Profiles

Recall that as a prerequisite to classification, the Galatean model of risk requires that each datum has an associated `value-mg` profile. The `value-mg` profile maps potential answer values to the level of activation or membership of the datum node. This section describes methods that were used to elicit `value-mg` profiles, and their associated representation format within the ST.

6.5.1 Collection of preliminary `value-mg` data

The generic tree annotation program described in Section 5.3.1 was adapted³ to serve as a `value-mg` profile elicitation tool (Figure 6.7). Depending on the contents of the `values` attribute (which contained the data type of the node), a suitable representation was constructed by the tool to collect `value-mg` data. Figure 6.7 depicts a representation for collection of `value-mg` data for the `scale` data type. For this data type, participants were presented with a grid on which to place points in order to sketch out a `value-mg` distribution (`scale` values being on the x -axis).

No restriction was placed on the number of data points within a `value-mg` profile. Therefore,

³The `value-mg` elicitation tool specifications were developed as part of PhD research, but the tool was realised by other members of the research team.

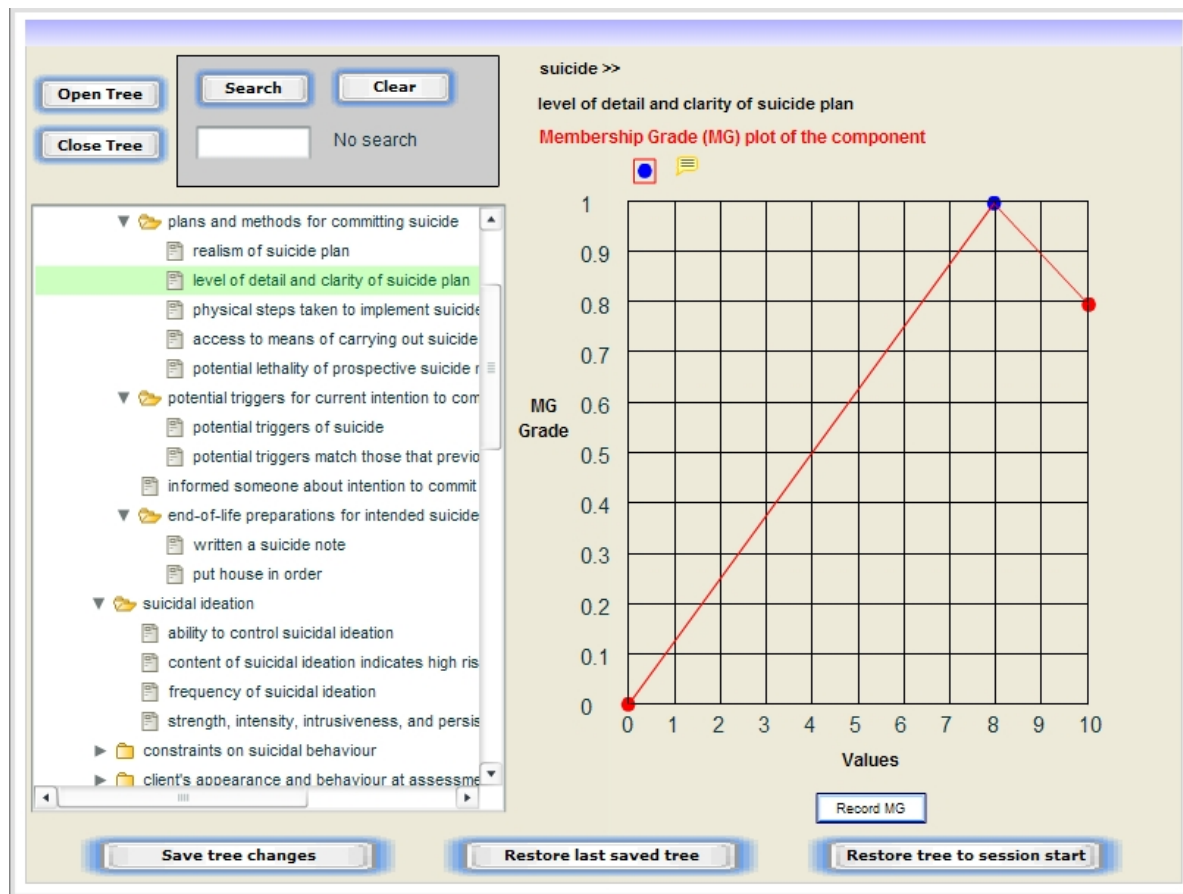


Figure 6.7: The value-mg profile elicitation tool being used to elicit a profile for a `scale` node. A graph is constructed by the user by dragging blue dots from the ‘pad’ at the top-left of the grid and placing them at the desired location. Red endpoints can only be moved along the y-axis.

profiles of arbitrary precision and complexity could be generated. The data points corresponding to the value-mg profile were represented within the ST node as an association list within a `value-mg` attribute as follows:

```
value-mg="((0 0) (8 1) (10 0.8))"
```

Value-mg profiles for nominal data types were also represented in a similar manner e.g.:

```
value-mg="((DOMESTIC 1) (FRIENDS-COLLEAGUES 1) (HEALTH-WORKERS 1)
(AUTHORITY-FIGS 1))"
```

Pilot value-mg elicitation was performed with experts using the focus group approach⁴ in order to arrive at a consensual value-mg profile for datum nodes. Members of the research team are currently formulating methodologies to analyse *individual* expert value-mg opinions and

⁴Focus groups were run by other members of the research team.

produce a consensual value-mg in a statistically robust way (Hegazy & Buckingham, 2010). This will allow the elicitation method to scale to the individual opinions of any number of experts.

6.6 Flexible Assessments Based on User Expertise Level

Section 6.3 detailed the reorganisation of the ST to remove redundancy. Sections 6.4 and 6.5 have established the ST machinery that will ultimately be used to conduct patient assessments. A key consideration not yet addressed, and which was one of the initial goals of the project, is one of molding an assessment to the needs of the clinician/user. Section 3.4.1 recognised the importance of this in the successful acceptance and integration of a new tool into existing workflows, e.g., Kawamoto et al. (2005); Sim et al. (2001). The present section expounds upon the machinery incorporated into the ST to consider the expertise of the assessor in order to produce a tailored risk assessment tool. This represents the first of a two-pronged approach that was implemented to meet individualisation needs of the practitioner (the second approach to be introduced in Chapter 10).

6.6.1 Practitioner expertise quantised as levels

The Structure Tree, by virtue of the way in which it has been engineered, is a comprehensive data set, which defines the concepts relevant to mental-health assessment. These concepts are broken down into their elemental components—lower level cues that even staff with no experience in mental-health can measure and provide information on. This makes any potential tool accessible to staff such as front-line services, who may not necessarily have the skills for making higher-level judgments. The reality however, is that many mental-health practitioners may want a shorter tool to match their level of experience and expertise.

To cater for varying levels of experience, three levels of mental-health experience were identified, with the aim of using the ST to drive the generation of tools applicable to each level, all from the same underlying validated knowledge structure. Levels of expertise identified were as follows:

Level 0 - Assessors with no mental-health background, such as police, housing officers, fire fighters etc.

6.6. FLEXIBLE ASSESSMENTS BASED ON USER EXPERTISE LEVEL

Level 1 - Assessors with a health background, but not mental-health, such as paramedics or accident and emergency nurses.

Level 2 - Assessors with mental-health training.

Level zero is considered to be the default, where all leaf (datum) nodes would be used to produce the assessment tool. These already have questions associated with them, formulated as part of the knowledge engineering activities described in earlier sections.

The graduated refinement of a concept that is a consequence of the ST's hierarchical structure means nodes higher up the tree encapsulate the meaning of descendent nodes. Therefore cutting off of a concept at a higher point in the tree is tantamount to representing the removed nodes in a more abstract form. Level one and Level two 'cut off' points can thus be defined via the incorporation of a `level` attribute:

```
level="1|2"
```

The presence of a `level` attribute against a node means that this is 'soft' cut-off point for tools that match the `level`. In such cases the cut-off point is effectively treated as a datum node (and nodes below it are not considered by the tool). The tool is thereby shortened as expert experience level increases.

The soft cut-off point introduced by virtue of the *levels* mechanism requires that a `question` and `values` be specified specifically for when the point is transformed into a datum in tools matching the `level`. Figure 6.8 depicts this for the *suicidal ideation* concept node, which has assigned a `level="1"` attribute, an accompanying question to be asked for the associated Level 1 risk assessment tool, and a `values="scale"` attribute to dictate the form of the accompanying answer. It means that a practitioner with experience matching the Level 1 tool requirements can make a direct judgement about the level of risk associated with *suicidal ideation*. Assessors using the full risk-screening tool would have to provide answers to the four lower-level questions and rely on the Galatean model to generate the level of risk associated with *suicidal ideation*.

```

<node label="suicidal ideation"
  filter-q="Is the person having suicidal thoughts or fantasies?"
  question="To what extent the person's suicidal thoughts/fantasies
  match those that would give you the most concern about suicide risk?"
  values="scale" level="1" value-mg="((0 0)(10 1))">
  <node label="How much suicidal ideation is verbalised"
    question="To what extent is the person talking about suicidal
    thoughts or fantasies?"
    values="scale" value-mg="((0 0)(10 1))"/>
  <node label="ability to control suicidal ideation"
    question="To what extent is the person able to control the suicidal
    thoughts or fantasies?"
    values="scale" value-mg="((0 1)(10 0))"/>
  <node label="content of suicidal ideation indicates high risk"
    question="To what extent does the content of the suicidal thoughts
    or fantasies raise serious concerns about suicide risk?"
    help="get away from it all ** harming others **
    harming themselves"
    values="scale" value-mg="((0 0)(10 1))"/>
  <node label="frequency of suicidal ideation"
    question="How often do the suicidal thoughts or fantasies occur?"
    values="nominal" value-mg="((DAILY 1)(WEEKLY 0.5)(MONTHLY 0.2)
    (LESS-THAN-MONTHLY 0))"/>
</node>

```

Figure 6.8: Part of the XML for holding information on *suicidal ideation* in the refined Structure Tree. It shows attributes holding the node names, the question to ask, the values the answer can take, the associated value-mg profile, the level of experience required to ask the question, and help information about the node semantics.

Algorithm for generating tools at each level

Algorithmically, a tool corresponding to a given level would be generated by traversing the tree as follows:

```

Conduct a depth first search of the tree, examining any level attributes.
IF a matching level OR a lower level is detected THEN
  Output the associated question and do not go down any further
ELSE
  Continue going down until the leaf node is reached
  (and thereby output the same question as a Level 0 tool would)
END IF

```

6.7 Conclusions

This chapter set out to describe the process by which the Structure Tree was refined and enhanced in order to make it amenable to driving a risk assessment tool. The transition from an *initial* ontological structure to one suited to the task of risk data gathering was marked by three phases:

1. Reorganisation to eliminate structural redundancy.
2. Specification of attributes to drive risk assessment tool creation.
3. Specification of a method to customise tools according to practitioner experience level.

Initially the ST required reorganising due to the fact that concepts and datums were being repeated across the tree even though they were essentially of the same generic structure. A categorisation of generic nodes into *g* and *gd* types was introduced due to the fact that internal weightings (Relative Influence values) were not always deemed to be constant across node instantiations. The distinction thus allowed the ST to remain as compact as possible, yet provided a way to flag up instantiations which would require their own weightings.

The ambition of this project was to go fluidly from knowledge elicitation to a validated knowledge representation to a risk assessment tool/decision support system directly specified by that knowledge structure. Essentially, the expert knowledge should directly drive tool creation. XML and XSLT-based technologies were specifically chosen for the flexibility and malleability of representation that they brought to the table. As preparation for incorporation of the knowledge into a risk assessment tool or CDSS, a number of constructs were introduced to aid tool construction/configuration. These were specified via new attributes. The attributes imbue the ST XML structure with the capability to stipulate data types, question groups, rapid screening questions, and Membership Grade activation profiles. The attributes embellish the raw scaffold of the knowledge with information about how it is to be used to generate the risk assessment tool.

Finally, the ability to customise according to practitioner experience was fashioned within the knowledge structure. The representation of the knowledge in terms intuitive to humans (i.e., concept hierarchies) played a pivotal role in facilitating this. The hierarchical ontological structure of the ST, in which knowledge is gradually decomposed into elemental components, allowed for this to be achieved via specifying soft cut-off points. This meant that experienced

practitioners would have a smaller ST driving construction of their tool, implying fewer but more abstract questions. The utility of the risk assessment would not be compromised even though the practitioner would be answering fewer questions because the underlying validated knowledge would be the same but would rely greater on the practitioner's expertise. This type of tailoring mechanism, where the quality of the assessment outcomes is maintained whilst shortening the tool, is an innovation in CDSSs in the area of mental-health.

Customisation according to clinicians' needs is an important issue, and this chapter has detailed the first of two approaches that were implemented to molding the tool towards the practitioner. The second approach, tailoring for different settings and patient demographics, is addressed in Chapter 10.

Before the ST can be used to drive a risk assessment and a decision support system, it requires that RIs be instantiated. Closer consideration of the initial strategy of removing structural redundancy within the ST alludes to a problem with regards to the instantiation of RIs. If the Structure Tree eliminates structural redundancy in locations where the redundancy does not extend to RI values, how are RI values to be elicited and recorded against the eliminated nodes? This topic is explored in the next chapter.

7

Representing Relative Influence

7.1 Introduction

The enhanced Structure Tree resulting from the reorganisation activities outlined in Chapter 6 is a normalised data structure, with all structural redundancy removed. It has a defined primary purpose:

- To serve as the tree that represents all the relevant concepts within the domain and how they break down and relate to each other - i.e., the domain model (Gruber, 1995).

The full decision support system based on the domain of application (i.e., mental-health) requires (Power, 2002):

1. The domain model itself.
2. Information on how to present the domain model knowledge as a data gathering tool and on how to collect and represent subsequent assessment data.

3. Uncertainty information for use by the Galatean model, in order that it is able to classify assessment data relating to the domain model.

The ST serves the first of these requirements directly, by virtue of its representing the domain model. The second requirement is also served, in the main, by the ST as a result of the additional enrichment process undertaken in Chapter 6. The third aspect however—that of representing uncertainty information for use by the classifier—is only part satisfied by the ST: Relative Influence (RI) values are not collected.

The present chapter details why the ST is not a suitable tree in which to collect RI data. It then proceeds to outline a method by which a derivative tree is generated from the ST, specifically for RI collection activities. Finally it uses the derivative tree to illustrate the concept of the Galatean Tree Hierarchy: a logical cascade of knowledge structures to help manage the complexity of the process of going from knowledge engineering to decision support system. Successive trees in the Galatean Tree Hierarchy will take the knowledge structure closer to the point where it can directly drive assessment and decision support.

7.2 The ST is Not Suitable for Recording RI Information

Recall from Section 2.3, that uncertainty information within a hierarchical domain is conceived by the Galatean model as being comprised of Membership Grades (MGs) and Relative Influence (RI) values. value-mg profiles for datum nodes are assignable directly within the nodes in the ST. This is possible even within *generic* datums because as explained in Section 6.3.2, the value-mg profile of a node should not logically change across instantiations of that datum. The value-mg maps a given value of a patient attribute (i.e., datum node), to the level of activation of the node when the node is considered *in isolation*. Therefore, by definition, value-mg profiles should not vary across node instantiations, and can thus be defined within the generic-datum definition.

Relative Influence values however, for reasons outlined in Section 6.3.1, can vary across generic *gd* concept and datum instantiations. In fact it is precisely due to this phenomenon that the notion of a *gd* concept/datum was introduced when removing redundancy from the ST. The `generic-type="gd"` attribute flags up nodes where there is an element of *distinctiveness* within an otherwise generic template for instantiation. The distinctiveness in RI values means that they cannot readily be accommodated within *gd* concept definitions.

7.2. THE ST IS NOT SUITABLE FOR RECORDING RI INFORMATION

One scheme that could be used to accommodate RIs is an adaptation of the method that was used in `multiple-tick` nodes. Here, a LISP-like list structure is used to record multiple questions within one attribute. Such a method could be used, for example, to store lists of complete paths to all to-be-instantiated locations along with the value-mgs those locations are to receive. However, this is neither a practical nor elegant solution to the problem, as it involves:

- A convoluted data structure within *each descendant* of the *numerous gd* concept definitions of the ST, especially when considering concept nesting rules such as rule 7 of Section 6.3.4.
- The requirement that each *gd* concept definition know *a priori* all the locations at which it is to be instantiated. This is as opposed to the more sensible notion of an instantiation location knowing the location of the *gd* concept definition; (*cf.* Mealling & Denenberg, 2002).
- Considerable bloating of the ST with repetitive data (paths) where the objective is to remove redundancy and retain clarity.

Direct incorporation of RIs within the ST was thus rejected.

Consideration of the nature of an RI value also presented a further reason as to why, even if it were possible to easily incorporate RIs within the ST, it would not be sensible to do so. Relative Influences are not part of the physical structure of the domain model, yet they are directly influenced by it by virtue of their consideration of sibling nodes. Consequently, physical changes to the ST as a result of future refinement will possibly render RIs incorrect. It therefore makes sense to not place RIs within the ST, leaving the ST to be modified and reworked without concerns of leaving it in an incorrect state. Of course, this does not mean the problem of maintaining RIs is remedied: it means instead that the problem is decoupled from any ST structural modification stage. Decoupling a problem into separate stages helps to manage its complexity (Pfleeger & Atlee, 2009). This approach to problem solving mirrors work in earlier phases of KE activities, where modular XSLT stylesheets were developed and chained together to achieve tree transformations, as opposed to using a monolithic approach. RI maintenance is considered in greater detail in Chapter 8. But before such considerations, the approach that was taken to represent RIs requires elucidation.

7.3 Generating the Relative Influence Tree

Originally, it was envisaged that all KE work would be carried out on the ST, with the ST then driving data gathering tools and the decision support system directly. However it became apparent that in order for the domain model to serve a decision support system, it required redundancies to be distilled. As described in the previous section, this has meant that issues such as RI elicitation are not straightforward using the ST.

It was decided that a new tree, a derivative of the ST, should be used for RI instantiation. If the ST were to be partially expanded essentially by way of *gd* concept instantiation, then each node that requires an individualised RI would be exposed. The new tree would then serve as the tree for recording RI information during RI elicitation rounds. It was hence named the Relative Influence Tree (RIT).

It is important to stress that the ST is not rendered defunct by the introduction of the RIT. Maintenance of the domain structure information is still solely within the remit of the ST, with automated RIT generation taking the domain knowledge to the next KE phase—the RI elicitation round. This is in effect, a layered approach to the development of the full ontology. The Heraclitus II ontology pyramid (Mikroyannidis & Theodoulidis, 2010) also adopts a similar approach, whereby layered ontologies are supplemented with additional information. However, whereas in the Heraclitus II approach, information introduced in one layer also propagates to an extent to the previous layer (due to each layer being of independent use to a different group of experts), the RIT approach only requires forward propagation of data from the ST. This is because both trees essentially model the same ontology.

7.3.1 Priming the ST for transformation

Given that more than one tree is now involved in the journey towards a decision support system, it was deemed prudent to conduct some preparatory remedial work on the existing ST. Specifically, existing node identifiers were considered inadequate to serve as identifiers and as keys across trees. Prior to the conception of the RIT, nodes were identified using their `label` attribute. This was however considered too problematic due to the following reasons:

- Verbosity - Labels were often long and primarily for the benefit of humans, e.g., for use by practitioners during knowledge elicitation and by the research team when working on the knowledge structure.

- Non-friendly to computers - Labels contained free-text, were liable to be constantly tweaked (and thus change), contained awkward characters etc. All these issues would need to be rectified and normalised each time a program were to use labels. This would needlessly add to development effort.

In essence, labels were there to make life easier for humans. Machines on the other hand can deal with simpler, more stable identifiers, that do not need to be visible to humans. The idea is similar to auto-generated primary key fields in databases, which are there primarily for the internal use of the DBMS application and are hidden from the end-user (T. Connolly & Begg, 2010). It was therefore decided to create an internal `code` attribute for each node, which was a stable, short form identifier of the node in question. For example, a node with `label="past and current suicide attempts"` would receive a new additional attribute: `code="suic-past-att"`. Codes were created based on abbreviated node labels specifically as a trade-off between space-efficiency/machine processability and human readability, the importance of which, is advocated by Hunt (2010). Pending processes would use the node's `code` where appropriate, yielding the ancillary benefit of allowing the `label` to be freely modified as part of future activities, whilst still making e.g., paths consisting merely of node-codes intelligible to the researcher.

7.3.2 RIT generation process

RIT-generation was automated using XSLT via a series of stylesheets to gradually effect the transformation from ST to RIT. Briefly, the transformation involved:

1. Converting “multiple-tick” nodes into a concept gated by a `filter-q`, and having each alternative answer as a child node. Effectively this operation removes the need for the `multiple-tick` construct, as the node has been expanded into a concept and child datums. Additionally, it allows RIs to be specified for each sub-component. (The `multiple-tick` conversion process is described in greater detail in Appendix C).
2. Instantiating all *gd* nodes throughout the tree.

3. Instantiating the children of the *direct risk children* pseudo concept inside each root-risk (note: *g* nodes are not instantiated, with paths still pointing to their location of definition).
4. Recalculating any paths to *g* node definitions as a result of *gd* node and *direct risk children* instantiation.
5. Fabricating a supplementary code for *gd* node descendants that have a `level` attribute—to be used in the future when the node (which may ordinarily be a concept) becomes a datum node for a data gathering tool matching that `level`. Effectively the node will be identified by a different `code` when it is to become a datum. This prevents problems where a node could theoretically have two data types (the data type depending on the level of the tool being used).^a
6. Removal of *gd* node definitions that have been instantiated, and of the *direct risk children* pseudo-concept.

^aThis step does not strictly need to be carried out in order to facilitate RI elicitation *per se*. However, it is implemented within the RIT generation algorithm as preparatory work for driving data gathering tools at multiple levels.

7.4 RI Elicitation

Once generated, the RIT was used to perform RI elicitation with mental-health experts. RI elicitation activities and any *additional* infrastructure required to support them was conducted by other members of the research team. This section is therefore only included for completeness, and does not form part of PhD work.

RI elicitation was performed using a tailored version of the Tree Annotation program developed for previous elicitation rounds. As in previous rounds, the infrastructure supporting RI elicitation was web-based, with annotated RITs being automatically uploaded and saved in the database for later analysis, and the activities themselves being coordinated via the website. The set-up of the task thus adhered to the requirement of allowing participation without geographical impediment whilst giving each participant the freedom to work at their own pace.

The specifics of the task involved experts being asked to weight each child node of a concept with respect to its siblings. This was achieved using a set of sliders that could be adjusted by experts to indicate the importance of each sibling. Importance was automatically normalised

across the siblings and presented as RIs and also as a stacked bar (chart) to aid visualisation. Figure 7.1 shows the ‘RI Elicitation’ tool being used to elicit RIs for the *past and current suicide attempts* concept.

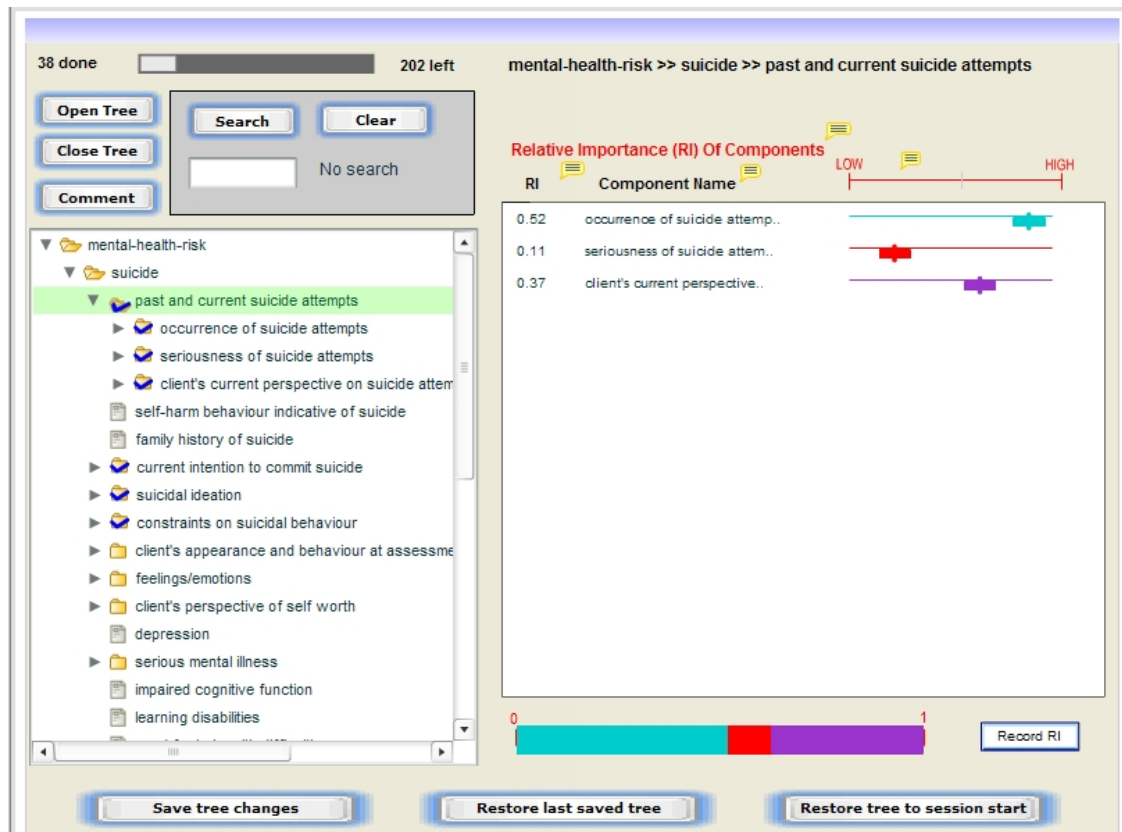


Figure 7.1: The RI elicitation tool. Slider positions are normalised to produce RI values and a stacked bar display at the bottom of the tool.

XSLT was used to collate RI values provided by each expert within a fresh RIT. A Bayesian framework is currently being developed by other members of the research team to infer consensual RI values for each node from the collated data. Additionally, an innovative method to reduce the number of nodes required to be evaluated by experts during RI elicitation has been developed (Hegazy & Buckingham, 2009b, 2009a). This means that the full RIT can be instantiated with RIs by only considering a subset of the nodes.

The following syntax was established to record finalised RIs within the RIT:

```
ri="value"
```

Instantiation of the finalised RI values marks the end of RIT enrichment activities.

7.5 The Galatean Tree Hierarchy

The journey from the ST to the RIT represents the expansion and enrichment of the underlying domain knowledge to a point where it can serve as a structure closer to enabling decision support. The RIT can be conceptualised as a new logical layer of enrichment that sits on top of the ST. It enables de-coupling of structural redundancy from RI elicitation, meaning trees are focussed on their own specialised tasks in isolation to each other. The ST and RIT together thus establish the *Galatean Tree Hierarchy*.

The idea of a hierarchy of trees is similar to that of a protocol stack, where each layer only needs to familiarise itself with the layers directly above and below (Blank, 2004). Layering of trees in this fashion means that the KE and software engineering aspects involved in the system as a whole can be quantised. Ultimately it represents a more clear, systematic and logical transition from expert data to expert system. Furthermore, similar to a conventional protocol stack, this type of organisation reduces duplication of logic over a monolithic tree design because (possibly multiple) end user tools only need to contend with the top layer. The top layer would be closer to meeting the needs of the tool than say the bottom layer, which in the context of the ST, would require (each) tool to implement tree expansion operations.

The Galatean Tree Hierarchy is a concept that will be revisited in future chapters as the decision support system that was developed is elucidated.

7.6 Conclusions

This chapter reiterated the rationale for a Structure Tree and identified the contribution the ST makes towards reaching the overall goal of a decision support system. It is recognised that the ST cannot directly drive a full DSS due to its not being able to support the recording of RI data. The chapter therefore introduced the notion of the Galatean Tree Hierarchy (GTH), consisting of derived trees. Each tree serves a specific purpose(s), whilst the aim of the tree stack as a whole is to smoothly bridge the gap between the domain model and the decision support system.

With regards to the issue of RI instantiation, the Relative Influence Tree (RIT) was introduced as a component of the GTH. The RIT would be automatically generated from the ST and serve as the platform on which to elicit the RIs, ready for use within a decision support system; a topic of Chapter 9.

The overall landscape of the journey from experts' knowledge to expert system as represented

in Figure 5.6 is now extended with the beginnings of the Galatean Tree Hierarchy as depicted in Figure 7.2. The GTH will be expanded in future chapters as the evolution of the system is explored.

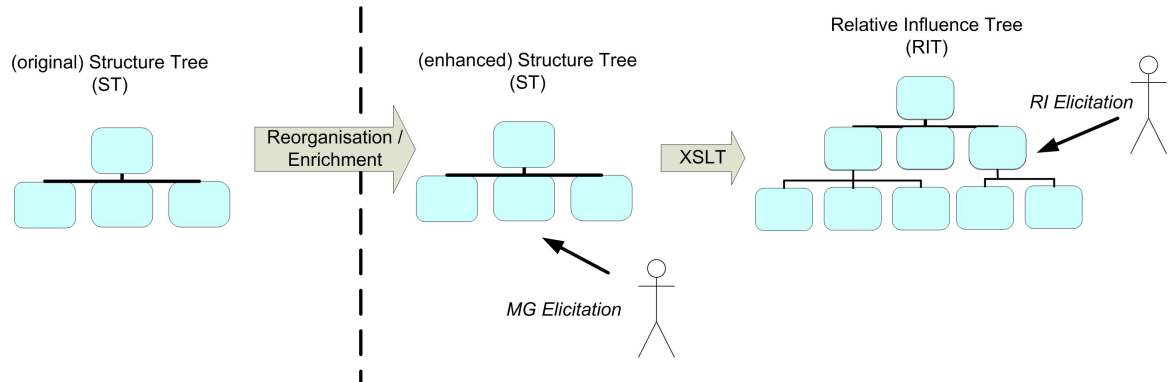


Figure 7.2: Going from the original ST to the beginning of the Galatean Tree Hierarchy. Reorganisation and enrichment activities on the ST mark the beginning of its inclusion within the GTH (the start of which is denoted by the dotted line). The ST is expanded and enriched to become the RIT (the second tree within the GTH).

8

Synchronising RITs and STs

8.1 Preamble

The RIT in its uninstantiated form, is a derivative of the ST. However, when the RIT is instantiated with RI values, it is no longer a pure derivative of the ST (since new information has been supplied to it). This poses a problem when modifications are required to be made to the knowledge structure contained in the ST. There needs to be a way to propagate RI values from the old RIT to the RIT that is generated from the updated ST, thus keeping the trees synchronised. This chapter specifies a method to automatically synchronise STs and RITs such that STs can be amended without unnecessary manual remedial work on the RIT.

8.2 Galatean Tree Roles

Each tree in the Galatean Tree Hierarchy serves a specific logical function.

- The Structure Tree (ST) describes the domain's knowledge structure. It eliminates structural redundancy and describes each concept fully only in one location.

- The Relative Influence Tree (RIT) (which is a derivative of the ST) essentially instantiates generic concepts whose components' weightings will depend on the location of instantiation (*gd* concepts). Because of this instantiation, the tree is amenable to having the weightings for each node specified. These weightings (or Relative Influences) can be entered into the RIT XML file directly or by using a GUI to the RIT.

8.3 The Knowledge Engineering Process is Iterative

Managing the complexity of the Knowledge Engineering process would be easier if different stages of the elicitation could be performed in an isolated manner, with each stage not needing revisiting upon its completion. In the context of the GTH, this would entail all the knowledge structure work being carried out within the ST. Upon perfection of the ST's structure, the RIT would be generated. The RI elicitation/instantiation work would only then be attempted.

The problem with the application of the Waterfall Model approach with respect to knowledge elicitation is that the process is inherently iterative. Each round of elicitation necessarily results in a re-evaluation of the existing knowledge structure. Although this is controlled to an extent by focussing the domain experts on the task at hand, there will invariably be suggestions for modifications to areas covered in earlier rounds as a result of:

- An aspect not having had adequate attention paid to it in an earlier round.
- Realisation and insight into the correct formulation of the aspect, only gained through evaluating it in the context of the current round.

To disregard these “auxiliary” outputs of e.g., the RI elicitation round means losing an opportunity to improve on the domain knowledge that will be used by the decision support system. That is, rounds beyond the ST formulation phase will invariably necessitate modifications to the ST.

8.3.1 GRiST and population diversity

It can be envisaged that GRiST may be required for the assessment needs of different segments of society, or “populations”. As the needs of each of these populations illuminate the knowledge structure from their own perspective, a better overall insight into the domain might be gained. Modifications would need to be made to GRiST to incorporate this knowledge. Logically, these

changes in the underlying knowledge structure should be made in the ST, since its remit is to efficiently describe the domain. Thus clearly, the initial formulation of the ST will need to be augmented as new populations are considered.

8.3.2 Organic evolution through clinical usage

Users will be in a position to fully evaluate the effectiveness of the decision support system when it is utilised in a *clinical* setting. An important outcome of any evaluation of the tool is directions on how to make the DSS more efficacious. One avenue of improving efficacy will be improving and fine-tuning the underlying knowledge structure upon which the DSS is based. Thus, praxis will most certainly inform changes required to the ST when the DSS is “in the field”.

8.4 The Problem Definition

Arguably, it is essential to make the knowledge engineering process both flexible and amenable to receiving information from any channel at any time without the need for significant rework to the system architecture. The previous section has established that there are necessary reasons for the application of retrospective modifications to the ST from various phases post ST generation. This raises the important issue of tree synchronisation.

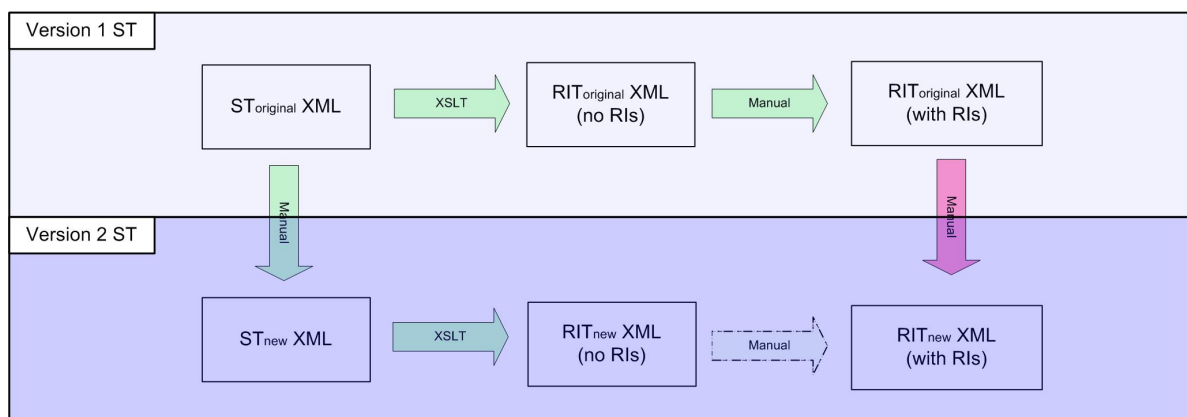


Figure 8.1: Automated and manual actions that are involved in tree generation/updating.

