DISTRIBUTED DEVELOPMENT

OF A LOGIC-BASED

CONTROLLED MEDICAL TERMINOLOGY

A DISSERTATION

SUBMITTED TO THE PROGRAM IN MEDICAL INFORMATION SCIENCES

AND THE COMMITTEE ON GRADUATE STUDIES

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Keith Eugene Campbell

June 1997

© Copyright by Keith Eugene Campbell 1997
All Rights Reserved

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

_____

Edward H. Shortliffe (Principal Advisor)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

_____

Glenn Rennels

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

_____

Terry A. Winograd

Approved for the University Committee on Graduate Studies:

_____

Abstract

A controlled medical terminology (CMT) encodes clinical data: patient's physical signs, symptoms, medication sensitivities, treatment plans, and diagnoses. Application developers and data analysts lack a robust CMT and the methodologies needed to coordinate terminology development within and between projects.

In this dissertation, I argue that if a formal terminology model is adopted and integrated into a change-management process that supports dynamic CMTs, then CMTs can evolve from being an impediment to application development and data analysis to a new state as a valuable resource that supports these activities.

My thesis states that such an evolutionary approach can be supported by using semantics-based methods for managing concurrent terminology development, thereby bypassing the disadvantages of traditional lock-based approaches common in current commercial database systems. By allowing many developers to work concurrently on any portion of the terminology while relying on semantics-based methods to resolve the “collisions” that are

inevitable in concurrent work, a scalable approach to terminology development can be supported.

This dissertation discusses the overall problem of CMT development in terms of three research topics:

- **Representation of Clinical Data:** Aggregation of clinical data is a primary reason for representing clinical data in a processible format. A logically-founded CMT supports aggregation by ensuring that the semantics of the terminology will be reproducible. This dissertation demonstrates that a logically-founded CMT can be evolutionarily developed using a participatory, consensus-driven approach.
- **Concurrency Control:** CMT development systems must support users who are distributed geographically, who work with different versions of the master CMT, and who may spend weeks working on the CMT before releasing their work publicly. The methods outlined in this dissertation can support such concurrent work. The CMT conflict detection described in this dissertation depends upon logically-based definitions of all terms. These definitions are used by a description-logic classifier to detect conflicting definitions based upon semantic equivalence rather than strictly syntactic equivalence. These semantics-based concurrency methods were evaluated in an actual development setting, and were shown to perform acceptably for routine CMT development.
- **Configuration Management:** This dissertation presents methods to manage CMT configurations using a change-set configuration-management model. These configuration methods support CMT verification and automated migration from one version of the CMT to another version. Finally, the configuration management methods minimize the

"local-update penalty" (the extra work required to migrate to a new version of a source terminology when local extensions have been added by developers to the previous version). The change-set configuration-management model properly supported the semantics-based concurrency-control methods evaluated in this dissertation, thus enabling distributed, concurrent CMT development.

Acknowledgments

Work such as this dissertation is not possible without a strong laboratory that provides a critical mass of faculty, staff, students, and computing resources. Ted Shortliffe, who has served as my principal advisor since the defense of my proposal, has made this lab possible. His guidance has provided the path for the successful conclusion of my dissertation, and has improved the exposition of my ideas. Glenn Rennels was instrumental in solidifying my preliminary ideas into a coherent proposal, and he has worked with me throughout my dissertation journey. He was readily available and showed enthusiasm for my work at a critical time in the development of the proposal. Mark Musen has provided important insights that helped form my early work, and he has introduced me to members of the informatics community, providing significant professional exposure for my ideas. One of the key elements in forming my early ideas has been interaction with other students within the laboratory. I particularly would like to thank Amar Das, Bill Detmer, John Egar, Diane Oliver, Malcolm Pradhan, Yuval Shahar and Walter Sujansky for valuable discussions and encouragement of my research.

One of the strengths of the Stanford medical informatics program is its curriculum. Stanford faculty from other departments provide a diverse background as well as exposure to new ideas. I was fortunate to meet Terry Winograd as part of that initial curriculum. His philosophy has provided me with a new way to think about how computers fit into the health-care environment. I am also grateful that he has participated as a member of my reading committee.

My dissertation project grew beyond the size supportable within an academic department. Completion of this project would not have been possible without the willingness of Kaiser Permanente to provide their electronic-medical record projects as a testing ground for my ideas. In particular, I wish to thank Simon Cohn, John Mattison, Jeff Rose, and John Dewy. They have been my champions inside Kaiser Permanente, and it is through their efforts that my dissertation project was funded. The Kaiser Permanente modelers were patient subjects for my experiment. I thank Bob Dolin, John Fedack, Bruce Fisch and Aaron Snyder for their participation in my experiment.

In addition to Kaiser Permanente, many people outside this laboratory have contributed to this work. Eric Mays sponsored me for an "academic visitors' position" at IBM's T.J. Watson Research Center, where I completed some early protype extensions to the K-Rep system. Bill Jensen sponsored me for the "clinical investigators' pathway" to eligibility for the internal medicine board at Santa Clara Valley Medical Center. He has provided an environment where I could work in alternating months on my dissertation project and on my internal medicine residency. The ability to combine these two pursuits has been most rewarding. Alan Rector came to Stanford for a short sabbatical as I neared the completion

of my original proposal. He has provided valuable insight about how my research relates to the GALEN project, and an important "reality check." Mark Tuttle and David Sheretz of Lexical Technology, Inc., helped to motivate the need for maintenance methods, and they have provided many examples of problems discovered by their UMLS experiences. Roger Côté and David Rothwell of the SNOMED editorial board have listened to ideas for SNOMED with an open mind, and have encouraged my work. John Sowa has generously given his time as I struggled with the semantics of conceptual graphs. Vimla Patel introduced me to Argumentation Diagrams which I present in Chapter 6. Amy Falkowitz and John Thomas have provided excellent editorial assistance.

I owe a special debt of gratitude to Susan Castillo. Susan's experiences with software development and configuration management at Hughes Aircraft Company, the Stanford Linear Accelerator Center, and with Apple Computer, convinced me of the need to look beyond the problems of *representing* clinical data to the problems of *developing* such a representation. The insight provided by her experiences have inspired the focus and content of this dissertation. I also owe Susan a second, and greater, debt of gratitude. Susan is my wife, and without her love, support, and encouragement to complete a Ph.D., this work would not have been possible.

This work was supported in part by the Agency for Health Care Policy and Research under grants HS06330 and HS08751, by the National Library of Medicine under grants LM05208, LM07033, and LM08751, and by the Kaiser Foundation Health Plan. I gratefully acknowledge this indispensable support.

Table of Contents

Abstract ..... **v**

Acknowledgments ..... **ix**

List of Figures ..... **xix**

List of Tables ..... **xxiii**

Chapter 1
Encoded Clinical Data:
Promises, Perils, and Challenges **1**

1.1 Promises 2

1.1.1 The Promise of CMTs for Decision Support 3

1.1.2 The Promise of CMTs for Monitoring Quality 5

1.1.3 The Promise of CMTs for Medical Research 6

1.1.4 The Promise of CMTs for Data Acquisition 7

1.2 Perils 8

1.2.1 Perils of Improper Conclusions from Incomplete Data 9

1.2.2 Perils of Performance Failures 10

1.2.3 Perils of Costly Premature Mandates 13

1.2.4 Perils of Incompatible Terminologies 14

1.3 Challenges 15

1.4 CMT Development and Dissertation Overview 17

1.4.1 Development Setting 18

1.4.2 Centralized Coordination, Local Control 19

1.4.3 Logical Representation of Clinical Data 23

1.4.4 Evolutionary Design 24

1.4.5 Thesis: Semantics-Based Concurrency Control 25

1.4.6 Configuration Management 26

1.5 Technology: Necessary but not Sufficient 27

1.6 Guide to this Dissertation 28

Chapter 2
Background:
Classification, Concurrency Control and Configuration Management 31

2.1 Clinical Data Classification 34

2.1.1 Common Terminology Structure 34

2.1.2 Historical Perspective 36

2.1.3 A Logical Foundation 40

2.2 Concurrency Control 43

2.2.1 Transactions 44

2.2.2 Traditional Concurrency Control 45

2.2.3 Semantics-Based Concurrency Control 49

2.3 Configuration Management 52

2.3.1 Configuration Management Models 54

2.4 Summary 61

Chapter 3
Representation:
Terminological Definitions, Information Models, and Patient Data 63

3.1 Terminological Definitions and Information Models 64

3.2 Terminological Definitions and Patient Data. 66

3.3 Representation of Terminological Definitions 68

3.3.1 Concept Definitions 68

3.4 Derivation of an Information Model. 71

3.5 Foundational Models 72

3.6 SNOMED International 74

3.6.1 Migration Path for SNOMED 78

3.7 Prospects. 79

Chapter 4
Computer Support for Collaborative Development 81

4.1 CMT Development Assumptions 82

4.2 CMT Development Examples 87

4.2.1 Nonunique-definition Conflict 87

4.2.2 Multiple-Definition Conflict 90

4.3 CMT Concurrency Control. 92

4.3.1 CMT Transactions 92

4.3.2 CMT Transaction Validity 93

4.3.3 Terminology Change Sets 94

4.3.4 Terminology-Specific Conflict Resolution Strategies 98

4.3.5 Configuration Management Challenges. 105

4.4 CMT Configuration Management 106

4.4.1 Custom Configurations 107

4.4.2 Version Naming 108

4.4.3 Version Merging. 109

4.4.4 Minimization of Local-Update Penalties. 111

4.4.5 Custom Configuration Examples. 114

4.5 Limitations of Conflict Detection 120

4.6 Summary and Discussion 122

Chapter 5
The Galápagos:
Applications to Study Evolutionary Terminology Development. 125

5.1 K-Rep: Classification Engine 126

5.2 Application Support for the Development Cycle 131

5.2.1 Isabella: Configuration Management and Conflict Identification . 133

5.2.2 K-Rep DE: Terminology Enhancement. 138

5.2.3 Cristobal: Filter Changes 142

5.2.4 Rhabida: Conflict Resolution 144

5.3 Prototype Application Summary 147

5.4 Future Needs for CMT Development Applications 149

Chapter 6
Evaluation 151

6.1 Proof-of-Concept 152

6.1.1 Conflict Detection 153

6.1.2 Conflict Review 154

6.2 Proof-of-Performance 158

6.2.1 Conflict Resolution and Evolutionary Design 164

6.2.2 Future Research on Conflict-Resolution Methods 171

6.3 Conclusion 172

Chapter 7
Conclusion 175

7.1 Generalizability and Limitations 176

7.1.1 Logic-Based Approach 177

7.1.2 Foundational Models 178

7.1.3 Evolutionary Enhancement 180

7.1.4 Domain-Specific Conflict Detection and Resolution 181

7.1.5 Configuration Management: No Free Lunch 183

7.1.6 Evaluation 184

7.2 Ancillary Lessons 185

7.2.1 Clearly Define the Boundaries of Collaboration 185

7.2.2 Limit Distribution of Errors of Commission 189

7.2.3 Independently Edit Top-Level Hierarchy. 191

7.2.4 Meaningless Identifiers 191

7.2.5 Spelling Correction 193

7.3 Future Work 194

7.3.1 Alternative Development Paradigms 194

7.3.2 Support for Locally Maintained Enhancements 200

7.4 Contributions 201

7.4.1 Medical Informatics. 202

7.4.2 Medicine 203

7.4.3 Experimental Computer Science 203

7.5 Final Remarks. 204

Appendix A
Transcript of Conflict Resolution Session 207

Appendix B
The Knowledge Representation System Specification. 241

Bibliography 245

List of Figures

Figure 1-1. A development scenario where a central body coordinates all changes to the CMT while local sites retain authority over if and when they will submit changes to the central body 21

Figure 2-1. An ICD-9-CM hierarchy and an analogous SNOMED hierarchy, showing classification of "pleural effusion" 35

Figure 2-2. Valid serial sequence of transactions. 46

Figure 2-3. A development scenario with no central coordination 53

Figure 3-1. Relationships between a terminology model and an information model . 65

Figure 3-2. Relationships between a terminology model and a patient (data) model . 67

Figure 3-3. An expanded set of terminological concepts. 68

Figure 3-4. SNOMED type hierarchy showing classification of "pleural effusion" . . 77

Figure 4-1. Graphical illustration of a nonunique-definition conflict 89

Figure 4-2. Graphical illustration of a multiple-definition conflict 91

Figure 4-3. A transaction set equivalent to the changes made by Modeler A 94

Figure 4-4. CMT version naming convention 109

Figure 4-5. Possible CMT states 112

Figure 4-6. Update path for sites A and C 116

Figure 4-7. Update path for site B 118

Figure 4-8. Update path for site D 119

Figure 5-1. K-Rep Engine components and selected API functions 128

Figure 5-2. Classification of terminological definitions. 129

Figure 5-3. Terminology Development Cycle 132

Figure 5-4. Isabella Architecture and representation of selected program functions. 134

Figure 5-5. Isabella's character-based interface. 135

Figure 5-6. Example entry from a conflict report 137

Figure 5-7. K-Rep DE Architecture 139

Figure 5-8. K-Rep DE's taxonomy view 140

Figure 5-9. Concept viewer displaying the definition of "infectious pneumonia" . . 141

Figure 5-10. Representation of Cristobal's filtering functionality 144

Figure 5-11. Rhabida architecture and representation of its functionality. 144

Figure 5-12. Rhabida tool demonstrating conflicting definitions 146

Figure 5-13. Representation of the terminology development cycle 148

Figure 6-1. Semantically equivalent changes. 154

Figure 6-2. Semantically conflicting changes 154

Figure 6-3. Semantically-conflicting changes from Kaiser Permanente development 162