Figure 8.1 illustrates the relationship, within the Galatean Tree Hierarchy, between the ST and the RIT. Furthermore, it shows where current automated and manual transformational processes are applied to these trees. The RIT is a derivative of the ST (and a necessary precursor

to a viable DSS tool). Once the RIT has been automatically generated from the ST using XSLT, it is (manually) instantiated with Relative Influence (RI) values solicited from domain experts. If there were to be no additional changes made to the ST, then the diagram would be considered to be complete at the *Version 1 ST* stage.

As argued earlier, retrospective changes will invariably be made to the structure of the ST. These are depicted in the *Version 2 ST* stage of the diagram. Manual changes by way of direct user edits on the old ST itself, or through the use of GUI-based tools will be made, as depicted by the green arrow crossing between the two stages. Once the updated ST, ST_{new} , has been created, an automated process can be used to generate an RIT as before. There is then a less substantial manual process to solicit RI values for new nodes that were introduced in *Version 2 ST*. This is indicated by the dotted arrow. However, there is also the issue of synchronising the data from the original RIT, RIT_{original} , with the as yet mainly uninstantiated new RIT, RIT_{new} . The original RIT data needs to somehow be incorporated into the new RIT. Since there is no guarantee that nodes in the original RIT will be located in the same position in the new RIT, a straightforward copy operation cannot be performed between the original and the new RIT. In fact, resultant RIT nodes may have been renamed, moved, deleted, added to, as part of the ST upgrade, making any sort of manual synchronisation (see pink arrow) of the new RIT a potentially laborious and error prone process. Put another way, the following two conflicting goals need to be reconciled:

1. having a DSS tool-chain grounded on the concept of derived trees, each being supplemented with its own dedicated information.
2. being able to upgrade a progenitor tree, and having those updates propagated to derivative trees.

This is akin to a Linux package management system having to upgrade a user's program, yet preserve the user's previous settings, taking into account the fact that setting definitions may have been deleted, been renamed, been added to (thus having an effect on existing settings) or even moved locations!

If the manual RIT synchronisation process (depicted by the pink arrow) is replaced by an automated method, source trees can be edited without having to be concerned about upgrading the rest of the Galatean Tree Hierarchy. A cursory glance at the problem might indicate that it could be solved by utilising some of the XML file change tracking algorithms and software that

are in the public domain. However, it quickly becomes apparent that existing methods are not suitable for the task at hand. The RIT is a derived tree, with many subconcepts potentially having the same internal structure, but differing in their internal RI values. This is one area that would confuse a generic XML file change tracking algorithm. If multiple instantiations of a *gd* concept have moved locations in the new RIT, there would be no reliable way of determining which one was which.

Furthermore, small changes in the ST could lead to magnified (i.e., multiple) changes in the (derived) RIT due to the tree expansion process that goes on as part of RIT generation. Again, because a comparison operation is performed on derived trees (and not on STs), even small ST changes can result in seemingly unconnected changes in disparate locations of the RIT when viewed by a “naive” XML comparator algorithm.

Additionally, the potential for a resultant RIT node to be renamed and moved to a different location, yet still logically be considered to be the same node places another seemingly impossible task on a naive XML comparator program.

It becomes clear that the output of any naive comparator algorithm would need manual verification of its results and manual accepting/rejecting of changes in order to ensure the right RIs are copied over to the right nodes. In order to ensure that RI values are reliably and robustly migrated over with no human verification, an intelligent automated process is required. Its application should result in the new ST being synchronised to the newly generated RIT with no re-entry of previously populated RI values required.

As a high-level overview, any intelligent automated process would thus need to overcome the following problems:

- Tracking the location of a specific RIT node across (old and new) trees irrespective of surface changes to it (i.e., changes to node attributes other than code).
- Determine whether its RI value is to be; copied over, re-calculated, or solicited from a domain expert; depending on the values of its resultant siblings.

8.5 Architecture for Tracking a Specific RIT Node

This section considers the architecture for tracking a specific node from its location in an “old” tree across to its location in a “new” tree. It lays out the machinery that will be required for node tracking, and the pathways that tracking will take across trees.

8.5.1 Unique node identifiers

In order to be able to track a node from its location in one XML tree to its location in a modified version of that tree, there needs to be a method of uniquely identifying it. In the context of the trees in the Galatean Tree Hierarchy, node identifiers can be regarded as being stored inside `node-code` attributes. However, using these `node-code` attributes for the purposes of tracking would not be viable because they do not serve to uniquely identify the node. Their function is to uniquely identify the node *semantically*. Thus for example, a node with the same `node-code` can appear in multiple locations due to its being a *g* or *gd* concept node or its being a direct risk child. Since these locations could theoretically have different RIs, it is not enough that the `node-code` is semantically unique. For the tracking task, the node's position needs to also be taken into account.

The location of a given node in an XML file can unambiguously be stated as an XPath statement giving the path to the node from the root node of the tree. With Galatean Trees, it would be most logical to use each intervening node's `node-code` when qualifying the path to the node in question. This is because one of the design goals of any proposed tracking mechanism is that it should accommodate changes being made to the node's name, i.e., its `label`. Using the `node-code` however, introduces the constraint that a given node's `node-code` must remain invariant. But this is acceptable given that the `node-code` semantically identifies the node, and that any change in semantics effectively means this is now a different node.¹ Furthermore, from its inception, the `node-code`'s primary role was that of an identifier to be used internally, in the mechanics of the GRiST, and not as information to be viewed by users of the system. Thus, the `node-code` is a stable attribute.

An XPath statement to the node needing to be tracked could be embedded as a "fingerprint" attribute within the node in question. A major limitation to this approach however, is that it would lead to considerable bloating of the tree. Furthermore, human readability of the XML structure of the tree would be significantly impaired, thereby diminishing the ability to do rapid prototyping work by way of direct tree editing.

The XPath fingerprinting scheme could be made more efficient by converting the path into

¹If it is envisaged that a node's semantics will change, it might be argued that a `node-code` change will be necessitated. Since the semantics of the node will change, it implies the pre-existing semantics of the node to be incorrect. The pre-existing RI values of the node and its descendants would therefore need to be reappraised in the new tree. Thus, a `node-code` change is effectively tantamount to the deletion of a concept and the creation of a new concept since pre-existing RI information can be discarded. So there isn't any increase in manual labour by enforcing an invariancy constraint on the `node-code`, as violation implies deletion and subsequent creation of a concept (and RI values would have needed to be reappraised anyway).

a cryptographic hash. This hash could be used as a more compact proxy for the path. Since the probability of hash collisions is too small to be of any concern, each node’s path will be guaranteed to generate a unique hash, and thus a unique fingerprint.

The remainder of this chapter adopts the MD5 hashing algorithm for fingerprinting purposes, due to its relatively small size (32 ASCII characters). Unless stated otherwise, it also adopts the convention of generating the hash using the following string representation of the path to the node in question:

```
root-node-code intervening-node-code-A intervening-node-code-B ...node-code
```

That is, all nodes from the root node to the node for which the fingerprint is to be generated (inclusive) have their node codes separated by a space character. This string is then used as the input to the MD5 function. The hash is stored in an attribute called `fingerprint` against the node for which it has been generated.

8.5.2 Node fingerprint audit trail

Recall that Figure 8.1 illustrated the current automated and manual paths taken in the transformation and manipulation of the ST and the RIT. In order that the (pink) arrow representing manual synchronisation of RIs be replaced by an automated process, node fingerprints in $RIT_{original}$ need to persist through to RIT_{new} . The required unbroken audit trail from these two endpoints can thus be expanded and represented by expression 8.1.

$$\begin{aligned}
 & RITnode_{original} \rightarrow RITnode_{new} \\
 \implies & RITnode_{original} \rightarrow STnode_{original} \rightarrow STnode_{new} \rightarrow RITnode_{new} \qquad (8.1)
 \end{aligned}$$

Initially the approach of directly generating $RIT_{original}$ node fingerprints was considered. Using this scheme, $RIT_{original}$ node fingerprints would be propagated from $RIT_{original}$ to $ST_{original}$. Since these would remain in ST_{new} , they could then be propagated down to RIT_{new} . However, this approach is suboptimal, as it requires a method of linking an $RIT_{original}$ fingerprint to a particular node of its $ST_{original}$ progenitor. So each node of $ST_{original}$ would therefore need an “ $ST_{original}$ fingerprint” of its own in order to provide the linkage. Furthermore, there would be the issue that there can be a one-to-many relationship between an ST node and an RIT node. This would make it impossible to disambiguate (many) RIT_{new} nodes that derive from an (one) ST_{new} node, without introducing further additional constructs in the ST—thus blurring the

segregation of ST and RIT.

A more efficient propagation method, which was ultimately adopted, obviated the need for $RIT_{original}$ fingerprints, and instead involved creation of $ST_{original}$ fingerprints. Since the ST data is the only data that is directly carried from the *Version 1 ST* phase through to *Version 2 ST*, it was decided that node fingerprints should be generated in this tree. This would then decompose the problem of generating an unbroken fingerprint audit into two sub-expressions, as denoted in expression 8.2.

$$\begin{cases} STnode_{original} \rightarrow STnode_{new} \\ STnode_x \leftrightarrow RITnode_x & \text{for } x \in \{original, new\} \end{cases} \quad (8.2)$$

Essentially, ST fingerprints acting as co-ordinators allowed for the problem to be broken down into the two sub-problems:

1. Tracking ST nodes across original and new STs.
2. Tracking an ST node in the RIT (and vice versa).

These issues are tackled below.

Tracking ST nodes across STs

The original ST has hashes calculated for its nodes and stored against `fingerprint` attributes, as outlined in section 8.5.1. As can be seen in Figure 8.1, the original ST, $ST_{original}$, is manually edited, after which it is redesignated as ST_{new} . Therefore, any node fingerprints generated in $ST_{original}$ remain in ST_{new} . Since these fingerprints relate to the ST when it was $ST_{original}$, the `fingerprint` gets renamed to reflect this, e.g. `fingerprint-orig`. Fresh node fingerprints can then be calculated for each node of ST_{new} . Because each node now has a `fingerprint` and a `fingerprint-orig` attribute, a relation can be made between this node in the original tree and in the new tree. Thus, node movements, additions and deletions (which will ultimately have a bearing on nodes of the RIT) can be tracked between the two trees.

Linking RIT nodes back to the ST – the enablement of *standard* node tracking

When a fingerprint-enriched ST is transformed into an RIT, `fingerprint` attributes will be propagated to nodes of the RIT. Thus, a given RIT node can be linked back to a node in the corresponding ST where there is a *one-to-one correspondence* between the ST node and the

8.5. ARCHITECTURE FOR TRACKING A SPECIFIC RIT NODE

RIT node. For the remainder of the discussion, such RIT nodes (where there is one-to-one correspondence with an ST node) will be referred to as “*standard*” nodes.

Once a linkage has been established between a *standard* RIT node and its corresponding ST counterpart, it is now a trivial set of lookup exercises to ascertain the position of a current RIT node in an RIT from a previous round. For example, consider a node “*past and current suicide attempts*” i.e., *suic-past-att*. It is a *standard* node present only in one location in the ST and RIT. In such a scenario, the “default’ *fingerprint tracking algorithm* would be used:

1. A **fingerprint** is generated in ST_{original} from *suic-past-att*’s path.
2. The **fingerprint** propagates to RIT_{original} .
3. An RI is added against this node in RIT_{original} by the user.
4. The user makes edits to ST_{original} that e.g., result in *suic-past-att* being relocated to a deeper level of the tree. ST_{original} is now ST_{new} .
5. All ST_{new} node fingerprints are moved to attribute **fingerprint-orig**. New fingerprints are calculated and stored in attribute **fingerprint**. Note that because *suic-past-att* has been moved, the two attributes will contain different fingerprint values.
6. RIT_{new} is generated by the system and **fingerprint** is propagated to RIT_{new} . (**fingerprint-orig** is not propagated to RIT_{new}).
7. The system observes that node *suic-past-att* in RIT_{new} does not have an RI. It notes its **fingerprint**, and then does a lookup of ST_{new} to ascertain its **fingerprint-orig**.
8. The system queries RIT_{original} for a node that contains a **fingerprint** that is the same value as **fingerprint-orig**. It has now identified the node in RIT_{original} and can copy over its RI.

8.5.3 Interim conclusions

Thus far, a mechanism has been detailed, by which RIT nodes can be uniquely identified by their path in the ST. This path is converted into a hash for convenience, and remains with the node as it is edited/moved by the user in the ST. The new ST recalculates all hashes and passes

8.6. ROBUST TRACKING OF NODES WITH ONE-TO-MANY MAPPINGS BETWEEN ST AND RIT

them on to a newly generated RIT. The new ST also retains information about the previous hashes, allowing the new RIT's nodes to be linked back to the nodes in the original RIT.

This *standard* node methodology is sufficient in cases where there is a one-to-one mapping between ST nodes and RIT nodes. However, in cases where there is a one-to-many relationship, there cannot be an unambiguous linkage from a node in the ST to an instance in the RIT. Since the ST's role is to remove redundancy, it naturally follows that there will be many cases where there will be one-to-many node relationships between the ST and RIT. These are considered in the next section.

8.6 Robust Tracking of Nodes with One-to-many Mappings Between ST and RIT

It can be recognised that not all nodes in an RIT can be classed as *standard* nodes. Some nodes are *g* concepts, which exist as one subtree in the ST, but which are translated into multiple instances in the RIT. The root node of a *g* concept will have a different RI wherever it is instantiated due to the fact that each instance can be placed amongst different siblings. The internal configuration of the *g* concept (i.e., its descendants and their attributes) will however, be the same across all instantiations of the *g* concept. With respect to node tracking, the RI values within all descendants of the *g* concept will be static across all instantiations. Thus, for the purposes of RI value tracking, descendant nodes of a *g* concept (but not the concept node itself) should be regarded as clones, which are fungible across instantiations of the *g* concept. It would therefore be sensible for all cloned nodes to have the same fingerprint, since any instance can be used for the tracking task.²

Without recapitulating all the various types of non-*standard* nodes that exist in an ST/RIT, it should be pointed out that for some classes of nodes (e.g., *gd* concept constituents) it is essential for instantiations to *not* all have the same fingerprint. This is because by definition, their RIs vary across instantiations, hence they should not be thought of as clones. (Assuming the node isn't a clone) the default method of generating a fingerprint against an ST node and propagating it down to the RIT will therefore be inadequate in situations where there isn't a one-

²The notion of clones having the same fingerprint is lent some support in nature in the manifestation of the fingerprints of monozygotic twins (Jain, Prabhakar, & Pankanti, 2002). However, whereas in nature, such twins' fingerprints are *similar* but not exactly the same (due to environmental factors), the word *clone* is used in the RI value and node fingerprint contexts to mean *exactly* the same.

8.6. ROBUST TRACKING OF NODES WITH ONE-TO-MANY MAPPINGS BETWEEN ST AND RIT

to-one correspondence between the node in the two trees. Table 8.1 summarises the various node configurations in the RIT and whether instantiations in each case should be treated as clones (i.e., have the same fingerprint) or as unique.³ The table also outlines whether fingerprints propagated from the ST to the RIT will serve to identify that node in the RIT to the degree required for tracking its RI.

It is clear from Table 8.1 that the majority of node configurations of the RIT can be adequately fingerprinted by mere propagation of fingerprints generated in the ST to the RIT. For example, consider the relatively simple case of a g concept root node (see case two). It is able to be fingerprinted in all instantiations of the RIT even though each instantiation will have a unique RI. This is because each location of the ST where the g concept is to be instantiated is a stub that contains the path to the location where the g concept is defined. Because stubs are themselves nodes, each stub will have its own fingerprint in the ST. This fingerprint will be propagated to the RIT, thereby allowing each instantiation of the g concept root node to be individually fingerprinted. The ST fingerprint is thus viable for this case. However, because the g concept's descendants are only defined in one place in the ST, with no stubs pointing to them, each such descendant will have a uniform fingerprint across all instantiations (is a clone). This is also desirable for RI tracking purposes, because RIs in such cases will also be uniform.

Table 8.1 outlines several cases where the fingerprint propagated from the ST is not viable. Revisiting the case of gd concept root node descendants described earlier, case five details the result for this scenario. It indicates that the RI uniqueness property of instantiations of such a node preclude the viability of the ST-generated fingerprint, thereby rendering the fingerprint not useable in the RIT. As in the case for g concept descendants, this result is a consequence of such nodes being defined in one location in the ST, with no stubs pointing to them. Such fingerprints are thus cloned across instantiations in the RIT. Unlike the case for g node descendants, this is not a desirable result, as each instantiation should be able to have a unique RI—a non-viable fingerprint would prevent tracking of this since a node instance could not be resolved without ambiguity. Variations of this problem also apply to the scenarios in cases 11 (generic datums of type gd inside gd concepts) and 12 (g concept roots inside gd concepts) of the table. Cases 15 and 17 also describe scenarios where the ST fingerprint is non-viable. However, this is due to the fact that the nodes in question are fabricated at RIT-generation time. So there is no ST

³All configurations (be they legal or illegal) are presented. This is to enable the specification of a robust fingerprinting algorithm, which would not require future remedial work if currently illegal configurations were to become valid possibilities.

8.6. ROBUST TRACKING OF NODES WITH ONE-TO-MANY MAPPINGS BETWEEN ST AND RIT

RIT Node instantiation /configuration	RI Uniqueness	Viability of ST fingerprint in RIT	Remarks
1) a <i>standard</i> node	unique	viable	
2) a node that is a <i>g</i> concept root node	unique	viable	
3) a node that is a descendant of a <i>g</i> concept root node	clone	viable	
4) a node that is a <i>gd</i> concept root node	unique	viable	
5) a node that is a descendant of a <i>gd</i> concept root node	unique	not viable	
6) a node that is a <i>gd</i> concept root node that itself is a descendant of a <i>g</i> concept root node	clone	viable	
7) a node that is a descendant of a <i>gd</i> concept root node that itself is a descendant of a <i>g</i> concept root node	clone	viable	
8) a node that is a generic-datum of type <i>gd</i>	unique	viable	
9) a node that is a generic-datum that is not of type <i>gd</i>	clone	viable	
10) a node that is a generic-datum of type <i>gd</i> that itself is a descendant of a <i>g</i> concept root node	clone	viable	
11) a node that is a generic-datum of type <i>gd</i> that itself is a descendant of a <i>gd</i> concept root node	unique	not viable	
12) a node that is a <i>g</i> concept root node that itself is a descendant of a <i>gd</i> concept root node	unique	not viable	
13) a node that is a descendant of a <i>g</i> concept root node that itself is a descendant of a <i>gd</i> concept root node	clone	viable	
14) a node that is a <i>gd</i> concept root node after translation from an ST <code>multiple-tick</code> node	unique	viable	
15) a node that is the descendant of a <i>gd</i> concept root node after translation from an ST <code>multiple-tick</code> node	unique	not viable	no ST fingerprint would be present in the RIT
16) a node that is a <i>gd</i> concept root node that itself is a descendant of a <i>g</i> concept root node after translation from an ST <code>multiple-tick</code> node	clone	viable	
17) a node that is the descendant of a <i>gd</i> concept root node that itself is a descendant of a <i>g</i> concept root node after translation from an ST <code>multiple-tick</code> node	clone	not viable	no ST fingerprint would be present in the RIT

Table 8.1: All hypothetical node instantiation configurations in the RIT. Against each configuration are details about whether RIs are unique or clones, and also, whether an ST fingerprint propagated to the RIT will accommodate the unique/clone status of the RI.

8.6. ROBUST TRACKING OF NODES WITH ONE-TO-MANY MAPPINGS BETWEEN ST AND RIT

fingerprint to carry through to the RIT.

The cases detailed in Table 8.1 that result in non-viable fingerprints in the RIT clearly highlight the need for remedial work on the RIT's fingerprints. Therefore, what is required to be adopted in the RIT generation phase, is a system where affected fingerprints are recomputed using an appropriate algorithm.

8.6.1 Fingerprint remediation in the RIT

Attempting to execute the default version of the *fingerprint tracking algorithm* (see section 8.5.2) would result in non-viable fingerprints for scenarios 5, 11 and 12 detailed in Table 8.1. These can be remediated by recalculating them in the RIT using the following *fingerprint remediation algorithm* (which would be a sub-algorithm to the *fingerprint tracking algorithm*, after fingerprints are propagated to an RIT):

1. For a given node of the RIT,
IF it is a descendant of a *gd* concept AND NOT a descendant of a *g* concept.
 - overwrite the node's fingerprint with a hyphen-separated amalgam of the outermost *gd* concept's fingerprint and the node's fingerprint.

The result of the above algorithm is that any descendant of an instance of a *gd* concept receives a unique fingerprint that reflects that it is contained in that instance of the *gd* concept. The fact that the *fingerprint remediation algorithm* utilises the outermost *gd* concept's fingerprint and not the innermost *gd* concept's fingerprint during remediation allows it to elegantly accommodate nested *gd* concepts. Were the innermost *gd* concept fingerprint to be used in these cases, instantiations of outer *gd* concepts would result in the inner *gd* concept's descendants having the same fingerprint across instantiations.

8.6.2 Analysis of revised *fingerprint tracking algorithm* efficacy

Fingerprint remediation in the RIT allows for each RIT node to be uniquely marked for the purposes of RI tracking. Node movements as a result of ST edits can theoretically cause a node to move anywhere in the tree, be deleted etc. Some of these operations can be particularly complex. For example, a *gd* concept descendant moving into a *g* concept, or its moving out

8.6. ROBUST TRACKING OF NODES WITH ONE-TO-MANY MAPPINGS BETWEEN ST AND RIT

of the *gd* concept altogether. These will result in structural changes to the node fingerprint; i.e., fingerprints will change their values such that they may no longer be hyphenated, or vice versa—making nodes more difficult to track. Specifically, the simplistic lookups performed in steps 7–8 of the “default” *fingerprint tracking algorithm* will not be adequate for complicated node movements. It is therefore important to examine how fingerprints change in possible edit scenarios. Design features of an augmented *fingerprint tracking algorithm* can then be illuminated in context to demonstrate the algorithm’s soundness in tracking the node in each scenario.

Table 8.2 details the types of changes that can be made to an ST, and an analysis on how fingerprints are affected. More accurately, it is an exhaustive list of all the transformational pathways a given fingerprint can take en-route from RIT_{original} to RIT_{new} . Each of the ten outlined pathways represents one or more theoretically possible ST edit scenarios. For example, the Type 3 pathway depicts a scenario where an RIT_{original} node is contained inside a *gd* concept (and is not descended from a *g* concept)—as denoted by the fingerprint `xxxxx-aaaaa`. An edit in ST_{new} has resulted in its no longer being inside a *gd* concept in RIT_{new} —as denoted by the non-hyphenated resultant fingerprint, `aaaaa/`.

In order to track a given node across RITs, a fingerprint tracking algorithm must be able to reconcile a given RIT_{new} fingerprint with the corresponding RIT_{original} fingerprint. This means it needs to:

- Examine the structure of the node’s RIT_{new} fingerprint,
- Use its knowledge of how fingerprints are constructed (i.e., its knowledge of the *fingerprint remediation algorithm*),
- Conduct diagnostic tree searches and node lookups to help determine which of the 10 fingerprint pathways apply in this case,

before finally retrieving the node’s original RI value(s). The *Unified Reconciliation Algorithm*, outlined in pseudocode in Appendix D, encompasses these methods in order to resolve the fingerprint pathway(s) that have been applied to a node of RIT_{new} . It is then able to retrieve its RI(s) from RIT_{original} .

8.6. ROBUST TRACKING OF NODES WITH ONE-TO-MANY MAPPINGS BETWEEN ST AND RIT

ST Edit Operation Type	RIT _{original} fingerprint	ST _{original} fingerprint	ST _{new} fingerprint-orig	ST _{new} fingerprint	RIT _{new} fingerprint	Example Scenario(s) (not exhaustive)
Type 1:	aaaaa	aaaaa	aaaaa	aaaaa/	aaaaa/	standard node relocated OR <i>g</i> concept descendant relocated
Type 2:	aaaaa	aaaaa	aaaaa	aaaaa/	xxxxx/-aaaaa/	standard node relocated into a <i>gd</i> concept
Type 3:	xxxxx-aaaaa	aaaaa	aaaaa	aaaaa/	aaaaa/	<i>gd</i> descendant moved outside of the <i>gd</i> concept, becoming a <i>standard</i> node or a <i>g</i> concept descendant
Type 4:	xxxxx-aaaaa	aaaaa	aaaaa	aaaaa/	xxxxx/-aaaaa/	<i>gd</i> concept descendant relocated due to <i>gd</i> concept being relocated OR <i>gd</i> concept descendant relocated within the <i>gd</i> concept
Type 5:	xxxxx-aaaaa	aaaaa	aaaaa	aaaaa/	yyyyy/-aaaaa/	<i>gd</i> concept descendant relocated into another <i>gd</i> concept
Type 6:	xxxxx-aaaaa	-	-	-	xxxxx/-aaaaa/	generated multiple-tick child relocated as a result of multiple-tick node (or its ultimate <i>gd</i> ancestor concept) being relocated
Type 7:	xxxxx-aaaaa	-	-	-	yyyyy/-aaaaa/	generated multiple-tick child relocated as a result of multiple-tick node being relocated into a <i>gd</i> concept
Type 8:	xxxxx-aaaaa	-	-	-	aaaaa/	generated multiple-tick child relocated as a result of multiple-tick node being relocated into a <i>g</i> concept
Type 9:	aaaaa	-	-	-	aaaaa/	generated multiple-tick child that is contained inside a <i>g</i> concept relocated as a result of the <i>g</i> concept being relocated
Type 10:	aaaaa	-	-	-	xxxxx/-aaaaa/	generated multiple-tick child that is contained inside a <i>g</i> concept relocated into a <i>gd</i> concept

Table 8.2: Relocation operations that can be performed on the ST, and their effect on node Fingerprints. Non-hyphenated fingerprints are represented as aaaa (pre-relocation) and aaaa/ (post-relocation). Hyphenated fingerprints are represented as xxxx-aaaa (pre-relocation) and xxxxx/-aaaa/ (post-relocation). Where a post-relocation hyphenated fingerprint is represented as yyy/-aaaa/, it implies the node is now not inside the (possibly relocated) xxxxx *gd* concept it was originally contained in, but instead, inside a different *gd* concept. The linkage of fingerprints from the RIT_{original} through to RIT_{new} (and vice versa) can be followed via traversing each table column. (Although ST_{original} fingerprints are generated before RIT_{original} fingerprints, RIT_{original} fingerprints are presented first in order to more easily follow the transmission of fingerprints from RIT_{original} to RIT_{new}).

8.6. ROBUST TRACKING OF NODES WITH ONE-TO-MANY MAPPINGS BETWEEN ST AND RIT

Demonstration: the *fingerprint tracking algorithm* applied to *gd* concepts

Examples of the *fingerprint tracking algorithm* (now augmented with *fingerprint remediation* and *unified reconciliation* algorithms) as applied to a given *gd* concept, will illustrate how a *gd* concept instantiation could be tracked across RITs. Furthermore, they will serve to highlight features of the algorithm that enable it to successfully track fingerprints for the other theoretical edit scenarios.

ST type 4 edit operation (see table 8.2) - *gd* concept descendant being relocated as a result of *gd* concept relocation

1. After propagation from ST_{original} , the node's fingerprint is recalculated in RIT_{original} using the *fingerprint remediation algorithm*. Note that since the node is descended from a *gd* concept, the node receives a hyphenated fingerprint in each instantiation of the concept. This ensures that each instantiation is uniquely fingerprinted, since it can have its own unique RI. An RI is then assigned to the node by the user.
2. The reference to the *gd* concept is relocated to a different location in ST_{original} , implying the *gd* concept instance (and its descendants) are to move. ST_{original} is now ST_{new} .
3. ST_{new} fingerprints are reassigned to **fingerprint-orig**, recalculated, and propagated to RIT_{new} as in the case for *standard* nodes. RIT_{new} fingerprints are overwritten where appropriate using the *fingerprint remediation algorithm*.
4. The system observes that the (*gd* concept descendant) node in RIT_{new} does not have an RI. (Unlike in the case of a *standard* node) the *unified reconciliation algorithm* notes its hyphenated **fingerprint**, e.g. **xxxxx'-aaaaa'**. The system needs to ascertain **fingerprint-orig** for this node if it is to be able to locate it in RIT_{original} . **fingerprint-orig** cannot directly be queried in ST_{new} because this instantiation of the node will not exist in ST_{new} . Therefore, the system exploits the fact that the **xxxxx'** component of the node's fingerprint is also the fingerprint of the *gd* concept instance it is contained in. ST_{new} can directly be queried for the **fingerprint-orig** corresponding to **xxxxx'**. Similarly, the system exploits the fact that **aaaaa'** corresponds to the fingerprint of the uninstantiated location of the node in ST_{new} . Thus, **fingerprint-orig** can be located for the uninstantiated location. Finally, these two **fingerprint-orig** values can be amalgamated to yield **xxxxx-aaaaa**.

8.6. ROBUST TRACKING OF NODES WITH ONE-TO-MANY MAPPINGS BETWEEN ST AND RIT

This corresponds to the fingerprint of the node in $RIT_{original}$ —the original fingerprint is thus reconstituted.

5. The system queries $RIT_{original}$ for a node that contains a fingerprint that is the same value as `xxxxx-aaaaa`. It has now identified the node in $RIT_{original}$ and can copy over its RI.

The ability of the algorithm to reconstitute/extrapolate an $RIT_{original}$ fingerprint from an RIT_{new} is key to node tracking in cases where *gd* concepts are involved. The outcome of a search of $RIT_{original}$ for the candidate fingerprint is also important in determining the ST edit operation that has indeed taken place on this node. A Type 4 edit should mean that `xxxxx-aaaaa` is a valid fingerprint in $RIT_{original}$. If on the other hand, a search does not yield `xxxxx-aaaaa`, it implies that it is possible that a different edit operation has taken place. For example, the node may have originally been a *standard* node that has been brought into a *gd* concept. It could also have originally been contained in a different *gd* concept. Therefore, further searches are carried out to identify the edit type(s), and to ultimately reconcile the fingerprint.

Relocation of *gd* concept descendant within the *gd* concept – The amalgamated fingerprint is comprised of the *gd* concept root node and the node in question. Therefore, internal movements of the node do not result in a different remedial fingerprint, thereby allowing the *gd* concepts example above to satisfy this case without modification. Note however, that there is thus a constraint implied by the *fingerprint remediation algorithm* that a *gd* concept cannot have multiple copies of a *gd* concept inside it. This is because their constituents could not be resolved without ambiguity since a remediated fingerprint is comprised of the root *gd* concept fingerprint and the descendant fingerprint. This is an acceptable constraint because according to the semantics of the knowledge structure, this is not a valid scenario.

Relocation of *gd* concept descendant to outside of the *gd* concept –

- If the descendant node is relocated into another *gd* concept, it can be tracked in a manner similar to that detailed above. However, the relocation is classed as Type 5 in Table 8.2, since (unlike a Type 4 relocation within the same *gd* concept) there is the difference that `yyyyy-aaaaa` will not exist in $RIT_{original}$. Thus, in this case, the *unified reconciliation algorithm* searches $RIT_{original}$ for all instantiations of fingerprint `aaaaa`, i.e., nodes with

8.6. ROBUST TRACKING OF NODES WITH ONE-TO-MANY MAPPINGS BETWEEN ST AND RIT

fingerprints ending with `-aaaaa` and those with fingerprints of `aaaaa`, since all their RIs will be desired.

- If the descendant node is relocated to outside of the *gd* concept and becomes a *standard* node or one inside a *g* concept, the node can be tracked in the same way a *standard* node can be tracked (see examples in section 8.5.2 and the present section). However, the *fingerprint reconciliation algorithm* step will not be able to find the node's fingerprint in RIT_{original} . This is due to the fact that whereas the node's fingerprint in ST_{original} will be non-hyphenated, the node's fingerprint in RIT_{original} will have been hyphenated as a result of fingerprint remediation. This fact will differentiate this ST edit type (Type 3) from a *standard* node relocation (Type 1). The RIs of all instantiations of this node in RIT_{original} are potentially required in RIT_{new} . The *fingerprint reconciliation algorithm* therefore searches RIT_{original} for all nodes with fingerprints ending in `-aaaaa`.

8.6.3 Conclusions and discussion on node tracking

Section 8.5 introduced a method of using node paths (abstracted as fingerprints) for tracking nodes across RITs. However, the method was limited to nodes where there is a one-to-one correspondence between the node in the ST and the RIT. This is because fingerprints are created in the ST and propagated to the RIT (since the RIT is a generated tree). The replication and RI instantiation of 'template' branches of the ST as part of RIT generation means copies of nodes exist in the RIT that do not exist in the ST. Thus, the established fingerprinting mechanism was inadequate for uniquely identifying instantiations where those instantiations each required RIs unique to them.

The present section has built on the fingerprinting mechanism used to track *standard* nodes across RITs by ensuring that fingerprints uniquely identify an instantiation where that instantiation will have a unique RI (*gd* concepts). Fingerprint remediation is a post-process that occurs in the RIT, and involves creating a hyphenated fingerprint that encapsulates the location of a *gd* concept descendent within the context of the *gd* concept instantiation.

This section has proceeded to identify and tabulate the types of relocation operations that can occur to a given node. This analysis, together with the property of hyphenated fingerprints of maintaining a unique node fingerprint that identifies the node as well as the node's context (i.e., the containing *gd* concept) has informed a reconciliation algorithm. This algorithm is able to deduce the type of relocation operation that has been performed on a given node *post-hoc*

and thereby determine which node(s) of the original RIT to query for RI values. The overall tracking algorithm parsimoniously accommodates arbitrary nestings of g and gd concepts. This is due to the fact that fingerprint remediation does not take place inside g concepts, and when it does occur, it only takes into account the ultimate gd concept ancestor. It also naturally handles the situation (in the ST) of arbitrarily nested g and gd concepts where a nested concept is referenced *independently* to the container concept, meaning that a given node can in fact satisfy more than one type of edit simultaneously. This is accommodated by virtue of the fact that each RIT node searches for its own RIs, taking into consideration its g or gd ancestor context (which is encapsulated in the fingerprint produced by the remediation algorithm).

Although the fingerprint tracking algorithm has been illustrated in the context of node relocations, it can be applied unmodified to node copy operations. This is because node lookups are performed on ST_{original} and RIT_{original} , which of course are the pre move/copy trees.

The use of fingerprints for node tracking bears some similarity to the work of Marian, Abiteboul, Cobéna, and Mignet (2001), in that a unique XID integer is assigned to each node; however, the node's placement is not used to calculate the XID. Thus, such an approach would have difficulties reconciling instances where a gd descendant has been moved out of the gd concept, and RI information from each and every original instantiation is required to be presented in the new location.

The current approach is also different to the node fingerprints used in Rönnau, Pauli, and Borghoff (2008), who employ MD5 hashes, but where the node hash is a combination of the node element name and its attributes, and *not* its location. Therefore, node instantiations may not readily be accommodated. The present approach of path-based hashed fingerprints combined with augmented *context-based* hashed fingerprints is well suited to the specialised task of tracking *derived* trees. Specifically, scenarios where derived trees are expansions containing repeating, yet individualised instantiations of the same underlying branch are naturally accommodated by this scheme.

8.7 RI Reappraisal

Recall from Section 8.4 that successful migration of RIs between RITs is a two-phase problem. Once a node has been tracked across RITs, it is not always sufficient that the old RI is merely copied over. The RI of a node is a weighting attached to the node, such that the weightings of all siblings sum to unity. Therefore, it follows that changes to siblings of a node under

consideration have an impact on the RI of that node. For example, deletion of a sibling node requires a rescaling of the RIs of the remaining nodes. The current section outlines the various types of change that require re-adjustment/solicitation of RIs. It discusses how such changes will be identified and appropriate methods of RI readjustment where possible. Where expert input is required, methods for flagging this up are described.

8.7.1 Changes that affect RIs and schemes for RI reappraisal

Consider an RIT_{new} concept node (i.e., a node containing at least one child node). The concept node, when being compared with itself in $RIT_{original}$ with respect to its children, will be in one of the following mutually exclusive states:

- *unchanged* – The concept’s children have not been added to or deleted from: it has the same children.
- *adopted* – The concept has at least one additional child that belonged elsewhere in $RIT_{original}$.
- *new* – The concept has at least one additional child newly introduced in RIT_{new} .
- *adopted-new* – A combination of states *adopted* and *new*.
- *removed* – The concept has one or more of its original children removed.
- *adopted-removed* – A combination of states *adopted* and *removed*.
- *new-removed* – A combination of states *new* and *removed*.
- *adopted-new-removed* – A combination of states *adopted*, *new* and *removed*.

RI reappraisal and recalculation schemes for each concept state are presented below.

- *unchanged*: concept will not require any adjustment of its children’s RIs.
- *adopted*: concept will require its additional children’s RIs to be validated/ratified by an expert before they are used. Therefore, the affected **ri** attributes containing the RI values that were obtained from the original RIT will be renamed to **ri-original**. Effectively, this will mean that newly adopted children will have no RI, and that the RIs of pre-existing children will not require modification. An RI elicitation tool could easily be used to look for nodes with no **ri** attributes, and present these to the user in a suitable manner (along

with information from any *ri-original* attribute). Once the user has assigned RIs to these nodes, the RIs of the siblings will automatically be adjusted, and the *ri-original* attributes deleted.

- *new*: concept will require its additional children's RIs to be supplied by an expert. Effectively, this will mean that newly added children will have no RI, and that the RIs of pre-existing children will not require modification (until RI elicitation). RI elicitation will be handled via the RI elicitation tool.
- *adopted-new*: refer to schemes for *adopted* and *new* states.
- *removed*: concept will require its remaining original children's RIs to be rescaled so that they sum to unity. This would be an automatic operation, not requiring expert intervention.
- *adopted-removed*: the scheme for the *removed* state should be applied first, ensuring that all the remaining *original* children have RIs that sum to unity. This leaves the remaining *original* children with a sensible default RI value. Following this, the scheme for the *adopted* state should be applied.
- *new-removed*: consistent with the rationale given in the *adopted-removed* state scheme, the *removed* state scheme should be applied first, followed by the *new* state scheme.
- *adopted-new-removed*: consistent with the rationale given in the *adopted-removed* state scheme, the *removed* state scheme should be applied first, followed by the *new* and *adopted* state schemes.

In summary, the above schemes rescale existing RIs among remaining nodes where nodes have been deleted. Furthermore, where there are also nodes that have been added, the RIs of these nodes are uninstantiated, and are left to be solicited from the expert. RI elicitation tools will help with this process, soliciting the missing RIs, providing access to previous RI values for such nodes, if available, and recomputing RIs of sibling nodes with respect to the new solicited values.

8.7.2 The application of heuristics in RI reappraisal

There is scope for reducing the RI reappraisal process further via the application of heuristics in certain cases. Consider the *adopted-removed* instance where all of the node's original

children have been removed, and all its adopted children were also siblings in RIT_{original} . The **adopted-removed** reappraisal algorithm in its default incarnation would demand ratification of the provisional **ri-original** values in RIT_{new} before their acceptance as legitimate RI values. However, in this case it is clear that the correct course of action is to forego ratification and to directly use the RI values. Should these RI values not sum to unity (due to only some of the original siblings being *adopted* into the node), the RIs should be automatically rescaled.

A similar optimisation can be applied to the *adopted-new-removed* instance where all the original children have been removed and the new children consist of a mixture of *adopted* original siblings and *new* nodes. The *adopted* siblings' Relative Influence values need not be ratified (but should be rescaled to sum to unity).

Further heuristics could be applied to reduce human RI reappraisal; for example, assigning default RI values to *new* nodes based on 'unused' RI as a result of sibling removal etc. However, the algorithms and strategies mentioned thus far err on the side of caution rather than trying to reduce human intervention to a point where it may become unsafe to do so. Indeed, two ST modification operations that have exactly the same mechanics may represent different intentions and focuses for the expert that would require different treatment of RIs in each case. An overly enthusiastic heuristic in such a scenario would treat RIs in the same manner in both cases, leading to an RI instantiation error that may not be easily discernable *post hoc*.

8.7.3 The mechanics of RI reappraisal

RI reappraisal is enacted as a post-process to RI reconciliation. Each concept of RIT_{new} is looked up in RIT_{original} . The children of each concept are then compared across RITs (via their node codes) to determine the state of the RIT_{new} concept. New and adopted children will be differentiable by the absence/presence of pre-existing RIs in RIT_{new} . However, further lookups will be necessitated by cases amenable to the application of heuristics detailed in section 8.7.2. Finally, **ri** and **ri-original** attributes are updated accordingly, ready for experts to fill in missing RIs or to select RIs in cases that a node has multiple associated RIs (as a result of e.g., being taken out of a *gd* concept).

8.8 Conclusions

The chapter has introduced the problem of synchronising (data in) derived trees where the progenitor tree has been altered. In the context of the Galatean Tree Hierarchy, this problem arises when an RIT that has been derived from an ST has been supplemented with further information not derivable from the ST. When the ST is subsequently altered, a regenerated RIT would not contain the supplemental information (i.e., RI values). Therefore the chapter has proceeded to elaborate on a method to reliably carry over supplemental information from the old RIT into the new RIT, bearing in mind that the trees could be radically different with regards to their structure.

The method employs tracking of nodes via their node paths (abstracted as fingerprint hashes). Where individual tracking is required for a node that has been instantiated in multiple locations as part of tree expansion, the fingerprint is augmented. Specifically, the fingerprint becomes an amalgam of the containing *gd* concept's fingerprint and the node's fingerprint. Thus, instantiation location information is encapsulated in the node's fingerprint.

ST edits result in a new set of fingerprints being generated and propagated to the newly derived RIT. The old set of fingerprints are also maintained in the modified ST. In simple cases, the fingerprint tracking algorithm outlined in this chapter uses the new fingerprints in the RIT to track back to the old fingerprints in the progenator ST. This is effectively the key to the node location in the old RIT. In more complex cases that involve node instantiations, the algorithm employs original fingerprint reconstitution and intelligent querying of the RIT to locate the node in the original RIT.

The innovative combination of path-based fingerprint hashes combined with an overriding scheme that encapsulates *identification* and '*location-tagging*' of node instantiations, is well suited to this problem. A node can be identified in its own right and it can also be simultaneously identified that it is contained in a particular instantiation of the concept. It is these two properties that drive fingerprint reconstitution and intelligent querying of the old RIT. Such a fingerprinting scheme is generalisable to any domain that requires tracking of nodes in derived trees that are expansions containing repeating instantiations of the same underlying branch.

Finally, having established the method for tracking RIT nodes, the chapter proceeds to define a set of rules for copying over and reappraising RIs. The rules ensure that RIs in the new tree are automatically instantiated where it is safe to do so, and only with sensible values.

The methods and algorithms developed in this chapter mean considerable time savings with regards to remedial work by experts when tree structures are modified. They allow the trees in the Galatean Tree Hierarchy to remain separate, and true to their *raison d'être*, yet minimise the complexities of integrating modifications from trees upstream. Ultimately, this makes GRiST CDSS knowledge maintenance an easier task for the knowledge engineer.

9

Deployment of a Preliminary Risk Assessment Solution

9.1 Introduction

This chapter considers operational issues surrounding the deployment of the initial risk assessment tool. It argues how the use of a monolithic tree to drive assessment would result in inefficiencies in the use of storage and network bandwidth. It therefore demonstrates the need for specialised trees that are derived from the RIT, and which underpin specific aspects of the assessment process, limiting data redundancy. These are:

The Client Assessment Tree (CAT) – A derivative of the RIT with all nodes instantiated. All nodes are stripped of everything but structure information.

The Question Tree (QT) – A derivative of the RIT, holding question data relating to nodes contained in the CAT.

The Answer Tree (AT) – Generated during the course of a patient assessment in order to store clinician-supplied data.

The second part of the chapter discusses the preliminary risk assessment tool infrastructure that was created, and how the specialised assessment trees were used to drive it. It introduces the three versions of the Galatean Risk Screening Tool (GRiST) that were created to run on top of this platform; each developed to cater for different stages of integration with Trusts, and each varying in its technical capability. These range from the non-technical paper-based GRiST, to the lightweight HTML version, through to the fully featured Java-based fat-client GRiST; all freely accessible on the GRiST website.¹

Finally, the reporting capabilities of the platform are showcased, with a discussion on the potential future directions along which reporting and analysis can be taken.

9.2 Rationale for Specialised Assessment Trees

This section explores the practical limitations associated with the RIT if it is to be directly used to drive assessment tools. Consequently, it establishes the requirement for a Client Assessment Tree, a Question Tree, and an Answer Tree.

9.2.1 The need for a Client Assessment Tree

Recall that the RIT expands generic concepts of type *gd* in order to allow for their instantiation with RI values. However, there is as of yet, no expansion of *g* concepts—concepts whose RI values do not vary with the location of their instantiation. Expanding such concepts in the RIT would have represented unnecessary bloating of the tree and duplication of work to instantiate the same RI values at each location during elicitation activities.

A patient assessment data gathering tool would however, require such *g* concepts to be instantiated. This is because the clinician should have the flexibility to view and answer the associated questions at any instantiated location, since logically, the questions form a part of that locale.

One approach to the instantiation of *g* concepts would be their enactment within the tool logic itself. However, this would not be an efficient solution for a number of reasons. The first is one of multiple tool implementations. There could theoretically be (and indeed the project did spawn) multiple tool implementations to suit different usage scenarios, e.g., a paper-based tool, a thin-client tool with minimal client-side logic, a fat-client tool with heavier client-side logic,

¹<http://www.galassify.org/grist>

etc. Each hypothetical implementation would therefore need to incorporate the logic to perform node instantiations, thereby increasing development overhead.

A further issue with direct *g* concept expansion within end-user tools is that of duplication of computation. With each and every invocation of an assessment, the RIT would be required to be dynamically expanded. On a modern desktop computer this may not necessarily be a hinderance, but with the proliferation of hand-held/mobile computing on low-powered CPUs, this might introduce a significant time penalty. Similarly, offloading to the server could result in a performance bottleneck during busy periods of activity.

Recall that the RIT contains ‘profiles’ for varying levels of clinician expertise. The system of `level` attributes within the tree help determine ‘soft’ pruning points corresponding to expertise grades. Following `level` attributes is again, work that adds additional complexity to end-user tools. Theoretically, this work can be done *a priori*, with derivative trees corresponding to each level being generated and stored/cached before use.

A fourth issue that demonstrates the need for a separate derivative tree is that of recording and presentation of the results of classification using the Galatean model. During classification, the Galatean model takes answer values provided for leaf nodes of the tree and uses them to calculate and percolate mgs up to the root risk nodes. That is, the intervening nodes have their mgs calculated, and it is the effective flow of mgs up the tree that is important when visualising how answers impact on risk. Therefore, the logical location to record the results of running an assessment through the classifier is within the tree instance itself. But recording within the RIT instance, (which has not had *g* concepts instantiated), means that percolation is difficult to visualise directly because it requires “piecing together” of the tree.

It is with the above reasons in mind that the requirement for a fully instantiated Client Assessment Tree (CAT) becomes apparent. The primary role of the CAT is thus, to provide a tree instance that can structure the assessment according to clinician expertise level, and which can subsequently be used to store/display classification results.

9.2.2 Optimising storage and bandwidth through separation of question data

The CAT, as envisaged, is a monolithic structure, and can in theory, directly drive an assessment data gathering tool/decision support system. It contains all the node structure data that is pertinent to the Level of the clinician, as well as question data, answer data type definitions, RIs and value-mg profiles. Furthermore, it can store mgs generated during classification of the

assessment in situ within the nodes.

The monolithic nature of the envisaged CAT does however introduce inefficiencies when the manner and environment in which it will be used is considered. Consider for example, a hypothetical ‘thin-client’ data gathering tool. This tool would rely on the server to run the Galatean classifier on the assessment data (as opposed to relying on client-side logic). During the start of an assessment, it will download a fresh copy of the CAT and proceed to record answer data against it as the assessment progresses. The clinician may wish to make a provisional classification of the data at any point in time. The easiest way to achieve this will be for the client program to send the answer-instantiated CAT to the server, whereupon it will be classified by the Galatean model and populated with mgs. The classified CAT will then be sent back to the client program for displaying. A clinician could make any number of provisional classifications as the assessment progresses. Furthermore, upon an assessment save or assessment completion, the server may wish to maintain a copy of the classified CAT within the database as a permanent record of that assessment.

The above scenario is clearly inefficient from the point of view of bandwidth because each classification requires the uploading and downloading of the complete CAT. Upload bandwidth tends to be severely restricted by ISPs (Huang, Li, & Ross, 2007). Similarly, a large CAT saved for each assessment impacts on storage. Therefore, the CAT should ideally be as lean as is practicable.

Question and question-related data within the CAT does not change throughout the lifetime of an assessment. It is thus, more efficient to have such data within a separate structure that is only downloaded once at the start of the assessment session. The solution to reducing CAT size that was adopted was therefore, one of stripping question and question-related data from the CAT and placing it within a separate data structure, i.e., the Question Tree (QT).

The QT is a data structure that is generated directly from the RIT along with the CAT. Unlike the CAT, whose hierarchy drives the hierarchy of the assessment tool, the QT does not have an implicit hierarchy. Its role is to serve as a lookup table or directory to the assessment tool whilst it is compositing the data gathering interface. The QT is thus a flat structure indexed by node `codes` which correspond to the codes against nodes within the CAT. Whenever the assessment tool wishes to ascertain the question data corresponding to a node of the CAT, it performs a lookup of the node in the QT via its `code`. A further benefit to lower transmission and storage overheads is the reestablishment of the removal of question-text redundancy. In

essence, the CAT and QT are used within the system as a relational database consisting of two tables, thereby merging the benefits of relational databases with the flexibility of XML trees.

9.2.3 Further optimisation through separation of answer data

Further consideration of the hypothetical thin-client assessment scenario outlined in Section 9.2.2 point to optimisations that can also be made with regards to the need for uploading the CAT itself. Upload bandwidth speed is a limiting factor, so it is important to further reduce this where possible.

The CAT, as conceived of so far, maintains the overall structure of the assessment, but does not itself include question data. Additionally it is used to store node answers in situ during the course of an assessment. The CAT is used as a transmission vehicle for those answers when a classification is required. Upon classification, the answers are returned within the CAT in addition to mgs. Thus, in the scenario, it can be observed that a costly upload of the CAT is performed primarily to transmit answers to the server (which then come back to the client unchanged within the classified CAT).

In theory, the server itself does not require anything more than the assessment answers, as it can easily load an ‘instance’ CAT from the database in order to populate it with computed mgs. It therefore implies that transmission efficiencies can be gained from storing answers separately to the CAT. If answers are stored separately, only a small Answer Tree (AT) data structure need be sent to the server. Once the server has reconciled answers to ‘instance’ CAT nodes, it can perform a classification and send down a classified CAT to the client. Upload bandwidth requirement is thereby reduced to a minimum.

Similar to the QT, the AT is a flat structure, indexed by a node’s `code`. Unlike the QT, the AT is not created on the server, or generated from the RIT. The AT is instead, generated by the assessment tool when needed (i.e., when there is answer data to record), and nodes are created on an *ad-hoc* basis.

9.3 Generating the Assessment Trees

This section covers the procedures for generating the CAT, QT and AT; the trees that will be used to drive the assessment. XSLT is once again used to effect all transformations. This means the transformation process is automated and the potential for human error as a result of direct manipulation of the tree structures is eliminated.

9.3.1 Generating the CAT

The CAT is essentially a fully instantiated RIT, stripped of the majority of its question-related attributes. Furthermore, whereas the RIT contains assessment information pertinent to all Levels of clinician expertise, the CAT is instead, a tree matched to each Level. Soft pruning points dictated by `level` attributes are enacted using XSLT to automatically generate CATs tailored to each Level. This yields the benefit that tool logic can be Level-agnostic, yet assessments at different Levels can be serviced within the tool by supplying the appropriate CAT. The ability of XSLT to easily treat soft pruning points as hard pruning points effectively results in simplifying the development and operation of the end-user tool(s). It is thus, another example where adopting XSLT to mold data structures as required ‘behind the scenes’ brings efficiency and flexibility to the system.

Briefly, the following transformations will have taken place to the RIT in order to generate a CAT at a given Level:

1. All generic concept nodes are fully instantiated in all locations of the tree.
2. The *generic nodes* pseudo-risk is removed as it has been rendered superfluous to requirements.
3. `values`, `question`, `value-mg`, and `help` attributes are removed. `layer` attributes are retained, and `filter-q` attributes emptied.^a `ri` attributes remain.
4. `level` attributes are executed in generating appropriate CATs for each Level using the algorithm detailed at the end of Section 6.6.1. Any `filter-q` attribute that is present at a Level matching the CAT’s Level is removed because the node will become a datum. All `level` attributes are then removed from the tree.

^a`filter-q` attributes are emptied but are not removed outright as a computational efficiency measure. Leaving such attributes in situ negates the need for the assessment tool to ascertain via a lookup of the QT, the filter or layer status of each node during tool rendition.

After using the above algorithm to generate a Level zero CAT, the *suicidal ideation* concept of Figure 6.8 will now look as depicted in Figure 9.1.

The corresponding representation within a Level 1 tool is presented in figure 9.2. Here, the *suicidal ideation* concept, by virtue of its having a `level="1"` attribute against it, has been hard-pruned. Consequently, the concept has been reduced to one datum node.

```

<node label="suicidal ideation" filter-q="" code="suic-ideation" ri="ri">
  <node label="How much suicidal ideation is verbalised"
    code="suic-id-verb" ri="ri"/>
  <node label="ability to control suicidal ideation"
    code="suic-id-control" ri="ri"/>
  <node label="content of suicidal ideation indicates high risk"
    code="suic-id-hi-risk" ri="ri"/>
  <node label="frequency of suicidal ideation" code="suic-id-freq" ri="ri"/>
</node>

```

Figure 9.1: Part of a Level 0 CAT XML representing *suicidal ideation*. Question-related attributes have been removed, thereby shrinking the concept definition. `code` attributes, although not depicted in the original ST version of the concept, were later introduced into the ST prior to RIT generation.

```

<node label="suicidal ideation" code="suic-ideation" ri="ri" />

```

Figure 9.2: Part of a Level 1 CAT XML representing *suicidal ideation*. Sub-nodes have been removed because the concept is a Level 1 concept. Consequently, the concept becomes a datum. This renders its original `filter-q` attribute also irrelevant.

Answers provided during an assessment are not stored in the CAT. Instead these are stored within the AT.

9.3.2 Generating the QT

The QT is generated directly from the RIT. It is a flat structure that will record *relevant* question information stripped out from the CAT. QTs will thus be matched with CATs, meaning that a CAT at a given Level will have a corresponding QT for that Level. Information not relevant to a CAT of a specific Level will therefore, not be present in the QT for that Level.

Figure 9.3 represents the typical composition of a QT. A number of rules govern when certain combination of attributes and values may appear within a given node. For example, `values="filter-q"` is a legal dyad, even though *filter-q* is not strictly a data type. The validity arises out of the fact that a QT node will never have both a question and a filter question to be asked. This is due to the QT's being matched to a tool Level. And at a given Level, a node will either have a `question` or a `filter-q`, but not both. Therefore, in cases where a filter question is to be asked, the QT node will have `values="filter-q"` and `question="The filter question text"` (the data type of the filter question is implied). A fuller list of rules for generating the QT are available in Appendix E.1.

```

<questions>
  <node code="datum1-code-name" question="question relating to datum 1"
    values="scale|integer|real|nominal|date-year|date-month|date-week|
      date-day|filter-q|layer"
    help="help info"
    value-mg="((x1 y1)(x2 y2)(xn yn))"/>
  <node code="datum2-code-name" question="question relating to datum 2"
    values="nominal" value-mg="((none 0)(some 0.7)(lots 1))"
    help="help information" />
  <node code="concept-node-code" question="question relating to concept"
    values="scale" layer="1"/>
  <node code="root-risk-code"
    question="question relating to risk node judgement" values="scale"/>
</questions>

```

Figure 9.3: The composition of the QT. The flat node structure lists question-related data for all CAT nodes that contain a question that is to be asked.

Relating back to the *suicidal ideation* ST concept example of Figure 6.8, the corresponding Level 0 CAT fragment of Figure 9.1 is complemented by the Level 0 QT fragment presented in Figure 9.4. This is generated via the algorithm in Appendix E.2.

```

<questions>
  <node code="suic-ideation" value-mg="((0 0) (10 1))" values="filter-q"
    question="Is the person having suicidal thoughts or fantasies?" />
  <node code="suic-id-verb" value-mg="((0 0)(10 1))" values="scale"/>
    question="To what extent is the person talking about suicidal
      thoughts or fantasies?"
  <node code="suic-id-control" value-mg="((0 1) (10 0))" values="scale"
    question="To what extent is the person able to control
      the suicidal thoughts or fantasies?"/>
  <node code="suic-id-hi-risk" value-mg="((0 0) (10 1))" values="scale"
    question="To what extent does the content of the suicidal thoughts
      or fantasies raise serious concerns about suicide risk?"
    help="e.g. get away from it all; harming others; harming themselves"/>
  <node code="suic-id-freq" values="nominal"
    value-mg="((DAILY 1)(WEEKLY 0.5)(MONTHLY 0.2)(LESS-THAN-MONTHLY 0))"
    question="How often do the suicidal thoughts or fantasies occur?"/>
</questions>

```

Figure 9.4: Part of a Level 0 QT XML representing *suicidal ideation*. Nodes are in a flat structure, each containing question-related attributes derived from the RIT, and no longer present in the corresponding CAT.

A Level 1 QT fragment corresponding to the *suicidal ideation* concept is presented in

Figure 9.5. Again, this QT is shorter than the Level 0 QT because the nodes beneath the *suicidal ideation* concept will have been pruned in the Level 1 CAT, and therefore, are not required here. This QT is generated via the algorithm in Appendix E.3.

```
<questions>
  <node code="suic-ideation" value-mg="((0 0) (10 1))" values="scale"
    question="To what extent do the person's suicidal thoughts/fantasies
      match those that would give you most concern about
      suicide risk?"/>
</questions>
```

Figure 9.5: Part of a Level 1 QT XML representing *suicidal ideation*. In the corresponding CAT, this concept has become a datum by virtue of its `level="1"` attribute being applied. Consequently, the node is no longer a filter node. Thus, the QT data is in reference to the `question` attribute originally contained in this node, and not the `filter-q` attribute.

9.3.3 Generating the AT

As was stated in Section 9.2.2, the AT is generated at “run-time”. Therefore, unlike the QT and CAT, which are prefabricated on the server via XSLT and cached, the AT is produced as and when needed by the assessment tool. When a clinician attempts to save an assessment, the tool creates an AT node for each node that has been answered by the clinician. Similar to the QT, nodes are indexed by `code` within a flat structure, with answers stored against them via `answer` attributes. Figure 9.6 depicts an AT fragment corresponding to *suicidal ideation* concept question answers.

```
<answers>
  <node assessment-status="in progress"/>
  <node date="08032010"/>
  <node code="suic-ideation" answer="yes" management="Provide counselling"/>
  <node code="suic-id-control" answer="3" comment="Could do better"/>
  <node code="suic-id-hi-risk" answer="5"/>
  <node code="suic-id-freq" answer="MONTHLY" comment="Doing well"/>
</answers>
```

Figure 9.6: Part of an AT with answers related to some *suicidal ideation* questions. In addition to answer data, the AT stores clinician-provided data relating to comments and also, the management of the aspect.

In addition to storing question answers, the AT stores any comments the clinician wishes to make against the node. Comments can be made against any node of the CAT, and thus are

not contingent to the node's having a question answered against it, or even, the node having a question to answer!

The **management** attribute is used to store information that the clinician may wish to record about how this aspect should be managed by the patient or any interventions that have been implemented. The **management** attribute is yet another example where the choice of XML for driving the system has added value and flexibility. This functionality was not conceived of from the outset by the research team, but was borne out of feedback from clinicians and best practice recommendations (Kettles & Woods, 2009). The extensibility of XML allowed the additional attribute to be incorporated within the AT file format. Crucially, the data from this attribute could be incorporated within the standard report of the assessment, or be used to drive a specialised report focussing on *management*, without much adaptation to the XSLT-driven reporting functionality (to be introduced later).

Coiera (1997) observes that electronic records systems are sometimes focussed on the process of data collection, yet neglect the context in which they will be used and by whom. The need to facilitate context and layered 'views' are echoed in a case study conducted by Fitzpatrick (2004). Here it is observed that paper-based records are used in a layered fashion with different clinicians using different coloured ink to overlay information on a single paper-based form. Fitzpatrick coins the term, 'working record' to denote a record that facilitates multiple clinicians' needs and layering. In essence, the extensible nature of the AT allows different types of data to be stored centrally within the file, yet easily supports specialised views of that data by using XSLT to select or ignore specific file attributes. The potential for this is that different groups of personnel along the care pathway can each augment or view the data in a manner that is useful for them, and ultimately, useful to the patient. This goes some way towards realising the 'working record' for a mental-health assessment.

9.4 Relating the CAT, QT and AT to the Galatean Tree Hierarchy

Section 7.5 introduced the notion of the Galatean Tree Hierarchy (GTH)—a set of derived trees that aim to make a systematic and managed transition from the domain model to the decision support system. Sections 9.2 and 9.3 introduced the CAT, QT and AT. These three trees specify a model for generating, conducting and representing assessments. They therefore represent the

9.4. RELATING THE CAT, QT AND AT TO THE GALATEAN TREE HIERARCHY

next phase of operations for the system as a whole, i.e., patient data gathering and assessment. It is therefore apt to revisit the GTH in order to visualise how the new trees are incorporated within it.

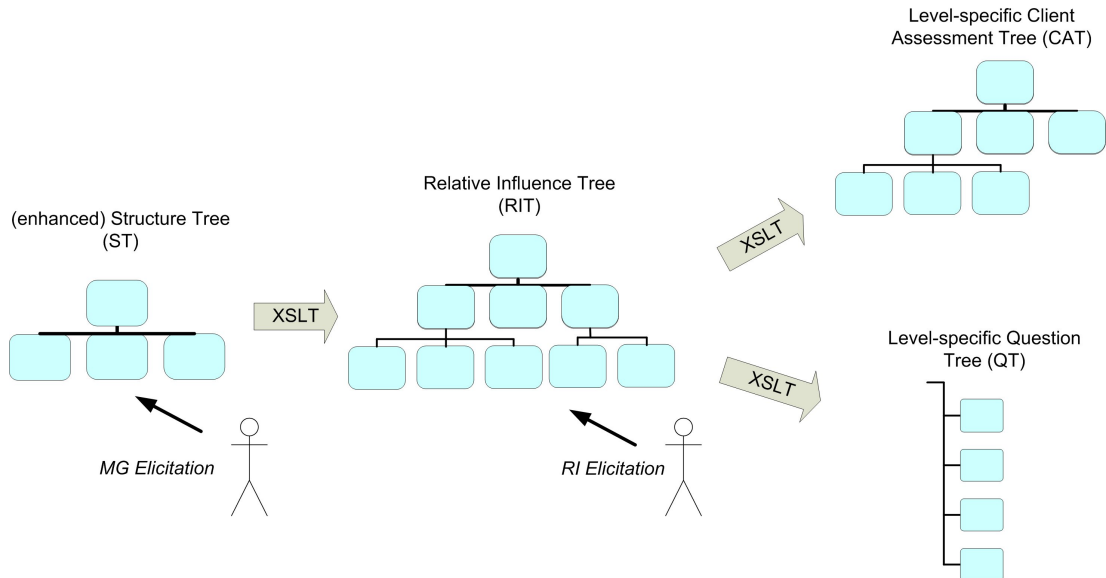


Figure 9.7: The Galatean Tree Hierarchy extended to incorporate CATs and QTs. CATs and QTs derive wholly from the RIT, and are matched to each other by expertise Level. XSLT is used to automatically generate CATs and QTs from the RIT.

Since the CAT is a tree that is derived from the RIT, taking the domain knowledge to a point where it can service assessment generation, it would thus take its place within the Galatean Tree Hierarchy as a child of the RIT. As can be seen from Figure 9.7, the CAT is a wholly generated tree, with XSLT being used to automate the transformational process. This applies to CATs defined for each Level of practitioner expertise.

Similarly, the RIT is the progenitor of the QT; again with XSLT enacting the transformation. QTs are matched to CATs according to Level; and together they drive generation of the assessment.

Not shown in the diagram is the AT. The AT is not strictly part of the GTH as it is a tree that is spawned as and when needed during assessment. However, nodes about which information will be stored in the AT, originate from the CAT. Therefore, the AT could be regarded as a child tree of the CAT were it to be represented within the GTH.

9.5 Deploying the Galatean Risk Screening Tool

Three end-user risk screening tool architectures were built to be driven by the data-gathering /assessment trees of the GTH. These architectures supported deployment of a paper-based tool, a HTML-based tool, and a Java tool. Together, they constituted the deployment of the initial Galatean Risk Screening Tool (GRiST) for working-age adults. This section covers the rationale for development of each of the three flavours of GRiST and introduces their features. Primary focus however, falls on the HTML-based tool, as this was the tool developed as part of PhD work.

9.5.1 Paper-based GRiST

When GRiST was first being readied for deployment, mental-health Trusts were still primarily paper based (A. Lewis, 2002; Ainsworth, 2007), and therefore wanted this format for GRiST. In effect, they initially wanted GRiST to be used simply as a paper-based data-gathering tool. This format needed to be relatively short due to the fact that it would be paper based, hence it required question redundancy to be eliminated. Furthermore, the paper-based format would not require any of the uncertainty information that was present in the CAT and QT. These two factors made the ST amenable to being used as the base tree for driving a paper-based GRiST.

Using XSLT to transform the ST into a web page to represent the paper-based GRiST form is one option that could have been adopted. Indeed, such an option would have obviated the need for creating a separate architecture for the paper tool, as the HTML-based tool (to be introduced in Section 9.5.2) could have been printed off instead. However, this approach to a paper GRiST was not considered further due to the crudeness of HTML and CSS in specifying high-quality print layout.² The situation is improved somewhat with the latest iteration of CSS, version 3, although this standard has not yet reached W3C Candidate Recommendation status, and browser implementations vary in quality and completeness (Schmitt, 2010).

XSL-FO is a markup-language that can more precisely describe how pages should be rendered for print (Holman, 2003). A further alternative is to use a typesetting language such as \LaTeX , which can be used to specify very precisely, the structure and formatting of a document (Mittelbach & Goossens, 2004). Therefore, XSLT was employed to transform the ST into appropriate \LaTeX code. After relatively minor hand-tweaking, the result was a \LaTeX document

²Cascading Style Sheets (CSS) is a markup language used in web pages to specify styling information.

that could be compiled to generate a Postscript or PDF version of what was dubbed the paper tool.³ This was made available on the project website and disseminated to Trusts. An excerpt from the paper tool is presented in Figure 9.8.



Figure 9.8: An excerpt from the paper-based version of GRiST. Rapid screening questions are presented up-front, along with references to additional question locations.

The paper incarnation of GRiST, albeit the simplest format, represents a number of benefits to Trusts that may have previously been reliant on in-house or bespoke risk assessment forms. These are borne out of the tool being derived from the GTH. The knowledge contained within the GTH means that the paper tool is a comprehensive data gathering/risk assessment form that has been developed and validated in consultation with mental-health experts. Shorter versions of the tool can be produced by leveraging the Levels functionality built into the system, thereby providing tools suited to the assessor. Furthermore, the tool’s mainly automated generation from the GTH means that any changes to the domain model can easily be reflected within the tool—keeping it current. Finally, the paper tool is synchronised with the more advanced computerised versions of GRiST because they too are built on top of the GTH. This provides Trusts with an upgrade path to the HTML and Java versions of GRiST once they have become accustomed to

³Paper versions of GRiST are freely downloadable at <http://www.galassify.org/grist>

the paper format, thereby unlocking the full potential of the GRiST Decision Support System. The next step in the upgrade path—the HTML tool—is explored in the following section.

9.5.2 HTML-based thin-client GRiST

The main goal of the research project was to provide an electronic risk assessment tool/decision support system whose expertise could be utilised as an online resource via the Internet. The web browser is ubiquitous across Internet-enabled consumer and business computing devices. In order to gain maximum penetration, it was therefore fundamental that a risk assessment solution be developed that could run within a vanilla install of the browser. Such a solution was not to rely on ancillary plugins such as Flash or Java, the availability of which cannot always be guaranteed. With the AJAX revolution⁴ ushering in an era of sophisticated applications development for the browser, a thin-client GRiST was deemed a viable solution.

Back-end architecture supporting the HTML tool

Developing what was ostensibly an HTML-based thin-client GRiST required employing a number of technologies on the server. These would work to translate the information contained within the CAT and the QT into a fully functioning web application to be run within the browser. The word ‘application’ in this context alludes to the fact that this would involve generating, saving, and reloading assessment information across sessions. In contrast to the Paper tool, where the job of the computer program is to merely produce read-only output, the infrastructure supporting the HTML tool would be required to maintain assessment state information, and the HTML tool required to dynamically reflect that state information.