Figure 6-4. Semantically-equivalent changes from Kaiser Permanente development 163

Figure 6-5. Non-unique definition conflicts. 164

Figure 6-6. Argumentation diagram of arguments and conflicting hypotheses and the conflicting classifications for flexion 167

Figure 6-7. Diagram of arguments and revised hypothesis and newly agreed classification 169

Figure 6-8. Semantically conflicting changes between Kaiser Permanente and SNOMED 3.3 170

Figure 7-1. A development scenario where a central body directs and coordinates all changes to the CMT 196

Figure 7-2. A development scenario with no central coordination 199

Figure A-1. Conflict report for “tremor”. 209

Figure A-2. Conflict report for “procedure order form” 210

Figure A-3. Conflict report for “flexion” 211

Figure A-4. Conflict report for “prone body position” 214

Figure A-5. Conflict report for “flaccidity” 217

Figure A-6. Conflict report for “intermalleolar straddle” 219

Figure A-7. Conflict report for “transient paralysis of limb” 221

Figure A-8. Conflict report for “Chemoreceptor function” 222

Figure A-9. Conflict report for “extension” 223

Figure A-10. Conflict report for “Cranial nerve XI exam”. 224

Figure A-11. Conflict report for “posture” 226

Figure A-12. Conflict report for “morning stiffness” 230

Figure A-13. Conflict report for “visual acuity testing” 231

Figure A-14. Conflict report for “myalgia” 232

Figure A-15. Conflict report for “eye and eyelid symptom” 232

Figure A-16. Conflict report for “peritoneal dialysis” 234

Figure A-17. Conflict report for “tetany” 235

Figure A-18. Conflict report for “skin rash”. 237

List of Tables

Table 3-1. Terms with corresponding definitions 69

Table 4-1. Change sets S3 and S4 with compensating change sets 100

Table 4-2. Change sets S1 and S2 with compensating Change sets 104

Table 4-3. Change sets to be combined to generate a new CMT reference version 111

Table 4-4. Version 1 and version 2 definitions of infectious-pneumonia and pulmonary-disease 114

Table 4-5. Version 1.A.1 definitions, version 2 definitions, and the change sets required to synchronize version 1.A.1 with version 2 117

Table 4-6. Version 1.B.1 definitions, version 2 definitions, and the change sets required to synchronize version 1.B.1 with version 2 118

Table 4-7. Version 1.D.1 definitions, version 2 definitions, and the change sets required to synchronize version 1.D.1 with version 2 120

Table 5-1. Concept forming operators and the terminological axioms of the K-Rep language 130

Table 6-1. Statistics for the 5 merges included in the 6 month evaluation period . . 159

Table B-1. Concept forming operators and the terminological axioms of the Knowledge Representation System Specification 242

Chapter 1
Encoded Clinical Data: Promises, Perils, and Challenges

The health-care industry and government agencies are looking to computer-based tools to reduce health-care expense, to assess the quality of health-care providers, and to deliver health-care services more efficiently. A core component of these tools will be a *controlled medical terminology* (CMT).[1] A CMT is used to encode clinical data: patient's physical signs, symptoms, medication sensitivities, treatment plans, and diagnoses. It is *controlled* in the sense that a formal process for adding new content to the terminology is followed, thus ensuring high standards of quality and functionality.

A CMT is more than a simple system component, however, because any tool's functionality is inseparably linked to a CMT's ability to represent relevant concepts and to engage the user in a discourse regarding those concepts. If a CMT fails to represent the concepts necessary for the tool to operate, the tool will fail. Similarly, if the coupling between the tool and the user does not facilitate natural interaction (perhaps because of a poorly

1. I use "terminology" rather than "vocabulary" throughout this dissertation to adhere to the recommendations of the International Standards Organization (1990).

designed interface or ambiguity in the CMT), the tool will again fail. Given this tight coupling between the functionality of a tool and the underlying CMT, the two must be discussed together. In this chapter, I briefly present the promised benefits and potential perils of tools that rely on a CMT, and then discuss the unsolved challenges that must be overcome if one is to develop a CMT to support these tools. I propose an evolutionary-development methodology that will answer many of these challenges. It is outlined in this chapter and refined throughout the document. As will become clear, I believe that effective creation, use, and sharing of clinical terminology requires a coordinated evolutionary approach and methods for assuring consistency and for resolving inconsistencies as it evolves.

My thesis,[2] states that such an evolutionary approach can be supported using semantics-based methods for managing concurrent terminology development, bypassing the disadvantages of traditional lock-based approaches common in current commercial database systems. By allowing many developers to work concurrently on any portion of the terminology while relying on semantics-based methods to resolve the "collisions" that are inevitable in concurrent work, a scalable approach to terminology development can be supported.

1.1 Promises

Many health-care applications rely on a clinical terminology to represent data about patients. The functionality of any data-management application is inseparably linked to its

2. Formally stated in Section 1.4.5.

underlying terminology. If the terminology cannot represent the distinctions that an application needs, that application will fail either because the data acquisition fails to support an appropriate discourse with the user or because the terminology lacks distinctions necessary for data analysis. The terminology must also have sufficient structure to allow application developers to reproducibly apply it, since an application's promised functions are enabled by algorithms that process information using an appropriate clinical terminology.[3] Examples of such tools are discussed in the following sections.

1.1.1 The Promise of CMTs for Decision Support

Computer-based decision-support systems are computer programs designed to help the user make decisions. Shortliffe (1990) classifies tools for clinical decision-support into three general types: tools for *information management*, tools for *focusing attention*, and tools for *patient-specific consultation*.

Examples of tools for information management include hospital information systems (Wiederhold & Perreault, 1990) and bibliographic-retrieval systems (Siegel, Cummings & Woodsmall, 1990). These tools aim to improve the quality and to decrease the cost of medical care by making relevant information available to the health-care provider so that proper actions can be taken in a timely manner.

Tools for focusing attention include laboratory information systems (Smith & Svirbely, 1990) that flag abnormal laboratory values and define "panic" values that must be immedi-

3. Terminological structures that support reproducible semantics include explicit defining relationships between terms within the terminological system, and prose definitions of those terms. In this sense, a terminology should be distinguished from related collections such as lexicons (the linguistic units of a language that cannot be divided into smaller meaningful parts), word lists, or phrase lists.

ately communicated to the health-care provider. Pharmacy systems may also provide similar functions by automatically checking a patient's list of medications for interactions (Tatro et al., 1975). Other systems, such as the Regenstrief medical record (McDonald, Blevins, Tierney & Martin, 1988) and the HELP system (Kuperman, Gardner & Pryor, 1991), provide similar reminders for other aspects of the patient's care. These reminders may advise that a patient is due for a routine mammogram or an annual occult-blood test (McDonald et al., 1984). Such tools promise to prevent mistakes and oversights by helping providers to filter out the important from the routine and also by reminding them of health-maintenance protocols.

Tools for patient-specific consultation are the most complex tools in this class and require the most data to provide proper advice. Some systems, such as those for computer-based ECG analysis, are widely used (Willems et al., 1991). Some, such as the Quick Medical Reference (QMR) (Miller, Masarie & Myers, 1986) and ILIAD (Bergeron, 1991) are commercially available. Miller (1994) provides an extensive review of medical diagnostic systems and predicts that these systems will proliferate. Such tools promise to reduce the number of overlooked diagnoses and to improve the efficiency of diagnostic workups. Their ability to suggest overlooked diagnoses has prompted consideration of their use as tools for quality assurance (Lau & Warner, 1992). Yet their successful use will require a terminology that represents appropriate diagnoses as well as all the signs and symptoms associated with these diagnoses. Further, if automated capture of the signs and symptoms is desired, the terminology within the quality-assurance tool, and the terminology within the application that originally captures the data (such as a data-acquisition tools discussed

in Section 1.1.4) must be compatible. A quality-assurance tool can be no better than its underlying terminology.

1.1.2 The Promise of CMTs for Monitoring Quality

Automated capture of clinical data promises to make risk-adjusted morbidity and mortality studies a routine part of assessing the quality of care delivered by a health care organization or an individual provider. Several studies have already tried to assess the quality of care delivered using existing administrative data (Bowen & Roper, 1987; Luft & Romano, 1993).

Such databases, collected routinely in our current medical system, are very attractive sources for this purpose (Flood, 1990) because:

- they cost less than specially collected data sets
- they allow quality to be examined in a nationwide context
- they have sample sizes that can measure small differences in effectiveness
- they can track patients over longer periods and look at the permanence of the effects being observed, and
- they can look at factors, other than identity of the provider, that may affect the quality of care.

As more clinical data are routinely collected, quality surveillance will utilize clinical databases in addition to administrative databases. Clinical databases (databases that expand upon administrative data by including relevant clinical information not routinely collected

for administrative purposes) promise to improve the reliability and utility of analysis based strictly on administrative data because clinical data can be used to risk-adjust mortality rates using clinical factors. Hannan and colleagues (1992) have demonstrated that clinical databases can perform substantially better than administrative databases when one is evaluating mortality following coronary-artery bypass graft surgery. Clinical databases can perform better at such tasks because they have an expanded terminology that is able to represent clinically significant data that an administrative database cannot. Clinical databases require a detailed CMT to represent this clinically significant data.

1.1.3 The Promise of CMTs for Medical Research

The development of large clinical databases promises to provide benefits other in addition to facilitating nationwide quality surveillance. If the clinical data within these databases are appropriately detailed and reliable, they will also become valuable resources for retrospective clinical research. If data pooling is made possible by uniform use of a CMT, these databases hold the promise of offering information similar to that obtained by randomized clinical trials but at much lower cost.

Since databases created by pooling data from routine encounters are not expected to eliminate the need for randomized clinical trials, another class of tools is being developed to support them (Musen, Carlson, Fagan, Deresinski & Shortliffe, 1992; Shortliffe & Hubbard, 1989). These tools are designed to determine automatically when patients are eligible for protocols prospectively and to help physicians follow these protocols after their patients are enrolled. These tools promise to improve the reliability of clinical trials by making data collection more complete (Kent, Shortliffe, Carlson, Bischoff & Jacobs,

1985) and to speed completion of clinical trials by preventing eligible patients from being overlooked and ensuring that enrolled patients are managed in accordance with protocol guidelines.

Large clinical databases, protocol-eligibility applications and protocol-management applications all depend upon CMTs to represent the data they act upon. As with other kinds of clinical applications, the functionality of these systems is inseparably linked to the robustness of the underlying CMT.

1.1.4 The Promise of CMTs for Data Acquisition

Requirements for documenting health-care encounters are increasing. Some of this increase is being driven by regulatory and reimbursement requirements. Other factors include the desire to improve the quality of medical care and to reduce its cost.

Irrespective of the causes for the increase in documentation, application developers are working to provide solutions that promise to improve the quality and completeness of clinical documentation, as well as to make secondary uses of this documentation (such as quality assurance and medical research) more efficient and less costly.

Many approaches have been, or are being, developed to capture or analyze computer-processible clinical data (Baud et al., 1993; Bell et al., 1992; Benoit et al., 1992; Bernauer, 1991; Johnson, Aguirre, Peng & Cimino, 1993; Kuhn et al., 1993; Lenert & Tovar, 1993; Naeymi-Rad et al., 1992; Poon, Fagan & Shortliffe, 1996; Rassinoux, Baud & Scherrer, 1992; Rector et al., 1991; Sager et al., 1993; Schröder, 1992). These approaches typically use menu selection, other types of structured data entry, or natural-language processing to

collect clinical data. Although there are important differences among these approaches, they all share a common feature: the need for a robust CMT to enable their use. Data acquisition applications must have a CMT that represents the concepts necessary to document an encounter, and, for that data to be useful for secondary uses, their CMT must be compatible with the terminologies of other applications such as those discussed in Sections 1.1.1-1.1.3. If the CMT is lacking, the collective promises of informatics applications may not be realized.

1.2 Perils

The promised benefits of automating access to clinical data are accompanied by serious perils. Most of these are directly related to an inadequate CMT, and include the following:

- **Improper conclusions**: misleading conclusions can be drawn from easily available but incomplete data that typically rely upon a limited CMT such as the Current Procedural Terminology (American Medical Association, 1995) or the International Classification of Diseases (National Center for Health Statistics, 1995). These terminologies are designed for limited purposes and are fraught with danger if attempts are made to use them for inappropriate analyses.
- **Performance failures**: difficulties acquiring data may cause expensive failures. The failure may be due to an impoverished terminology that providers will not use because it poorly characterizes their patients; on the other hand, an overly detailed terminology may be excessively time-consuming to navigate.

- **Premature mandates**: regulatory pressures may force adoption of an incomplete or inappropriate CMT that is difficult to maintain and is inadequate for its intended use.
- **Incompatible terminology**: if CMTs are incompatible, sharing of clinical data will never become a reality.

These four perils are discussed in more detail below.

1.2.1 Perils of Improper Conclusions from Incomplete Data

The studies of the quality of care delivered using administrative or clinical data cited previously in Section1.1.2 allow only limited conclusions because the data employed were coded using the International Classification of Diseases, Ninth Revision, with Clinical Modifications (ICD-9-CM) (National Center for Health Statistics, 1995). These codes were originally developed to classify causes of death and have been modified subsequently to track disease prevalence worldwide. They were never intended for assessing the quality of care delivered at hospitals or clinics. The categories are too broad and there is no reliable mechanism for classifying the severity or acuity of a condition. Some studies have tried to group patients with the same diagnoses into equivalent risk groups based on their age, sex, and race, but the descriptions are simply too limited to stratify patients properly so that outcomes can be validly compared (Jollis et al., 1993). Even specially collected datasets that contain clinical data can have substantial biases, preventing valid inferences about risk-adjusted quality-of-care measures (Blumberg, 1991). Clearly, the limitations of the data must be understood. Otherwise improper—and possibly damaging—conclusions may be made.