XSLT embedded within a LAMP stack were the technologies of choice for developing the HTML tool and supporting the assessment process. The LAMP stack is a collection of open source software that can be installed on a computer so that it can be used as a viable webserver (Lee & Ware, 2003). In the context of the HTML tool application,

- **Linux** was the operating system used to host the website and thus, the HTML tool infrastructure.
- **Apache** was the webserver software used to serve the HTML tool to the client browser.

⁴AJAX is the use of scripting capabilities on the web browser to generate interactive applications that update their user interface without page reloads. These applications use behind-the-scenes calls to the server to fetch/send data—often as XML.

- **MySQL** would be used to maintain cached copies of fresh CATs (uninstantiated with answers) and QTs, ready to drive tool creation. MySQL would also serve to store ATs and corresponding classified assessment CATs, ready for reloading or archiving.
- **PHP** would work in tandem with a server-side XSLT processor to translate information from CATs, QTs (and ATs) dynamically into HTML, javascript and CSS markup that rendered the current assessment state as a live HTML tool.

The procedure below provides an overview of how the HTML tool is generated and serviced when a clinician initiates an assessment using their web browser. These steps are visualised in the accompanying diagram (Figure 9.9).

1. A new assessment is requested by the clinician using a web browser.
2. The PHP/XSLT-based tool generation engine loads a new CAT and QT at the appropriate skill Level from the GTH database.
3. A previously saved AT is also fetched if the assessment is a resumed or repeated assessment.
4. The tool generation engine loads a pre-coded XSL stylesheet, whose function will be to transform the data within the assessment trees into the HTML tool.
5. The tool generation engine invokes the server-side XSLT processor and passes assessment trees and the XSL stylesheet over to it.
6. The XSLT processor in combination with PHP generates appropriate HTML/CSS/Javascript files.
7. The generated files are sent to the web browser to display as the HTML version of GRiST.
8. When the assessment is saved/completed, an Answer Tree is stored in the database.
9. Reports can then be generated using the CAT, QT and saved AT, and displayed on the browser.

The tool generation engine, and step 6 in particular, is an interesting example where the benefits of both XSLT and PHP are married together to produce a transformation that would

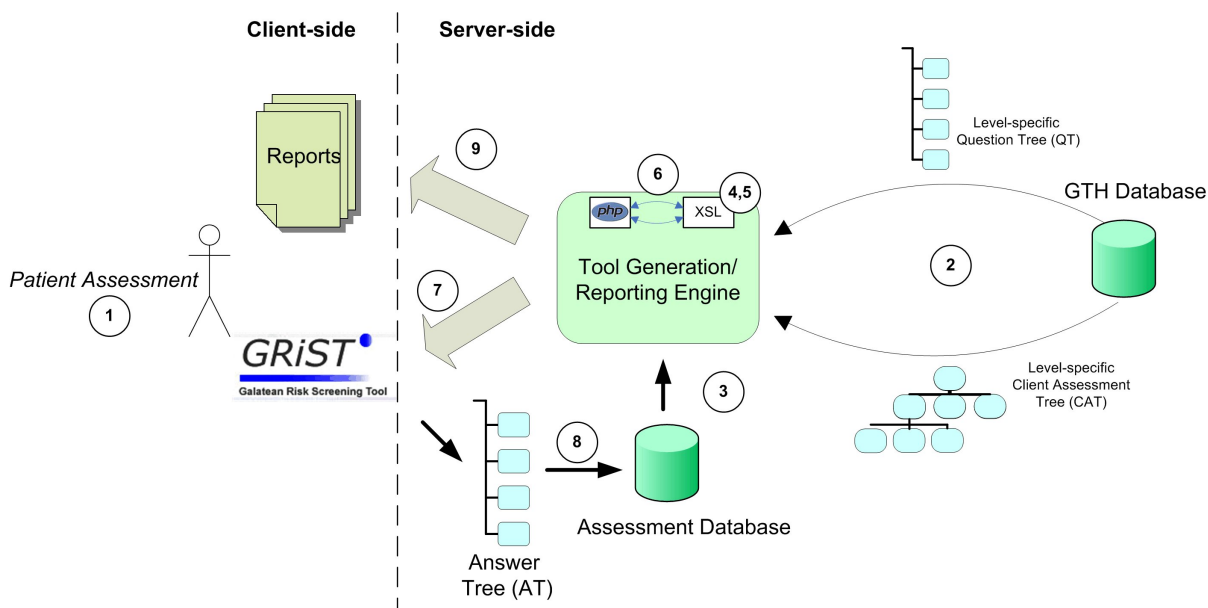


Figure 9.9: Overview of the back-end processes involved in generating the HTML version of GRiST and its associated assessment reports. The tool-generation/reporting engine utilises a combination of PHP and XSLT to transform assessment trees into GRiST.

otherwise have been very difficult to do using either language exclusively. XSLT on its own is eminent at processing XML. However, it is very cumbersome to use when manipulating and formatting text, and does not have any native capability for querying databases (Mangano, 2006). PHP, on the other hand, excels at string manipulation, complex logic, and can directly interface with MySQL (Nixon, 2009).

PHP's XSL extension adds the capability for PHP to register PHP functions with the server's XSLT processor at run-time. This means that functions can be coded using PHP and these can then be invoked directly from within an XSL stylesheet as if they were native functions. The idea is similar to that of using functionality exposed by DLLs in Windows or Shared Objects in Linux (Beazley, Ward, & Cooke, 2002), where the client program is not restricted to being coded in the same language if it is to utilise them.

The function registration feature was utilised heavily within the tool generation engine: where the XSL stylesheet required database access, string manipulation or any other complex processing, it deferred to the PHP side of the engine. This resulted in considerable developmental time-savings.

The HTML GRiST front-end

GRiST is launched via the *Assessment Management Page* within the GRiST website. This page is a dynamic Control Panel-type application which allows patients to be selected and assessments and report statuses to be viewed (Figure 9.10). From this page GRiST can be configured to launch a new, resumed or repeat assessment of the patient via the appropriate button.

Client to Assess: Search for Client: (start typing here...)

Client Information You have selected client No: **bua temp**. There are 7 previous assessments held in the database for this client:

Assessment Date	Status	Tool Used	Options	Reports	Delete
20/11/2010 02:03:36	in progress		Resume	Client Answers Example Classification	<input type="checkbox"/>
19/10/2010 06:49:39	in progress		Resume	Client Answers Example Classification	<input type="checkbox"/>
08/03/2010 13:10:44	completed		Repeat	Client Answers Example Classification	<input type="checkbox"/>
15/01/2010 17:15:51	completed		Repeat	Client Answers Example Classification	<input type="checkbox"/>
15/01/2010 17:13:43	in progress		Resume Repeat	Client Answers Example Classification	<input type="checkbox"/>
14/01/2010 14:18:15	completed		Repeat	Client Answers Example Classification	<input type="checkbox"/>
14/01/2010 13:56:55	in progress		Resume Repeat	Client Answers Example Classification	<input type="checkbox"/>

Figure 9.10: GRiST assessment management interface.

The HTML version of GRiST produced by the tool generation engine was an AJAX application coded in accordance to W3C standards. With only minor tweaks it yielded a uniform experience within the major modern browsers.⁵ Figure 9.11 presents a screenshot of HTML GRiST running within Firefox. The tool is organised into two panels: the left panel contains the tool proper, whilst the right panel contains assessment controls and search functionality.

The screenshot in Figure 9.11 depicts a hypothetical repeat assessment of a patient taking place. Filter questions, when answered in the affirmative, result in the “branch” opening up to reveal further indented questions relating to the concept. In this way, the underlying hierarchy of the CAT is revealed as the assessment progresses and along the lines of questioning.

It is recognised that the number of questions within GRiST and the fact that not all are visible on launch means that search functionality will be of use, particularly to neophytes. The search panel contains an incremental search drop down box with which, users can search all concept labels. Once the concept of interest has been selected, a ‘breadcrumb’ of ancestral concepts leading down to the concept is displayed. Currently visible ancestors are represented

⁵Internet Explorer 7+, Firefox 3+, Chrome 1+, Safari 3+, Opera 10+

Figure 9.11: A screenshot of the HTML tool being used to conduct a repeat assessment of a patient.

as a clickable link, which takes the user to the location of the concept within the tool. The search feature is ‘live’, meaning that any consequences of opening and closing of branches are immediately reflected within the search results panel.

Questions and their associated controls are rendered as specified within the CAT and QT that drive the tool. Nodes can have comments attached via the yellow comment icon against questions. The contents of `help` attributes are rendered as purple ‘tooltip’ icons. If the assessment is a repeat assessment of the patient (as is the case in Figure 9.11), previous answers are available for reference purposes in grey text adjacent to the question.

A fundamental enhancement that HTML GRiST brings to the assessment process over the paper tool is the ‘live’ colour-coding of answers. Once a user has inserted or updated a node answer, the tool performs an evaluation of the answer with respect to the value-mg profile graph that has been defined for the node. Conversion of e.g., dates into durations and any requisite interpolation work is carried out instantaneously and behind the scenes via javascript to calculate and update the colour of the control. This provides the clinician with immediate feedback about the severity of the answer value as the assessment is being conducted.

The scale control is able to take colour-coding one step further by reflecting any arbitrary

value-mg profile directly within the scale prior to a value's selection. For example, Figure 9.12 depicts a *hypothetical* U-shaped relationship between answer value and membership grade driven via the `value-mg` attribute. *cf.* Figure 6.5, which represents a linear relationship.

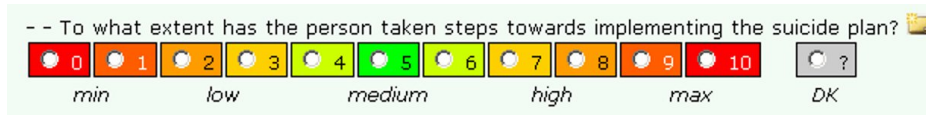


Figure 9.12: A hypothetical U-shaped value-mg profile represented within a scale control. Once a value is selected, the whole scale control changes colour to that of the selected value.

In addition to the search functionality, the right panel contains buttons to save, suspend, or finish the assessment. Clicking on any of these buttons initiates an answer validation process, which when successfully completed, results in an AT being generated and saved in the database.

The answer validation process is a client-side process that verifies that each answer value accords with the type defined in the QT. It also performs sanity checks such as the legality of dates and their correct formatting. Figure 9.13 shows a validation run being performed as part of the save process. The clinician is taken to each error and given direction on how to correct it. Once all errors have been corrected, the save is performed and an AT stored in the GRiST database.

Launching a repeat assessment warrants further discussion, as it exposes some of the complexities involved with maintaining ATs. ATs as have been conceptualised so far, only retain information for the current assessment. A repeat assessment however, involves the display and possible utilisation of information from the previous assessment. Furthermore, a repeat assessment does not have any control over the previous assessment—meaning an assessment can be repeated by a clinician, the repeat assessment then suspended, and the original assessment subsequently deleted by the clinician, ignorant to the fact that this would cause a side-effect on the newer assessment.

To resolve the issue of side-effects, the AT becomes a *composited* tree on assessment repeat initiation. The AT as saved on initial assessment completion acquires `answer-previous` attributes on repeat of the assessment. The `answer-previous` attributes store their own local copy of all the node answers from the initial assessment. This means that once an assessment is repeated and stored on the system, the previous assessment's answers are still available to it even if the previous assessment is subsequently deleted.

- When was the last suicide attempt? 🗓️ ➕
(Please enter a date in the format ddmmyyyy, mmyyyy or just yyyy)

This month does not have the number of days you specified. Please enter a valid date in the format ddmmyyyy, mmyyyy or just yyyy. After correcting the value, press the RETURN key to be taken to the next error.

30022008 DK

- Has there been more
 yes no DK

- When was the fir
(Please enter a date in the format ddmmyyyy, mmyyyy or just yyyy) 041986 DK

- Approximately how many suicide attempts have there been? 🗓️ ➕
(Please enter a number in figures)

The supplied value is not a whole number. After correcting the value, press the RETURN key to be taken to the next error.

dsf DK

- How have the suicide attempts been changing in frequency over the last two years? 🗓️ ➕
 decreasing same increasing DK

- To what extent were the suicide attempts well planned? 🗓️ ➕ 🛔 ⓘ
 0 1 2 3 4 5 6 7 8 9 10 ?
 min low medium high max DK

return to top

save data ⓘ

suspend ⓘ

submit form ⓘ

Search Panel
(type in box below)

Path to Result

Figure 9.13: A data validation run being performed as part of the assessment save process.

Complementing the making available of previous answers for reference is the notion of *answer persistence*. This is a feature that was developed as a result of feedback from users with respect to streamlining the assessment process. The initial position of GRiST was to provide previous answers only for reference. That is, the repeat assessment’s answer controls would not be pre-populated with the previous answers. This would guard against the possibility of the clinician overlooking questions that had already been pre-populated, and whose applicability to the current assessment may no longer be guaranteed. However, what was apparent in practice was that clinicians were spending vast amounts of time re-keying information during repeat assessments which had essentially remained unchanged.

As a compromise to the opposing requirements of maintaining repeat assessment data accuracy and reducing re-keying, two-tiered answer persistence is introduced—*hard persistence* and *soft persistence*. Questions whose answers are deemed to seldom change, e.g., historical information such as *date of birth* or *number of dependents*, are designated as *hard persistent*. These answers are automatically pre-populated on repeat assessment, and flagged as such with a gold padlock icon. Questions whose answers are deemed to be relatively stable are designated as *soft persistent*. These are also pre-populated but are flagged using a silver padlock. Furthermore, clinicians are instructed to verify silver padlocked answers on repeat assessment to ensure they

remain applicable to the current assessment. All other questions operate as per the original specification, and do not have their answers pre-populated.

The above persistence mechanism is implemented within the trees of the GTH as a

```
persistent="hard|soft"
```

attribute against the designated nodes in the ST. These then filter down through the RIT into the QT, where they can be used by the tool generation engine.

The current persistence mechanism reduces the number of questions that are not pre-populated, yet helps to guard against the introduction of stale assessment data. As yet unimplemented extensions to the persistence mechanism have been proposed in order to allow a greater number of questions to be designated as persistent, and streamline repeat assessments further. One such example is that of designating only certain answer values of a (previously non-persistent) question as being persistent. This could apply in cases where once an event has occurred, it cannot regress from the occurred state, thereby becoming persistent.

HTML GRiST Conclusions

HTML GRiST demonstrates how the GTH trees can successfully be used to create a computerised assessment tool. Specifically, assessment trees are used to drive a tool generation engine developed using XSLT and PHP. The engine uses the knowledge structure and node specifications contained within the assessment trees to generate an appropriate tool. With the tool's being a thin-client, it does not incorporate the Galatean classifier, and therefore does not on its own, realise the full potential of GRiST. What the tool does is provide real-time calculation and visualisation of MGs for leaf nodes. This on its own is a compelling reason to forego/upgrade from a paper-based assessment tool as it allows clinicians to see at a glance, hot spots requiring attention. The other compelling reason is that reports on the assessment can be generated and printed off as and when needed. These reports can focus on different dimensions of the assessment. Aspects to the assessment can be extended as and when the need arises due to the extensible nature of the XML file formats and the ease with which XSLT can perform filtering of them. Reporting capabilities are further discussed in Section 9.6.

The ultimate aim for the PHP tool is for it to be used as a front-end to the Galatean classification engine once RIs have been finalised. The engine can perform classification and the results subsequently incorporated within reports/sent back to the PHP tool for appropriate display.

9.5.3 Java-based fat-client GRiST

The specification of IT hardware is increasing to the point where a modern desktop PC is effectively idling under normal use. Furthermore, as support for the legacy Windows XP and IE6 browser is withdrawn by Microsoft, the next upgrade cycle will bring with it faster computers, better browsers and a ubiquity of standard browser plugins such as Java, Flash, Silverlight, and AIR. This upgrade in IT infrastructure will also to some extent, be fuelled by the delivery of NPfIT across the NHS—A national IT strategy for the NHS aimed at connecting all GPs and hospitals, and the maintenance of electronic patient care records (Department of Health, 2002; Hendy, Reeves, Fulop, Hutchings, & Masseria, 2005).

Concomitant to the advances in IT, Trusts and end-users will also wish to utilise the full power of GRiST. This will entail dynamic, and interactive evaluations of risk being presented to the user as the assessment progresses.

Non-PhD work has involved the creation of a fat-client to provide the upgrade step to Trusts once they are fully accustomed to HTML GRiST. The fat-client is developed using Java and unlike the HTML-based tool, incorporates a Galatean classifier within. A screenshot of the Java tool is presented in Figure 9.14.

In common with the other versions of GRiST, the Java tool is driven by the trees of the GTH, and launched in a manner similar to the HTML tool. In fact, both the Java tool and the HTML tool share a common abstracted data pathway within the system, which was developed to service electronic versions of GRiST.

On starting up of an assessment using the Java tool, rapid screening questions (refer to Section 6.4.2) are determined and displayed for answering by the clinician. The clinician can elect to answer these questions or navigate to the full set of questions.

The Java tool reintroduces the tree structure as a method of orienting the user as to their location within the assessment. The node `label` hierarchy within the CAT is used to enable the clinician to rapidly traverse the tree in order to answer appropriate questions. This can be contrasted with the HTML tool, which requires the user to consider filter questions. Common to both tools however, is the search feature. This enables any node to be readily located within the context of its environs.

Whereas the HTML tool will rely on a server-side version of the Galatean classification engine, the Java tool will make use of the built-in classifier. This means that risk levels can be calculated and displayed directly on the rendition of the CAT. Colour-coding the percolation of

The screenshot shows the GRIST - Java Edition interface. At the top, it displays 'GRIST - Java Edition - Version: 4 Update: 4'. Below this, patient information is shown: 'Patient ID: 178', 'Assessment Type: New', and 'Population Used: working-age'.

On the left, a tree view shows the assessment structure under 'mental-health-risk'. The 'suicide' folder is expanded, and 'past and current suicide attempts' is selected.

The main area contains 'Questions related to past and current suicide attempts':

- Question 1: 'Has the person ever made a suicide attempt? If yes, the questions about them should be answered with reference to the attempts in general rather than any specific one, unless otherwise stated.' with radio buttons for yes, no, and DK.
- Question 2: 'When was the last suicide attempt?' with dropdowns for Year (2009), Month (Sep), and Day (06), and a radio button for DK.
- Question 3: 'Has there been more than one suicide attempt?' with radio buttons for yes, no, and DK.
- Question 4: 'When was the first suicide attempt?' with dropdowns for Year (2006), Month, and Day, and a radio button for DK.
- Question 5: 'Approximately how many suicide attempts have there been?' with a text input field containing '6', an 'Enter' button, and a radio button for DK.
- Question 6: 'How have the suicide attempts been changing in frequency over the last two years?' with radio buttons for decreasing, increasing, same, and DK.

At the bottom, there is a navigation bar with buttons for Search, Preferences, Full Screen, Risk Judgement, Save Answers, and Finish.

Figure 9.14: A screenshot of the Java tool being used to conduct a new assessment of a patient.

risk along the tree will complement the already present leaf node colouration to provide a full classification of the degree of risk. This feature will be operable once RIs have been finalised.

9.6 Reporting in GRiST

The assessment trees of the GTH, once instantiated, can be used to drive reports based on the assessment. Figure 9.9 demonstrates the process by which assessment trees can be used to generate a generic report. The reporting engine transforms the CAT and AT into a web page that when rendered, represents the report. In the main, the reporting engine is agnostic of the version of GRiST that has been used to generate it (HTML/Java), and of the Level of the tool. This is borne out of the fact that the report is effectively built on top of the trees of the GTH. This has the advantage of reduced development effort, and of a uniform reporting experience from a user perspective.

9.6.1 Client answers report

Figure 9.15 presents a screenshot of the primary report that has been developed to run on top of the GTH. It corresponds to the hypothetical assessment depicted in Figure 9.11. In common with the HTML tool, XSLT is guided by the CAT for the structure of the overall report. Filtering and transformation of data into a suitable representation for reporting is also performed in parallel using XSLT and PHP.

The screenshot shows a web browser window titled 'User Comments - Mozilla Firefox'. The address bar shows the URL: <https://www.grist.galassify.org/panel/mhexperts/mh-demo-dss-portal/reports/client-answers/client-answers.php?patient>. The report content is as follows:

Partner sharing: **Yes**
 Number of non-dependents sharing accommodation: **One**
 Ethnicity: **White irish**

RISK	SUMMARY	COMMENTS
Suicide	Risk judged to be 4 (Medium risk)	<i>This is an example comment</i>
Self-harm	Risk judged to be 6 (Medium risk)	
Harm to others or damage to property	Risk judged to be 1 (Very low risk)	
Self neglect	Risk judged to be 5 (Medium risk)	
Risk to dependents	Risk judged to be 3 (Low risk)	
Vulnerability of service user	Risk judged to be 10 (Max risk)	
Information relating to all risks		

Key:
 service-user-supplied data is presented in bold italics and colour coded by risk level
 supplied comments appear in grey italics immediately underneath

Elapsed time is calculated in relation to assessment date, e.g., **25 days prior** implies 25 days prior to assessment date of 12/06/2010

A PDF version of this report (for colour printing) is available [here](#).
 A PDF version of this report (for monochrome printing) is available [here](#).

Mental health risk

Each risk heading presented below can be clicked on to collapse or expand its underlying information. Please note that some users may need to "allow blocked content" if presented with a yellow security bar at the top of the page.

suicide
 Risk judged to be 4 (**Medium risk**)
This is an example comment

Past and current suicide attempts: **Yes**
 Most recent suicide attempt: **9.3 months prior (6 Sep 2009)**
 Pattern of suicide attempts: **Yes**
 First time suicide attempt occurred: **3 years prior**
 How many suicide attempts: **Six**
 Suicide attempts escalating in frequency: **Increasing**
 How much planning was generally involved in the suicide attempts: **Very low risk**
 Family history of suicide: **Yes**

Figure 9.15: An example report generated from a hypothetical patient assessment. Answer values are automatically formatted by the report to be more informative. Colour-coding according to value-mg profiles associated with each question are calculated and applied.

In common with the HTML and Java tools, the report maintains a hierarchy of nodes to provide context. Unlike the HTML tool, the report does not reiterate each question, but rather,

refers to each node using its `label` text. This provides a compact representation of the node. Similarly, the display of answers is simplified as answer controls are no longer required.

Answers are formatted to make them more informative and human readable. This includes calculating elapsed times from raw dates, and formatting according to the magnitude of the duration and the specific data type of the date. Presently, colours are applied to leaf nodes according to calculated mgs in exactly the same manner as within the tools. A future incarnation of the report will augment this capability to include the display of mgs that have been calculated and instantiated in the CAT by the classifier. This will then enable the report to indicate the Galatean model's calculation of the patient's risk.

The difference in fidelity between screen and print versions of a report, particularly with respect to colour, mean that PDF versions are also required. Furthermore, this gives the clinician the option of maintaining local copies of patient assessments. The open source *wkhtmltopdf* utility⁶, which relies on the *webkit* browser engine⁷, was used to generate PDFs. This represented a considerable saving of programming effort over e.g., using XSL-FO, since the program effectively rendered the HTML version of the report on the server and then “printed” the rendition to PDF. The dynamically generated PDF is obtainable via a link from the web version of the report (see Figure 9.15).

Also available is a black and white version of the report. This version substitutes colours for font sizes and effects amenable to being represented on monochrome printers. The separation of styling information from content by appropriate use of CSS within the report generation engine meant that the transformation was achieved by simply linking to a different CSS stylesheet.

9.6.2 The GTH and specialised reports

The XML-based nature of the assessment trees lends them to augmentation with new capabilities merely through the addition of new attributes. In a project where it is not known *a priori* exactly how end-users will wish to use the system, or where new possibilities emerge *post hoc*, it is important to leave scope for expansion. Section 9.3.3 describes one such scenario involving the `management` attribute of the AT.

Incorporating risk management information within the AT via the `management` attribute means that this information can now be processed (or not) using existing reporting infrastructure. It also makes conceivable the building of various reporting ‘modes’ within the reporting

⁶<http://code.google.com/p/wkhtmltopdf/>

⁷<http://webkit.org/>

engine. These modes could correspond to specialised reports; each focussing on a different aspect of the assessment data.

The choice of XSLT as an integral component of the reporting engine again complements the use of an expandable set of attributes for recording data. XSLT stylesheets can easily be created to include or exclude information from the GTH assessment trees when generating reports.

Such a facility opens up the possibility of e.g., layered workflows, where different members of staff can fill in different segments of the assessment. Experts at each stage of the assessment could access combined or specialised reports based on data that is already present and use these to guide the input of data for the next stage.

Advanced auditing capabilities also become a possibility. These would allow every change to be owner and timestamped. Log data could be mined to see how assessments are being conducted and to inform e.g., changes to the assessment structure to make it more efficient.

Database systems such as Oracle's *Database 11*, the Apache Foundation's *Xindice* and to some extent, MySQL 6 enable users to perform sophisticated querying over XML files (Loney, 2009; Sarang, 2006; Vaswani, 2009, respectively). This opens up a new dimension to assessment reporting, as reports of the patient can for example, incorporate a temporal element. Where current reports are based on the present assessment, more advanced reporting could use the assessment trees corresponding to all of a patient's previous assessments and chart progress via graphs etc.

The potential for XML-based GTH assessment trees combined with XSLT-based reporting engines and the right database tools to improve mental-health assessment is huge. And the reporting features implemented thus far represent a significant first step in this direction.

9.7 Conclusions

The work described in this chapter has focussed on extending the Galatean Tree Hierarchy to the point where it is capable of efficiently driving risk assessments. This has meant the introduction of the CAT and QT. These assessment trees structure and specify the assessment in a generic fashion, which can then be realised via the assessment generation engine. The notion of an AT is also introduced to store answers and other data supplied as part of the assessment.

With the incorporation of the CAT and QT into the GTH, a clear and systematic path from the initial domain model through to the specification of an assessment is encapsulated by the trees. At the heart is the ST, whose pure form is enhanced using attributes that ultimately

drive the decision support system. Through successive automated transformations using XSLT and enrichment with uncertainty information at appropriate stages, the domain model becomes capable of driving the assessment.

The assessment trees have been used as a platform for generating multiple versions of GRiST. The chapter has demonstrated three such versions, each developed with different use-cases in mind and varying in technical capability.

The paper tool, intended to be used within existing paper-based workflows is a comprehensive data gathering tool. It aims to capture all the data pertinent to the risk assessment, yet represents a low-tech solution to risk assessment as it does not result in any automated determination of risk.

An upgrade path is provided from the paper tool by way of the HTML version of GRiST. This is an electronic data gathering/assessment tool, which consequently, has the built in capability to provide calculations of risk for individual questions. This tool will ultimately act as a front-end to the Galatean classifier on the server, which will use assessment data to perform a full classification.

The third version of GRiST, the Java tool, extends the data gathering and assessment capabilities of HTML GRiST further by incorporating the classifier within the client. Once RIs have been finalised, this tool will have the ability to perform risk quantifications dynamically as the assessment progresses.

All three tools share the same underlying heritage by virtue of the GTH. The assessment trees of the GTH also drive reporting functionality. This means interchanging of assessments between say the HTML tool and Java tool is theoretically possible, and reports are uniform.

What the XML-based GTH in combination with the transformational capability of XSLT and PHP provides, is a tailored assessment tool, customisable to experts at different levels of experience, with a choice of interface. This assessment tool is always current, because updating of the underlying domain knowledge filters down to the tool via the GTH. The tool does not need to be re-programmed since it is driven by the trees. Furthermore, the extensibility of the underlying XML lends itself to exploitation by the easy facilitation of new features through additional attributes. Examples such as answer persistence and the recording of management information have been realised via this method. These qualities on their own represent a powerful and organic system for risk assessment in heterogenous environments.

Coupled with the Galatean classification engine, the tool is imbued with the ability to provide

calculations of risk, reported in real time and available as a web resource without geographical or temporal impedance. This increases the utility of GRiST even further, making it into a full decision support system that can be used by clinicians and members of the public alike.

The extensibility of XML and the flexibility that XSL transformations can bring to GRiST are exploited further in the next chapter to bring even greater customisation and tailoring to the user. Specifically, via the introduction of user *populations*, each of which can be given their own unique version of GRiST, and each of which, is coordinated via the GTH.

10

Further Customisation of GRiST for Different Populations and Contexts

10.1 Introduction

Chapter 6 introduced a powerful mechanism with which, the GRiST assessment could be shortened for more experienced users. The Levels feature of Section 6.6 enables soft tree-pruning points to be programmed into the root knowledge structure driving GRiST, i.e., the ST. The hierarchical organisation of concepts within the ST means that a judgment elicited directly at a pruning point subsumes any underlying questions that would otherwise needed to have been considered individually. In this way, the experience of clinicians (codified by their designated Level) can directly translate to more vigorous or deeper cuts in the CAT, and therefore, the question set.

This chapter aims to complement the ‘vertical’ customisation of GRiST that is afforded by Levels by introducing a principled method for ‘lateral’ customisation. In its simplest form, lateral customisation may involve the renaming of labels and question phraseology to suit particular

organisational nomenclature. In its ultimate form, it could involve substantial re-specification of the ST so that it is appropriate to the assessment of e.g., different user groups, specialities and contexts.

In order to cater for lateral customisation, the GTH will be augmented with a new tree: the Super Structure Tree (SST). This tree, and how it facilitates lateral customisation objectives will be the main focus of the chapter.

The second half of the chapter will explore a computer-assisted tree management system developed to support the SST and the trees of the GTH. This system simplifies upload, generation, and maintenance of GTH trees. Additionally, it provides facilities for troubleshooting trees and for verification of their correctness.

10.2 The Need for Lateral Customisation

The Levels functionality offered in the ST means that clinicians or Trusts have a degree of choice with respect to the length of the tool¹ that is to be used. However, as far as the structuring and coverage of GRiST are concerned, there is as of yet, little choice for organisations, clinicians, or patients. Essentially, what the preliminary GRiST offers is the same tool in varying sizes.

10.2.1 The organisational perspective

A common reason identified for lukewarm user-reception of a new system is the discrepancy between its conceptualisation of the problem domain and the actual domain as viewed by users (Rouse & Morris, 1986); e.g., Bartis and Mitev (2008); J. Scott, Rundall, Vogt, and Hsu (2005). It is precisely for this reason that many a system has failed in its efforts; meaning at best, considerable re-work (Fitzgerald & Russo, 2005), and at worst, embarrassing white elephants (Hougham, 1996; Rogers & Mead, 2004; Gauld, 2007). Decision support systems are by no means immune to this phenomenon, and several examples exist where mismatches with user requirements have resulted in difficulties with user acceptance (Anumba, Dainty, Ison, & Sergeant, 2006; Dowding et al., 2009; Stewart, Lawrence, & Edwards, 2010).

With the ambitious research aim of GRiST to be pervasive across a number of assessment settings, it is especially important to consider mismatches between GRiST and those settings, and methods for their amelioration. GRiST could be used in settings ranging from NHS Trusts

¹Tool in this sense refers to the structure of the GRiST ontology and associated questions, and does not mean the GRiST software architectures.

to police stations or prisons. Each type of setting brings with it its own unique requirements and challenges. For example, the prison service may be familiar with asking questions in a particular format, or indeed particular questions that are not as readily useful in say, an NHS hospital. This may mean that the ‘default’ GRiST would be perceived as an awkward or cumbersome tool to be used within a prison, whereas it may be better received in a hospital.

Compounded to the differing needs of diverse services, is the fragmented nature of assessment practices within settings of the same type (Higgins et al., 2005). For example, different NHS Trusts may each have their own processes and tools for the assessment of patients (Hawley et al., 2006).

Both the above organisational issues mean that the potential for mismatches in expectations/performance of the tool is augmented, particularly for mental-health assessment.

10.2.2 The clinical perspective

The initial version of GRiST is generally used for assessing adults of working age, and consequently, is fine for the assessment of the majority of people. It is however, acknowledged that not all members of the general population fall into the working-age category. Nevertheless, the underlying domain knowledge of GRiST is comprehensive enough to be relevant in the most part, to the general population. Although, clearly, a better strategy for making GRiST more appropriate to the diverse spectrum of patients is to stratify the general population into groups.

Identifying different sub-populations can pave the way for modifications to the GRiST knowledge structure such that the assessment of patients of those sub-groups can be more clinically relevant. For example, consider a ‘children or adolescents’ population group. Some of the concepts relevant to the adult population, e.g., those in relation to dependents, will not apply to children. By the same token, other concepts not relevant to the working-age population may benefit the child-adolescent group by their introduction.

The ability to make GRiST assessments more relevant to the clinical context would increase the quality of the assessment of the patient over using the ‘default’ version of GRiST. The tailoring of GRiST in this way would also narrow the gap between the clinician’s expectations of GRiST and its offerings. This flexibility of GRiST would therefore increase its acceptance as an ‘all-weather’ mental-health tool.

10.2.3 The patient perspective

The traditional view of the mental-health assessment is one where the administering is performed by a mental-health professional or other front-line staff. GRiST can of course, readily be used within this assessor-patient dynamic.

The decomposition of GRiST's knowledge structures into elemental patient cues (i.e., those of the Level 0 tool) does also imply that someone without any mental-health training can make effective use of GRiST. Coupled with GRiST's being a web-based resource, what this means for the patient is that GRiST can be used in a self-service manner—the service-user could potentially assess themselves. This has indeed been one of the research goals of the project from the outset.

Opening up GRiST to the service-user would invariably necessitate bespoke tailoring work to the default structure of the knowledge and questions. Although the datum items are low-level enough to be collected by members of the general public, GRiST's terminology is pitched towards those users that are more routinely exposed to mental-health assessment than the layperson. Furthermore, questions are framed in the third person perspective, which would need to be adjusted to cater for the second person assessment scenario.

Reporting facilities targeted at the layperson also require consideration. However, given that GTH trees drive reports as well as the tools, tree customisation work would automatically benefit reporting aimed at the service-user.

10.3 Rationale for a Super Structure Tree

To meet the customisation demands of the various organisational, clinical and patient perspectives, it was decided to adopt a segmentation approach to the problem. This would involve identifying a core set of user-groups or organisation types, each of which would require modifications to their version of GRiST. Identifying a handful of 'populations' is a compromise position that helps manage complexity, whilst at the same time, guards against fragmentation of the GRiST tool due to over-flexible customisation.

Figure 9.7 depicts the current conceptualisation of the GTH. The ST effectively specifies a domain model that is appropriate for a single default population—working-age people. Given that the ST is specifying the population model, the focus of tree customisation activities should therefore, logically be the ST.

The most straightforward way to customise the ST is to have a version of the ST for each

population group. This has the benefit that no remedial work is required of the tools as no new constructs are introduced into the assessment trees. Although, from the point of view of auditing changes and maintenance of the various STs, the solution is a short-sighted one. Changes between the STs and the default working-age ST would be difficult to track, and evolution of the working-age ST would need to be replicated across the trees. In short, it would pose an unacceptable administrative burden with the maintenance of the system of STs.

To address these two issues, what was instead proposed was the introduction of a new tree that would sit at the root of the GTH. The Super Structure Tree (SST) would serve as a meta tree containing the default population model in addition to customisation information for each and every other population. XSLT would then be in a position to be employed to generate STs corresponding to each of the populations that have been defined.

Using the SST and XSLT route to generating bespoke STs, the following benefits are realised:

1. An audit trail of customisations is maintained via their specification in the SST.
2. Changes to the default population model are immediately reflected in models for the other non-default populations. This is because XSLT will use the default population model that forms the foundations of the SST, and apply the population-specific customisations to that model in order to arrive at the population-specific model.
3. No changes to tools are required since the syntax of the generated ST and derivative trees remains the same. The SST is simply another tree in the GTH stack, of which, the tools can be agnostic.

10.4 Super Structure Tree Syntax

Recall that the current ST effectively describes the model for the working-age ‘population’. This is the default population model for GRiST and should therefore serve as the basis for the SST. Put another way, the SST, when containing definitions for only one population group, i.e., working-age, is equivalent to the current conception of the ST.

The remainder of this section introduces the modifications and additions that are required to the SST over and beyond those inherited from the original ST. Once the SST has been augmented with the new constructs, it can incorporate customisation information for multiple populations.

10.4.1 The populations attribute

The SST contains a `populations` attribute in the root node of the tree. This contains within parentheses, a list of population groups for which the SST contains information. The list should contain *all* the known population groups (since the SST will be used to generate the corresponding STs), e.g.,

```
populations="(working-age child-adolescent older military service-user)"
```

Furthermore, implied in the definition of a `populations` attribute of the SST is the directive that the *first* element of the list contained within `populations` will contain the name of the population that is to be considered the default population.

10.4.2 The enhanced layer and order attributes

The `layer` attribute of the SST is one of the mechanisms involved in driving ST customisation for different populations. In addition to the standard form of `layer` attributes (i.e., a numerical value that is applicable to *all* population types), which may be present in the SST, there is also an enhanced form (specifically for the SST). The enhanced form uses the syntax of an association list e.g.,:

```
layer="((working-age 0)(child-adolescent 0)(older 1))"
```

This is an access control list (ACL) denoting all the population types that the `layer` attribute *applies and only applies to*. The first element of each sublist is the population name. The second element is an integer. The integers have the same meaning as that which has been established for standard form `layer` attributes (refer to Section 6.4.2), but will only apply to the corresponding organisation type.

The use of ACLs is common in computer filesystems where different users and groups are to be assigned different permissions to a filesystem object (Sandhu & Samarati, 2002). Typically, the ACL specifies an *object*, a *subject* and an *operation*. The enhanced form of the `layer` attribute implements the ACL as a LISP association list, with *subject* being the population and *operation* relating to the original definition for `layer` attribute values. The *object* is indicated by the fact that the attribute is located in situ, against the node under consideration. This makes the syntax very simple relative to more heavyweight approaches, which can separate access control information to outside of the document and rely on XPath; e.g., Damiani, Vimercati, Paraboschi,

and Samarati (2000) and Fan, Chan, and Garofalakis (2004). Indeed, ACLs mirroring the LISP list formula are used extensively in the remaining SST constructs that are to be introduced.

It is recognised that different populations may require different question orderings. For example, an organisation may be more focussed on assessing *self-harm* as opposed to e.g., *suicide* risk, and therefore would wish to have *self-harm* questions at the forefront. Where the default structural ordering of sibling nodes implied by their relative positions is not desired, an `order` attribute is defined for use in the SST. The `order` attribute can be used to indicate ordering precedence. In the same manner as the `layer` attribute, the `order` attribute contains population ACLs, with integer values denoting ordering precedence relative to siblings. A value of 0 indicates highest precedence, with precedence decreasing as the value increases. Below is an example of the `order` attribute:

```
order="((working-age 1)(child-adolescent 0))"
```

10.4.3 The SST Enhanced question, filter-q, label and help attributes

The attributes whose values are under consideration in this section are: `question`, `filter-q`, `label` and `help`. For simplicity, these attributes will in the remainder of this section, be referred to as ‘question’, or as ‘question(s)’ where there is no noteworthy differentiation in their treatment.

Most questions’ values will be the same for all populations. Where *all* populations share the same question text for a given node, the already established definition of the question attribute is used. Earlier discussions (refer to Section 10.2.3) imply that some populations will benefit from revised questions. Where there is at least one population group whose text should differ, an *SST enhanced* definition applies for the given node. This makes all the relations between populations and question texts explicit within the node:

```
question="(((list of populations) &quot;Question string&quot;);
           ((list of populations) &quot;Alternative question string&quot;);
           ....
           )"
```

10.4.4 Inverting value-mgs

A given population’s tailored question might be phrased in such a way that it would necessitate the associated value-mg definition’s lateral inversion. For example:

- the MG associated with a value of 2 would instead be required to be associated with a value of 8 (assuming a 0-10 value scale);
- the `value-mg` profile, `((0 0)(4 0.2)(7 0.8)(10 1))` would be required to become `((0 1)(3 0.8)(6 0.2)(10 0))`.

Algorithmically, this lateral inversion (or reversing of polarity) is as follows:

1. Replace each value with its complementary value i.e.,

$$\text{value}_i \rightarrow (\text{value}_{\max} - \text{value}_i) + \text{value}_{\min}$$
2. To maintain convention, re-order all elements (i.e., `value-mg` pairs) in numerically ascending value order.^a

^aThis operation is a matter of reversing the position of the elements of the `value-mg` list.

A given SST node may have an `rp-for` attribute indicating the populations that are to have the reverse polarity operation performed on the associated `value-mg` attribute prior to its transmission onto the ST. The syntax for this attribute is as follows:

```
rp-for="(list of populations on which to perform rp operation)"
```

10.4.5 Population-specific pruning of nodes

Not all populations will require every branch of the SST for their version of the tool. To facilitate this, there is a mechanism by which a given node (and its descendants) can be marked-up as *not* being required for a target population. These branches would then be omitted from the target population's ST. This facility is implemented by a `prune-for` attribute, containing a list of populations for which the node and its descendants should be omitted, e.g.,

```
prune-for="(list of population groups for which to omit)"
```

The `prune-for` attribute is an example of a tree construct where it is important to incorporate appropriate validation within any process that performs tree transforms. In this specific instance, this is necessitated by virtue of the SST's being an unexpanded tree. Consequently, it may be that there are path references to the to-be-pruned node from other locations. Pruning a node that is/contains a node that is referenced by a node located outside of the to-be-pruned node's hierarchy will result in broken paths. Depending on the preference of the knowledge engineer, these should either be flagged up as part of the validation process, or a cascade delete

operation be effected as part of the transformation. The present implementation adopts the former option; leaving the latter as a future improvement to the system.

10.4.6 Adding additional nodes

The addition of new concepts and nodes specific to a population is not currently directly supported by way of an *add-for* construct. The main reason for this is that it could encourage the introduction of model data of questionable provenance. As has been demonstrated in earlier chapters, the knowledge contained in the default SST population, i.e., the working-age population, is the product of a rigorous elicitation process. This can be traced right back to the original expert panel members by following the audit trail. This data therefore represents the validated mental-health domain model and can thus be used for assessment and decision support.

In order to carry over the authenticity and validity of the working-age tool to GRiST tools tailored for other populations, it is important not to encourage the introduction of additional concepts to the core domain model. Nevertheless it is acknowledged that there may be some situations that dictate the introduction of new population-specific concepts.

The `prune-for` attribute can be repurposed to facilitate the addition of population-specific concepts by specifying all the population groups the concept should *not* appear in. That is to say, all other populations (including the working-age population) are listed within the `prune-for` attribute.

10.4.7 Adding new populations to the SST

The nature of the Access Control List approach is such that adding a new population to the SST is not simply a matter of amending the `populations` attribute in the root node of the tree. Reconfiguration work needs to be undertaken on many nodes with pre-existing ACLs to inform them of the new population. However, if the addition of a new population is first decomposed into a *clone population* operation and then one of tailoring the cloned population with specific customisations, then the reconfiguration work can be automated as part of the clone operation.

When adding a new population to the SST, the knowledge engineer is provided with an XSLT utility on the GRiST website to clone a population. The knowledge engineer is able to choose the pre-existing population to clone and the name of the newly cloned population. The XSLT utility handles updating of the `populations` attribute and all the affected ACLs in the tree. The resulting SST can then be worked on to customise the cloned population.

10.5 The SST and its Incorporation into the GTH

With the SST serving as the new master tree, which is ultimately used to drive assessments, it is appropriate to revisit the GTH in order to incorporate the new tree. Figure 10.1 depicts the updated GTH, with the SST now serving as the base tree. The SST is responsible for generating multiple STs, which in turn generate RITs and the assessment trees. In common with the rest of the derived trees, each population-specific ST is now generated using XSLT stylesheets. That is to say, the SST is pruned appropriately, and all population-specific lateral customisation operations applied.

The generated ST is exactly the same with respect to syntax as the original conception of the ST—save for the presence of a `populations` attribute at the root node indicating the population this ST serves. This means all processes utilising the ST and *ipso facto* the Galatean risk assessment tools, are left requiring no modifications.

The incorporation of the SST into the GTH serves to enrich the system with the capability to provide lateral customisation of assessments in addition to the already present vertical customisation capabilities. This renders the system exceptionally flexible with respect to meeting differing contextual demands.

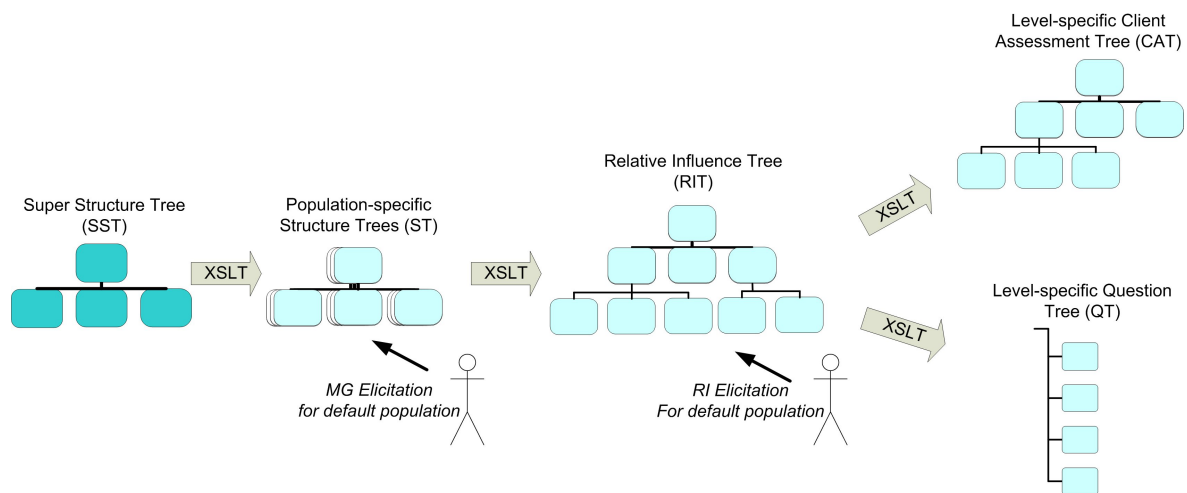


Figure 10.1: The Galatean Tree Hierarchy augmented with the prepending of the Super Structure Tree (SST). This tree contains population-specific customisation information, which is then transformed using XSLT into individually tailored STs. These are further transformed to ultimately generate the trees that drive assessment.

10.6 Fingerprint Reconciliation and the SST

Heretofore, the fingerprint reconciliation algorithms described in Chapter 8 have assumed that the base tree from which all trees are derived is the ST. It has been assumed that experts evolve the ST by way of direct edits upon it (see Figure 8.1), and these are propagated to the derived trees. This is in fact now a simplification, and it is actually the Super Structure Tree (SST) where direct tree edits are made. It thus follows that fingerprints that have been calculated and embedded in the ST should in fact be generated in the SST and propagated to the ST instead.

Unlike the transformation from ST to RIT, the transformation from an SST to an ST does not involve any tree expansion. Concepts are not instantiated, meaning that each necessarily unique node of the SST will still be a unique node of the ST. There will be a one-to-one relationship between ST and SST nodes. It is because of this equivalence that fingerprints originating in the SST can be treated as native ST fingerprints for the purposes of node tracking. Therefore, the amended fingerprinting scheme instead generates `fingerprint` and `fingerprint-orig` attributes in the SST, and these propagate to the ST. From here, nodes can be linked across STs as before, without further consideration of the SST.

It is interesting to note that depending on the sophistication of the translation directives from SST to ST, the propagated fingerprint may no longer be the same value as a fingerprint were it to be generated natively in the ST. That is to say, were a level of nodes to be removed as part of ST generation and all the children shifted up, their paths in the SST would be different to those in the ST. However, this does not cause a problem, as the value of the fingerprint *per se* is not relevant in this application—only that it is unique to the node in question.

10.7 Machinery for Generating and Organising STs and Derivative Trees

The SST was gradually augmented with lateral customisation information for a number of populations. XSLT stylesheets were developed to process the SST in order to arrive at STs corresponding to each defined population. These could then be processed as normal in order to arrive at the rest of the trees that constitute the GTH.

10.7.1 Computer-assisted tree management

The initial deployment of the assessment tools utilised a rather simplistic approach to tree transformation and management. There was only the working-age population, and consequently, trees were individually generated using the XSLT processor and stored in a database. All tools utilised the same set of derived assessment trees. Furthermore there was little validation of trees to ensure correctness.

With the introduction of the SST, it became apparent that the rudimentary approach to tree management was too inefficient. As the number of populations grew, there was a rapid proliferation of generated trees. Currently, there are seven populations that have been incorporated into the SST. Assuming three Levels of expertise, a total of 56 trees are therefore required to be derived as a consequence. Factoring into account development versions of SSTs, the total number of trees required to be generated can potentially be in the hundreds. Clearly there is a need for better infrastructure to support and manage tree-generation.

To meet this demand a facility was set up on the GRiST website to allow Administrator users to upload an SST directly to the website. Upon its upload, the system would read the populations listed within the SST and proceed to generate all the requisite derivable trees. A screenshot of an example derivation process is depicted in Figure 10.2.

All trees were generated via XSLT stylesheets being run on the server and co-ordinated using PHP. Databases were automatically reconfigured via the system to store all trees. Parallel to the trees' being stored in their entirety as binary data, trees were dynamically examined by the system. Database columns were created to store assessment data corresponding to each node. The idea was that assessment data would be available in its original XML-based tree form and also as normal table fields inside a tuple. This would open up the possibility of mining the data using the standard (non-XML) database machinery present in the RDBMS, or a combination of the two.

10.7.2 GTH Tree Validation

XML files, by their nature, bridge the gap between human readability and machine operability. This is no less of a truth for the trees of the GTH. Working with the SST using tools similar to the Flash tools developed during knowledge elicitation simplifies its editing. Nevertheless the SST is simple enough to be worked on directly using a text editor. The possibility of human error in these cases cannot be ignored. This is especially true as nodes in one location may not

10.7. MACHINERY FOR GENERATING AND ORGANISING STS AND DERIVATIVE TREES

SST Name	SST Description	Created
sst-17-11-10	tree with new labels for service users	2010-11-17

SST

Population Name	ST (PHP tool)	RIT	CAT (java tool)	QT
working-age	ST-working-age	RIT-working-age	CAT-L0-working-age CAT-L1-working-age CAT-L2-working-age	QT-L0-working-age QT-L1-working-age QT-L2-working-age
child-adolescent	ST-child-adolescent	RIT-child-adolescent	CAT-L0-child-adolescent CAT-L1-child-adolescent CAT-L2-child-adolescent	QT-L0-child-adolescent QT-L1-child-adolescent QT-L2-child-adolescent
older	ST-older	RIT-older	CAT-L0-older CAT-L1-older CAT-L2-older	QT-L0-older QT-L1-older QT-L2-older
military	ST-military	RIT-military	CAT-L0-military CAT-L1-military CAT-L2-military	QT-L0-military QT-L1-military QT-L2-military
oats	ST-oats	RIT-oats	CAT-L0-oats CAT-L1-oats CAT-L2-oats	QT-L0-oats QT-L1-oats QT-L2-oats
service-user	ST-service-user	RIT-service-user	CAT-L0-service-user CAT-L1-service-user CAT-L2-service-user	QT-L0-service-user QT-L1-service-user QT-L2-service-user
iapt	ST-iapt	RIT-iapt	CAT-L0-iapt CAT-L1-iapt CAT-L2-iapt	QT-L0-iapt QT-L1-iapt QT-L2-iapt

Figure 10.2: Automatically generating GTH trees from an uploaded SST. The SST contains lateral customisation data for seven populations and three Levels of vertical customisation. This yields 21 sets of assessment trees (CATs & QTs) and 56 trees in total. Grey buttons launch the respective tree for viewing. Yellow triangle icons, which contain exclamations, indicate an internal inconsistency identified within the tree. These can be clicked on to further investigate. Blue arrow icons provide the facility to upload a revised tree.

necessarily be independent of nodes in other locations: dependencies can be overlooked.

An important feature that was incorporated into the new computer-assisted tree management system was that of tree correctness and internal consistency checking. This included:

- Legality checks on the combination of attributes present in the various trees.
- Verification of paths referencing internal tree locations.
- Ensuring pre-requisites to certain node configurations were satisfied; e.g., one such constraint was that a node designated as `persistent` must also have its ancestral questions designated as such.

Many of these features were free in terms of implementation effort. This was because the tests

10.7. MACHINERY FOR GENERATING AND ORGANISING STS AND DERIVATIVE TREES

were essentially being carried out anyway as part of tree generation. What the tree management system provided was clear and cogent reporting on the problems that were identified. Reports were accessible via the yellow exclamation icon that was presented against erroneous trees. An example report is presented in Figure 10.3. The report uses error data embedded inside `error` attributes created as and when needed as part of the tree generation process. Therefore, XSLT again plays a prominent part in both identification and rendition of errors.



Figure 10.3: An example error report associated with a generated GTH tree.

The decision to use XSLT for tree validation activities represents a pragmatic and flexible approach to realising correct trees. The existing tree transformation machinery is re-purposed, or produces as a by-product, validation information. This could be contrasted with say, using the tools associated with XML Schema Definition Language² or RelaxNG³. Such tools would require the the trees' precise specification in these languages for the ostensible benefit of running the trees through a validator (*without* obviating the need to build tools to generate the derivative trees). Furthermore, facilities such as transforming human-friendly node-paths into equivalent XPath statements and dynamically executing them to test validity are beyond the capabilities of XML Schema Definition. Therefore, the schema approach would nevertheless have necessitated hybridisation with XSLT to effect a full validation.

²<http://www.w3.org/TR/xmlschema-0/>

³<http://www.relaxng.org/>

10.7. MACHINERY FOR GENERATING AND ORGANISING STS AND DERIVATIVE TREES

Once the GTH tree specifications have reached stability, in the interests of wider adoption and interoperability, it would then be prudent to re-examine the benefits of using a formal markup to define the syntax of the trees. These would complement the more human-readable specifications that currently document the GTH file formats and semantics.

10.7.3 Incorporating amended trees back into the GTH

Recall that although the RIT is a generated tree, its purpose is to enable the incorporation of Relative Influence values into the domain model. Consequently, the RIT requires instantiation and hence, updating post-tree-generation. The tree management system facilitates the updating of individual trees (refer to Figure 10.4) within the GTH. Once an updated tree is uploaded, the system will automatically regenerate downstream trees if the Administrator has indicated this as the mode of operation. Associated database housekeeping is also effected.

You have chosen to upload a revised *RIT-child-adolescent* tree. Please supply your revised file.

Regenerate trees derived from the tree that is being replaced?

SST Name	SST Description	Created
sst-17-11-10 <input type="checkbox"/>	tree with new labels for service users <input type="checkbox"/>	2010-11-17 <input type="checkbox"/>

Population Name	ST (PHP tool)	RIT	CAT (java tool)	QT
working-age	<input type="button" value="ST-working-age"/> <input type="button" value="⊕"/>	<input type="button" value="RIT-working-age"/> <input type="button" value="⊕"/>	<input type="button" value="CAT-L0-working-age"/> <input type="button" value="⊕"/> <input type="button" value="CAT-L1-working-age"/> <input type="button" value="⊕"/> <input type="button" value="CAT-L2-working-age"/> <input type="button" value="⊕"/>	<input type="button" value="QT-L0-working-age"/> <input type="button" value="⊕"/> <input type="button" value="QT-L1-working-age"/> <input type="button" value="⊕"/> <input type="button" value="QT-L2-working-age"/> <input type="button" value="⊕"/>
child-adolescent	<input type="button" value="ST-child-adolescent"/> <input type="button" value="⊕"/>	<input type="button" value="RIT-child-adolescent"/> <input type="button" value="⊕"/>	<input type="button" value="CAT-L0-child-adolescent"/> <input type="button" value="⊕"/> <input type="button" value="CAT-L1-child-adolescent"/> <input type="button" value="⊕"/> <input type="button" value="CAT-L2-child-adolescent"/> <input type="button" value="⊕"/>	<input type="button" value="QT-L0-child-adolescent"/> <input type="button" value="⊕"/> <input type="button" value="QT-L1-child-adolescent"/> <input type="button" value="⊕"/> <input type="button" value="QT-L2-child-adolescent"/> <input type="button" value="⊕"/>
older	<input type="button" value="ST-older"/> <input type="button" value="⊕"/>	<input type="button" value="RIT-older"/> <input type="button" value="⊕"/>	<input type="button" value="CAT-L0-older"/> <input type="button" value="⊕"/> <input type="button" value="CAT-L1-older"/> <input type="button" value="⊕"/> <input type="button" value="CAT-L2-older"/> <input type="button" value="⊕"/>	<input type="button" value="QT-L0-older"/> <input type="button" value="⊕"/> <input type="button" value="QT-L1-older"/> <input type="button" value="⊕"/> <input type="button" value="QT-L2-older"/> <input type="button" value="⊕"/>

Figure 10.4: Uploading a revised RIT. Affected derived trees are highlighted, and the option is provided to effect their regeneration upon upload of the new RIT.

To provide extra flexibility, the selective tree updating facility is available for any of the trees of the GTH. Although, uploading customised versions of derived trees is discouraged, due to the possibility of GTH trees then becoming inconsistent relative to each other. However, in a real-world system there may be occasions when a rapidly developed temporary solution to

a problem necessitates modifications to an individual tree. It is envisaged that when a more stable solution to a problem is developed, appropriate modifications are made to the SST and this is uploaded instead. Derivative trees can then be generated afresh and the modifications propagated down.

A limitation with the current tree management system is that downstream trees are regenerated without consideration of the data in the previous versions of those trees. This raises issues with data migration where new information has been introduced into a derivative tree, and that derivative tree is later required to be regenerated—as in the case of the RIT. Chapter 8 and Section 10.6 develop a method by which an SST can be revised without the need to re-instantiate unaffected RIs. Once this method has been incorporated into the tree management system, revised SSTs can be uploaded, with minimal rework required to the newly generated RIT.

10.8 Conclusions

For GRiST to be successfully deployed across diverse clinical contexts, disciplines, and client populations the tool needs to be malleable to their individual requirements. This has to be achieved without compromising the validity of the GRiST assessment. It also has to be achieved in a controlled and principled manner so that individual requirements can be serviced both in the short term and strategically as GRiST grows.

Vertical customisation of GRiST, mainly for the benefit of assessors, was introduced in Chapter 6 by way of the Levels mechanism. The present chapter has greatly enhanced customisability by supplementing vertical customisability with lateral tailoring capabilities. This has involved augmenting the GTH with the Super Structure Tree (SST)—a meta tree for describing STs.

The SST enables lateral customisation by encapsulating the default GRiST model data together with tree modifiers that are to be applied to each “population”. XSLT can then be used to generate tailored STs, which ultimately power the GRiST tools via the GTH.

Supporting the SST and the GTH is a computer-assisted tree management solution. This has the capability to automatically generate all requisite population trees and associated database infrastructure for the conducting of assessments. It has tree validation features and associated error reporting capabilities, which help to assist with tree maintenance. All of this facility is provided via the website without the user requiring any knowledge of underlying technologies, thereby making the system easy to administer.

The use of one SST to generate custom tools for populations means both the clinicians and patients are better served by GRiST, without compromising the validity of the assessment—each customised tool is based on the same underlying knowledge, which has been rigourously validated by experts. Furthermore, as the GRiST knowledge is refined, changes are propagated to each tailored version, keeping them current.

In addition to serving different populations, the GTH architecture of GRiST can also help break barriers between them. An assessment performed by a clinician (using terminology appropriate to that clinician) can easily be made intelligible to an assessor with a different background or the patient themselves. That is, assessment data from one population can be mapped onto another population. In many cases, this can be achieved simply by loading the assessment data (from an AT) and presenting it against the backdrop of the population tree that is most appropriate to the user. In other words, the ST driving assessment data-gathering can be decoupled from the ST driving reporting.

Adding in the potential for multi-layered workflows and reporting, and a choice of tool for conducting the assessment, together with decision support capabilities, GRiST represents a complete assessment solution for mental-health, irrespective of context.

The next chapter presents a case study of the successful deployment and utilisation of the GRiST solution within two NHS Trusts.

11

Full Deployment within NHS Trusts: A Case Study

11.1 Introduction

Chapter 10 developed more flexible knowledge customisation facilities, together with a tree management system to support assessments based on tailored trees. The present chapter demonstrates how these facilities were utilised to deploy the GRiST system within two NHS Trusts.

The chapter initially considers the governance and technological issues surrounding implementation of a new system within NHS organisations. These considerations are then used to develop a generic specification and API, which Trusts can use to connect to GRiST and conduct or manage assessments. Finally, the impact of the deployed system is evaluated with respect to its utilisation and efficacy.

11.2 Introducing Participating NHS Trusts

At time of writing, GRiST has been rolled out and integrated within two large NHS Trusts. These two Trusts therefore represent an ideal case study for exploring deployment of GRiST

within NHS-type organisations. This section briefly introduces the two Trusts together with any technical detail relevant to a GRiST deployment. The Trusts are given fictitious names to preserve their identity.

11.2.1 Holbrook NHS Foundation Trust

The Holbrook NHS Foundation Trust serves a population of 600,000. It comprises four hospitals, three of which provide mental-health services. These are complemented by a number of smaller teams working within the community.

Holbrook utilises iSOFT's *i.Patient Manager* (iPM)¹ Patient Administration System (PAS) to manage patient information. iPM is a shared electronic record that maintains information on the patient and their progress through the system from point of contact to discharge (Brennan, 2007).

iPM is proprietary in nature, and thus, does not allow external programs to modify or write to its database tables. The Trust does however, have independent access to the data contained within iPM's database via daily backup dumps. These dumps can be accessed directly and used as 'feeder' databases for databases of other systems the Trust may have.

Initially, Holbrook was using the GRiST paper tool, but later expressed an interest in leveraging the power of the electronic versions of GRiST. Interest was primarily in deploying tools for working-age and older patients.

11.2.2 Cradlemere Partnership NHS Foundation Trust

Cradlemere Partnership NHS Foundation Trust serves a population of almost 500,000. Cradlemere provides specialist services, including mental-health. These services are provided mostly within the community, and are available through Community Mental Health Teams through referral by the GP.

Similar to Holbrook, Cradlemere also uses iPM as the back-end system for managing patient information.

Cradlemere expressed interest in using the electronic versions of GRiST to assess patients across the age spectrum, i.e., child-adolescent, working-age and older population groups.

¹<http://www.isofthealth.com/en-GB/Solutions/UK%20Patient%20Management/iPM.aspx>

11.3 Deployment Considerations Generic to Trusts

Discussions with Holbrook and Cradlemere (and with other Trusts) resulted in a number of commonalities identified as being generic to NHS Trust-type organisations. These considerations were collated and would form the basis of a requirements specification for the interface that was developed to GRiST. The requirements could be broadly classified as falling under:

- Information governance issues – Maintaining the security of patient data.
- Database issues – Logistics of linking PAS data to GRiST.
- Interface integration issues – Seamless integration of the Trust’s interface with that of GRiST.

11.3.1 Information governance issues

From the outset, it was a design decision to (at least in the research and development stages) retain control of the GRiST system with respect to hosting of the tools and the assessment data. That is to say, GRiST was envisaged as being offered as *Software as a Service* (Laplante, Zhang, & Voas, 2008). By offering GRiST as a service as opposed to ‘installing’ it within a Trust, a number of benefits could be realised (Waters, 2005):

- Reduced database integration work required within each Trust.
- No need for dedicated computing hardware, software, backup processes etc., within each Trust.
- GRiST could easily be modified and upgraded since the GTH trees and the GRiST software would be resident on the GRiST server.
- Assessment data could be mined and used (in anonymised form and with the consent of Trusts) to help improve the performance of GRiST, informing its parameter values etc.

Hosting GRiST outside of the Trust does however, raise a number of information governance issues.

- Personally identifiable information (Pii) – Data that identifies the patient should not be maintained outside the Trust’s systems. This means GRiST cannot for example, store the NHS patient identifier within the GRiST database. It should instead, use a proxy.

- Availability of data – Assessment data should be available in a portable format that can be retained by the Trust, independent of the GRiST system. Furthermore, the assessment data should clearly indicate the patient to which it corresponds. This implies Pii will be required to be synthesised with assessment data at some point (since Pii should not be stored on the GRiST server).
- Unauthorised data access – Patient and assessment data should be protected in transit to prevent unauthorised third parties from gaining access.
- Auditing/accountability – Assessments should be able to maintain logs of saves made by individual clinicians, and reports should contain a clinician identifier.

11.3.2 Database issues

All Trusts invariably use a PAS as part of operations (Beaumont, 2008). This may be one that allows direct access to its database e.g., ePEX², or (more likely), a relatively closed system such as iPM, Lorenzo³ or TotalCARE.⁴ Fortunately, the degree to which GRiST will depend on writing to custom tables within the Trust's (PAS) database (dump) is significantly reduced due to the nature of the GRiST deployment. The offering of GRiST as a hosted service to the Trust ensures that the bulk of the database work will be carried out on the GRiST server, and not at the Trust. What is clear however, is that information from the PAS cannot be totally ignored by GRiST, since the most elementary function of the PAS data is to identify the patient. Therefore, the following should be considered:

- Access to the PAS patient identifier (or a proxy) – At the very least, the interface to GRiST must have read-only access to the NHS patient identifier or a proxy. This *may* involve working on a copy of the PAS database, raising issues of time-lags between creation of a PAS patient and a database dump.

11.3.3 Interface integration issues

Trust staff and GPs are accustomed to accessing the PAS software through web-based dashboard-type applications (Keogh, El-Sayed, & Pilkington, 2008). Application modules relating to a

²<http://www.ascribe.com/cgi-bin/ascribe/info.html?domain=info&name=ePEX%20-%20Community%20Healthcare>

³<http://www.isofthealth.com/en-GB/Solutions/UK%20Lorenzo.aspx>

⁴<http://www.mckesson.co.uk/primary/aboutus/newsroom/newsarchive.cms?newsArticleYear=2005&newsArticleMonth=7&newsArticleID=1&newsArticlePicker=20050014>

patient can be launched through the web-browser once the clinician has logged into the PAS. To reduce training overhead and promote a seamless experience, the GRiST interface would need to consider:

- Handoff to GRiST - Launching from within the dashboard application, with the dashboard performing login and redirection into GRiST behind the scenes.
- Patient selection - The clinician should not need to manually inform GRiST as to the patient that is under consideration. The GRiST session should automatically be configured with the (proxy) patient identifier corresponding to the patient that was selected from within the PAS.
- Appropriate display of personally identifiable information - Pii and assessor data will need to be composited within the assessment display so as to avoid any confusion as to the identity of the parties involved. In accordance with information governance diktats, the data must not be maintained on the GRiST server. Therefore, the relevant Pii must be securely and automatically passed to GRiST as part of *every* login handshake.

11.4 The Generic Trust Interface to GRiST

Taking into consideration information governance, PAS database openness, and interface integration issues, a generic solution to launching GRiST from within Trusts was developed. A corresponding specification document, an API to GRiST, and example code was disseminated to the Trusts. This formed a toolkit with which GRiST could be “deployed” within the Trusts.

Figure 11.1 depicts the life cycle of an interaction with GRiST from within the Trust using the generic interface. The corresponding walk-through of the interaction serves to illustrate how the various deployment considerations discussed in Section 11.3 are addressed.

- a) *Launching a GRiST window.* The clinician requests the GRiST program for a selected patient via, for example, a button within the PAS dashboard. The button launches the *GRiST connection page* (which resides on the Trust’s server) in a new window/iframe^a that will ultimately run the GRiST assessment. The new window starts up with a splash page displaying e.g., “GRiST loading”. The *GRiST connection page* is supplied with the NHS id of the patient under consideration.

^aAn iframe is an html tag that enables one html page to host another autonomous “guest” page inside itself.

- b) The *GRiST connection page* contains ancillary functions to perform:
- 1) *Login into GRiST*. Each Trust is assigned with a designated username and password. These credentials are used to perform a login into the GRiST server, which in turn returns a unique session identifier string (SID). The session identifier can subsequently be used to access GRiST services. As part of the login procedure, a clinician identifier is also passed to GRiST. This is used by GRiST for auditing purposes.
 - 2) *Compute/retrieve a proxy patient ID for use by GRiST*. Rather than sending the patient's normative NHS identifier to GRiST, a substitute is determined. The general solution involves applying a one-way hash function such as SHA-256 to a salted version of the NHS identifier. In the simplest case the proxy identifier is calculated dynamically. In more robust implementations, a separate database table is used to calculate and maintain proxy identifiers and salt values at the Trust. Additional fall-back mechanisms exist for generating a proxy identifier in the absence of a patient record, e.g. in cases where the Trust is using a non-up-to-date dump of the PAS database.
- c) *Redirect to GRiST Assessment Management Interface*. The patient ID proxy and the GRiST session identifier are supplied to the GRiST Assessment Management Interface. The interface is similar to that depicted in Figure 9.10, with the only difference being that the patient is locked to the supplied proxy ID. This means the patient is pre-selected. In cases where GRiST has not observed this proxy id before, the patient record is dynamically created just-in-time. The Assessment Management Interface is also supplied with the patient name for use during rendition of the assessment. This piece of Pii is maintained only as a session variable and not stored within the GRiST database, thereby satisfying information governance requirements.
- d) *Perform assessment and end session*. A GRiST assessment of the patient is performed as normal. Once assessment management is completed, the clinician is able to log out from the Assessment Management Interface. This results in the session id being nullified, along with the deletion of temporary session variables containing the clinician and patient names. The clinician is then redirected back to the PAS dashboard.