Too often, decision makers rely only on easily accessible data. Consider this real example: an insurance company was rating providers by the fees they charged, with only minor attention, if any, being paid to quality. Based on cost alone, the insurance company rated an ophthalmologist as a preferred provider, not knowing that the doctor had been barred from surgery by the state Medical Disciplinary Board because of using misleading advertising, altering medical records, and operating with negligence (Flores, 1993). This is just one example of the possible unanticipated consequences of deploying a new technology imprudently. As deployment proceeds, vigilance for these unanticipated effects is essential.

1.2.2 Perils of Performance Failures

Although the arguments for CMT-reliant tools are compelling, the financial and organizational costs associated with achieving acceptable performance from these tools may be overwhelming. There have been previous costly failures, and current efforts need to be viewed with appropriate skepticism.

A well known example, developed by Dr. Lawrence Weed, the Problem-Oriented Medical Information System (PROMIS), failed to achieve long-term acceptance by its intended users. This system, a computer-based, problem-oriented medical record (Weed, 1969), was deployed at the Medical Center Hospital of Vermont in the early 1970s. Weed's work, which proposed problem-oriented progress notes, has dramatically improved the quality of documentation in paper-based medical records, and has been widely accepted. Despite the success of his paper-based system, the computer-based implementation was rejected, largely because Weed and his coworkers assumed that physicians would be willing to

adhere to dogmatically applied problem-oriented principles. The inability of the system to adapt to the desires of its physician users was the most important reason for its failure to gain acceptance (Fischer, Stratmann, Lundsgarrde & Steele, 1980).

Part of the PROMIS system's failure was also due to immature interface technology and a limited understanding of human-computer-interface design at the time of its implementation in the 1970s. Since then, developers have been incorporating new methods into the design process to enhance user acceptance and thus increase the chances for success of their software systems. These include user-centered design, information-flow studies, increased attention to insights from cognitive psychology, and sociological analysis of the workplace (Norman & Draper, 1986; Winograd & Flores, 1986). Computer technology has also evolved to allow users to interact with computers with less effort than had previously been possible.

Just as there have been previous failures secondary to a poor understanding of human-computer-interface design, today developers are experiencing failures secondary to poor understanding of the requirements of terminology-based applications. Such poor understanding usually takes one of several forms: underestimating the costs associated with acquiring data in machine processible form, underestimating the costs associated with the development and maintenance of the terminology, and failing to validate the appropriateness of the terminology for a particular task.

One project that recently suffered from several such terminology-related problems is the Uniform Clinical Data Set (UCDS). This is a standardized set of clinical data elements and associated tools used to collect and analyze them. The Health Care Financing Administra-

tion (HCFA) had planned nationwide implementation of UCDS by 1996, and asked that all the requested data for medicare patients be collected by manual chart review after the patient was discharged from the hospital.

The developers of the UCDS promised that its use would improve the accuracy, reliability, and validity of the Medicare Peer Review Organization (PRO) review process by ensuring national uniformity. Initial experiments, however, have suggested that UCDS data abstraction methods would be too costly. The average time to review a chart using the UCDS was 93 minutes, compared to the average of 23 minutes for manual review, indicating an increase in the cost of labor in the vicinity of 400 percent (Audet & Scott, 1993). In addition, statistics describing the positive and negative predictive values for detecting quality problems using UCDS tools are not available. Currently, deployment of the UCDS is on hold due to the aforementioned problems. Nevertheless, the desire to realize the promises of automated data analysis remains high. Undoubtedly, future initiatives will be proposed. Hopefully, a robust CMT will be available to support these applications when they are deployed.

The limitations of medical terminologies were also painfully apparent during my development of IVORY, a clinical data-acquisition tool (Campbell et al., 1993). IVORY was designed to collect data directly from the health-care provider using structured data entry for selecting terms from a controlled medical terminology. IVORY produced a structured SOAP note[4] for the provider and the patient's chart as well as a structured representation

4. A progress note structured with sections in the following order: Subjective, Objective, Assessment, and Plan. The SOAP note structure is from Weed's problem oriented medical record (Weed, 1969).

of the clinical data entered by the provider. Despite the successful proof-of-concept, the inability to identify a suitable controlled medical terminology for IVORY led to significant difficulties in implementing it in a more demanding setting (Musen, Weickert, Miller, Campbell & Fagan, 1995). My frustration with this constraint on a system I had built accounts in part for my interest in addressing the problems of CMT development.

Since current standard representations are not sufficient for the needs of most clinical applications, developers are forced to create their own terminologies (Campbell & Musen, 1992a). Problems of coordinating local changes with a nationally evolving standard may overshadow the potential benefits of adopting such a standard. Significant problems, for example, have been encountered just trying to synchronize different versions of the terminology used by IVORY with the T-Helper decision-support system (Musen et al., 1992). These local problems will be magnified if development of a national terminology is attempted.

1.2.3 Perils of Costly Premature Mandates

The sense of urgency to develop CMT standards may prompt premature or poorly considered actions. The Agency for Health Care Policy and Research (AHCPR), for example, was given the responsibility to develop standards for the automated medical record by the Omnibus Budget Reconciliation Act of 1989. These standards were to include uniform definitions for clinical data and common reporting formats. The Health Care Financing Administration (HCFA) was not satisfied by the progress of standards development, and helped draft the Medical and Health Insurance Information and Reform Act of 1992 (which failed to be enacted by the 102nd Congress). This bill would have required that

certain standards be in place by certain dates, and if they were not, would have authorized the Secretary of Health and Human Services (HHS) to declare the standards and disseminate them.

Although the Medical and Health Insurance Information and Reform Act of 1992 failed to become law, a subsequent bill has: the Health Insurance Portability and Accountability Act of 1996 (United States Congress, 1996). This law has equivalent requirements for standards, and authorizes the Secretary of HHS to declare the standards and to disseminate them. If such standards are arbitrarily developed and applied, however, they may not meet the needs of applications other than those of HHS itself. Further, if standards are developed without the means to evolve as the needs of HHS change, those standards might not even meet the needs of HHS.

California provides another example of the rush to develop new clinical-data standards. The State of California has directed the Office of Statewide Health Planning and Development (OSHPD) to prepare an annual report on the quality of care delivered by all of the state's hospitals (California Legislature, 1991). OSHPD was directed to examine administrative and discharge data already collected and to determine if these data were sufficient to perform valid quality assessments. If the data were found lacking, OSHPD was to recommend what elements needed to be included in the analysis to make risk adjusted outcome measurements possible. If all of the states followed California's example, 50 different state-specific requirements would result, thereby placing an excessive burden on hospitals and information-system vendors, and the national comparability of such data would be severely compromised.

1.2.4 Perils of Incompatible Terminologies

Premature legislative mandates are not the only way that isolated and incompatible CMTs come into being. Such incompatibilities unfortunately are typical of most CMT-based tools. Diagnostic tools, such as QMR (Miller et al., 1986) and Iliad (Bergeron, 1991), for example, use incompatible CMTs even though the two applications have the same purpose (to suggest patient diagnoses from an input list of symptoms) and focus on the same domain (Internal Medicine). Similarly, laboratory systems and pharmacy systems from different vendors can also be expected to have incompatible CMTs.

Kahn (1993) characterized the information-systems infrastructure of American health-care providers as one of isolated, one-of-a-kind, and incompatible systems. Not only does this information system infrastructure threaten to be an ever-increasing maintenance burden, it is also severely compromising current efforts to provide integrated applications that can efficiently process data from a variety of sources. Without such integration, users are often required to re-enter data that already exist on one system (such as a laboratory system) into another (such as the QMR or Iliad diagnostic tools). This manual data re-entry is rightly cited as a significant barrier to physician acceptance of clinical decision-support tools.

1.3 Challenges

Standards capable of representing clinical data do not exist (United States General Accounting Office, 1993). The Institute of Medicine has recommended that such representations be developed over the next decade (Dick & Steen, 1991). The urgency of policy

makers' drive to solve health-care delivery problems, combined with a frequently incomplete understanding of current technology, increases the pressure to find immediate solutions.

The initial challenge for the medical informatics profession is to propose limited solutions that can be applied immediately. These proposals must not ignore these limitations, however, but incorporate frameworks for evolutionary enhancement to overcome them. Such frameworks are clearly important for long-term viability, but are also critical for the initial deployments.

Winograd and Flores (1986) state that one clear design objective is to "anticipate the forms of breakdown and provide a space of possibilities for action when they occur" (p. 165). This is important for an initial deployment because any artifact introduced into a new environment may create unanticipated breakdowns (failure of an artifact to perform as expected). If these breakdowns follow a recurrent pattern, they can be defined and classified. Rules for managing these breakdowns must be developed as well.

Therefore, the corollary challenge to proposing immediate solutions is to anticipate their inevitable breakdowns. Effective anticipation will include the development of mechanisms to manage them. Since a CMT is the foundation upon which the tools described in Section 1.1 are built, breakdowns in a CMT will create associated breakdowns in the dependent tools. Without a mechanism to manage these CMT breakdowns, the utility of derivative tools will be compromised.

Likely CMT breakdowns include what may be called "errors of omission" as well as "errors of commission." The former occur when necessary concepts are not included in a CMT or when incomplete definitions of the CMT lead to data-aggregation errors. The latter occur when incorrect statements within a terminological definition cause applications to retrieve information inappropriate to the aggregated class. In this dissertation, I propose mechanisms for managing these breakdowns. An overview of the CMT development process and how to manage the terminology changes required to address breakdowns follows immediately in Section 1.4. A detailed description of how to manage terminology changes, with examples, is provided in Chapter 4.

1.4 CMT Development and Dissertation Overview

This dissertation focuses on a methodology for identifying conflicts in concurrent work by individuals seeking to develop a CMT. My evaluation of the approach has taken place in an actual work setting, supported by advances in the application of the principles of configuration management to the development process as well as by a supportive social structure. This structure made it possible for a meaningful dialog about conflicting design decisions to take place among developers who were working on competing projects.

Because of the complex relations among the components that form the foundations of my work, this dissertation is necessarily an experimental effort. It seeks not just to advance a theory about how distributed terminology development can be realized, but also to evaluate the applicability of that theory to practical environments. Moreover, my work has been motivated by the need to coordinate terminology development on a national scale. It has

been a sizeable project, therefore, to construct a meaningful evaluation of the performance characteristics of the thesis of this dissertation as well as to describe the interplay of theoretical work with the demands of complex social organizations.

1.4.1 Development Setting

Kaiser Permanente, the nation's largest health maintenance organization, has several EMR development and implementation efforts in progress. A robust CMT is critical to the success of these EMR efforts. As part of its high-level commitment to have comparable data across all of its EMR projects, Kaiser Permanente has sought to use methods that I have developed (Campbell, 1994) to coordinate its CMT work and to allow an evaluation of the methods employed.

Although Kaiser Permanente has provided a challenging setting, thanks to the performance pressures associated with actual development, it has provided an ideal test bed for evaluating the distributed development ideas I describe in this dissertation. It is not only a real-world setting, but also is of sufficient scale so that lessons learned during the process are generalizable to other large-scale settings.

Although Kaiser Permanente is investing centrally in coordinating its CMT work through its national offices, the actual development work is being undertaken as part of competitive EMR efforts co-sponsored by Kaiser Permanente and development partners.[5] The national office focuses on coordinating terminology work and on providing an infrastructure that

5. The competitive efforts, while sharing Kaiser Permanente as an intended customer, are competing with one another in the commercial marketplace. Competitive pressures thus limit the scope of possible collaboration to very narrow areas, and require constant vigilance to maintain consensus.

fosters collaboration and exchange. Decisions about the CMT modeling are determined locally by the needs of the individual EMR development efforts. This organizational structure can be described as one of "centralized coordination, local control," the essential characteristics of which are described in Section 1.4.2.

1.4.2 Centralized Coordination, Local Control

A paradigm of centralized coordination, local control development may be appropriate whenever several organizations agree to collaborate to develop or maintain a CMT. Each organization (or development site) may already be developing a CMT for its own internal uses, but the collaborators recognize the importance of working together toward a common standard. They may obtain some partial external funding for their collaboration that will support coordination of efforts, but much of the development cost is likely to be locally funded, as was the case for Kaiser Permanente.

Under this paradigm, the local sites form a central steering committee that includes representatives from each site as well as representatives from a coordinating body. Within Kaiser Permanente, this group was known as the CMT Oversight Group and has had representatives from four of the regions actively participating in CMT development—Colorado, Mid-Atlantic, Northern California, and Southern California—as well as representatives from Kaiser Permanente's national offices.

The steering committee is responsible for defining the objectives of the collaboration. The resources necessary to develop the CMT are still controlled by each local site, and are significantly influenced by the needs of their respective EMR projects. At Kaiser Permanente,

each of the four regions had been working on competing projects, either internally developed or with EMR-development partners such as IBM, Oceania, and Oacis Healthcare Systems.

The steering committee seeks to find common goals from these objectives and to share work so that duplicate efforts are minimized—while assuring that confidential and proprietary information inherent in a development project is not compromised and seeing to it that resources of the collaborative are not unduly aligned with the needs of a specific vendor. Although relationships among the local sites are typically harmonious, inevitably there are differing priorities at each site. Maintaining clarity and consensus regarding the collaborative resources and goals is bound to be challenging.

Figure 1-1 illustrates a scenario that embodies the paradigm of centralized coordination, local control.There are two sites, A and B, in the illustrated collaboration. These sites begin with a common version of a CMT, version 1. Site A faithfully submits its changes to a central coordinator that integrates these changes into new reference versions. Site A does not always immediately incorporate these new reference versions. Changes to the local terminology may impact existing applications in ways that prevent immediate adoption of new reference versions. In the case illustrated in Figure 1-1, site A did not incorporate version 2 immediately, but instead waited until version 3 was available.

Site B was not been quite as cooperative as site A. Site B initially offered all its local changes and incorporated the second reference version immediately. Local requirements, however, demanded that it concentrate next on solving a significant local problem. Because all efforts were focused on the local problem, no changes were submitted for ver-

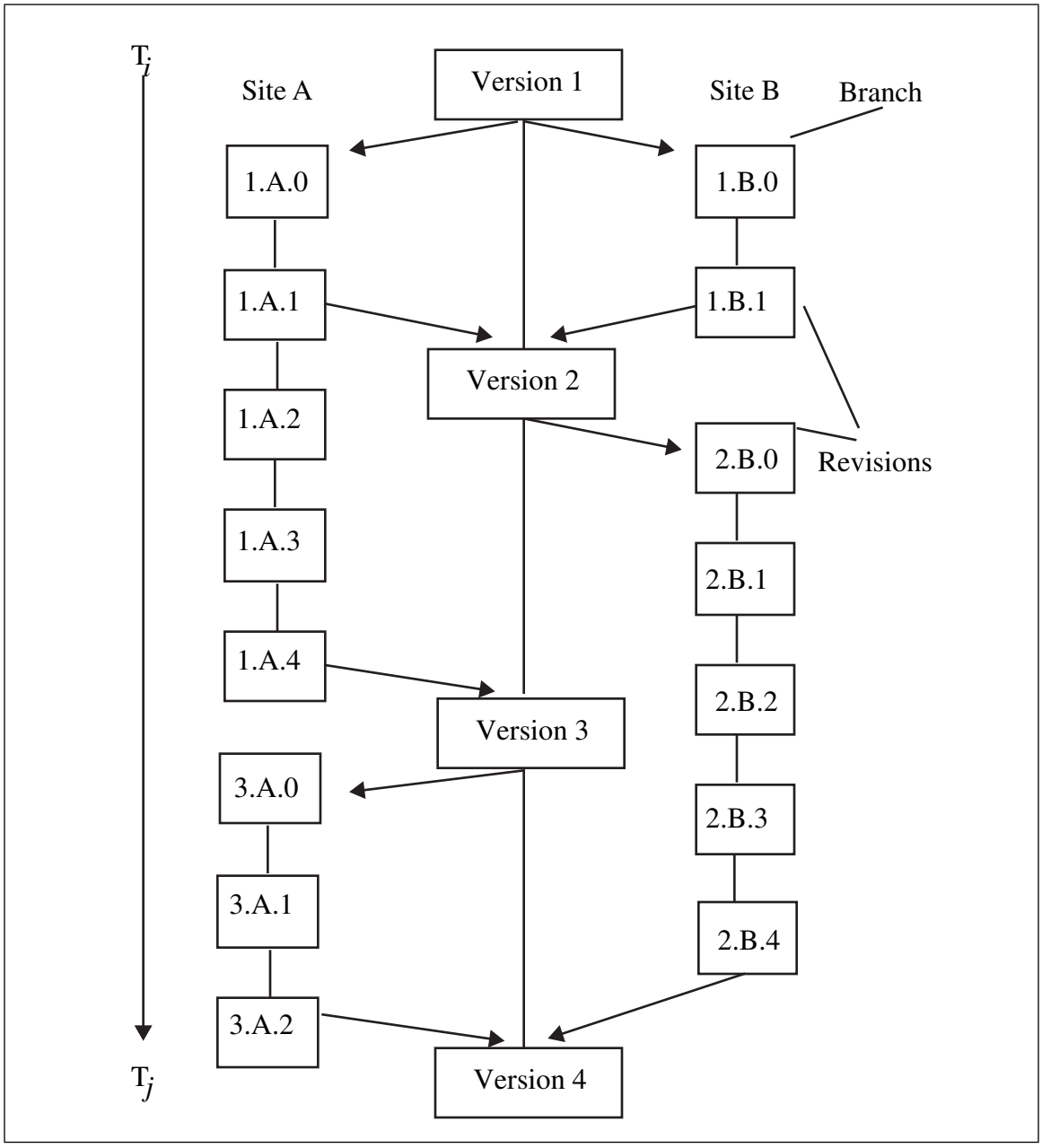


Figure 1-1. A development scenario where a central body coordinates all changes to the CMT while local sites retain authority over if and when they will submit changes to the central body, and if and when they will synchronize their local version with new reference versions. Left-right arrows indicate sharing changes between a local site and the central coordinator. Boxes below each site indicate serial revisions of a "branch" of the terminology.

sion 3, and only when it came time to create reference version 4 was site B able to contribute again.

Under the paradigm of centralized coordination, local control, a central steering committee recommends an overall direction, but only the local sites make actual commitments, sometimes to the detriment of the collaboration. The scenario illustrating this paradigm presents examples of where the collaborators failed to synchronize their work, thus increasing the complexity of future integration. Although the motives of this scenario are hypothetical, each of the failures of synchronization described by it actually occurred at least once during the Kaiser Permanente CMT project, despite the best intentions to have the participants synchronized at all times.

Clearly then, coordination of tasks between diverse groups can be challenging. Before such collaborations can be attempted, there must be some initial agreements about the nature of the terminology that the collaboration is intending to produce. One fundamental agreement regards the semantics of the underlying representation.

The identification of terminological conflicts is the focus of the thesis that I describe in this dissertation, and since these conflicts are an inevitable part of any distributed-development process, a description logic foundation is proposed to automate their identification. This foundation supports an enforced consistency within a terminology by providing modelers with a well-defined set of operations they can incorporate into their definitions, and by allowing the terminologies themselves to be interchangeable within a class of development tools that are also founded upon the semantics of description logic (Brachman et al., 1991; Brachman & Schmolze, 1985; Brill, 1993; Mays et al., 1991; Moser, 1983).

Currently, description logic has not been used extensively for medical terminologies, but as the demands placed upon these terminologies increase, their lack of formal semantics

will prove an insurmountable burden. CMTs require a logical foundation to live up to the demands of applications that depend upon them to deliver on the promises of supporting decisions, monitoring quality, expanding support for medical research, and improving the quality and reducing the cost of medical documentation.

1.4.3 Logical Representation of Clinical Data

Typical medical terminologies, such as SNOMED International (Côté et al., 1993) and ICD-9-CM (National Center for Health Statistics, 1995) use a hierarchical structure that organizes concepts. These kinds of hierarchies, however, have serious limitations (Campbell, Das & Musen, 1994). A simple hierarchical categorization neither sufficiently defines what a term represents nor tells how one term differs from another. Terminologies that use only concept hierarchies to categorize terms usually lack formal definitions for the terms in the system.[6]

Many research teams have sought to bring increasing formality to medical terminologies, some by developing logical definitions for the terms in a particular terminology, others by formalizing linguistically-derived relationships in the terminology (Bernauer, 1991; Cimino et al., 1994; Evans, Cimino, Hersh, Huff & Bell, 1994; Friedman, Cimino & Johnson, 1993; Masarie et al., 1991; Rector et al., 1993). I propose the use of description logics to define individual terms explicitly, thereby formalizing the defining relationships among these terms.[7]

---

6. Limitations of current CMTs as well as issues related to description logic as a foundation for CMTs are discussed further in Section 2.1.
7. The terminological definitions required and the relationships between those definitions on the one hand and information models and patient data on the other are presented in Chapter 3.

Formalization of a robust CMT is a challenging task. Candidate CMTs for formalization, such as the Systematized Nomenclature of Medicine—SNOMED (Côté et al., 1993)—are large, with typically over 100,000 terms. Terminologies of this size cannot be managed by a single developer, nor can all the design decisions necessary to complete the formalization be made in advance. Therefore, an evolutionary design approach is required.

1.4.4 Evolutionary Design

Evolutionary design is becoming more prevalent as an approach to software development. Rapid prototyping and user-centered design are representative examples. Evolutionary design has three fundamental principles (drawn from discussions surrounding Darwinian evolution (Dennett, 1995)):

1. **Evolution without pre-ordained design.** Although traditional design methods may be an efficient starting point for an evolutionary process, there is an explicit recognition that the design is not complete, and only through a development and feedback process can the product evolve to meet intended needs.
2. **Accumulation of design.** Throughout the development cycle, individuals may have developed insights into tasks that are manifested in their work. Such work should be archived, thoroughly analyzed, and incorporated, even in the event that such work conflicts with the work of another modeler.
3. **Heterogeneity**. By allowing and, indeed, encouraging a diverse set of approaches to focus on the development problem, the design efficiency of the development process is increased.

Evolutionary design may not be as efficient as traditional design, assuming that the design task is sufficiently small that a completed design is attainable. The heterogeneity of approaches found in evolutionary design, however, may allow the resulting medical terminology to meet the needs of a broader group of modelers. Moreover, a terminology designed by an evolutionary means, with participation from a broad consortium of model-

ers, may be preferable thanks to the broader base of participation in the development process and a greater sense of ownership among the modelers.

Evolutionary design can be made more efficient if computer applications are specifically tailored to support the evolution of a terminology. To make evolutionary design a realistic option, the first problem that must be overcome is to identify methods to allow modelers to work concurrently, to identify conflicting design decisions, to resolve these conflicts, and to disseminate the resolutions. There are two complementary classes of methodologies that, taken together, can make it possible for applications to support distributed CMT development: concurrency control and configuration management.

1.4.5 Thesis: Semantics-Based Concurrency Control

Traditional concurrency control schemes, such as those used by banking and airline reservation systems, are unable to support the especially demanding concurrency requirements of a nationally developed CMT. Specifically, new applications are required that will support development by users who are distributed geographically, who may be working with different versions of the master CMT, and who may spend weeks working on portions of the CMT before releasing their work publicly. Some aspects of this problem relating to use and enhancement of the Unified Medical Language System (Lindberg et al., 1993), have been described by Tuttle et al. (1991).[8]

In this dissertation I present a methodology for using *semantics-based* concurrency-control methods to support distributed development of a logically-based CMT. Such seman-

8. Section 2.2 of this dissertation provides a background of relevant concurrency control work.

tics-based methods use the underlying meaning behind terminological definitions to identify conflicts, and then seek to resolve such conflicts by optimizing the accumulation of design decisions from individuals working concurrently. By contrast, traditional methods of concurrency control force the acceptance of one individual's conflicting work at the expense of another's. Semantics-based concurrency-control methods can use an Aristotelian type hierarchy [9] to detect terminological conflicts created by concurrent development.

The fundamental thesis of this dissertation is that *Aristotelian classification can be used to identify concurrent-development conflicts within such description-logic systems. Further, the conflicts so identified are of sufficient frequency and terminological importance to warrant routine support for conflict-resolution in distributed description-logic development environments*.[10] Implementation of an Aristotelian classification is not sufficient, however. To successfully manage a distributed CMT development process, the Aristotelian classification methods must be combined with appropriate configuration-management methods. Together, they provide a viable framework for supporting distributed CMT development.

1.4.6 Configuration Management

Configuration management has been defined as "the process of identifying and defining the items in the system, controlling the change of these items throughout their lifecycle, recording and reporting the status of items and change requests, and verifying the com-

9. Discussed in Section 2.1.2 of this dissertation and further defined in Section 4.1.
10. Initial sections of Chapter 4 of this dissertation present the methodologies upon which Aristotelian classification can be used for semantics-based concurrency control.

pleteness and correctness of items" (ANSI/IEEE Standard 729, 1983). The development of any complex system requires configuration-management techniques to coordinate and control its construction: many of the basic principles have been developed for hardware engineering, large building construction, and software systems (Whitgift, 1991).

Concurrency-control methods for managing developmental changes that occur to terms in a CMT are an essential part of configuration management. Concurrency-control methods, however, are not sufficient to manage all configuration-management problems. CMT configuration management will also require naming conventions to identify items and versions of items, methods to verify the completeness and correctness of CMT configurations, and additional methods to create new CMT configurations.

Considerable work has been done developing computer-based tools for managing software configurations (Feiler, 1991) and hardware configurations (Barker & O'Connor, 1989). Some of the concepts created for these applications can be directly applied to CMT configuration management. New methods for managing CMT-specific problems are also needed.[11] I will describe how those advanced configuration-management methods can be combined with semantics-based concurrency control to manage CMT configurations.[12] Together, these methods will support CMT verification and automated migration from one version of the CMT to another.[13]

---

11. Section 2.3 of this dissertation presents relevant background work in configuration management.
12. See the discussion in the later sections of Chapter 4 of this dissertation.
13. Chapter 5 of this dissertation presents the prototype applications that incorporate distributed development support using a combination of Aristotelian classification as a basis for semantics-based concurrency control, and an advanced configuration management methodology. Chapter 6 presents an evaluation of how these combined prototype applications have performed within the Kaiser Permanente CMT project.

1.5 Technology: Necessary but not Sufficient

Although there are opportunities for computer-based tools to support the development process, such technological solutions must fit within an appropriate social framework. This framework largely determines the development process, and in turn, the quality of the product depends on the development process.

A real-world development effort will inevitably have more complex motivations, responsibilities, and processes than those found in the scenario illustrated above for the central coordination, local control paradigm.[14] A social framework must be developed for each effort. The purpose of this framework will be to control formally the problems of competing priorities, differing perspectives, failed performances, and consensus development.