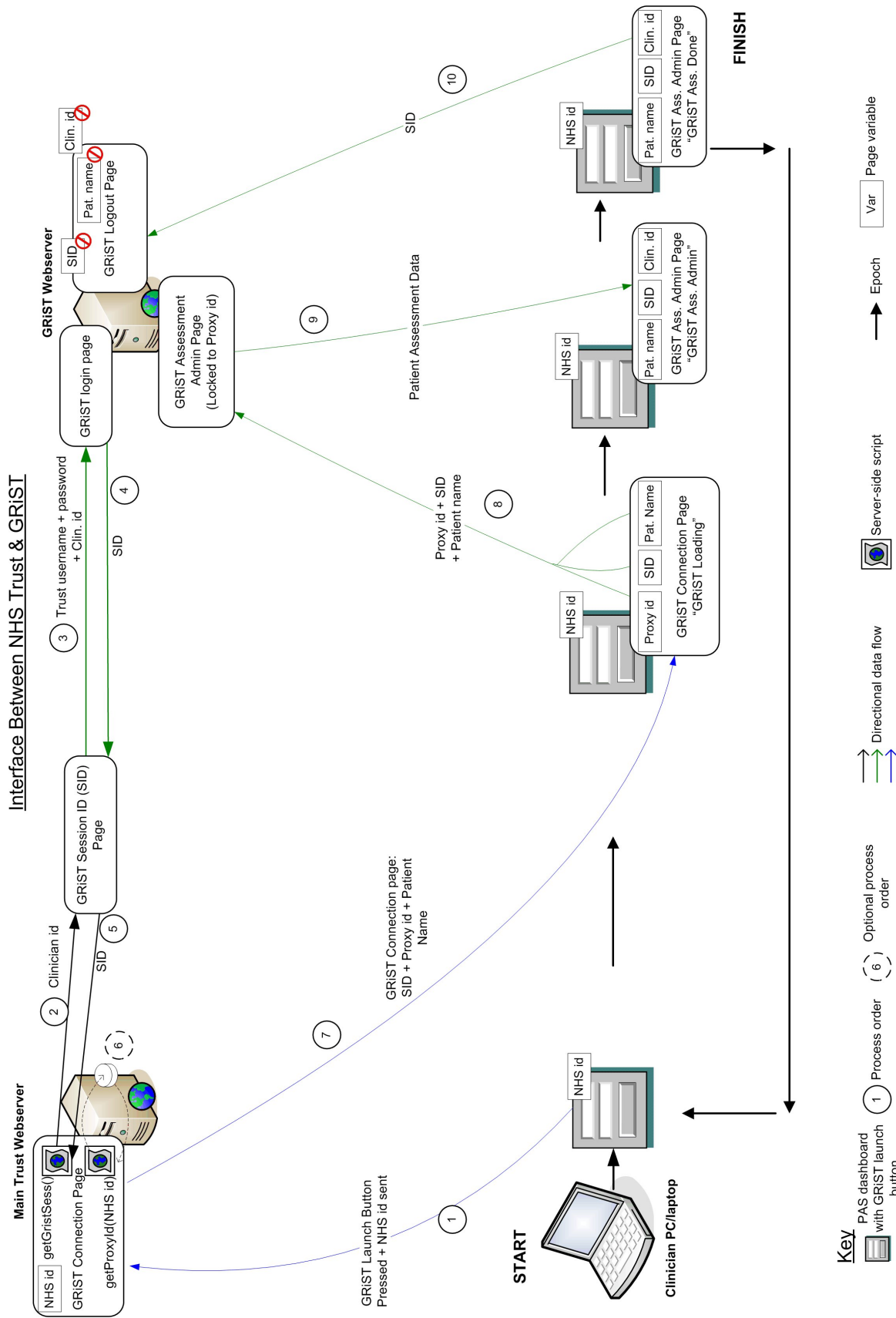


Figure 11.1: Illustrating the interaction that takes place between the NHS Trust and the GRIST server when conducting an assessment.

The Trust interface as explicated is fit for purpose primarily due to the following mechanisms inherent in the solution:

- All transactions with the GRiST server being conducted over an encrypted HTTPS⁵ channel.
- A seamless login into the GRiST server without the clinician's requiring a separate username and password.
- The GRiST server receiving a stable proxy identifier for the patient. This cannot be reverse engineered at the GRiST end, ensuring that the GRiST database does not have access to, and is not able reconstruct Pii data.
- The use of ephemeral Pii data for composition into assessment displays and reports. Since these are dynamically received, session variables corresponding to Pii data are not required to be maintained beyond the session's lifetime.
- Assessment save logs stamped with Trust-supplied clinician identifiers.

11.5 Flexibility Through API Features

Supplementing the interface to GRiST is an expanding API to enable remote maintenance operations such as patient deletion/renames/merges etc. Moreover, Trusts have the ability to specify the tools they wish to use, and the populations with which to use them. This means all versions of GRiST can potentially be accessed by the clinician. The clinician is thereby empowered by having the right tool for the assessment task in hand.

GRiST is set up such that Trusts can choose the extent to which they utilise its API, with GRiST using sensible defaults where information is not available. For example, initially, Cradlemere and Holbrook chose a standardised setup for their clinicians. All clinicians would use the same version of the default working-age tool. Later on, as clinicians became accustomed to using the software, more API features were utilised, gradually exposing newer functionality to the Trust's clinicians.

The development of the API is an incremental effort. The partnerships with Cradlemere and Holbrook Trusts have lead to the creation of numerous extensions to augment the functionality offered by GRiST, improving the clinician's experience.

⁵HTTPS is a protocol for encrypted communications with web servers. A web browser, when communicating with a web server in this fashion, displays a padlock icon to indicate that the page has been securely downloaded.

One such API effort is that of elimination of data duplication across the PAS and GRiST. PAS software such as ePEX typically maintains some information that is also solicited as part of a GRiST assessment. For example, certain demographics information is likely to be contained within the PAS—the patient’s date of birth being one such item. Due to the loose coupling of the two systems, this information would typically be required to be rekeyed into GRiST by the clinician. From the clinician’s perspective of the system as being a cohesive whole, this is illogical, requires extra keying effort, and introduces the possibility of data inconsistencies.

A development version of the API eliminates these problems with the introduction of an XML file format that contains “preset” data. This is created by the Trust for consumption by GRiST and can be transmitted by the Trust as part of assessment launch. Appropriate GRiST assessment fields are pre-populated on assessment start as a result.

11.6 Impact of Electronic GRiST Deployment in Partner Trusts

GRiST went live in Cradlemere in April 2010, with Holbrook following in July 2010. Within three months of deployment at Holbrook, 95 members of staff received training on GRiST, with another 86 to follow. Corresponding data for Cradlemere was not available, but it is envisaged that the number of users is similar, given the relative sizes of the two Trusts. Formal training provided by instructors was supplemented by the built-in electronic help provided by the tool interface, making the tool easy to pick up.

At time of writing, the two Trusts have used the GRiST system in the region of 26,000 times (measured by login count). This equates to 136 logins per day. Approximately 45 GRiST assessments were completed by Trust staff each day over this period. Initially, these comprised working-age GRiST assessments. As new population trees were made available for use by Trust accounts, the proportion of working-age assessments decreased, with the difference being made up by assessments using ‘older’ and ‘child-adolescent’ population tools.

Feedback/bug report forms for GRiST were placed in prominent locations within the Assessment Management Interface. Trust IT department staff and management were also directly able to communicate issues through email, conference telephone calls and meetings. Figure 11.2 summarises the type of feedback received via feedback/bug report forms for HTML GRiST.⁶

⁶Feedback data received via other channels was not available for analysis. Hence these are not included.

11.6. IMPACT OF ELECTRONIC GRIST DEPLOYMENT IN PARTNER TRUSTS

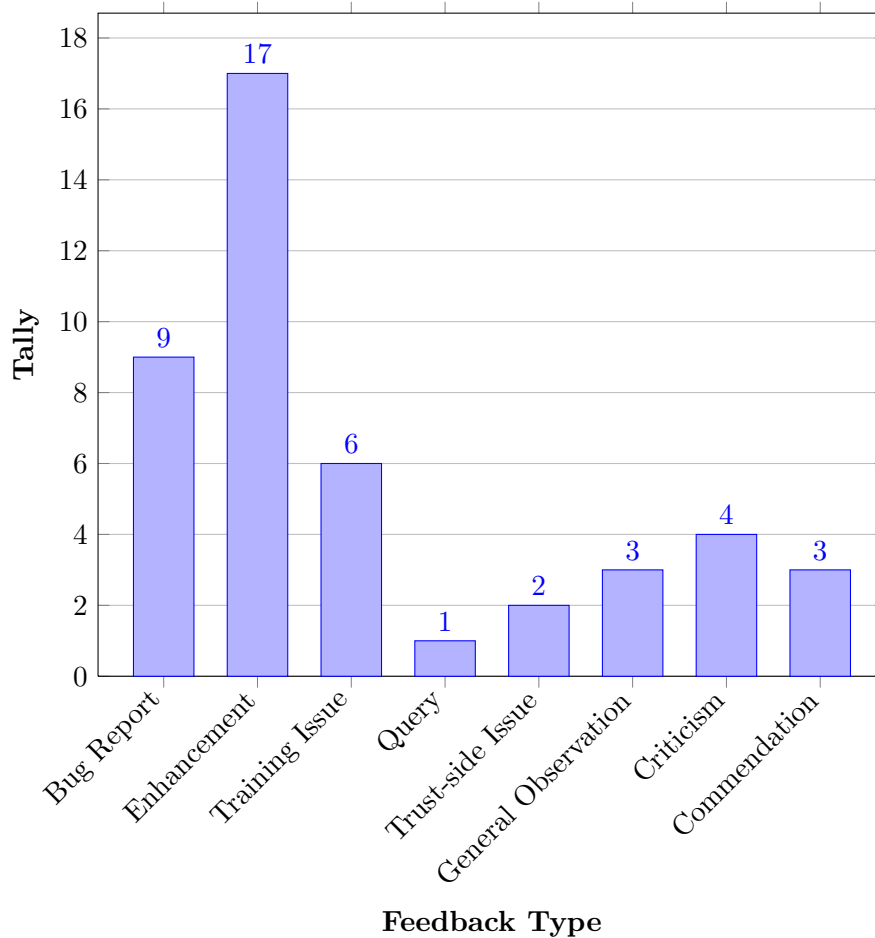


Figure 11.2: Feedback that was received for HTML GRiST via online forms.

Surprisingly, only a handful of negative comments/issues were reported by clinicians during their day-to-day use of the tools. This was impressive considering GRiST had been used 26,000 times. Of the 45 items of feedback that were received, 20% were relating to minor technical issues during early phases of the deployment. These were remedied as and when they were diagnosed. Issues that could be resolved through user-training accounted for 13% of feedback. Many comments also fell into the category of feature requests (38%), and a small number were commendatory (7%).

Arguably, the importance of incorporating the input of end-users in developing a decision system is vital to its success. Particular attention was thus paid to incorporating this input where practicable. Table 11.1 presents examples of the types of feature requests that were relayed to the research team, and the changes to the system that resulted. Without user-feedback, substantial features such as; persistent answers, display of historical comment data, management reports;

11.6. IMPACT OF ELECTRONIC GRIST DEPLOYMENT IN PARTNER TRUSTS

would not have arrived quite so early, if at all. Clinicians' experiences of GRiST would have been poorer for it, and there would not have been the same sense of ownership within those clinicians.

Enhancement Request	Tally	Action Taken
Copying over of previous answers on repeat assessment.	4	Development of the notion of answer-persistence as a balance between the need to reduce re-keying and the need to maintain data accuracy.
Date of birth information should be retrieved from PAS.	2	Provided an API for Trusts to dynamically pass demographic data to GRiST at runtime.
A mechanism for creating an action plan for the patient.	1	New fields for recording action plans and management information were created along with associated reports.
A mechanism for printing reports in colour.	1	Provided the option for generating PDF reports—these faithfully reproduce colour information when printed.
Spell-checking facility.	1	Not implemented. All modern browsers, barring IE, have this facility built-in. It is envisaged that IE will follow suite in a future iteration.
Reports should clearly indicate clinician's name.	1	Implemented.
Reports should have a space for clinician signature.	1	Not implemented. This is a relic from paper-era reporting. The clinician's name appearing on the generated report is sufficient evidence that the clinician produced the assessment.

Table 11.1: A selection of enhancement requests from HTML GRiST users, and the corresponding action that was taken to augment GRiST.

It is human nature for effortful feedback to be meted out more so when a product or system is deemed unsatisfactory (Day, 1977). Consequently, a positive impact tends to be regarded with only tacit acceptance and little overt pronouncement. Judging by the small number of comments received, of which, relatively few were criticisms, it can be concluded that where operational feedback from clinicians is concerned, GRiST has been very well received.

The above conclusion is reinforced further when considering the verbal feedback that has been relayed by managers working within the Trusts. A selection of feedback from Holbrook,

recently presented at a GRiST workshop in the British Science Festival are reproduced below.

“GRiST started out as a paper questionnaire. Holbrook NHS Foundation Trust was one of the first Trusts in the country to use an electronic version. GRiST provides a platform that for us hasn’t been around before. Risk information can be clearly communicated. We have had very positive feedback from clinicians and most disciplines have been trained. Clinicians can complete GRiST in about half an hour so it is much better than the previous tool. The advantage of the electronic form over the paper form is that it enables modification, can be filled in by more than one person on a team and creates persistent data so the patient is not continually asked the same questions.”

11.7 GRiST Usage Outside Trusts and in the Wider Community

This chapter has focussed on deploying GRiST within NHS Trust-type organisations. The ultimate ambition of the research project of course is to fully leverage the flexibility of GRiST—to make it equally relevant and applicable outside the Trust and within the wider spectrum of the community.

The interface to GRiST explicated in this chapter is suitable for large organisations such as Trusts, who engage with many patients at a time. These Trusts retain in-house IT staff as a matter of routine. The modest technical requirements for deploying GRiST can therefore be met by Trust IT staff. With a high throughput of patients that can benefit from GRiST, economies of scale help to mitigate the labour cost associated with this deployment.

It is worth re-iterating that with GRiST’s being a web-based tool, accessible through any modern browser, its use is not restricted to large NHS Trusts who have invested in an interface. Independent hospitals and small organisations, right down to individual GPs’ surgeries can make use of GRiST using a single user mode. These organisations can request a standalone account on the GRiST website, which enables them to log in through the web browser. The user can proceed to create and assess patients as normal. In lieu of an API, the standalone account has configuration options allowing the clinician to use tools corresponding to each population. The combination of configurable options working in tandem with the GTH that powers the underlying system means almost all of the flexibility available to the large Trusts is also available to the smaller outfit.

Taking this work to its logical conclusion, non-PhD efforts are currently being directed towards self-registration access to GRiST for members of the general public. This initiative,

known as *my-GRiST*, is based on a service-user tree, tailored to make mental-health risk concepts understandable to the layperson. In common with the other methods of engaging with GRiST, the underlying technologies driving the system are the GTH and the various tool/reporting engines, with the Galatean model providing risk quantifications. Therefore, with little extra effort, the GRiST infrastructure is optimised for consumption by the wider population.

11.8 Conclusions

This chapter has taken the GRiST platform developed over the course of this thesis, and explored its deployment within a real-world setting—NHS Trust-type organisations.

Consultations with partner Trusts lead to an exposition of interfacing considerations relevant to a hosted service dealing with NHS patient data. The outcome of these discussions was the development and implementation of a suitable generic interfacing solution and API to GRiST.

The API enables Trusts to access the gamut of tools offered by the GRiST platform, and seamless integration between Trust systems and GRiST. The flexibility offered by GRiST's underlying *populations/Levels* paradigms, realised via the GTH, means clinicians are better placed to conduct an assessment tailored to their needs and to those of the patient. This compelling feature of GRiST is one that can only be crudely emulated using a heterogenous set of tools or short/long versions of the same tool. Indeed, many organisations standardised on a conventional assessment tool, being limited to one version, have not had this flexibility at all.

Alternatives to GRiST, such as HoNOS (Wing et al., 1998) do address the need for tool polymorphism to a certain extent, with multiple versions for different populations, e.g., a version for those with learning difficulties (Roy, Matthews, Clifford, Fowler, & Martin, 2002). However, with as little as twelve items, such tools do not have the depth and breadth of coverage as GRiST. Furthermore, due to their paper-based nature, they are not inherently able to translate an assessment across population groups, transcending e.g., barriers between the clinician's understanding of the assessment and that of the patient.

The GTH platform's ability to provide tailored versions of the tool to Trusts together with its decoupling of the interface and answers from the domain model could be regarded as a significant factor in making GRiST a success within the Trusts presented in this case study. It represents an ability for the tool to be molded for organisation and audience, leaving even the choice of tool engine up to the end-user—a degree of choice and flexibility not seen in any other computerised mental-health assessment system.

Of course, customisability of the system represents only one ingredient to the blend of qualities that are required for successful adoption of GRiST. For example, the substantive and comprehensive nature of GRiST's domain knowledge is a vital factor in the tool's gaining credibility with its user-base; arguably, even more so than flexibility. Inadequate or inappropriate questions would certainly have led to GRiST's rejection by clinicians at even the pilot stages of any deployment.

The quality of the overall system, and its allotropic approach to addressing end-user needs has resulted in two very successful large scale deployments. Daily login counts and assessment tallies are a testament to the system's accomplishments. Furthermore, clinicians' approval of the system has been confirmed by Trust managers and via the volume and type of feedback being received directly via the website. Indeed, interest in GRiST is now spreading to a number of other Trusts in the UK and organisations in North America and Australia, with a few already using the paper version.

GRiST's position as a serious contender in the computerised mental-health risk assessment arena is hereby established. Although, resting on the laurels of success is not an option, given the manifold directions in which GRiST can be evolved. The concluding chapter of this thesis will evaluate this research as a whole and examine some of the avenues yet to be taken by GRiST.

12

Conclusions and Future Work

12.1 Introduction

The thesis is concluded with a summary and evaluation of this body of work in light of the objectives of the research, and broader issues in knowledge engineering and CDSSs. Avenues for extending the work, both in terms of building directly on the research, and integrating with work conducted further afield, are discussed.

12.2 Review of Thesis Objectives

The GRiST project set out with the ambitious goal of creating a platform for comprehensive computerised mental-health risk assessment and decision support open to all. GRiST's knowledge was to be built from the ground up, utilising the Galatean model of classification for the calculation of risk.

It is often the case that there is a level of disparity between knowledge engineers' understanding and conceptualisation of a domain and that of domain experts. The knowledge engineer does

not always know in advance the nature of the knowledge and the sometimes complex relationships inherent in it. The domain expert may have this understanding, but will not be able to transcribe it in terms understandable to computers. Therefore, in order for a system to truly represent the domain knowledge, the two parties must continually work together in cementing its crystallisation, both crossing over into each other's territory.

Once a system based on the domain knowledge has been developed, there is the uncertainty that it may fail to gain widespread user acceptance because of mismatches with user/situational requirements. These may be due to the poor quality of the underlying knowledge (as conceptualised in the system) or the possibility that there may be wide-ranging and conflicting expectations from different user groups as to the system's role. One way that tool creators have mitigated these effects is by developing small and very specialised tools. It is due to this rigidity and specificity of many extant tools that organisations have turned to using multiple assessment solutions, with there being little standardisation in the field (M. Scott, 2009).

With GRiST's remit of being a universal tool, accessible to both clinicians and the layman, yet remaining applicable to all spheres of society, this translated into the following primary research objectives:

1. To arrive at a knowledge engineering methodology that would be adaptable to the unknown and changing requirements associated with documenting the mental-health risk areas.
2. To use the fruits of knowledge engineering (the comprehensive mental-health domain model) to directly drive mental-health risk assessment according to the needs of the assessor and the target of the assessment.

12.3 A Flexible Toolchain for Mental-health Assessment and Decision Support

The main theme that emerges from the thesis objectives is that of flexibility. Flexibility in the elicitation process is crucial in order to accommodate idiosyncratic modes of engagement with domain experts, unanticipated changes to elicitation requirements, and an evolving cognisance of the nature of relations emerging within the domain knowledge itself.

Flexibility built into the tool creation methodology is essential in order to successfully accommodate the wide remit of simultaneous suitability to multiple settings, assessors and

patient groups.

The body of work presented in this thesis addressed the research objectives via the development of a comprehensive toolchain for the creation of mental-health assessment/decision support tools from raw domain knowledge solicited from experts. Flexibility was injected into each stage of the process, with XML and XSLT playing a pivotal role in facilitating this.

Chapter 5 developed the idea of maintaining simplistic GUI tools for knowledge elicitation and validation activities. These could be developed at low cost, and would be more accessible to domain experts (due to their lower associated learning curve). *Ad-hoc* augmenting of these tools was facilitated via structured comments that could be made using existing functionality. Simple modular XSLT stylesheets imbued the process with the power to automatically transform comments into directives and finally into actual operations, thereby simplifying tool logic.

The ST resulting from validation activities was reorganised and enhanced in Chapter 6 to make it more amenable to driving risk assessment. The initial work consisted of removing structural redundancy by distilling generic concepts into a designated region of the tree. This reduced the size of the tree considerably, and decreased the likelihood of internal inconsistencies. The second phase of activities involved enriching the ST with constructs relating to question configuration during an assessment, e.g., data types. Provisioning for flexible modes of assessment was initiated with the introduction of the layers and Levels constructs within the ST. These would facilitate screening assessments and assessments of differing lengths according to clinician expertise, respectively. Therefore, the assessment length could theoretically now be controlled by the degree of question abstraction. Finally, uncertainty measures in the form of Membership Grades and associated risk profiles were incorporated.

Uncertainty measures in the form of Relative Influence values could not directly be recorded within the ST, due to its being a distilled tree whose primary remit was to record the structure of the domain knowledge. As part of work in Chapter 7, the RIT, a tree designed primarily for the collection of RIs, was generated automatically from the ST using XSLT. Unlike the ST, the RIT would have generic concepts that were to receive individualised RIs across instantiations expanded to facilitate their collection. The RIT, by nature of its expansion, also brought the knowledge structure closer to one amenable to directly facilitating assessments. It therefore extended the evolving concept of a Galatean Tree Hierarchy for driving assessments.

By designating RI elicitation activities to a separate derived tree, and decoupling from knowledge structure elicitation work (designated to the ST), the issue of RI transferral across RIT

versions required resolving. Chapter 8 developed a method by which a newly generated RIT could carry over RIs manually injected into the previous RIT. The implication of this work was that structural modifications could theoretically be made to an ST, with most of the associated RI housekeeping being carried out automatically. This would make knowledge maintenance activities considerably easier.

Chapter 9's focus was on the generation of assessments. In order that the GRiST knowledge structure could directly and efficiently drive an assessment, the GTH was expanded with the introduction of the CAT and QT. Both were trees derived from the RIT, and matched to each designated Level. The CAT expanded all generic concepts so that the resulting tree structure mirrored the basic assessment structure. The QT was an efficient method by which to maintain question and configuration data related to each CAT node. Assessment answers were to be stored in a separate AT. A major benefit of segregating answer data was that an assessment conducted using one CAT and QT need not have been required to be tied to that tree pair during report generation or reassessment. This would in a later chapter, have implications for making an assessment conducted by/for one speciality more accessible to another.

Various tool generation engines developed to run on the GTH platform were also showcased in Chapter 9. Tools ranged from a generated paper version, through an electronic HTML version, to the fully-featured Java version utilising the Galatean model. Users therefore had an upgrade path available to them, providing them with the flexibility to unlock more of GRiSTs features as and when they were ready. Because all tools would run on the same underlying GTH platform, these tools could easily be made to interoperate with regards to assessment data. Furthermore, updates to the underlying knowledge structure could immediately be propagated to the tools and associated reports since it is the (generated) CAT, QT (and the AT) that drive tool and reporting engines. This made the system very maintainable. The fact that the GTH is XML-based, with tools being generated via languages such as XSLT, also made the system both extensible and adaptable. New features could be added via new attributes, and reports could be configured to include/omit/synthesise data from these attributes.

Building on the user-facing flexibility of multiple tool engines, each capable of producing tools matching the user's Level of experience, was the powerful concept of *populations*. Chapter 10 developed this construct as a consequence of the notion that different user groups would be better accommodated by a tool specifically tailored to each group. This would mean that questions applicable to a specific segment of the general population could be limited to being available

only for assessment of that group, e.g., questions relevant only to children. Similarly, question organisation and configurations suited to particular organisation/clinician specialities and even the patient themselves could be represented as a population.

The SST was introduced into the root of the GTH as a meta-ST to record population-specific customisations. XSLT would be used to automatically generate population-specific STs from the SST. These could be used to generate population-specific assessment trees. The benefit of this approach was that tool engines required no modification due to the GTH's restricting the complexity associated with multiple-population data to the SST. The approach, when combined with the decision to segregate question and answer data from assessment structure data (i.e., from the CAT), resulted in an additional level of flexibility to GRiST—an assessment conducted using one population CAT could trivially be made to be viewable using the CAT of another population. This would mean for example, that an assessment conducted using terms understandable to a clinician could be made to be reported in terms understandable to the patient—fortifying GRiST with a level of inclusiveness not previously available to mental-health assessment systems.

Finally, deployment of the improved GRiST architecture within two NHS Trusts was explored as a case study, presented in Chapter 11. This required the development of a generic interface to GRiST, which took into consideration issues related to information governance and the PAS software used within Trust-type organisations. User feedback was indicative of GRiST's welcome reception. This was also supported by login and assessment statistics, which implied GRiST (in its many configurations) to be a heavily used resource.

Building on the success of the large GRiST deployments, other efforts to make GRiST accessible to smaller organisations and members of the public were briefly described. The result of these efforts is that the expertise embodied in GRiST is potentially available to anybody wishing to use it—in a format understandable to them.

12.4 Benefits of GRiST's Approach to KE

One of the main impetuses for the development of Protégé, and its predecessor, Opal, was a desire to include domain experts more directly in the creation of knowledge bases (Gennari et al., 2003). This motivation is also shared by the knowledge acquisition aspects of the GRiST toolchain (described in Chapter 5). It is fitting therefore, to compare GRiST's knowledge acquisition features with those of Protégé. Table 12.1 provides this comparison, beginning with attributes

12.4. BENEFITS OF GRIST'S APPROACH TO KE

that facilitate easier inclusion of domain experts, through to generic attributes beneficial to any knowledge acquisition systems.

Feature	GRiST	Protégé
Allowing multiple domain experts to simultaneously work on the knowledge.	The KE activities were web-based, and allowed experts to log on and work on an individualised copy of the knowledge structure. A report on all edits could be viewed by knowledge engineers in order to decide which were to persist.	Efforts such as the collaborative Protégé plugin (Tudorache, Noy, Tu, & Musen, 2008) mean that experts are able to simultaneously work on an ontology. This offers a real-time updated view of the consensual knowledge structure.
Auditing of changes in knowledge.	GRiST's approach provides end-to-end auditing from initial expert interview through to the item's inclusion in the final version of the ST.	The collaborative Protégé plugin has the capability to record the author of a change in the evolving knowledge structure.
Appropriateness of learning curve to domain experts.	The GRiST approach is one of removing as much computer science terminology as possible. Furthermore, KE tools are limited in feature-set, allowing e.g., addition, renaming, deletion of nodes etc. This makes it easy for domain experts to master the tools.	Being a meta-tool for the development of knowledge acquisition tools, Protégé employs a very rich array of features, leaning heavily on computer science terminology. Protégé's UI is complex due to the feature-set it supports. Thus, it is difficult for domain experts to become proficient in its use.
Visualisation of domain model.	KE tools are able to display GRiST's tree-like knowledge structure using built-in XML viewing components.	The OWLViz plugin to Protégé enables display of OWL class hierarchies (Lanzenberger, Sampson, & Rester, 2010).
Portability of file format.	GRiST uses a bespoke, hierarchical, XML file format, with an emphasis on human readability, and ease of translation into other formats via XSLT.	Files are stored in OWL. This is also an XML language, but does not utilise the inherent hierarchy of XML. The expressivity of the language makes it complicated to read and parse without tool assistance. Nevertheless, OWL is well known in the KE community and a W3C standard.
Expressibility of underlying representation language.	Contains relatively few constructs, and exploits the inherent hierarchy of XML.	Contains a large number of constructs to model many scenarios.
Verification of the correctness of the knowledge structure.	Tree verification functionality built into the GTH generation process.	A reasoner such as RACER (Haarslev & Möller, 2003) can be used to perform verification.
Inference of new facts from existing facts.	The Galatean model provides classification of risk, but not the inference of new concept relations.	RACER can be used to perform inference.

Table 12.1: A comparison of GRiST's knowledge acquisition approach with that of Protégé.

The aim of the GRiST project was to allow domain experts to partake in KE activities without feeling overwhelmed. The comparison in table 12.1 reveals that GRiST's strength does indeed lie in the simplicity of its representation and of its knowledge engineering tools. Nevertheless, it does also show that more work could be done in increasing the portability of GRiST's file format; a topic that will be revisited in Section 12.6.5.

12.5 Contribution to CDSS Best-practice Theory

In addition to the development of a CDSS in mental-health, the GRiST project has provided an opportunity to perform some real-world validation of the ideas advanced in the literature for improving decision-support. This section reflects on some of the issues put forward by Sittig et al. (2008) and Bates et al. (2003), and outlines GRiST's contribution to the discussion.

12.5.1 Strategic Challenges

Section 3.3 describes some of the challenges set by Sittig et al. (2008) that are yet to be addressed by CDSSs. GRiST's main contribution in this area falls under the creation of *an architecture for sharing CDSS modules*. GRiST's API makes it very easy for a Trust's PAS to connect and invoke GRiST. Message-passing behind the scenes, as described in Chapter 11, allows the two systems to seamlessly integrate, and eliminates re-keying where data contained in the PAS is also required by GRiST.

One key issue that has been overcome is that of allowing Trusts to subscribe to an external provider (GRiST) without necessarily having to give up confidential patient data in the form of Pii. It is however, acknowledged that there is still a lot of work to be done in developing standardised interfaces to CDSS services. Development of GRiST can help in this respect via the dissemination of a mature API, so that PAS developers and CDSS providers alike can identify the types of features that would be required of an interface. By the same token, GRiST's data format could draw from standardised medical terms systems such as SNOMED (Spackman, Campbell, Côté, et al., 1997) and UMLS (Aronson, 2001). This would lay the foundations for greater portability of GRiST's output data, reducing vendor lock-in.

The *human-computer interface* aspect of Sittig et al.'s challenges are met primarily through the dynamic use of colour during a GRiST assessment. Areas requiring attention are highlighted in oranges and red, and are a means of drawing the clinician's attention in a non-blocking manner. A data validation run is performed only on assessment submission, whereupon the clinician is

taken through any errors/inconsistencies step-by-step. The steps cannot be bypassed, meaning that error validation is a self-contained activity. Whilst error validation leads to the correction of errors, it also acts as a training exercise for the clinician—the clinician learns why certain errors are being generated and will in time modify entry behaviour accordingly.

12.5.2 GRiST and the “ten commandments” for effective CDSSs

Section 3.4.1 describes ten “commandments” proffered by Bates et al. (2003) for the development of successful CDSSs. These have been garnered through first-hand experience in developing numerous decision support systems. The insight gained from the development of GRiST contributes to these theories, as applied to mental-health CDSSs, in the following ways:

1. **Speed is everything** – GRiST’s response times are instantaneous when updating risk levels for a given answer. However, assessment initialisation, verification and saving are not instantaneous due to the computational and network demands of these activities. In fact, on slower computers and browsers, these operations can take upwards of three or four seconds. GRiST mitigates these factors by updating the UI to communicate to the user that an operation is taking place. This creates a perception of speed through the fact that something is always seen to be happening, as opposed to the tool appearing “frozen”. Consequently, none of the feedback received from users indicated speed to be a problem. Thus, the GRiST project supports the idea that the *perception* of speed through UI responsiveness is also very important where real speed cannot be obtained (Tabbers, Kester, Hummel, & Nadolski, 2004).
2. **Anticipate needs and deliver in real-time** – Currently, GRiST provides instantaneous feedback as to the severity of the provided answer value. This has been a well-received improvement to the HTML tool over paper-based approaches. Future improvements to the system will mean that classification will occur dynamically as the assessment progresses (refer to Section 9.5.3). This will pave the way for the system to automatically ascertain when sufficient information has been gathered to deliver a confident risk prediction. This would be useful as a time-saving measure, as the clinician could then be prompted to move to a different top-level risk or finish the assessment.
3. **Fit into the user’s work-flow** – The importance of this aspect has also been evidenced in this project. For example, the *management* attribute’s genesis was through discussions with

one of the Trusts. Here, clinicians were wishing to use GRiST to record not only patient attributes, but also how these would be addressed via any treatment. Furthermore, the decision-support aspect (delivered through colouration of answer values) has meant that GRiST not only fits into the clinician's work-flow, but also augments it—GRiST is not simply being used as a documentary tool, but rather, as an aid to assessing the patient.

4. **Little things can make a big difference** – Given that a clinician can spend up to half an hour on an assessment, and will be routinely using GRiST, usability has been an important consideration in the development of GRiST. Tweaks have ranged from automatically providing a sensible file name for a generated PDF report, to reworking the tool's functionality to reduce re-keying. Not all of this was achieved on the first try, and only through user feedback did features such as answer persistence and improved recording of comment data crystallise. These allowed the clinician to “do the right thing” faster.
5. **Recognise that clinicians will strongly resist stopping** – Even though GRiST has screening questions that should logically be answered before the main set of questions, it was a conscious decision not to enforce any kind of rigid progression through the tool (refer to Section 3.4.2). This leaves the clinician to work through the questions as desired, leaving the clinician in ultimate control. Furthermore, the answer validation functionality highlights erroneous answers, and importantly, gives feedback on how the answer should be corrected. This is akin to providing an opinion together with a reason for that opinion, which is more effective at inducing a change in the clinician's behaviour.
6. **Changing direction is easier than stopping** - This aspect was not applicable to GRiST, and so, did not receive in any real-world testing in this project.
7. **Simple interventions work best** – GRiST is designed to be operated with minimal training. Information is conveyed through a combination of colour and icons. Reports only detail questions that have been answered and provide the node label rather than the question text, thereby reducing the amount the clinician is required to read. Crucially, different reports can be generated for different perspectives. The benefit therefore, is that operation of/output from GRiST is simple, reducing information overload, and the potential for important information to be buried in the data.
The simplicity aspect was also crucial to the KE phases of GRiST's development. Simple tools designed to be used by clinicians with relatively little IT experience were essential

in gaining participation in the project. XML and XSLT have both been invaluable in providing this simplicity due to their malleability. Future implementers should therefore apply this “commandment” not only to the CDSS, but also to the KE process.