Laying the appropriate groundwork for the Kaiser Permanente CMT project, including an agreement on an appropriate approach to development, took about two years and the dedicated effort of many committed individuals (often over the objections of other equally committed individuals). Undoubtedly, attaining a similar agreement at a state or federal governmental level will prove even more challenging.

The goal of the methodologies described within this dissertation is to support these social processes. It is the combination of appropriate technologies in support of a social commitment that will allow the promises of CMT based applications to be realized while the potential perils are avoided.

---

14. See Section 1.4.2.

1.6 Guide to this Dissertation

This chapter has provided motivation for, and an overview of, this dissertation. The remainder of this dissertation expands these ideas with concrete examples and specific methodologies. Chapter 2 provides background. There, I present material necessary to understand the development tasks and the existing work in other fields that can be applied to CMT development. This review incorporates a discussion of clinical-data classification, software-management principles, and concurrency-control and configuration management methods.

The next two chapters describe my approach for representing clinical data and for developing a terminology suitable for such a representation. In Chapter 3, I demonstrate examples of how a CMT can be logically represented using description logic, how a CMT participates in information models, and how patient data is represented using a combination of a CMT and an information model. In Chapter 4, I present formal methods for semantics-based concurrency control and change-set configuration management that can be applied to support the development of a logical-based CMT. This support includes methods for resolution of anticipated breakdowns in such a way that an evolutionary conservation-of-design development approach is practical.

The subsequent two chapters describe my implementation of semantics-based concurrency control and an evaluation of that implementation. Chapter 5 presents the applications that were used to demonstrate the semantics-based concurrency control and change-set configuration management. Chapter 6 contains an evaluation of those applications and how they supported the development process in an actual development setting.

Chapter 7 presents a discussion of this work, including ancillary lessons and directions for future work, and my concluding remarks.

Chapter 2
Background:
Classification, Concurrency Control and Configuration Management

Developing large systems, such as a controlled medical terminology (CMT), is a complex process. All such projects face two general challenges: to ensure that the proposed system is technically feasible and to manage development in such a way that acceptable levels of productivity and quality are maintained. These two challenges are closely related because a system's technical aspects will directly affect how easily it can be managed.

For any conceptual framework to be viable, it must be supported by a development environment that is cognitively, computationally, and organizationally scalable. A development environment is cognitively scalable if it supports the developer by computing inconsistencies and new relationships that might be otherwise overlooked. It is computationally scalable if the resources necessary for computation remain readily available as the size and complexity of the terminology within the classification grow. It is organizationally scalable if the management effort necessary to coordinate and integrate the individual work grows in a manageable way as the number of modelers increases. A description-logic classifier can preserve computational scalability by using a computationally tractable

sub-language of predicate logic with the properties of decidablilty and completeness. If efficiently implemented, it can also support cognitive scalability by computing coherence of concepts and by utilizing a classifier to identify inconsistencies and new relationships. Implementations of these classifiers have traditionally ignored organizational scalability, however, and they have had little—if any—support for more than one concurrent modeler, and none for identifying conflicting design decisions made by individual modelers.

The diversity of developers' needs and the tight coupling of needs with features in applications make support for pluralistic design essential. Moreover, since terminologies are inherently dynamic, computer-based support for their concurrent development will reduce the cost of migrating to new versions by offering a set of supported processes and tools that will make such migrations routine.

Several CMTs have long histories. For example, the International Classification of Diseases, was first initiated in 1853 (National Center for Health Statistics, 1995), the Medical Entities Subject Headings date back to 1960 (National Library of Medicine, 1992), and the Systematized Nomenclature of Medicine (Côté, Rothwell, Palotay, Beckett & Brochu, 1993) was derived from the Systematized Nomenclature of pathology, which itself began in 1965 (Wells, 1965). These vocabularies have served diverse needs for many years without having either formal processes for computer-supported distributed development or formal languages for logically defining the relationships of terms contained within one CMT to those found in the others. These CMTs are becoming more complex as they evolve, and the demands placed upon them by the pursuit of the Electronic Medical Record (EMR) are

enormous. Existing management infrastructure is inadequate to ensure continued growth of these important CMTs.

CMTs are becoming increasingly expressive due to increases in size. They are also becoming increasingly difficult to analyze due to inconsistencies in their use and the complex interrelationships between terms that are often not explicitly defined. To address these problems, CMTs will need to be enhanced to allow them to be used consistently and to make it possible to define interrelationships among terms explicitly. Management of this enhancement will be a complex process, involving the participation of many groups with competing priorities. Creation of an appropriate management infrastructure for carrying out CMT enhancement is an accordingly essential first step.

Fortunately, such an infrastructure need not be developed from scratch. General principles of management can be directly applied to CMT development, and many specialized principles of software development can also be adapted and applied.

Many large project-management techniques are surprisingly consistent, as the general applicability of such management principles as quality control to diverse fields such as manufacturing, marketing, research and development, and health care demonstrates (Berwick, Godfrey & Roessner, 1990). Although the fields are diverse, they all accomplish productive work by following an appropriate process. Shewhart (1931) recognized that the quality of a product is inseparably linked to the quality of the development process, and therefore recommended that management focus on analyzing and improving the process rather than focusing exclusively on the product.

2.1 Clinical Data Classification

Collecting clinical data is expensive and time consuming. Currently, only minimal data are routinely encoded. Any analysis requiring new data elements consumes significant resources to (1) develop a sampling plan for the data, (2) develop data collection instruments, (3) train the data abstracters, (4) collect the data, and (5) analyze the data.

In today's cost- and quality-conscious atmosphere, there is increasing interest in routinely encoding more data, such as that found in admission histories, physical examinations, progress notes, and discharge summaries. Some researchers have been working on applications for collecting these data as part of the care process (Bell, Greenes & Doubilet, 1992; Benoit et al., 1992; Campbell, Wieckert, Fagan & Musen, 1993; Kuhn, Zemmler & Heinlein, 1993; Naeymi-Rad, Almeida & Trace, 1992; Rector, Nowlan & Kay, 1991). Other researchers have been developing applications for extracting these data from narrative text (Baud et al., 1993; Lamiell, Zbigniew & Isaacks, 1993; Lenert & Tovar, 1993; Sager, Lynman, Tick, Ngô & Bucknall, 1993). Agencies such as the Health Care Financing Administration (HCFA) recognize that standardizing and automating the representation of medical information are essential to increasing their programs' efficiency (Audet & Scott, 1993; United States General Accounting Office, 1993). Unfortunately, there is now no standard that is capable of representing this kind of detailed clinical data.

2.1.1 Common Terminology Structure

Typical medical terminologies, such as SNOMED International and ICD-9-CM (National Center for Health Statistics, 1995), use hierarchical structures that organize concepts into

concept hierarchies. As an example, SNOMED and ICD-9-CM classifications of "pleural effusion" are presented in Figure 2-1.

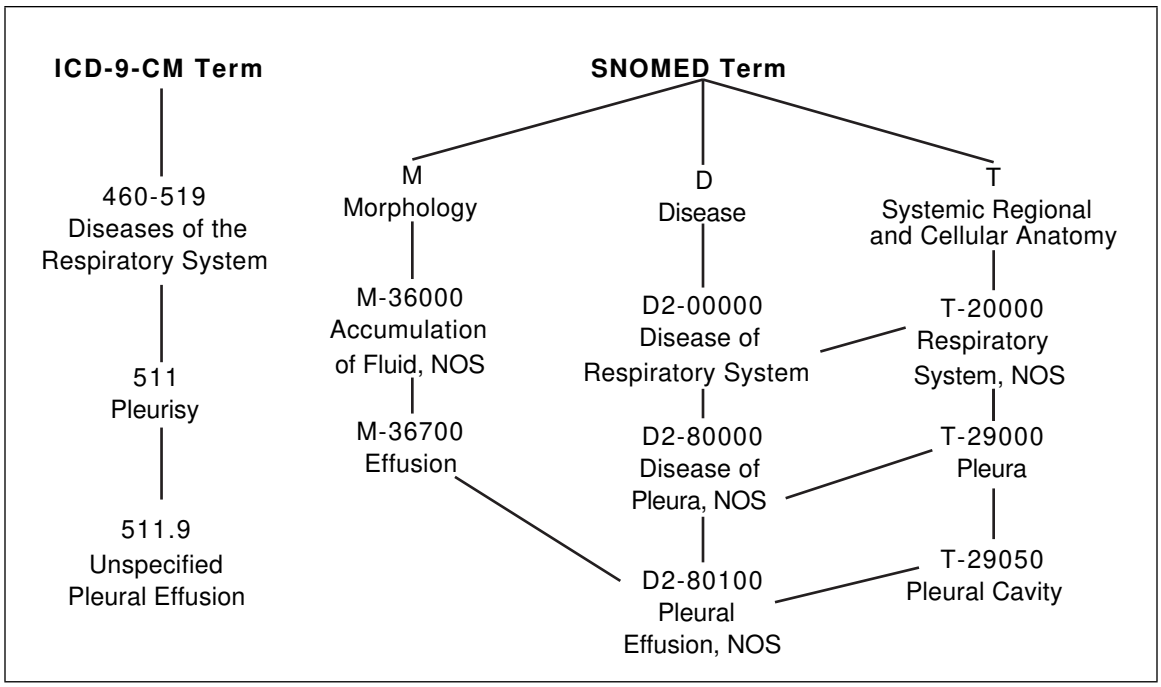


Figure 2-1. An ICD-9-CM hierarchy (left) and an analogous SNOMED hierarchy (right), showing classification of "pleural effusion." The dashed lines represent cross-references provided by SNOMED. The ICD-9-CM terms have no cross-references. NOS is an abbreviation for "not otherwise specified." This abbreviation is used in SNOMED to indicate that the term is general, and there are children of the term that are more specific.

Concept hierarchies provide a mechanism for organizing a terminology systematically, but they have significant shortcomings. This form of categorization neither sufficiently defines what a term represents nor indicates how one term differs from another. Terms that have the same parents (e.g. asthma and bronchitis may have the same parent: pulmonary disease) are obviously related, but since they were not given the same termcode, they are also different from one another in some undefined way. This kind of classification scheme works well for a hierarchy in which (1) a term needs only one parent, (2) the hierarchy uses only terminology that can be unambiguously interpreted by anyone using it, and

finally, (3) minimal automated processing of the terms in the hierarchy is required. In all other cases, the organizational mechanism has serious limitations. These limitations may contribute, for instance, to the significant random errors in ICD-9-CM's classification of hospital discharge diagnoses. Error rates between 20 and 29 percent have been reported in the literature (Hsia, Ahern, Ritchie, Moscoe & Krushat, 1992; Smith, 1989; van Walraven, Wang, Ugnat & Naylor, 1990). If a terminology could be subjected to automated checking for internal consistency, these errors might be reduced.

The most serious limitation of terminologies that use only concept hierarchies is their lack of formal definitions (explicit representations of meaning, either in prose or in logic) for each term. If each term in a terminology were formally defined, it could be used more consistently. Some medical-terminology systems, such as the Medical Subject Headings (MeSH) (National Library of Medicine, 1992) and Diagnostic and Statistical Manual of Mental Disorders (DSM-III) (American Psychiatric Association Task Force on Nomenclature and Statistics, 1980), have prose definitions intended for the human reader. If these definitions could also be processed by a machine, tools could be developed to handle the processing consistently. Because the current definitions of terms in hierarchies such as ICD-9-CM and SNOMED are not formalized, each meaning is subject to broad interpretation by users, and automated tools that could reliably use these terminologies cannot be developed.

2.1.2 Historical Perspective

Concept hierarchies, similar to the ones used by most medical terminology systems, were introduced by the ancient Greek philosopher Aristotle around 300 B.C., with his theory of

categories (Hutchins, 1952). Aristotle also developed a method for defining new types within the type hierarchy by *genus* (the category of classification for a term) and *differentia* (the elements, features, or factors that distinguish one term from another), and for using deductive arguments to analyze the inheritance of properties of these new types. This kind of type hierarchy is called *Aristotelian*. The concept hierarchies in Figure 2-1 already define the genus of each term by the lines connecting each term to its parent. To make these concept hierarchies Aristotelian, differentia for each term must be specified. A definition that contains both the genus and differentia for the term PLEURAL-EFFUSION, for example, can be represented by the phrase "a pleural effusion is an effusion located in the pleural cavity and caused by a disease." The genus is represented by the statement "a pleural effusion is an effusion" and the differentia is represented by the modifier "located in the pleural cavity and caused by a disease."

A little more than 300 years ago, the German philosopher and mathematician Gottfried Leibniz developed the first system capable of *computing* the elementary concepts from which more complex terms are composed. This system, the Universal Characteristic, represented primitive concepts as prime numbers. It created compound concepts by multiplying primitive concepts together. If PLEURAL were represented by 3 and EFFUSION were represented by 7, their product, 21, would represent PLEURAL-EFFUSION. Leibniz's system was certainly visionary, and led to his development of the first mechanical computer capable of doing multiplication and division.

The Universal Characteristic was essentially the first mechanical implementation of a multiple-inheritance type hierarchy. Because it implemented only a type hierarchy and was

not able to represent machine-processible definitions using differentia, it suffered from the same limitations as other concept hierarchies. The only logical relations permissible in such a system are conjunctions of primitives. To represent the logical relations necessary to define a system for medical concept representation, more complex relations are needed, such as the ability to use defined relations such as AFFECTS and PART-OF relationships. Although having an expressive language helps us make statements that closely approximate our understanding of real-world conceptual entities and their relationships to one another, there is an associated tradeoff in complexity. The more expressive the system, the greater the chance that statements cannot be proven to be consistent with one another because proof of theorems in complete first-order logic is, in the worst case, intractable, that is to say, even the fastest computers combined with the best algorithms may never be able to answer certain questions. Using an efficient description logic based on a subset of first-order logic, however, can provide an expressive language with properties of decidablilty and completeness.