8. **Additional information should be asked for only when needed** - Asking for information that can automatically be obtained or inferred by GRiST represents an inefficiency and is an annoyance for the clinician. Indeed, feedback from clinicians has led to its being pointed out that demographic data such as patient date of birth is needlessly being asked of the clinician. This is why API features such as automatic passing of demographic data from the PAS (refer to Section 11.5) are being developed. Persistent data for repeat assessments is also another area that was developed as a result of clinician feedback.
9. **Monitor impact, get feedback, and respond** – Bates et al. emphasise the importance of altering automatic notifications and alerts according to whether they are serving a useful purpose. This way, only important alerts remain, and are less likely to simply be “clicked-through”. Although this aspect was not directly testable using GRiST since it does not issue alerts as such, a related issue of *listening to users* was found to be very relevant. This is discussed further in the next section.
10. **Manage and maintain the knowledge-based system** – The system of trees in the GTH and the tree synchronisation mechanisms described in Chapter 8 have been designed to make maintenance of the knowledge that drives GRiST very easy for the user. This means GRiST can continually be improved through incorporation of new findings in the literature, clinician feedback, and automated “self-improvement”. One of the mechanisms by which GRiST aims to self-improve is through the comparison of clinician-supplied ratings of top-level risk with ratings generated via GRiST’s classification process. These will then enable RIs and MGs to be adjusted to better match how clinicians weight concepts in a clinical setting.

Another strategic avenue of self-improvement is through the use of case-based reasoning: the data accumulated from tens of thousands of assessments could be used to more quickly inform the outcome of the current assessment. Thus, implementers should also look at automatic ways of maintaining and improving the system via intelligent use of the rich data that is accumulating within it.

12.5.3 The “eleventh commandment” for effective CDSSs

The GRiST project has demonstrated that most of Bates et al.’s (2003) “commandments” are applicable to decision support systems for mental-health risk assessment. It has also provided further insight, particularly in relation to the first commandment: *speed is everything*. Speed, although important, may not be as crucial as Bates et al. decree. What is important is *responsiveness*. Responsiveness can mitigate the effects of going over the sub-second threshold that Bates et al. regard as the limit of tolerability for the user.

The present research also informs another “commandment” not adequately addressed by Bates et al.

11. Listen to the users

It is not enough to passively monitor and obtain low-level feedback from tool-usage (i.e., click counts etc.) as advocated in “commandment” nine. Communicating directly with system users and making provision for them to air views and concerns is paramount to moulding an effective CDSS. The system receives clinical usage in the hands of these users, so they are in the best position to give feedback on what is working, what needs improvement, and why (refer to Section 11.6 for examples of this). Users were involved throughout the development of GRiST, and many important features and efficiencies were developed directly as a result of open dialogue with these users.

12.6 Future Work

Numerous directions for improving and extending GRiST are open to exploration. Many of these can be categorised as seeking to address limitations in the current GRiST platform or to extend its functionality. More strategic improvements involve further opening up GRiST to the wider research community, allowing its benefits to be maximised. This section seeds future work by introducing some of these avenues.

12.6.1 Incorporation of the Galatean model into the server

At present, the only mature implementation of the Galatean classification engine resides within the Java version of GRiST. Once RIs for GRiST’s knowledge structure have been finalised, the Java tool will be able to fully calculate risk values for the assessment in real-time, as the

assessment progresses. With respect to the generation of reports of risk, the *status quo* implies that an assessment will be required to be performed using the Java tool in order for the requisite risk information to be available to reporting engines.

Future work should incorporate an additional Galatean classifier within the server for the use of report generation engines. The benefits of this are two-fold:

- Full risk reports can be generated for the user, regardless of the tool that has been used to collect data.
- The potential for e.g., the HTML version of GRiST to make use of ad-hoc or near-real-time risk calculations is opened up. This could be via AJAX calls to the classifier on the server and suitable interface modifications to the tool GUI.

With the Galatean model's ability to calculate the accumulation of risk over concepts, additional work would need to be carried out on displaying risk. Specifically, research into the display of percolation of data within hierarchical structures.

12.6.2 Augmenting of the *populations* framework

Chapter 10 conceptualised *populations* as representing organisational, clinical, and patient perspectives on the assessment. This approach is well suited to the development of customised versions of GRiST where a user group falling into *one* of these categories is recognised. However, the chapter also acknowledges that the *populations* mechanism (in its current incarnation) is a compromise situation between flexibility and complexity. Indeed, as GRiST grows in terms of number and type of deployments, it is inevitable that scalability problems with the present *populations* mechanism will surface.

It is entirely plausible that a hypothetical real-world GRiST customisation could more logically/optimally be described in terms of an interaction between *more than one* perspective. For example, the prison service may dictate a specific ordering for questions that is consistent with the needs of this type of organisation. Within the prison service there may be different age groups, who might be best served by a tool tailored to that age group.

Given that the current *populations* framework treats perspectives as mutually exclusive, the only way to realise the scenario of population interactions would be to create a new population definition specifically for each interaction. This would give rise to a combinatorial explosion of populations. In the case of the hypothetical prison service offering, the gamut of default

clinical population definitions would each need to be replicated, and the replications modified specifically.

Future research would need to consider the issue of population interactions and how best to accommodate them within the SST. It may be that the generalised attribute, `populations`, would need to evolve into specialised attributes such as `organisational-populations`, `clinical-populations` and `patient-populations`. An accompanying inheritance/precedence scheme incorporating conflict resolution mechanisms would also be required where more than one specialised population group would be applicable to a node.

Language Localisation

As international interest in GRiST grows, language localisation may also need to be incorporated into the SST. This may form part of the upgraded *populations* framework via a new additional perspective. Alternatively, a parallel framework for recording translated strings may be necessitated due to the complexity introduced by the potential interactions of the other population perspectives.

12.6.3 Bilateral mappings with other risk assessment tools

Risk assessment practice across the NHS organisations is not standardised (Higgins et al., 2005; Hawley et al., 2010). Consequently, organisations employ a variety of assessment tools (Hawley et al., 2006). Many of these tools are specialised or consist of a relatively small number of items, e.g., HCR-20 for assessing violence risk (Douglas & Webster, 1999), and RFL-OA for suicide in older adults (Edelstein et al., 2009). GRiST on the other hand, covers all five of the risk areas identified in the NHS's best practice guide for mental-health risk management (National Risk Management Programme, 2007), is comprehensive in its data collection, and incorporates a classification engine for the quantification of risk.

In the interests of providing choice and synergistic benefits to Trusts, it would be interesting to investigate the idea of developing bilateral mappings between GRiST and other tools. In this scenario, the GRiST representation of an assessment, being more comprehensive, could be considered to be a superset of the items contained in other published tools. In principle, the data from a GRiST assessment could automatically be translated and scored as an assessment of a third-party tool. The assessor would then be able to compare results from both the GRiST perspective and the perspective of the other tool. This would help the assessor to make a more

informed decision in determining a care pathway for the patient.

Mappings from other extant tools to GRiST could also be created to help generate GRiST risk quantifications using the third-party data. This feature could, for example, be combined with the “preset” answers API discussed in Section 11.5.

Finally, as bilateral mappings between *multiple* tools and GRiST develop, this would place GRiST in a position where it could serve as an intermediary format between tools. In essence, an assessment conducted with one tool could be translated into an assessment for a different tool, with the GRiST format serving as a tool-agnostic representation. Such a facility could be published as a web service, which could be utilised in its own right without any commitment to directly using GRiST for assessment.

12.6.4 Application of the GRiST toolchain to new domains

Although GRiST has been developed to assess mental-health risk areas, this does not preclude the architecture from being used in other domains.

The Galatean model is a general purpose model of classification, developed independently to GRiST. It can model and explain a variety of psychological phenomena in the field of human classification decision-making, and has in the past, been applied to domains as diverse as horse-racing! Therefore, it is both theoretically and demonstrably capable of serving as a classification engine beyond the mental-health domain.

When considering the representation of domain knowledge, any area whose concepts are refinable in a step-wise fashion (Wirth, 1971) is amenable to being represented using GRiST’s hierarchical ST format. Once codified as an ST, the rest of the GTH for the new domain could be generated using existing XSLT stylesheets, and the pre-existing engines used for driving “assessments” and reporting.

12.6.5 Automated mappings from the SST file format to OWL DL

Chapter 4 eschewed using the established ontology language, OWL DL, for developing the GRiST knowledge structure, and instead advocated using a hierarchical XML format, which eventually coalesced into the SST. Several reasons were outlined for this decision:

- The inherent hierarchy within an XML file is naturally suited to Psychologists’ conception of how expert knowledge is organised (Larkin, 1980; Eylon & Reif, 1984), and to the Galatean model’s conceptualisation of risk percolation.

- Although a rich arsenal of constructs is available within OWL DL for describing concepts and their inherent relations, the guarantee of decidability in reasoning in OWL DL means certain restrictions on the language's expressivity (Hoekstra, 2009). Given the scope of the SST—to describe the domain model as well as to ultimately drive flexible tool construction—it was not possible *a priori* to assess whether and how this would impact on the project.
- The complexity of the OWL DL language itself would create an artificial barrier with respect to participation of domain experts, and the need for specialised tools in order to efficiently and correctly manipulate the underlying files. OWL DL is a general purpose ontology language, and is therefore aimed at serving a much wider range of ontology applications than the SST syntax.
- An OWL DL reasoner was not required, since the Galatean model would preform computation of risk.
- OWL DL tool support e.g., in the area of collaborative model editing and auditing, was at a less advanced stage of development at the inception of the research project.

Given that the syntax of the SST is now relatively stable, a fuller understanding of the mereological relationships governing concepts within the tree can be gained. Furthermore, the involvement of domain experts is no longer of primary concern, assuming that knowledge elicitation phases are complete. These factors remove some of the hindrances that discounted OWL DL as a representation format to use during GRiST knowledge elicitation.

In the interests of making GRiST's knowledge more accessible to, and reusable within the wider research community, it is at this juncture that OWL should be considered as an additional representation format. Indeed, the genesis of OWL was out of a need to share and amalgamate disparate islands of knowledge in an intelligent way (Lacy, 2005). Formulating an OWL representation of the knowledge could for example, benefit recent ontology creation efforts in related areas such as mood disorders (Haghighi, Koeda, Takai, & Tanaka, 2009) and psychosis (Kola et al., 2010), and contribute to grand unifying schemes as described in Dumontier (2010), which envisage repositories that span the life sciences.

With the recent ratification of the OWL 2 DL standard (W3C OWL Working Group, 2009), the development of design patterns for the modelling of mereology relevant to GRiST's hierarchical format (Rector & Welty, 2005), and interesting work conducted in the use of XSLT to

convert XML structures into RDF (D. Connolly, 2007) and OWL (Bohring & Auer, 2005), this is now an appealing avenue to exposing GRiST's knowledge outside of the tools and further afield.

12.7 Epilogue

Revisiting the fundamental question of flexibility, this thesis has successfully demonstrated this quality throughout its approach to developing the toolchain for mental-health assessment and decision support. The GTH, its representation as XML, and transformations via XSLT have been the underlying mechanisms for achieving this. The outputs of this toolchain are; a validated representation of mental-health risk knowledge, and the Galatean Risk Screening Tool.

References

References

- Abernethy, M. A., Horne, M., Lillis, A. M., Malina, M. A., & Selto, F. H. (2005). A multi-method approach to building causal performance maps from expert knowledge. *Management Accounting Research*, 16, 135-155. (page 61)
- Adams, I., Chan, M., Clifford, P., Cooke, W., Dallos, V., Dombal, F. de, et al. (1986). Computer Aided Diagnosis Of Acute Abdominal Pain: A Multicentre Study. *British Medical Journal (Clinical Research Edition)*, 293(6550), 800-804. (page 39)
- Ainsworth, J. (2007). The challenges of clinical e-Science: Lessons learned from PsyGrid. In *Proceedings of the uk e-science all hands meeting 2007*. (page 147)
- Anderson, J. (1982). Acquisition of cognitive skill. *Psychological review*, 89(4), 369-406. (page 38)
- Anumba, C., Dainty, A., Ison, S., & Sergeant, A. (2006). Understanding structural and cultural impediments to ICT system integration: A GIS-based case study. *Engineering, Construction and Architectural Management*, 13(6), 616-633. (page 165)
- Aronson, A. (2001). Effective mapping of biomedical text to the UMLS Metathesaurus: the MetaMap program. In *Proceedings of the amia symposium* (p. 17). (page 201)
- Ayyub, B. M. (2001). *Elicitation of expert opinions for uncertainty and risks*. London: CRC Press. (page 70)

-
- Baader, F., & Nutt, W. (2003). Basic description logics. In *The description logic handbook* (pp. 43–95). (page 52)
- Bartis, E., & Mitev, N. (2008). A multiple narrative approach to information systems failure: a successful system that failed. *European Journal of Information Systems*, 17(2), 112–124. (page 165)
- Bates, D., Kuperman, G., Wang, S., Gandhi, T., Kittler, A., Volk, L., et al. (2003). Ten commandments for effective clinical decision support: making the practice of evidence-based medicine a reality. *Journal of the American Medical Informatics Association*, 10(6), 523–530. (pages 43, 46, 47, 201, 202, 204, and 205)
- Beaumont, R. (2008). *Types of Health Information Systems (IS)* (Tech. Rep.). Available from http://www.fhi.rcsed.ac.uk/rbeaumont/virtualclassroom/chap12/s2/systems_new.pdf (page 184)
- Beazley, D., Ward, B., & Cooke, I. (2002). The inside story on shared libraries and dynamic loading. *Computing in Science & Engineering*, 3(5), 90–97. (page 151)
- Bergman, L., & Fors, U. (2008). Decision support in psychiatry – a comparison between the diagnostic outcomes using a computerized decision support system versus manual diagnosis. *BMC Medical Informatics and Decision Making*, 8(1), 9. (page 48)
- Berlin, A., Sorani, M., & Sim, I. (2006). A taxonomic description of computer-based clinical decision support systems. *Journal of Biomedical Informatics*, 39(6), 656–667. (pages 40, 41, and 42)
- Berners-Lee, T., & Hendler, J. (2001). Scientific publishing on the semantic web. *Nature*, 410, 1023–1024. (page 51)
- Blank, A. G. (2004). *TCP/IP Foundations*. John Wiley & Sons. (page 110)
- Boegl, K., Adlassnig, K. P., Hayashi, Y., Rothenfluh, T. E., & Leitich, H. (2004). Knowledge acquisition in the fuzzy knowledge representation framework of a medical consultation system. *Artificial Intelligence in Medicine*, 30, 1-26. (page 57)

-
- Bohring, H., & Auer, S. (2005). Mapping XML to OWL Ontologies. In *Leipziger Informatik-Tage, volume 72 of LNI* (pp. 147–156). (page 210)
- Bouch, J., & Marshall, J. J. (2005). Suicide risk: structured professional judgement. *Advances in Psychiatric Treatment, 11*, 84-91. (page 22)
- Bray, T., Tobin, R., Thompson, H. S., Hollander, D., & Layman, A. (2009, December). *Namespaces in XML 1.0 (third edition)* (W3C Recommendation). W3C. (<http://www.w3.org/TR/2009/REC-xml-names-20091208/>) (page 66)
- Brennan, S. (2007). The biggest computer programme in the world ever! How's it going? *Journal of Information Technology, 22*(3), 202–211. (page 182)
- Brewster, C., & O'Hara, K. (2007). Knowledge representation with ontologies: Present challenges–Future possibilities. *International Journal of Human-Computer Studies, 65*(7), 563–568. (pages 45 and 46)
- Brooker, C., & Fox, C. (2009). *Health needs assessment of children in secure settings in the east midlands* (Tech. Rep.). University of Lincoln. (page 89)
- Buckingham, C. D. (1992). *Galatean model of human classification implemented in a decision support system*. Unpublished doctoral dissertation, University of Birmingham, UK. (page 24)
- Buckingham, C. D. (2002a). Psychological cue use and implications for a clinical decision support system. *Medical Informatics and the Internet in Medicine, 27*(4), 237-251. (pages 24, 31, and 34)
- Buckingham, C. D. (2002b). Psychological cue use and implications for a clinical decision support system. *Medical Informatics and the Internet in Medicine, 27*(4), 237-251. (page 62)
- Buckingham, C. D., Ahmed, A., & Adams, A. E. (2007). Using XML and XSLT for flexible elicitation of mental-health risk knowledge. *Informatics for Health and Social Care, 32*(1), 65–81. (pages 20 and 75)
- Buckingham, C. D., & Birtle, J. (1997). Representing the assessment process for psychodynamic psychotherapy within a computerized model of human classification. *British Journal of Medical Psychology, 70*, 1-16.

(pages 24 and 34)

- Buckingham, C. D., & Chan, T. (2002). *Developing a mental-health risk-screening tool* (Tech. Rep.). Surrey Hampshire Borders NHS Trust. (page 31)
- Buckingham, C. D., Kearns, G., Brockie, S., Adams, A. E., & Nabney, I. T. (2004). Developing a computer decision support system for mental health risk screening and assessment. In J. Bryant (Ed.), *Current perspectives in healthcare computing 2004* (p. 189-194). Swindon BCS HIC. (page 30)
- Buzan, T. (2003). *The mind map book*. BBC Consumer Publishing: London. (page 60)
- Cannon, D., & Allen, S. (2000). A comparison of the effects of computer and manual reminders on compliance with a mental health clinical practice guideline. *Journal of the American Medical Informatics Association*, 7(2), 196. (page 41)
- Care Services Improvement Partnership. (2006). *10 high impact changes for mental health services*. London: Department of Health. (page 21)
- Castle, K., Duberstein, P. R., Meldrum, S., Conner, K. R., & Conwell, Y. (2004). Risk factors for suicide in blacks and whites: An analysis of data from the 1993 national mortality followback survey. *American Journal of Psychiatry*, 161, 452-458. (page 22)
- Cerri, D., & Fuggetta, A. (2007). Open standards, open formats, and open source. *Journal of systems and software*, 80(11), 1930–1937. (page 51)
- Clark, J. (1999, November). *XSL transformations (XSLT) version 1.0* (W3C Recommendation). W3C. (<http://www.w3.org/TR/xslt>) (pages 57 and 73)
- Coiera, E. (1997). *Guide to Medical Informatics, the Internet and Telemedicine*. London: Chapman and Hall Medical. (page 145)
- Connolly, D. (2007). *Gleaning resource descriptions from dialects of languages (GRDDL)* (W3C Recommendation). W3C. (<http://www.w3.org/TR/2007/REC-grddl-20070911/>) (page 210)

-
- Connolly, T., & Begg, C. (2010). *Database systems: a practical approach to design, implementation, and management*. London: Addison-Wesley.
(pages 84 and 107)
- Cosmides, L., & Tooby, J. (1996). Are humans good intuitive statisticians after all? Rethinking some conclusions from the literature on judgment under uncertainty. *Cognition*, 58, 1-73.
(page 39)
- Crockford, D. (2006). JSON: The fat-free alternative to XML. In *Proc. of xml*. Boston.
(page 96)
- Damiani, E., Vimercati, S. de Capitani di, Paraboschi, S., & Samarati, P. (2000). Design and implementation of an access control processor for XML documents. *Computer Networks*, 33(1-6), 59-75.
(pages 169 and 170)
- Day, R. (1977). Extending the concept of consumer satisfaction. *Advances in Consumer Research*, 4(1), 149-154.
(page 191)
- Del Fiol, G., Rocha, R. A., Bradshaw, R. L., Hulse, N. C., & Roemer, L. K. (2005). An XML model that enables the development of complex order sets by clinical experts. *IEEE Transactions on Information Technology in Biomedicine*, 9(2), 216-227.
(page 57)
- Department of Health. (1999). *National Service Framework for Mental Health*. London: DH Publications.
(page 22)
- Department of Health. (2002). *Delivering 21st century IT support for the NHS: national strategic programme*. London: Department of Health.
(page 157)
- Department of Health. (2004a). *Choosing health: making healthier choices easier*. London: DH Publications.
(pages 21 and 22)
- Department of Health. (2004b). *The National Service Framework for Mental Health - 5 years on*. London: DH Publications.
(pages 21 and 22)
- Department of Health. (2004c). *NHS improvement plan: putting people at the heart of public services*. London: DH Publications.
(pages 21 and 22)

- Department of Health. (2006, April). *Lowest suicide rate since records began*.
www.dh.gov.uk/PublicationsAndStatistics/PressReleases/. (accessed October 22nd)
(page 22)
- Dexter, P., Perkins, S., Overhage, J., Maharry, K., Kohler, R., & McDonald, C. (2001). A computerized reminder system to increase the use of preventive care for hospitalized patients. *The New England journal of medicine*, *345*(13), 965.
(page 46)
- Dong, J., Du, H. S., Lai, K. K., & Wang, W. (2004). XML-based decision support systems: case study for portfolio selection. *International Journal of Information Technology and Decision Making*, *3*(4), 651-662.
(page 57)
- Dotsika, F. (2003). From data to knowledge in e-health applications: an integrated system for medical information modelling and retrieval. *Medical Informatics and the Internet in Medicine*, *28*(4), 231-251.
(page 57)
- Douglas, K., & Webster, C. (1999). The HCR-20 violence risk assessment scheme. *Criminal Justice and Behavior*, *26*(1), 3.
(page 207)
- Dowding, D., Mitchell, N., Randell, R., Foster, R., Lattimer, V., & Thompson, C. (2009). Nurses' use of computerised clinical decision support systems: a case site analysis. *Journal of Clinical Nursing*, *18*(8), 1159–1167.
(page 165)
- Doyle, M., & Dolan, M. (2000). Violence risk prediction. *American Journal of Psychiatry*, *177*, 303-311.
(page 22)
- Dumontier, M. (2010). Building an effective Semantic Web for health care and the life sciences. *Semantic Web*, *1*(1), 131–135.
(page 209)
- East, T., Heermann, L., Bradshaw, R., Lugo, A., Sailors, R., Ershler, L., et al. (1999). Efficacy of computerized decision support for mechanical ventilation: results of a prospective multi-center randomized trial. In *Proceedings of the amia symposium* (p. 251).
(page 40)
- Edelstein, B., Heisel, M., McKee, D., Martin, R., Koven, L., Duberstein, P., et al. (2009). Development and Psychometric Evaluation of the Reasons for Living–Older Adults Scale: A Suicide Risk Assessment Inventory. *The Gerontologist*.

- (page 207)
- Estes, W. K. (1986). Memory storage and retrieval processes in category learning. *Journal of Experimental Psychology: General*, *115*(2), 155-174. (page 29)
- Eylon, B., & Reif, F. (1984). Effects of knowledge organization on task performance. *Cognition and Instruction*, *1*(1), 5-44. (pages 30 and 208)
- Fan, W., Chan, C., & Garofalakis, M. (2004). Secure XML querying with security views. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data* (pp. 587-598). (page 170)
- Fernández-López, M., & Gómez-Pérez, A. (2002). Overview and analysis of methodologies for building ontologies. *The Knowledge Engineering Review*, *17*(02), 129-156. (page 50)
- Ferraté, A. (2010). *Google Wave: Up and Running*. Sebastopol, CA: O'Reilly Media. (page 66)
- Fiedler, K., & Armbruster, T. (1994). Two halves may be more than one whole: Category split effects on frequency illusions. *Journal of Personality and Social Psychology*, *66*, 633-645. (page 39)
- Fiske, S. T., & Taylor, S. E. (1991). *Social cognition* (2nd ed.). New York: McGraw-Hill. (page 29)
- Fitts, P. (1964). Perceptual-Motor Skill Learning. In *Categories of human learning*. Academic Press Inc. (page 38)
- Fitzgerald, G., & Russo, N. (2005). The turnaround of the London ambulance service computer-aided despatch system (LASCAD). *European Journal of Information Systems*, *14*(3), 244-257. (page 165)
- Fitzmaurice, D., Hobbs, F., Murray, E., Holder, R., Allan, T., & Rose, P. (2000). Oral anti-coagulation management in primary care with the use of computerized decision support and near-patient testing: a randomized, controlled trial. *Archives of Internal Medicine*, *160*(15), 2343. (page 45)
- Fitzpatrick, G. (2004). Integrated care and the working record. *Health Informatics Journal*, *10*(4), 291.

- (page 145)
- Freyhof, H., Gruber, H., & Ziegler, A. (1992). Expertise and hierarchical knowledge representation in chess. *Psychological Research*, *54*(1), 32–37. (pages 31 and 62)
- Garg, A., Adhikari, N., McDonald, H., Rosas-Arellano, M., Devereaux, P., Beyene, J., et al. (2005). Effects of computerized clinical decision support systems on practitioner performance and patient outcomes a systematic review. *Journal of the American Medical Association*, *293*(10), 1223–1238. (pages 38, 40, 41, and 45)
- Garner, W. R. (1974). *The processing of information and structure*. Hillsdale, NJ: Erlbaum. (page 28)
- Gauld, R. (2007). Public sector information system project failures: Lessons from a New Zealand hospital organization. *Government Information Quarterly*, *24*(1), 102–114. (page 165)
- Gennari, J. H., Musen, M. A., Ferguson, R. W., Grosso, W. E., Crubézy, M., Eriksson, H., et al. (2003). The evolution of Protégé: an environment for knowledge-based systems development. *International Journal of Human-Computer Studies*, *58*(1), 89 - 123. (pages 39, 64, and 199)
- Gigerenzer, G. (1994). Why the distinction between single-event probabilities and frequencies is important for psychology (and vice versa). In G. Wright & P. Ayton (Eds.), *Subjective probability* (p. 129-161). New York: Wiley. (page 39)
- Gigerenzer, G., & Hoffrage, U. (1995). How to improve bayesian reasoning without instructions: Frequency formats. *Psychological Review*, *102*(4), 684-704. (page 39)
- Griffiths, T., Chater, N., Kemp, C., Perfors, A., & Tenenbaum, J. (2010). Probabilistic models of cognition: exploring representations and inductive biases. *Trends in cognitive sciences*, *14*(8), 357–364. (page 28)
- Gruber, T. (1995). Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human Computer Studies*, *43*(5), 907–928. (pages 39 and 103)
- Haarslev, V., & Möller, R. (2003). Racer: An owl reasoning agent for the semantic web. In *Proceedings of the international workshop on applications, products and services of web-based support systems, in conjunction with the* (pp. 91–95).

-
- (page 200)
- Haghighi, M., Koeda, M., Takai, T., & Tanaka, H. (2009). Development of clinical ontology for mood disorder with combination of psychomedical information. *Journal of medical and dental sciences*, *56*, 1–15. (page 209)
- Halcomb, E. J., & Davidson, P. M. (2006). Is verbatim transcription of interview data always necessary? *Applied Nursing Research*, *19*(1), 38 - 42. (page 61)
- Hanson, R. K. (2005). Twenty years of progress in violence risk assessment. *Journal of Interpersonal Violence*, *20*(2), 212-217. (page 22)
- Harold, R. E., & Means, W. S. (2002). *XML in a nutshell* (2nd ed.). Sebastopol,CA: O.Reilly. (page 73)
- Hawley, C. J., Gale, T. M., Sivakumaran, T., & Littlechild, B. (2010). Risk assessment in mental health: Staff attitudes and an estimate of time cost. *Journal of Mental Health*, *19*(1), 88–98. (pages 22 and 207)
- Hawley, C. J., Littlechild, B., Sivakumaran, T., Sender, H., Gale, T. M., & Wilson, K. J. (2006). Structure and content of risk assessment proformas in mental healthcare. *Journal of Mental Health*, *15*, 437-448. (pages 22, 166, and 207)
- Hegazy, S. E., & Buckingham, C. D. (2009a). iARRIVE: an incremental algorithm for robust relative influence values elicitation. In *eTELEMED* (p. 239-244). IEEE Computer Society. (page 109)
- Hegazy, S. E., & Buckingham, C. D. (2009b). 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. (page 109)
- Hegazy, S. E., & Buckingham, C. D. (2010). Modulating membership grades to gain consensus for fuzzy set uncertainty values in a clinical decision support system. In *Advances in human-oriented and personalized mechanisms, technologies and services (centric), 2010 third international conference on* (p. 40 -45). (page 98)

-
- Hendy, J., Reeves, B., Fulop, N., Hutchings, A., & Masseria, C. (2005). Challenges to implementing the national programme for information technology (NPfIT): a qualitative study. *British Medical Journal*, *331*(7512), 331–336. (page 157)
- Higgins, N., Watts, D., Bindman, J., Slade, M., & Thornicroft, G. (2005). Assessing violence risk in general psychiatry. *Psychiatric Bulletin*, *29*, 131-133. (pages 22, 166, and 207)
- Hoekstra, R. (2009). *Ontology representation: Design patterns and ontologies that make sense*. Amsterdam, Netherlands: IOS Press. (pages 55 and 209)
- Holdsworth, N., & Dodgson, G. (2003). Could a new Mental Health Act distort clinical judgement? a Bayesian justification of naturalistic reasoning about risk. *Journal of Mental Health*, *12*(5), 451-462. (page 22)
- Holman, G. K. (2003). *Definitive XSL-FO*. Upper Saddle River, N.J: Prentice Hall. (page 147)
- Hougham, M. (1996). London Ambulance Service computer-aided despatch system. *International Journal of Project Management*, *14*(2), 103–110. (page 165)
- Huang, C., Li, J., & Ross, K. (2007). Can internet video-on-demand be profitable? In *Proceedings of the 2007 conference on applications, technologies, architectures, and protocols for computer communications* (pp. 133–144). (page 139)
- Hunt, T. (2010). Natural or artificial primary key? Using the Mifrenz children’s email application as a case study. *New Zealand Journal of Applied Computing and Information Technology (NZJACIT)*, *14*(1), 16–23. (page 107)
- Iliffe, S., Austin, T., Wilcock, J., Bryans, M., Turner, S., & Downs, M. (2002). Design and implementation of a computer decision support system for the diagnosis and management of dementia syndromes in primary care. *Methods of information in medicine*, *41*(2), 98–104. (page 45)
- Jacobson, D., & Jacobson, J. (2002). *Flash and XML: a developer’s guide*. London: Addison-Wesley. (page 64)

-
- Jain, A. K., Prabhakar, S., & Pankanti, S. (2002). On the similarity of identical twin fingerprints. *Pattern Recognition*, 35(11), 2653 - 2663. (page 121)
- Jovanović, J., & Gašević, D. (2005). Achieving knowledge interoperability: an XML/XSLT approach. *Expert Systems with Applications*, 29, 535-553. (page 57)
- Kahneman, D., & Tversky, A. (1973). On the psychology of prediction. *Psychological review*, 80, 237-251. (page 39)
- Kawamoto, K., Houlihan, C., Balas, E., & Lobach, D. (2005). Improving clinical practice using clinical decision support systems: a systematic review of trials to identify features critical to success. *British Medical Journal*, 330(7494), 765. (pages 43, 44, 45, and 98)
- Kay, M. R. (2001). *XSLT programmer's reference*. Wrox press. (page 57)
- Keogh, B., El-Sayed, D., & Pilkington, T. (2008). *NHS Number Standard for Secondary Care (England) – Appendix SC-E1: Exemplar Site Report: North Bristol NHS Trust* (Tech. Rep.). NHS Connecting for Health. Available from <http://www.connectingforhealth.nhs.uk/systemsandservices/nhsnumber/staff/documents/scappbristol.pdf> (page 184)
- Kettles, A. M., & Woods, P. (2009). The theory of risk. In P. Woods & A. M. Kettles (Eds.), *Risk Assessment and Management in Mental Health Nursing* (pp. 49–76). Chichester, UK: John Wiley and Sons. (page 145)
- Kline, P. (2000). *Handbook of psychological testing* (2nd ed.). London: Routledge. (page 92)
- Knublauch, H., Tetlow, P., Wallace, E., & Oberle, D. (2006, March). *A semantic web primer for object-oriented software developers* (W3C Note). W3C. (<http://www.w3.org/TR/2006/NOTE-sw-oosd-primer-20060309/>) (page 53)
- Koehler, D. J. (2000). Probability judgment in three-category classification learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 26(1), 28-52. (page 39)
- Koehler, D. J., White, C. M., & Grondin, R. (2003). An evidential support accumulation model of subjective probability. *Cognitive Psychology*, 46(2), 152-197.

-
- (page 39)
- Kola, J., Harris, J., Lawrie, S., Rector, A., Goble, C., & Martone, M. (2010). Towards an ontology for psychosis. *Cognitive Systems Research*, 11(1), 42–52. (page 209)
- Kroll, L., Bailey, S., Myatt, T., McCarthy, K., Shuttleworth, J., Rothwell, J., et al. (2003). *Mental Health Screening Tool: SIFA*. (www.youth-justiceboard.gov.uk/nr/rdonlyres/fl1eda350-70db-437c-8f3f-5d19fd57e533/0/sifa.pdf) (pages 67 and 88)
- Kruschke, J. K. (2006). Locally Bayesian learning with applications to retrospective revaluation and highlighting. *Psychological Review*, 113(4), 677–698. (page 29)
- Kruschke, J. K. (2010). Bridging levels of analysis: comment on McClelland et al. and Griffiths et al. *Trends in cognitive sciences*, 14(8), 344. (page 28)
- Lacy, L. (2005). *OWL: Representing information using the web ontology language*. Crewe, UK: Trafford Publishing. (page 209)
- Landeta, J. (2006). Current validity of the delphi method in social sciences. *Technological Forecasting and Social Change*. (page 70)
- Lanzenberger, M., Sampson, J., & Rester, M. (2010). Ontology visualization: Tools and techniques for visual representation of semi-structured meta-data. *Journal of Universal Computer Science*, 16(7), 1036–1054. (page 200)
- Laplante, P., Zhang, J., & Voas, J. (2008). What's in a Name? Distinguishing between SaaS and SOA. *IT Professional*, 10(3), 46–50. (page 183)
- Larkin, J. (1980). Skilled problem solving in physics: A hierarchical planning model. *Journal of Structural Learning*, 6(4), 269–294. (pages 30 and 208)
- Lee, J., & Ware, B. (2003). *Open source Web development with LAMP: using Linux, Apache, MySQL, Perl, and PHP*. London: Addison Wesley. (page 149)

-
- Leopold, J., Coalter, A., & Lee, L. (2009). A Generic, Functionally Comprehensive Approach to Maintaining an Ontology as a Relational Database. In *Proceedings of the ICOSE 2009: International Conference on Ontological and Semantic Engineering*. ICOSE. (page 54)
- Lewis, A. (2002). Health informatics: information and communication. *Advances in Psychiatric Treatment*, 8(3), 165. (page 147)
- Lewis, G., Sharp, D., Bartholomew, J., & Pelosi, A. (1996). Computerized assessment of common mental disorders in primary care: effect on clinical outcome. *Family Practice*, 13(2), 120. (pages 41 and 45)
- Loney, K. (2009). *Oracle database 11g: the complete reference*. New York: McGraw-Hill Professional. (page 161)
- Luciano, J., & Stevens, R. (2008). OWL: PAX of mind or the AX? Experiences of Using OWL in the Development of BioPAX. *OWL: Experiences and Directions, Gaithersburg, MD, USA*. (page 40)
- Maden, A. (2001). Practical application of structured risk assessment. *The British Journal of Psychiatry*, 178, 479. (page 22)
- Maden, A. (2003). Standardised risk assessment: why all the fuss? *Psychiatric Bulletin*, 25, 129-131. (page 22)
- Maden, A., Scott, F., Burnett, R., Lewis, G., & Skapinakis, P. (2004). Offending in psychiatric patients after discharge from medium secure units: prospective national cohort study. *British Medical Journal*, 328, 1534. (page 22)
- Mangano, S. (2006). *XSLT cookbook* (2nd ed.). Sebastopol, CA: O'Reilly. (page 151)
- Marian, A., Abiteboul, S., Cobéna, G., & Mignet, L. (2001). Change-centric management of versions in an XML warehouse. In *Proceedings of the international conference on very large data bases* (pp. 581–590). (page 130)

-
- Maviglia, S., Zielstorff, R., Paterno, M., Teich, J., Bates, D., & Kuperman, G. (2003). Automating complex guidelines for chronic disease: lessons learned. *Journal of the American Medical Informatics Association*, *10*(2), 154. (page 46)
- Mays, N., & Pope, C. (2000). Qualitative research in health care: Assessing quality in qualitative research. *British Medical Journal*, *320*, 50-52. (page 63)
- McClelland, J., Botvinick, M., Noelle, D., Plaut, D., Rogers, T., Seidenberg, M., et al. (2010). Letting structure emerge: connectionist and dynamical systems approaches to cognition. *Trends in Cognitive Sciences*, *14*(8), 348-356. (page 28)
- McGuinness, D. L., & Harmelen, F. van. (2004, February). *OWL web ontology language overview* (W3C Recommendation). W3C. (<http://www.w3.org/TR/2004/REC-owl-features-20040210/>) (page 52)
- Mealling, M., & Denenberg, R. (2002). *RFC 3305: Uniform Resource Identifiers (URIs), URLs, and Uniform Resource Names (URNs): Clarifications and Recommendations* (Tech. Rep.). IETF. Available from <http://tools.ietf.org/html/rfc3305> (page 105)
- Medin, D. L., & Schaffer, M. M. (1978). Context theory of classification learning. *Psychological Review*, *85*(3), 207-238. (page 28)
- Meyer, M. A., & Booker, J. M. (2001). *Eliciting and analyzing expert judgment: A practical guide*. Philadelphia: SIAM. (page 80)
- Mikroyannidis, A., & Theodoulidis, B. (2010). Ontology management and evolution for business intelligence. *International Journal of Information Management*. (page 106)
- Miller, G. A. (1956). The magical number seven plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, *63*, 81-97. (page 92)
- Mittelbach, F., & Goossens, M. (2004). *The LaTeX companion* (2nd ed.). Boston: Pearson. (page 147)
- Monahan, J., Steadman, H. J., Appelbaum, P. A., Robbins, P. C., Mulvey, E. P., Silver, E. R., et al. (2000). Developing a clinically useful actuarial tool for assessing violence risk. *British Journal of Psychiatry*, *176*, 312-319.