Although Leibniz was limited by having only mechanical computers capable of doing multiplication and division, there are now far fewer current constraints on what can be computed. Modern computing power makes it possible to implement concept-representation systems that define terms by genus and differentia, which in turn will allow automated methodologies to be developed for processing a terminology. As a rule, the more complete the differentia defined by such a system, the more powerful the tools that can be developed. The ideal system would allow the differentia to be defined in a *complete* system of logic, and would be *efficient* in its processing abilities. This, however, is a paradoxical requirement. Since completeness and efficiency cannot be achieved at the same time,

developers must chose where their applications should fit on a continuum of efficiency and completeness.

Some current medical classification systems, such as ICD-9-CM, continue to use only simple hierarchical structures. Other current medical classification systems, however, such as SNOMED and MeSH, use what are known as multiple-inheritance hierarchies. These hierarchical structures often define the genus of each term, or some other relationship between two terms. They do not formally define the differentia of each term, however, and therefore do not define an Aristotelian hierarchy.

Although comprehensive, *logic-based approaches* have yet to be adopted by the principal medical naming systems, researchers have been exploring how other formal systems could be used for medical terminology. For example, Cimino and colleagues have constructed the Medical Entities Dictionary using a *semantic network* (Cimino, Clayton, Hriscsak & Johnson, 1994). Masarie and colleagues have described a *frame-based approach* for mapping equivalent concepts between clinical vocabularies (Masarie, Miller, Bouhaddou, Nunzia & Warner, 1991). In another project, Rector and colleagues have described an approach that uses their *Structured Meta Knowledge (SMK) formalism* (Rector, Nowlan, Kay, Goble & Howkins, 1993). These systems can also be described using first-order

logic.[1] All applications stand to benefit if logic is adopted as a common syntactic foundation for representing medical concepts.

2.1.3 A Logical Foundation

Further evolution of controlled medical terminologies is needed so that they become capable of expressing structured knowledge. They should also provide methods for accessing and reasoning with that structured knowledge in a principled way. A natural first step is to formalize the representation of controlled medical terminologies using a description logic.

Description logics are a class of languages used to express knowledge about concepts and concept hierarchies. They use declarative semantics (they provide statements of meaning that are independent of any particular method of processing said meaning) with formal foundations derived from predicate logic (description logics must be faithful to predicate-logics methods of deriving a conclusion from a set of assumptions). Description logics allow a terminology modeler to declare a set of primitive concepts (such as DISEASE and ORGANISM) and relationships (such as CAUSED-BY), and to define new concepts (such as INFECTIOUS-DISEASE) using existing concepts and relationships supported by a particular description logic (thus INFECTIOUS-DISEASE can be defined as a DISEASE CAUSED-BY an ORGANISM).

1. Schubert (1991) appeals to the knowledge representation community to consolidate ideas, discarding artificial distinctions and inconsistent terminology. He argues that all knowledge representation schemes which aspire to cope with large, general propositional knowledge bases, such as “frame-based systems,” “semantic databases,” “blackboard systems,” and “semantic networks” are in fact just a set of first-order predicate calculus formulas, in a “slightly altered terminological guise,” (p. 99) and therefore are examples of artificial distinctions derived from individuals who find it “faster to rediscover something within the framework of one’s colony than to glean it from the writings of another” (p. 96). In his exposition, he demonstrates first-order logic predicates capable of representing semantic networks.

Since they are founded on predicate logic, description logics are relevant to any area of reasoning. They are topic neutral (Spencer-Smith, 1991), that is to say, they can be applied to problems in any domain with equal validity. In certain fields, such as linguistics and knowledge representation, practitioners use this kind of logic extensively. Computers use logic at their lowest level, in circuit switching, at their highest level, in logic programming languages such as Prolog, and to represent knowledge in expert-system shells such as KL-ONE (Brachman & Schmolze, 1985). As a class of unifying formalisms, description logics provide formal underpinnings to frame-based systems, semantic networks, KL-ONE derivative languages, and object-oriented representations.

Although logical formality forms the basis of the work undertaken in this dissertation, some previous attempts to use it have been problematic. Early artificial-intelligence systems ignored logical soundness, in some cases because early researchers were overwhelmed by problems of syntax and semantics (Sowa, 1984). Other researchers failed to understand logic properly, confusing it with a particular programming system or syntax (Hayes, 1977). Today, logic has been making a resurgence, thanks to the acceptance of a family of newly-defined description-logic languages (Brachman, Fikes & Levesque, 1983; Brachman, McGuinness, Patel-Schneider, Resnick & Borgida, 1991; Moser, 1983).

In addition to these systems, new standards for interchanging information are based on logic. These include the Knowledge Interchange Format (KIF) (Genesereth & Fikes, 1992) being developed for the Defense Advanced Research Project Agency (DARPA), the Information Resource Dictionary Standard (IRDS) developed by the American National Standards Institute (ANSI) (Perez & Sarris, 1993), and the Knowledge Representation

System Specification (Patel-Schneider, Swartout & KRSS working group of the DARPA Knowledge Sharing Effort, 1993).

Since, as noted, full first-order logic is computationally intractable in the worst case, it might seem reasonable to ask why various efforts are adopting an intractable formal system to serve as a basis for their standards. Basically, the answer is that despite the worst scenario, logic has proven its value over time. One prominent example of a well-established standard based on first-order logic with set-theoretic extensions (to allow manipulation of collections of elements) is the Standard Query Language (SQL) (Ullman, 1988). The benefits of having a logically consistent data-modeling and data-retrieval mechanism far outweigh the problems associated with being able to create a series of relational queries that may not run to completion. Moreover, intractable queries can be eliminated—if desired—by limiting the expressivity of the query model, while maintaining its properties of logical soundness and logical completeness (Mays et al., 1996).

Logical notation allows relationships between terms in a system to be formalized, so that applications can make valid inferences using those relationships (determining that Aspirin is contraindicated in patients who have ulcers for example). For any system, such as database systems, where these relationships need to be processed, these formal relationships provide the most powerful argument in favor of using logic. Without this formality, ad hoc approaches (e.g., “rules of thumb”) must be used that may result in questionable conclusions because the soundness of these methods cannot be proved.

Description logics can readily be applied to terminological definitions. Consider the definition of the term PLEURAL-EFFUSION previously given in Section 2.1.2. It can be stated using logic as

> For all x, x is a pleural effusion if
> x is an effusion, and
> for some pleural cavity y, and for some disease z,
> x is located in y and x is caused by z.

Using predicate calculus notation, the definition is restated as

$$\forall x \text{ PLEURAL-EFFUSION}(x) \equiv \text{EFFUSION}(x) \wedge \exists y \text{ PLEURAL-CAVITY}(y) \wedge \exists z \text{ DISEASE}(z) \wedge \text{LOCATED-IN}(x,y) \wedge \text{CAUSED-BY}(x,z).$$

This dissertation has employed a a specific description logic, the Knowledge Representation System Specification (KRSS) (Patel-Schneider et al., 1993), and it is the syntax upon which the prototype tools to be described in Section 5.1 were based. Using KRSS, the definition of the term PLEURAL-EFFUSION can be restated as follows:

```
(define-concept PLEURAL-EFFUSION
    (and EFFUSION
        (some LOCATED-IN PLEURAL-CAVITY)
        (some CAUSED-BY DISEASE)))
```

It should be noted that the semantics of the KRSS definition are the same as that of the predicate calculus definition.

2.2 Concurrency Control

It is inevitable that a CMT will be developed incrementally by multiple developers. This will require breaking down the actions of individual developers into semantically atomic units called *transactions*, each representing a single operation on a CMT term. These will

usually be what might be called long-lived transactions because they will be created over several weeks by individual developers before they are made public.

Development of advanced-database applications (Barghouti & Kaiser, 1991) able to support distributed development of a CMT will require improving two current classes of techniques to support long-lived transactions. First, they will need to support merging of long-lived transactions that are developed concurrently. These *concurrency-control techniques* must support the integration of many sets of long-lived transactions in a manner that preserves as much individual work as possible, yet ensures the consistency of the resulting CMT. Second, they must integrate the concurrency-control techniques with others designed to assure version and configuration control.

2.2.1 Transactions

All operations are seen by the database, and therefore by the overseeing concurrency control scheme, as a series of read and write operations. These operations are grouped together into ordered sets of operations referred to as transactions. Grouping operations into transactions serves three purposes (Lynch, 1983):

- Operations are grouped together to form a complete task.
- Sequential execution of well-formed transactions preserves the consistency of the database.
- Grouping forms a logical “all” or “none”: executing all, or none of the operations will preserve the consistency of the database.

If a transaction fails after starting execution (either because of a software or hardware failure) database consistency can be restored by undoing all the operations of transactions that have not yet been completed.

2.2.2 Traditional Concurrency Control

Traditional schemes for managing consistency are designed to be general purpose. They lack, therefore, any information about the application or the semantics of the database operations created by the application. Transactions from multiple users, or multiple transactions from a single user, are executed sequentially according to a schedule. Figure 2-2 shows a serial schedule (one that requires completion of one set of actions before another set of actions can begin) in which two users can sequentially operate on the same definition, preserving the consistency of the database. In this example, the serial schedule is ensured by the locks placed on each term before modification and released after committing the actions.

Traditional concurrency control schemes (Barghouti & Kaiser, 1991) rely on locking mechanisms or optimistic nonlocking mechanisms (see descriptions below) to create a serial schedule of transactions. If a serialization of all transactions cannot be found, one or more transactions are aborted to allow the others to complete. The work involved in creating the aborted transactions must be repeated.

Locking Mechanisms

Two-phase locking (Eswaran, Gray, Lorie & Traiger, 1976) is the standard mechanism of concurrency control in conventional database management systems. Two-phase locking

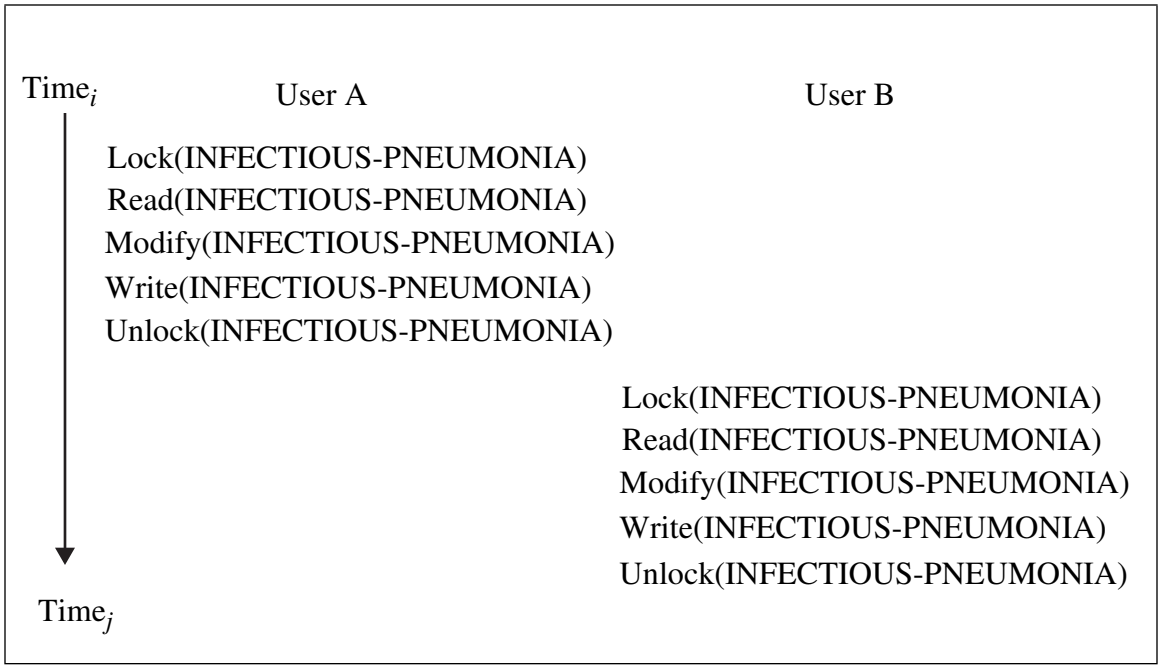


Figure 2-2. Valid serial sequence of transactions. Valid serialization is assured with lock and unlock operations.

guarantees the serializability of transactions (ordering of transactions into a sequence of non-interfering operations where one transaction must complete before another is allowed to begin) in a centralized database when transactions are executed concurrently. Transactions that utilize two-phase locking are divided into a growing phase and a shrinking phase. During the growing phase, transactions sequentially lock all of the data elements that may be read or modified. Once transactions have acquired all the necessary locks, the data elements are processed. The shrinking phase begins when transactions release the locks one by one until the transactions are completed. Although this ensures consistency of the database, a transaction is required to wait until all of the necessary data elements are released by other transactions.

Locking mechanisms are not appropriate for CMT development because a CMT developer may work with a set of data elements for weeks before releasing them publicly. During

these long-lived transactions, all other developers would be prevented from working on that set if locking mechanisms were in place. Another argument against locking mechanisms is their inability to support parallel pilot projects, in which a group may wish to work on a project until they have proven the merits of their approach (even though reliable access to a central database is available), and therefore seek to delay the integration of their work with the concurrently developed work of others until these merits have been convincingly demonstrated.