- (page 22)
- Motik, B. (2007). On the Properties of Metamodeling in OWL. *Journal of Logic and Computation*, 17(4), 617. (page 52)
- Murphy, G., & Lassaline, M. (1997). Hierarchical structure in concepts and the basic level of categorization. In K. Lamberts & D. Shanks (Eds.), *Knowledge, concepts and categories* (pp. 93–131). Cambridge, MA: MIT Press. (pages 31 and 62)
- Musen, M., Shahar, Y., & Shortliffe, E. (2001). Clinical decision-support systems. In E. Shortliffe, L. Perreault, G. Wiederhold, & L. Fagan (Eds.), *Medical Informatics: Computer Applications in Health Care and Biomedicine. Second ed.* (pp. 573–609). New York: Springer. (pages 38 and 39)
- National Confidential Inquiry. (2006). *Avoidable deaths: Five year report of the National Confidential Inquiry into suicide and homicide by people with mental illness.* (<http://www.medicine.manchester.ac.uk/psychiatry/research/suicide/prevention/nci/reports/avoidabledeathsfullreport.pdf>) (page 22)
- National Risk Management Programme. (2007). *Best practice in managing risk.* London: Department of Health. (pages 23 and 207)
- Neuendorf, K. (2002). *The content analysis guidebook.* London: Sage. (page 61)
- NHS Executive. (1999). *Mental Health National Service Framework.* London: Department of Health. (page 21)
- Nixon, R. (2009). *Learning PHP, MySQL, and JavaScript* (1st ed.). Sebastopol, CA: O'Reilly. (page 151)
- Nosofsky, R. M. (1986). Attention, similarity, and the identification-categorization relationship. *Journal of Experimental Psychology: General*, 115(1), 39-57. (page 28)
- Nosofsky, R. M. (1992). Exemplars, prototypes, and similarity rules. In A. F. Healy, S. M. Kosslyn, & R. M. Shiffrin (Eds.), *From learning theory to connectionist theory. Essays in honor of William K. Estes* (Vol. 1, p. 149-167). Hillsdale: New Jersey.

-
- (page 28)
- Nosofsky, R. M., Kruschke, J. K., & McKinley, S. C. (1992). Combining exemplar-based category representation and connectionist learning rules. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 18(2), 211-233. (page 28)
- Novak, J. D. (2003). The promise of new ideas and new technology for improving teaching and learning. *Cell Biology Education*, 2, 122-132. (page 61)
- Patel, A., Harrison, A., & Bruce-Jones, W. (2009). Evaluation of the risk assessment matrix: a mental health triage tool. *British Medical Journal*, 26(1), 11. (page 89)
- Peleg, M., Boxwala, A., Bernstam, E., Tu, S., Greenes, R., & Shortliffe, E. (2001). Sharable representation of clinical guidelines in GLIF: relationship to the Arden Syntax. *Journal of Biomedical Informatics*, 34(3), 170-181. (page 38)
- Peleg, M., & Tu, S. (2006). Decision support, knowledge representation and management in medicine. *Methods Inf Med*, 45(Suppl 1), 72-80. (pages 38 and 45)
- Pfleeger, S. L., & Atlee, J. M. (2009). *Software Engineering: Theory and Practice (4th Edition)*. NJ: Prentice Hall. (page 105)
- Posner, M. I., & Keele, S. W. (1968). On the genesis of abstract ideas. *Journal of Experimental Psychology*, 77, 353-363. (page 29)
- Power, D. (2002). *Decision Support Systems: Concepts and Resources for Managers*. Westport, CT: Quorum Books. (page 103)
- Rector, A., & Welty, C. (2005). Simple part-whole relations in OWL Ontologies [W3C Working Draft]. (<http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/>) (page 209)
- Roaldset, J., Hartvig, P., & Bjørkly, S. (2010). V-RISK-10: Validation of a screen for risk of violence after discharge from acute psychiatry. *European Psychiatry*. (pages 67 and 88)

-
- Rogers, A., & Mead, N. (2004). More than technology and access: primary care patients' views on the use and non-use of health information in the Internet age. *Health & Social Care in the Community*, 12(2), 102–110. (page 165)
- Rollman, B., Hanusa, B., Lowe, H., Gilbert, T., Kapoor, W., & Schulberg, H. (2002). A randomized trial using computerized decision support to improve treatment of major depression in primary care. *Journal of General Internal Medicine*, 17(7), 493–503. (page 41)
- Rönnau, S., Pauli, C., & Borghoff, U. (2008). Merging changes in XML documents using reliable context fingerprints. (page 130)
- Rottenstreich, Y., & Tversky, A. (1997). Unpacking, repacking, and anchoring: Advances in support theory. *Psychological Review*, 104, 406–415. (page 39)
- Rouse, W., & Morris, N. (1986). Understanding and enhancing user acceptance of computer technology. *IEEE Transactions on Systems, Man, and Cybernetics*, 16, 965–973. (page 165)
- Roy, A., Matthews, H., Clifford, P., Fowler, V., & Martin, D. (2002). Health of the Nation Outcome Scales for People with Learning Disabilities (HoNOS-LD). *The British Journal of Psychiatry*, 180(1), 61. (page 193)
- Ruland, C., & Bakken, S. (2002). Developing, implementing, and evaluating decision support systems for shared decision making in patient care: a conceptual model and case illustration. *Journal of biomedical informatics*, 35(5-6), 313–321. (page 43)
- Saint-Andre, P. (2005). Streaming XML with Jabber/XMPP. *IEEE Internet Computing*, 82–89. (page 66)
- Sandhu, R., & Samarati, P. (2002). Access control: principle and practice. *Communications Magazine, IEEE*, 32(9), 40–48. (page 169)
- Sarang, P. (2006). *Pro Apache XML*. Berkley, CA: Apress. (page 161)
- Schmitt, C. (2010). *CSS cookbook* (3rd ed.). Sebastopol, CA: O'Reilly.

- (page 147)
- Schriger, D., Gibbons, P., Langone, C., Lee, S., & Altshuler, L. (2001). Enabling the diagnosis of occult psychiatric illness in the emergency department: A randomized, controlled trial of the computerized, self-administered PRIME-MD Diagnostic System. *Annals of emergency medicine*, 37(2), 132–140. (page 41)
- Scott, J., Rundall, T., Vogt, T., & Hsu, J. (2005). Kaiser Permanente's experience of implementing an electronic medical record: a qualitative study. *British Medical Journal*, 331(7528), 1313–1316. (page 165)
- Scott, M. (2009). *Simply Effective Cognitive Behaviour Therapy: A Practitioner's Guide*. New York, NY: Routledge. (page 196)
- Seibel, P. (2005). *Practical common lisp*. Berkeley, California: Apress. (page 95)
- Shortliffe, E. (1976). *Computer-based medical consultations: MYCIN*. New York: Elsevier. (page 38)
- Siau, K., & Tan, X. (2005). Improving the quality of conceptual modelling using cognitive mapping techniques. *Data & Knowledge Engineering*, 55, 343–365. (page 61)
- Silverman, D. (Ed.). (2004). *Qualitative research: theory, method and practice*. London: Sage. (page 72)
- Sim, I., Gorman, P., Greenes, R., Haynes, R., Kaplan, B., Lehmann, H., et al. (2001). Clinical decision support systems for the practice of evidence-based medicine. *Journal of the American Medical Informatics Association*, 8(6), 527. (pages 38, 43, 45, and 98)
- Simon, S., Kaushal, R., Cleary, P., Jenter, C., Volk, L., Orav, E., et al. (2007). Physicians and electronic health records: a statewide survey. *Archives of internal medicine*, 167(5), 507. (page 43)
- Singleton, N., Bumpstead, R., Lee, A., & Meltzer, H. (2003). Psychiatric morbidity among adults living in private households, 2000. *International Review of Psychiatry*, 15(1), 65–73. (page 21)
- Singleton, N., & Lewis, G. (2003). *Better or worse: a longitudinal study of the mental health of adults living in private households in Great Britain*. Department of Health.

- (page 21)
- Sittig, D., Wright, A., Osheroff, J., Middleton, B., Teich, J., Ash, J., et al. (2008). Grand challenges in clinical decision support. *Journal of Biomedical Informatics*, 41(2), 387–392.
(pages 42, 45, and 201)
- Skegg, K. (2005). Self-harm. *Lancet*, 366, 1471-1483.
(page 22)
- Smith, J. D., & Minda, J. P. (1998). Prototypes in the mist: the early epochs of category learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 24, 1411-1430.
(page 29)
- Smith, J. D., & Minda, J. P. (2000). Thirty categorization results in search of a model. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 26, 3-27.
(page 29)
- Spackman, K., Campbell, K., Côté, R., et al. (1997). SNOMED RT: a reference terminology for health care. In *Proceedings of the amia annual fall symposium* (p. 640).
(page 201)
- Stallings, W. (2009). *Operating systems : Internals and design principles*. Upper Saddle River NJ.
(page 66)
- Stewart, A., Lawrence, A., & Edwards, D. (2010). *Uptake of Decision Support Systems in the Forestry Sector in Great Britain* (Tech. Rep.). Forestry Commission. Available from [http://www.forestry.gov.uk/pdf/Uptake_of_DSS_Scoping_Report_Sept2010.p%df/\\$FILE/Uptake_of_DSS_Scoping_Report_Sept2010.pdf](http://www.forestry.gov.uk/pdf/Uptake_of_DSS_Scoping_Report_Sept2010.p%df/$FILE/Uptake_of_DSS_Scoping_Report_Sept2010.pdf)
(page 165)
- Tabbers, H., Kester, L., Hummel, H., & Nadolski, R. (2004). Interface design for digital courses. *Integrated E-learning: Implications for pedagogy, technology and organization*, 100–111.
(page 202)
- Tamblyn, R., Huang, A., Perreault, R., Jacques, A., Roy, D., Hanley, J., et al. (2003). The medical office of the 21st century (MOXXI): effectiveness of computerized decision-making support in reducing inappropriate prescribing in primary care. *CMAJ: Canadian Medical Association Journal*, 169(6), 549–556.
(page 40)
- Tamineé, O., & Dillmann, R. (2003). KaViDo. A web-based system for collaborative research and development processes. *Computers in Industry*, 42, 29-45.

-
- (page 57)
- Tidwell, D. (2008). *XSLT, Second Edition*. Sebastopol, CA: O'Reilly. (page 73)
- Timm, J., & Gannod, G. (2005). A Model-Driven Approach for Specifying Semantic Web Services. In *Proceedings of the IEEE International Conference on Web Services* (pp. 313–320). (page 40)
- Tomić, B., Jovanović, J., & Devedžić, V. (2006). Javadon: an open-source expert system shell. *Expert Systems with Applications*, 31, 595-606. (page 57)
- Trivedi, M., Daly, E., Kern, J., Grannemann, B., Sunderajan, P., & Claassen, C. (2009). Barriers to implementation of a computerized decision support system for depression: an observational report on lessons learned in "real world" clinical settings. *BMC Medical Informatics and Decision Making*, 9(1), 6. (pages 47 and 48)
- Tudorache, T., Noy, N., Tu, S., & Musen, M. (2008). Supporting collaborative ontology development in protégé. In *Proceedings of the 7th international conference on the semantic web* (pp. 17–32). (page 200)
- Tversky, A., & Kahneman, D. (1982). Evidential impact of base rates. In D. Kahneman, P. Slovic, & A. Tversky (Eds.), *Judgement under uncertainty: Heuristics and biases* (p. 153-160). Cambridge: Cambridge University Press. (page 39)
- Vaswani, V. (2009). *MySQL Database Usage & Administration*. New York: McGraw-Hill Professional. (page 161)
- Vickers, B. (1994). Designing layered functionality within group decision support systems. *Decision Support Systems*, 11(1), 83–99. (page 89)
- W3C OWL Working Group. (2009). *OWL 2 web ontology language document overview* (W3C Recommendation). W3C. (<http://www.w3.org/TR/2009/REC-owl2-overview-20091027/>) (page 209)
- Waters, B. (2005). Software as a service: A look at the customer benefits. *Journal of Digital Asset Management*, 1(1), 32–39.

- (page 183)
- Watts, D., Bindman, J., Slade, M., Holloway, F., Rosen, A., & Thornicroft, G. (2004). Clinical assessment of risk decision support (CARDS): The development and evaluation of a feasible violence risk assessment for routine psychiatric practice. *Journal of Mental Health*, *13*(6), 569-581. (pages 67, 88, and 89)
- Weibel, S. (1997). The Dublin Core: a simple content description model for electronic resources. *Bulletin of the American Society for Information Science and Technology*, *24*(1), 9-11. (page 52)
- Weingart, S., Toth, M., Sands, D., Aronson, M., Davis, R., & Phillips, R. (2003). Physicians' decisions to override computerized drug alerts in primary care. *Archives of internal medicine*, *163*(21), 2625. (page 42)
- Weir, C., Lees, K., MacWalter, R., Muir, K., Wallesch, C., McLelland, E., et al. (2003). Cluster-randomized, controlled trial of computer-based decision support for selecting long-term anti-thrombotic therapy after acute ischaemic stroke. *QJM: monthly journal of the Association of Physicians*, *96*(2), 143. (page 45)
- Whiteford, H. (2003). Responding to the burden of mental illness. In J. Sussex (Ed.), *Mental health economics and policy in a global context* (p. 5-8). Office of Health Economics. (page 21)
- Wing, J., Beevor, A., Curtis, R., Park, S., Hadden, S., & Burns, A. (1998). Health of the Nation Outcome Scales (HoNOS). research and development. *The British Journal of Psychiatry*, *172*(1), 11. (page 193)
- Wirth, N. (1971). Program development by stepwise refinement. *Communications of the ACM*, *14*(4), 221-227. (page 208)
- Wright, A., Sittig, D., Ash, J., Sharma, S., Pang, J., & Middleton, B. (2009). Clinical decision support capabilities of commercially-available clinical information systems. *Journal of the American Medical Informatics Association*, *16*(5), 637. (page 45)
- Zadeh, L. A. (1965). Fuzzy sets. *Information Control*, *8*, 338-353. (page 30)

Appendices

A

Tree Manipulation Keywords Used Within Structured Comments

Focus groups were used to review and continue restructuring of risk trees that had been returned with comments by the individual experts. Some of the changes required were additional to the functionality provided by the Flash program and so keywords denoting these changes were put into the comment boxes of nodes that the focus groups determined should be altered in the manner defined by the keyword. XSLT was then used to detect the keywords and execute the instructions accordingly, which bypassed having to change the Flash software. This appendix provides the keyword definitions.

- **GENERIC** [*path to generic node*] means the concept is defined generically and the new definition will replace the current one. Display node name and the statement that the node is generically defined elsewhere. Do not display any subcomponents. The XSLT transformation deletes subcomponents and instead inserts a `generic="path to generic node"` attribute.

-
- **GENERIC-DATUM** [*path to generic node*] means the datum repeats and is held elsewhere, within a *generic nodes* section of the tree. Used for purposes of copying over question text from the generic location via XSLT, so that a question is not continually repeated manually in all instances.
 - **DELETE LEVEL** Take all the retained children of this node and attach them directly to the parent node.
 - **COPY:** [*path to node to copy*] define this node by the node at the given path following the copy keyword. It overwrites the subcomponents of the node with those at the path.
 - **ADD:** *name [new name]; path [path]; description [description]*. For nodes that must be added as subcomponents to the added node, either from node at given path, or as described. If it is desired to add a node that exists elsewhere but where the subnodes are required only as help for the added node, then *path* keyword should be replaced by a lower-case *add-help* keyword. Note that it is therefore not possible to have both a *path* and an *add-help*.
 - **REMOVE** For nodes that are no longer part of the sense of the parent concept because of some change to that concept. So the node is not to be part of the help box and must be completely removed from the structure, compared to other deleted nodes which become part of the parent concept definition (as help).
 - **REORDER** [*n*] Put this node in *nth* position with respect to the siblings (or place at the end).
 - **HELP** [*path to node that is to be included as part of the defining help for the node*]. There may be several nodes to include, in which case the paths are separated by a double asterisk, as follows: `HELP [generic concepts >> appearance indicators of self neglect >> skin >> colour ** generic concepts >> appearance indicators of self neglect >> skin >> dehydrated]`. “Phantom” node names (free text or deleted nodes) can also be added.
 - **HELP INSTRUCTIONS** [*some instructions*]. Free form text about the help attribute.
 - **LINK** [*path to node*] Used to show links between nodes that will be accommodated eventually by ri-modifiers.

-
- RELOCATE [*instructions*]. Used if it is easier to explain moves of a concept rather than using the more laborious *add* and *delete* approach.
 - VALUES [*data type or values to include for node*] Nodes with qualitative answers that do not lie on a range are shown by this code word, followed by the values it can take (which must be mutually exclusive). These have automatic mgs of 1 or 0 assigned, depending on whether they are present or absent. If there is no values keyword in the comment box, it is to be assumed that values are along a scale. Additionally defined data types include *integer*, *date-year*, *date-day* etc.
 - [*g*] To denote generic nodes that are to be kept as cohesive wholes: “g” for “generic”. These nodes will have the same name wherever they occur.
 - [*gd*] for generic concepts that are clearly not homogenous may be kept but with names that reflect their different locations. This is the case for feelings/emotions and, possibly, social context. These nodes will be labelled “gd” for “generic distinct”. They will have names that reflect their locations, but their structures will be the same. The difference will be in the internal RIs.

B

Stages Involved in Enacting Tree Changes Using XSLT

Enacting of the tree-manipulation directives derived from structured comments was carried out using XSLT. An algorithmic approach was taken to achieving this, with the manipulations broken down into modular stages. Stylesheets were created to implement the stages. The Knowledge hierarchy XML file was then transformed by passing it through each of the stages. This appendix provides a brief description of each stage of transformation.

STAGE 1: REORDER (phase 1) : Preparation for reorder. Involves creating temporary attributes.

STAGE 2: REORDER (phase 2) Applies the re-order directive and removes all the re-order and temporary attributes.

STAGE PRE-COPY: Convert user-friendly paths to XPath paths. Escape all the apostrophes from node labels and create temporary attributes with the actual `label`, and actual `renamedLabel`. This is because XPath statements that are to be created and executed dynamically (e.g., for copy operations) are problematic if the statement has an apostrophe in it.

STAGE 3: COPY (pass 1) : This involves selecting each node element having a `copy` attribute. These elements' sub-elements are then deleted. The descendants of the node referred to in the `copy` attribute are then copied over to the present element. The `copy` keyword is no longer included as an attribute of the current element.

STAGE POST-COPY: Undo that which was done by the PRE-COPY stage.

STAGE 4: ADD: This involves converting all the “added-node” elements into “node” elements and creating a (or prepending to any pre-existing) `comment` to indicate this as being a user created node. Additionally appending to the comment anything that may have been contained in a `description` attribute. The result of this is that added nodes will become normal nodes and will not have a `description` attribute.

STAGE PRE-COPY: *As described earlier.*

STAGE 5: COPY (pass 2): This involves selecting each “node” element having a “copy” attribute (there will be some because “added-node” elements will not have been touched by COPY (pass 1)). These elements' sub-elements are then deleted. The descendants of the node referred to in the `copy` attribute are then copied over to the present element. The `copy` keyword is no longer included as an attribute of the current element. NOTE: in the intermediate xml file (which will have been generated prior to STAGE 1), any paths that were part of the “added-node” definition will have been transformed into `copy` attributes. So COPY (pass 2) will have the effect of completing the ADD operation.

STAGE POST-COPY: *As described earlier.*

STAGE PRE-COPY: *As described earlier.*¹

STAGE 6: HELP (phase 1): Parse all paths from the `help` attributes and replace attributes with ‘help’ nodes containing the content of the trees referenced by the paths. (NB: this is slightly different from a COPY because here the referenced node and its descendants are also copied, whereas in COPY it is just the descendants of the referenced node). Also NB: there are two types of content that the stylesheet can expect for a `help` attribute: *a*) a *path*, *b*) a short-form path to a sibling node i.e., just the name of a sibling node. Both types will be processed as described above. Multiple paths within a `help` attribute can

¹This stage and the one preceding, although strictly not necessary, have been introduced for reasons of maintaining modularity.

be separated by ‘ ** ’. However, within the `help` attribute there must only be paths of either type *a* or type *b* (but not both).

STAGE POST-COPY: *As described earlier.*

STAGE HELP-CLEANUP: Newly created ‘help’ nodes may have inside them, nodes that are marked as deleted or removed. Since these are descendants of a help node, these are not in fact to be deleted. Therefore the `delete` and `remove` attributes from descendants of ‘help’ nodes should simply be deleted.

STAGE 7: REMOVE: Take out all of the nodes that have a `remove` attribute and also, their descendants.

STAGE 8: HELP (phase 2): Check if ALL of any node’s ‘node’ children have a delete attribute. If so, then the node is (at a later stage) going to be a datum (because of the delete operations), hence add the subtree (i.e., nodes marked for deletion) of this node to the current node as a ‘help’ node.

STAGE HELP-CLEANUP: *As described earlier.*

STAGE HELP-TO-ATTRIBUTES: There isn’t a straightforward way to tell the flash tools’ XML tree component to ignore ‘help’ nodes. Therefore, this stage is introduced to take out the ‘help’ node and replace it with an attribute that contains paths of labels to each leaf node of the now defunct ‘help’ node. Each path is separated by a ‘ ** ’. At some future date, these paths could be extracted and trees built from them (if need be).

STAGE 9: DELETE: Delete all the nodes (and their sub-trees) that have a `delete` attribute.

STAGE PRE-COPY: *As described earlier.*

STAGE 10: GENERIC: Anything presently under the node marked by this attribute AND where the attribute is not equal to `g|gd` is deleted. The instance’s name is also changed to reflect the to-be-copied-from generic node’s name in cases where the to-be-copied-from generic node has a `rename` attribute against it. Generic nodes are instantiated wherever they appear via looking at the path in the ‘generic’ attribute of the current node and copying over the contents below that location (i.e., nothing from that location or above it) to the current location. In cases where there is a `g|gd` instead of a path, no instantiation is performed.

STAGE POST-COPY: *As described earlier.*

STAGE 11: DELETE LEVEL: Take all the retained children of this node and attach them directly to the parent node.

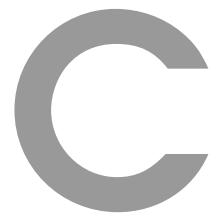
STAGE 12: RENAME: the rename directive is applied.

STAGE 13: COMMENT-CLEANUP: Any comments in the tree are removed.

NB: Relocates should be written in terms of *adds* and *removes*

NB: At the end of each stage, if an attribute type has become defunct it is removed from the XML file.

NB: Other attributes such as `help-instructions` and `link` will remain in the file untouched, since no transformation of these attributes would be logical.



Algorithm for expanding “multiple-tick” nodes

The `multiple-tick` attribute is one that applies to datum nodes that have multiple answers and that are subsequently converted into a concept for the RIT with each answer as a child. An example is the “*targets of harm to others*” node where there can be several targets, e.g.:

```
<node label="targets of harm to others" values="nominal"
  code="hto-targets"
  multiple-tick="(&quot;Has the person targeted any particular group of
  people rather than complete strangers?&quot; (
    (DOMESTIC &quot;Has the person harmed within a domestic setting?&quot;);)
    (FRIENDS-COLLEAGUES &quot;Has the person harmed friends/colleagues?&quot;);)
    (HEALTH-WORKERS &quot;Has the person harmed any health workers?&quot;);)
    (AUTHORITY-FIGS &quot;Has the person harmed any authority figures?&quot;);)
  )"/>
```

The algorithm below details the conversion process.

1. Create a `filter-q` attribute with the value equal to the first question in the `multiple-tick` attribute value.
 2. Create child nodes with `labels` and `codes` equal to labels contained in the association list that is within the `multiple-tick` attribute.
 3. Create and instantiate respective `question` attributes within each child from associated question data that is contained in the `multiple-tick` attribute.
 4. For each child, give it a `value-mg` attribute value of `"((yes 1)(no 0))"` and a `values="nominal"` attribute.
- Note: an empty `multiple-tick` attribute is retained within the concept so that data gathering tools can still render the nodes using checkboxes should they choose.

Application of the algorithm to the *"targets of harm to others"* node results in the following expansion:

```
<node label="targets of harm to others" code="hto-targets" values="nominal"
  multiple-tick=""
  filter-q="Has the person targeted any particular group of people
    rather than complete strangers?"
  <node label="domestic" code="domestic"
    question="Has the person harmed anyone within the
      domestic setting?"
    values="nominal" value-mg="((yes 1) (no 0))" />
  <node label="friends colleagues" code="friends-colleagues"
    question="Has the person harmed friends/colleagues?"
    values="nominal" value-mg="((yes 1) (no 0))" />
  <node label="health workers" code="health-workers"
    question="Has the person harmed any health workers?"
    values="nominal" value-mg="((yes 1) (no 0))" />
  <node label="authority figs" code="authority-figs"
    question="Has the person harmed any authority figures?"
    values="nominal" value-mg="((yes 1) (no 0))" />
</node>
```



Unified Reconciliation Algorithm

The algorithm outlined in pseudocode below is used to reconcile a fingerprint in RIT_{new} with the corresponding fingerprint(s) in $RIT_{original}$, and then retrieve associated RIs. The algorithm determines which of the edit cases outlined in table 8.2 the RIT_{new} fingerprint belongs to. Once the case(s) have been determined, the algorithm is in a position to query the reconciled node(s) for its RI value(s).

```
IF (able to find aaaaa' in STnew) THEN
  //this is not a multiple-tick child node
  IF (RITnew fingerprint is NOT hyphenated) THEN
    //RITnew fingerprint has the form aaaaa'
    IF (able to reconcile aaaaa' with aaaaa) THEN
      IF (RIToriginal contains a node with fingerprint aaaaa) THEN
        //this is TYPE 1
        retrieve the matched node's RI value
      ELSE
        //this is TYPE 3
        search the RIT for all fingerprints that end in -aaaaa.
```

```

        //each location of the fingerprint is an
        //instantiation of that node, hence
        retrieve all the matched nodes' RI values.
    END IF
END IF
ELSE
    //RITnew fingerprint has the form xxxxx'-aaaaa' | yyyyy'-aaaaa'
    //for simplicity, writing this as xxxxx'-aaaaa'
    IF (able to reconcile aaaaa' with aaaaa) THEN
        find xxxxx' in STnew
        IF (able to reconcile xxxxx' with xxxxx AND
            RIToriginal contains a node with fingerprint xxxxx-aaaaa) THEN
            //TYPE 4
            retrieve the matched node's RI value
        ELSE
            //TYPE 2 and/or 5
            search the RIT for all fingerprints ending in -aaaaa or aaaaa.
            //each location of the fingerprint is an
            //instantiation of that node, hence
            retrieve all the matched nodes' RI values.
        END IF
    END IF
END IF
ELSE
    //this case is a multiple-tick node's generated child
    MTChildCode = this node's node-code
    IF (RITnew fingerprint is hyphenated) THEN
        //this is TYPE 6, 7 or 10
        find xxxxx' in STnew
        IF (able to reconcile xxxxx' with xxxxx AND RIToriginal contains a
            node with MTChildCode and fingerprint beginning with xxxxx-) THEN
            //TYPE 6
            retrieve the matched node's RI value
        ELSE
            //TYPE 7 or 10
            search the RIT for all node instances with MTChildCode
            //each location is an instantiation of that node, hence
            retrieve all the matched nodes' RI values.
        END IF
    ELSE
        //TYPE 8 or 9
        search the RIT for all node instances with MTChildCode
        //each location is an instantiation of that node, hence
        retrieve all the matched nodes' RI values.
    END IF;
END IF

```



Rules and Algorithms for Generating QTs

This appendix details the general rules that govern combinations of attributes within a QT node. It then details the algorithms that are used to generate QTs at different Levels.

E.1 Rules Governing QT Attributes

The following rules should be used as guidance when generating QT nodes. These rules help to clarify any ambiguity in relation to the exact attributes that are to be present in specific use-cases.

1. An RIT will be used to generate separate QTs appropriate to each CAT Level.
2. An RIT node must contain a `filter-q` or a `question` for there to be an entry against the node in the QT.
3. The `layer` attribute only exists for questions that should be displayed for asking at the beginning of the assessment. The lower the value, the higher up the order of presentation of the question.

4. A QT node will never have both a question and a filter question to be asked. This is due to the QT's being matched to a tool Level. Therefore, in cases where a filter question is to be asked, the QT node will have `values="filter-q"` and `question="The filter question text"`.
5. An RIT node may contain a `filter-q` as well as a `layer` attribute at the same time. In these cases, the QT node will have `values="filter-q"`. This is because the node is to be treated exactly like a filter question node.
6. An RIT node may contain a `layer` attribute but NO `filter-q` attribute. If this is the case, the contents of the corresponding QT node's `values` attribute depends on whether the RIT node is calculated to be a concept or a datum node in the final CAT: if it is a concept node, then it will be `values="layer"` (implying a yes/no answer is to be solicited by the tool); if it is a datum node, then `values` will take the contents of the `values` attribute from the RIT node.

E.2 Algorithm for Generating a Level 0 QT

Generating a Level 0 QT from an instantiated RIT requires consideration of the `question`, `filter-q`, `level` and `layer` attributes. Deciding which nodes to include in the QT and what the QT `question` and `values` attributes will contain is determined by the algorithm below, which will operate on each RIT node.

```

IF (exists filter-q attribute)
  Include this node in the QT;
  QT question attribute = "RIT filter-q attribute text";
  QT values attribute = "filter-q";
ELSE IF (exists question attribute)
  IF (exists layer attribute);
    QT question attribute = "RIT question attribute text";
    IF (node will be a datum node after any instantiation that it may
      require has taken place)
      QT values attribute = "RIT values attribute text";
    ELSE
      //node will be a concept
      QT values attribute = "layer";
    END IF;
  ELSE IF (exists level attribute)
    //do nothing with this node, but carry on processing its children
  ELSE
    QT question attribute = "RIT question attribute text";
    QT values attribute = "RIT values attribute text";
  END IF;
END IF;

```

E.3 Algorithm for Generating QTs at Level 1 and Above

Generating a QT for a Level greater than zero from an instantiated RIT requires consideration of the `question`, `filter-q`, `level` and `layer` attributes. Deciding which nodes to include in a given Level's QT and what the QT `question` and `values` attributes will contain is determined by the following algorithm¹, which will operate on each RIT node.²

¹Note: the algorithm contains references to `level-question` and `level-code` attributes, which are intermediate attributes automatically created during RIT generation to aid in generation of CATs /QTs at multiple levels.

²The algorithm for the Level 0 QT outlined in Section E.2 is essentially a simplified version of the algorithm outlined here for the Level 1+ QT.

E.3. ALGORITHM FOR GENERATING QTS AT LEVEL 1 AND ABOVE

```
IF (exists a level attribute that <= QT level)
  IF (exists level-question attribute)
    QT question attribute = "RIT level-question attribute text";
    QT code attribute = "RIT level-code attribute text if it exists.
      Otherwise, RIT code attribute";
    QT values attribute = "RIT values attribute text if the values
      attribute exists.
      Otherwise, 'filter-q' if a filter-q
      attribute exists.
      Otherwise, 'layer' if the layer attribute
      exists.";

  ELSE IF (exists question attribute)
    QT question attribute = "RIT question attribute text";
    QT values attribute = "RIT values attribute text if the values
      attribute exists.
      Otherwise, 'filter-q' if a filter-q
      attribute exists.
      Otherwise, 'layer' if the layer attribute
      exists.";
    QT code attribute = "RIT code attribute text";
  ELSE
    error: no question associated with this level;
  END IF;
  Do not go further down this subtree;

ELSE IF (exists filter-q attribute)
  Include this node in the QT;
  QT question attribute = "RIT filter-q attribute text";
  QT values attribute = "filter-q";

ELSE IF (exists question attribute)
  IF (exists layer attribute);
    QT question attribute = "RIT question attribute text";
    IF (node will be a datum node after any instantiation that it may
      require has taken place)
      QT values attribute = "RIT values attribute text";
    ELSE
      //node will be a concept
      QT values attribute = "layer";
    END IF;
  ELSE IF (exists level attribute)
    //do nothing with this node, but carry on processing its children
  ELSE
    QT question attribute = "RIT question attribute text";
    QT values attribute = "RIT values attribute text";
  END IF;
END IF;
```