Optimistic Nonlocking Mechanisms

Kung and Robinson (1981) proposed a mechanism for optimistic concurrency control that does not rely on locking mechanisms. Their version of optimistic concurrency control requires that all transactions have a read phase, in which all data are written to local copies, and a validation phase, in which the application must show that committing this transaction will not violate the serialization of all previously committed transactions (the application verifies that the data used or modified by this transaction will not interfere with the data requirements of another transaction and that no other transaction has modified the data used by this transaction during its duration).

This mechanism can provide concurrency control with less overhead (computer time associated with administrative tasks) than two-phase locking requires. It is most useful when serialization conflicts are uncommon. If a transaction violates serialization, it must be aborted, and its work must be repeated. Moreover, this mechanism requires that all users operate on a central database so that serialization of transactions can be validated before a

transaction is allowed to commit (the process of finalizing the transaction and writing the resulting data to the database).

Kung and Robinson's mechanism does offer an improvement over locking mechanisms for CMT development. Because there are no locks, it allows multiple developers to work with the same data elements. Despite this improvement, it still relies on traditional serialization before allowing a transaction to commit. If serialization is violated, the work involved in creating the transaction is lost. This is an important consideration because *if developers fear that their work will be lost, they will be reluctant to participate in distributed development*. A better solution would be one that can resolve violations of serialization after the fact without forcing transactions to abort.

Shortcomings of Traditional Concurrency Control

All traditional database applications implement concurrency control by requiring serialization of transactions—either before the transaction begins (with two-phase locking), or before the transaction commits (under optimistic concurrency control). In addition, if serialization is violated, the transactions are aborted and must be redone. Practically speaking, this means the human thought required to create the aborted transaction must be repeated.

There would be several adverse consequences of relying upon traditional concurrency control for CMT. First of all, it might be necessary to wait months for long-lived transactions to commit, or else applications will require manual recreation of any transactions that abort because of violations of serialization. Second, CMT applications would need to be connected to a central database during the creation of transactions. This requirement

alone would prevent the creation of local enhancements or using distributed development in which developers regularly submit their local enhancements for inclusion in a master CMT.

Traditional concurrency control schemes then do not support long transactions and or cooperative development among developers. Two developers who want to exchange work produced independently (i.e., while not connected to a central database) are seeking a form of cooperative development that Yeh and colleagues (1987) referred to as synergistic interaction—a process that CMT development should support—but they may very well have produced contributions that cannot be serialized.

2.2.3 Semantics-Based Concurrency Control

Traditional concurrency control schemes can be improved by using domain-specific knowledge. If an application is specialized for a domain in which the semantics (the intended purpose) of the transactions are known, a nonserializable, but consistent, schedule can be constructed for these transactions. For example, if it is known that two transactions are intended to add synonyms to the same concept, those transactions might be allowed to run concurrently since the addition of one synonym does not invalidate the other (as long as both synonyms are *really* synonymous).

Garcia-Molina (1983) observed that the serializability requirements used by concurrency-control techniques could be replaced by constraints based on requirements of semantic consistency if an application possesses semantic information about the transactions. He proposed that a semantically consistent schedule be sought rather than a strictly serializ-

able schedule to preserve the consistency of the database. This scheme allows a class of semantically consistent (but not serializable) transactions, yet it still suffers from many of the problems of the traditional concurrency control schemes. Transactions that violate the constraints of semantic consistency must still be rolled back, and the labor used to create them must be repeated.

In a later paper, Garcia-Molina and Salem (1987) proposed that long-lived transactions be organized into what they called *sagas* (a set of transactions with a known purpose) that could be interleaved (no serialization is required) with other transactions. The execution of sagas is managed by a saga execution component (SEC) that uses the application's traditional transaction management. The SEC executes the saga transactions one by one, keeping a log of all the actions taken on behalf of the saga. The SEC may sequentially execute all of the saga's transactions without error, or it may partially execute a saga, discover unrecoverable conflicts, and then roll back the saga, by removing all of the saga transactions from the database. The saga is rolled back not by the traditional cascading rollback of conventional concurrency control systems, but rather by relying on compensating transactions that will restore the database to a semantically consistent state. This has the advantage of not requiring the rollback of other transactions by the database that occurred during the aborted saga's execution. The compensating transactions are domain- and task-specific, and require semantic knowledge about the saga's transactions to be properly applied.[2]

The use of sagas and compensating functions that remove the effects of aborted transactions from databases would solve some of the problems of traditional concurrency control.

Their use would prevent databases from unnecessarily rolling back concurrently executing transactions by applying the compensating functions and by allowing interleaving of transactions in an order that may not be strictly serializable.

Sagas do not, however, provide conservation-of-design mechanisms.[3] Anytime a conflict is detected, at least one of the competing transactions must be aborted and manually recreated from a new base configuration. A better solution would be to use semantic knowledge to compensate for conflicts in a way that does not require choosing one transaction at the expense of another (assuming that there are appropriate enhancements in both transactions). Such a solution, using semantically equivalent sets of transactions (as will be described in Chapter 4) is preferable because the work involved in creating both transactions can be preserved.

---

2. The compensating transactions described by Garcia-Molina (1987) are specific to undoing the actions of an aborted saga. Specifically, if a saga is composed of individual transactions T_i, then for each transaction there must be a compensating transaction, C_i, that will remove the effects of the transaction from the database, restoring it to a semantically consistent state. If a saga consisted of the sequence

   $$T_1, T_2, T_3, \ldots T_n.$$

   and for each transaction, a compensating transaction was defined

   $$C_1, C_2, C_3, \ldots C_n.$$

   either the entire saga sequence,

   $$T_1, T_2, T_3, \ldots T_n.$$

   is executed, or the sequence

   $$T_1, T_2, \ldots T_j,\ C_j, \ldots C_2, C_1$$

   for some $0 \leq j < n$ will be executed.

3. See Section 1.4.4.

2.3 Configuration Management

Coordinating changes made by multiple developers on a single product can be complex. Figure 2-3 illustrates such a scenario. Two developers independently modify version 1 of a product, and periodically swap their changes with each other to create version 2 of the same product. There are many potential problems under such a scenario. How are changes to be represented? How are those changes to be exchanged? What if changes by one developer conflict with, or negate, the changes of another? Without a plan for handling such problems, it would be risky to allow more than one person at a time to work on a product.

After the locally developed changes are merged into a new reference release, the process is assumed to start over. Each institution will create new branches from the new reference CMT, and the cycle will begin again. This cost-free resumption of the development cycle overlooks one important step, however. All local changes have been successfully merged into a new reference version, but the local versions have not been synchronized with the new reference version. The extra effort required to synchronize local and reference versions is a *local-update penalty*. If the institution had not created local enhancements, it would not incur this penalty.

Mechanisms for managing these problems have been devised for software development, and these mechanisms are continuing to evolve. The concurrency-control mechanisms described above can broker strict control, often preventing conflicts, but these strict controls are unrealistic for software development. Software is typically developed by teams working in parallel rather than by individuals working serially. Moreover, software must often work in different environments in parallel, and so development for these different

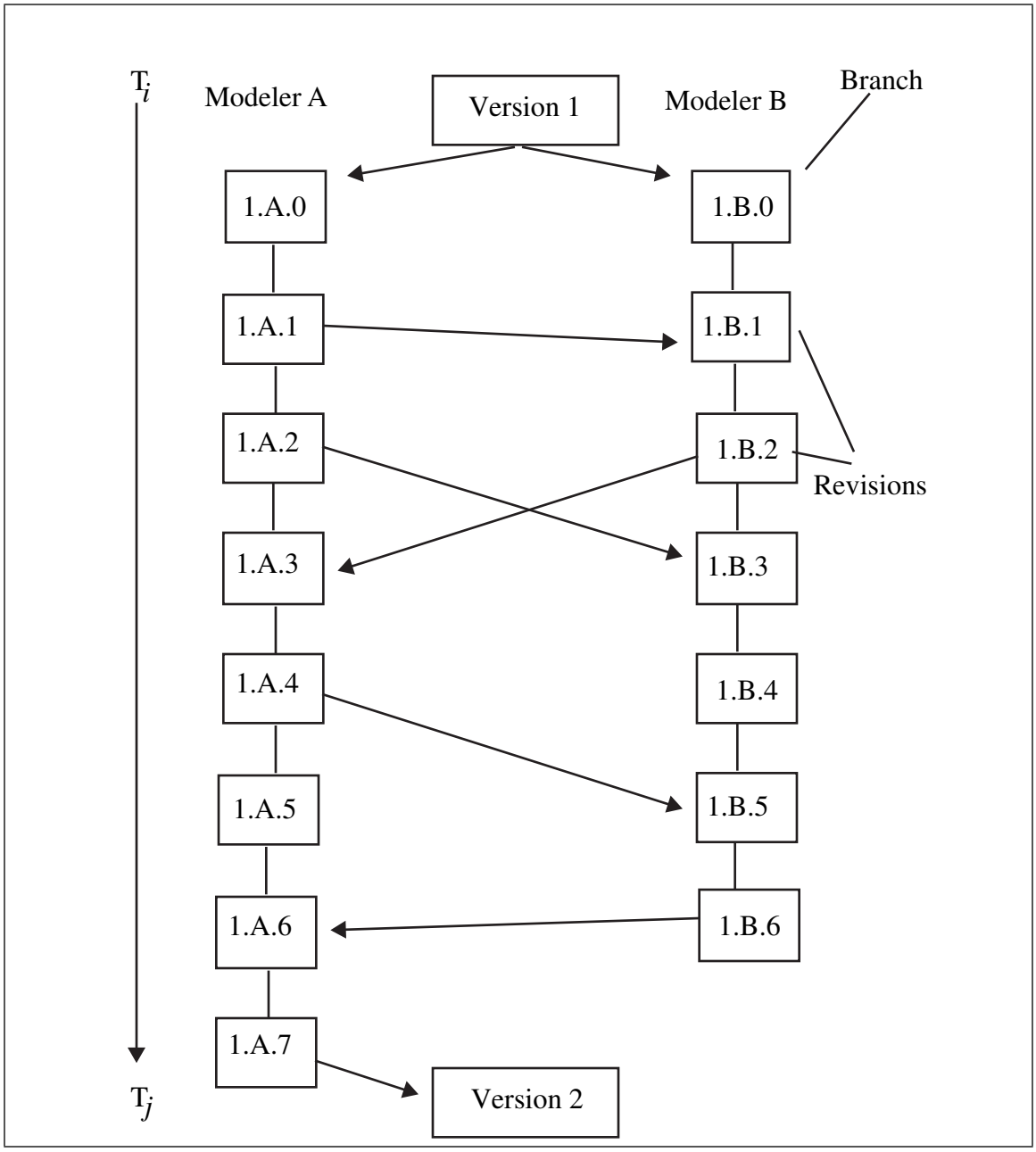


Figure 2-3. A development scenario with no central coordination. Modelers A and B each begin working with Version 1 of some terminology. Left-right arrows indicate sharing changes between two modelers. Boxes below each modeler indicate serial modifications of the terminology.

environments is also typically done in parallel. Rather than allowing only one version of a product to exist, with only one modification allowed at any time, software developers have chosen to allow multiple versions of a product to exist and have developed configuration

management mechanisms to manage product versions. The remainder of this section discusses three such models.[4]

2.3.1 Configuration Management Models

Configuration management models have been developed in a variety of disciplines in which objects must be combined to create a final product. Although these models often have common underpinnings, there are domain-specific differences as well. These differences are necessary because domain objects (objects specific to a particular discipline such as source code for software or a circuit board for electronic hardware) may have different physical and logical characteristics, different procedures for creation and use, and different lifecycle characteristics (Dart, 1992). Despite these differences, the potential exists for synergistic sharing of ideas between disciplines. Dart (1992) and Katz (1990) have described the common themes that are found in software development and computer-aided design (CAD) environments. CMT development also has concerns that are similar to those of the software development and CAD environments.

The common link among all disciplines concerned with configuration management is the cataloging of components that may evolve over time. Some rudimentary means of dealing with this evolution is present in all configuration management systems. More advanced systems offer additional functionality. Feiler (1991) examined commercial software configuration-management systems, describing their similarities and differences. The models he described represented not only the spectrum of system functionality found in commer-

4. The term *model* is frequently used in the software configuration management literature. In that context, as well as my use in this dissertation, the term refers to "a description of a system or theory that accounts for all of its known properties" (from the American Heritage Dictionary, Second College Edition).

cial systems but also the evolution of configuration management concepts since the development of the Source Code Control System (SCCS) (Rochkind, 1975) in the early 1970s. This section presents three of the models Feiler described: the checkout/checkin model, the long-transaction model, and the change-set model. These represent the spectrum of changes in configuration management concepts from the initial checkout/checkin model for cataloging and storing software versions to the most recent change-set model that captures changes as identifiable components which can be applied to selected configurations of other components to create a new configuration.

Checkout/Checkin Model

The checkout/checkin (CO/CI) model is the model most familiar to developers, but is also the most limited. It is used by the UNIX tool Revision Control System (RCS) (Tichy, 1985) and many others. CO/CI tools provide a repository where developers can store versions of software, source code, and documentation. Creation and maintenance of the repository is a responsibility of the developers who are to check appropriate components into, and out of, the repository. In addition to repository functions, CO/CI tools typically provide a status accounting of the components in the repository. Typical accounting functions include a record of all persons who checked components into or out of the repository, and a binary "snapshot" of each version checked into the repository. Any previous version of any component can be retrieved from these records, with simple library functions serving to prevent two developers from checking out the same component at the same time.

Thanks to these library functions, CO/CI tools can enforce a concurrency control model that will ensure a serial sequence of changes by requiring that only one developer at a time

can check out a component. The discussion of traditional forms of database concurrency control (see Section 2.2.2) has described how such serial sequences ensure valid sequences of transactions (using lock-based techniques or optimistic nonlocking techniques). As noted, however, such serial sequences are too restrictive and will not meet the requirements of CMT development. Software developers have also known that strict serialization is not acceptable in many situations, and they require the CO/CI tools to allow multiple developers to check out the same components for concurrent modification. When this occurs, the tools allow the creation of a version branch, a new version of a component that will subsequently evolve independently from the original component. Any changes in the version branch that must later be incorporated into a subsequent version of the original component must be integrated manually.

The repository component is fundamental to the CO/CI model, but it is not sufficient for configuration management. It simply creates an audit trail of all the checked-in versions of components. It has no notion of how these components can be combined to create valid configurations. Software developers depend on additional tools to specify relationships between different components of a system, to create valid configurations of components, and to control the process of building the software application. Typically, developers rely on tools like UNIX's *make* (Feldman, 1979).

Make is a tool that is able to read a file of specifications about a software product and generate a set of commands that can create a new, valid product configuration. *Make* generates these commands from a set of build commands and file dependencies, supplied by the developer in a file called the *makefile*. The burden of proper execution is placed on the

developer. Not only must the developer properly specify all the file dependencies and build commands to generate a product properly, she must also make sure that all the compilers and linkers specified by the build commands are properly installed and available to the *make* tool. The CO/CI model places these burdens on the developer because it does not provide a complete development environment. The CO/CI then can be seen as a combination of special-purpose tools targeted at specific problem areas in the process of building software, but it is not a comprehensive solution. The burden of integrating these tools, and of providing additional automated solutions where possible, is placed upon the developer, rather than on the environment.

One reason for the limitations of the CO/CI model is that its tools do not support the process of changing a component. CO/CI tools become aware of the existence of modifications only after a developer commits those changes by checking a file into the repository or when a changed file has been saved to disk. All file modifications are generated using tools, such as text editors, that are external to the CO/CI model. Newer configuration management models, such as the long-transaction and change-set models, seek to have all component changes occur within a single environment, thus allowing the environment to provide additional functionality.

Long-Transaction Model.

The long-transaction model focuses on the evolution of an entire system. Any time a change occurs to a system component, the configuration management environment will record a corresponding transaction. A system therefore evolves from an initial, or base configuration to a new state by executing a set of transactions that represent apparently

atomic changes of system components. Often several developers will be working on the same system. Each of their changes will be represented by transactions that must be sequentially applied to a base configuration. These transactions are coordinated by a concurrency-control scheme that is internal to the development environment, rather than external (as in the CO/CI model's branching and manual merging).

Environments that support the long-transaction model must provide each developer with a workspace, a base configuration on which all transactions will be applied, and mechanisms for recording and applying transactions to this configuration. The workspace shields the developer from changes in other workspaces until they are committed. The concurrency control scheme brokers the transactions between different workspaces when the transactions are committed.

Incorporating a concurrency-control scheme into a configuration management environment has potential advantages and drawbacks. One advantage is that if a strict concurrency-control policy is adopted, existing database concurrency-control models can be applied. Using these models will ensure that properly sequenced transactions can be validly applied to the base configuration, and will also allow developers to utilize traditional database shells for configuration management applications. An advantage from one perspective can, however, be a hindrance from another perspective. As discussed above, traditional concurrency-control schemes fail to give adequate support to the long-lived transactions typical of any development process because they either rely on locking mechanisms, which may prevent developers from accessing large portions of the database, or on

optimistic nonlocking mechanisms, which, if a conflict is detected, will accept the changes of one developer only at the expense of another.

Long-transaction environments could support CMT development. Implementing this concurrency-control model would allow multiple developers to operate on the same base configuration, make local changes, and later submit these changes for integration into a new reference version. Using such a concurrency-control scheme would solve some of the challenges facing CMT development, in particular, those of distributed development. Long-transaction environments, however, will not provide a solution to local-update penalties (Tuttle et al., 1991).[5]

The local-update penalty could be eliminated if a set of transactions that can be recombined to create individually-tailored sequences could be developed as a by-product of merging transactions. These sequences could be sent back to individual developers, providing a set of transactions that can be applied to the local configuration and thereby synchronize local terminology with the new reference version. The long-transaction model cannot support creation of these individually tailored sequences, however, because the transactions exist only within the context of a workspace and have no validity outside it. They do not exist as named independent entities that can be extracted and applied to other system configurations.

5. The local-update penalty refers to the extra work required to migrate from one version of a standardized terminology to another version when developers have made local enhancements to that terminology to allow it to properly function within their applications.

Change-Set Model.

The change-set model is a natural extension of the long-transaction model that will support the creation of individually tailored sequences of changes. In the long-transaction model, a sequence of changes (a series of transactions that applied to one version of a system will generate the next version of the system) is captured during the development process. The change-set model allows these transactions to be independent, named entities. As a result, a series of transactions can be combined to create a *change set*, which contains the logical changes that have occurred to a component. These change sets can then be applied to other configurations of the system, allowing for checking for possible conflicts, whenever the same set of logical changes are desired.

Change sets are created from long transactions that represent a configuration's preserved and committed changes. Once the change set has been created, new configurations can be built by adding change sets to other existing configurations. Not all combinations of existing configurations and change sets will result in valid configurations. The validity of any configuration corresponds to the validity of the schedule of transactions applied to the base configuration. Traditional and semantics-based mechanisms for validating a transaction schedule can be employed for this purpose.

The change-set model can also support distributed, concurrent change without centralized coordination. Individual sites can generate change sets independently. These change sets can then be exchanged, allowing individual sites to combine change sets independently. Exchanging change sets in this way allows system evolution to take place at both sites, while also allowing local control over the process. To realize such distributed—and inde-

pendent—development, however, a mechanism must be provided to prevent significant conflicts from occurring or to resolve them when they occur.

2.4 Summary

There is significant prior work that can be applied to problems of clinical data representation. This chapter has presented relevant background material from existing CMTs, knowledge representation, concurrency control, and configuration management. This material was presented together to emphasize an important point. Any viable solution must be a holistic one. A sound knowledge-representation formalism is not adequate to solve representation problems on a national scale unless there are associated methods for managing the development and maintenance of the representation.

The following chapters will expand the background material presented here to provide solutions capable of supporting development and maintenance of a large-scale CMT. Chapter 3 will describe how the CMT combines with information models to represent clinical data. Chapter 5 will discuss extensions I have developed to support distributed CMT development.

Chapter 3
Representation: Terminological Definitions, Information Models, and Patient Data

Clinical data is a complex, intertwined representation of terminological definitions, information models, and patient data. Often the distinctions among these components are lost to casual observers. Even more sophisticated observers and practitioners blur the distinguishing features of information models and terminological definitions, often including terminological definitions in information modeling activity. If the number of terminological definitions is small or limited in sophistication, including these definitions directly in the information model may be expedient and harmless, but as the magnitude and complexity of the definitions increase, their generalizability is compromised and it quickly becomes impractical to include them in information models.

Rector, however, has described a three tiered model to illustrate distinctions among the three components found in representations of clinical data (Rector, 1993) that has been adopted by and expanded for this dissertation. Understanding these fundamental distinctions is essential for distinguishing between CMT development as described in this dissertation and development of information models, an activity that lies outside its scope.

That terminological definitions can be successfully standardized using description logics is a fundamental assertion of the thesis of this dissertation. To test this assertion, Kaiser Permanente CMT project has formalized an existing terminology system, SNOMED International (Côté et al., 1993), by recasting existing relationships with a description logic.

3.1 Terminological Definitions and Information Models

Information models describe the entities and relationships captured within a database.Typically, they are coarse descriptions of data that incorporate information that is necessary to store that data in a database. These descriptions also facilitate the work of analysts seeking to devise ways of retrieving desired data from the database. Depending upon the application, such coarse descriptions may be sufficient. Certainly, numerical data in databases require few or no terminological definitions; the defined rules of algebra and statistics are sufficient to describe and analyze numerical fields.

During analysis, however, it is common to aggregate data based on its non-numerical features. A simple example is linking an average salary computation to the sex or race of employees. For personnel databases, a simple terminological model consisting of 30-50 terms describing race, sex, and job function may well be sufficient and will not tax information modeling tools. A medical database intended to allow description of known diseases and physical findings, however, can easily reach over 100,000 different terms which will exceed the capacities of most information modeling tools. In such cases, the alternative is to have the information tools model certain regular features of clinical data, such as

relationships among patients, diseases, treatments, and complications. Complex defining relationships of the terms themselves need to be managed in a system optimized for management of terminological definitions. Figure 3-1 illustrates such a relationship between a terminology model and an information model.

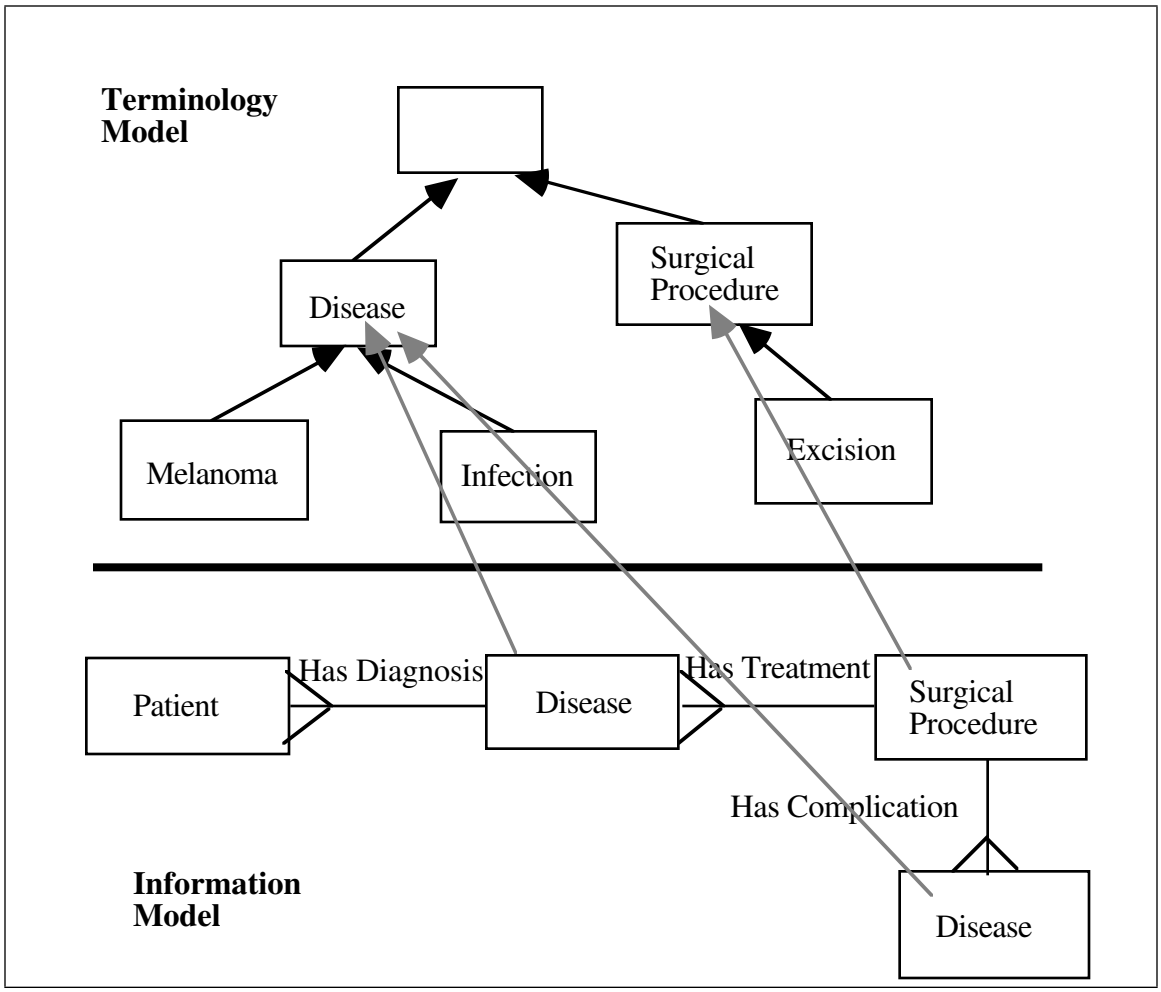


Figure 3-1. Relationships between a terminology model (top) and an information model (bottom). Grey arrows represent "is-instance-of" relationships. Figure from Rector (1993).

Although as this example shows, both the information and the terminology models share high level concepts such as DISEASE and SURGICAL-PROCEDURE, the more specific concepts such as MELANOMA, INFECTION, and EXCISION are only represented within the termi-

nology model. These specific concepts, however, can be use to populate DISEASE and SURGICAL-PROCEDURE items within the information model.

3.2 Terminological Definitions and Patient Data

Patient data contain complex descriptions. The flexibility of subsequent data analysis depends upon inclusion of appropriate details in the terminology model. If patient data are limited to the detail described in the information model, important information will be missing. Figure 3-2 illustrates a specific instance of patient data that is more detailed than the information contained in the information model. Accurate representation and analysis of patient data depends on tight coupling of the terminology and information models.

When the terminology and information models are left uncoupled, the semantics of the two models are more difficult to coordinate when analyzing data. To ask the general question "how many melanoma excisions at this institution resulted in an infectious complication or excessive scar formation?" an analyst would first of all need to review all the different terms that could be used to represent an infection or scar formation in the terminology model, analyze the information model to determine where the desired terms were stored, and only then formulate a query to retrieve the desired information.

Although integrated modeling of terminology and information models is not a focus of this dissertation, continued improvements toward coupling these two tiers of data representation into integrated applications can be anticipated.

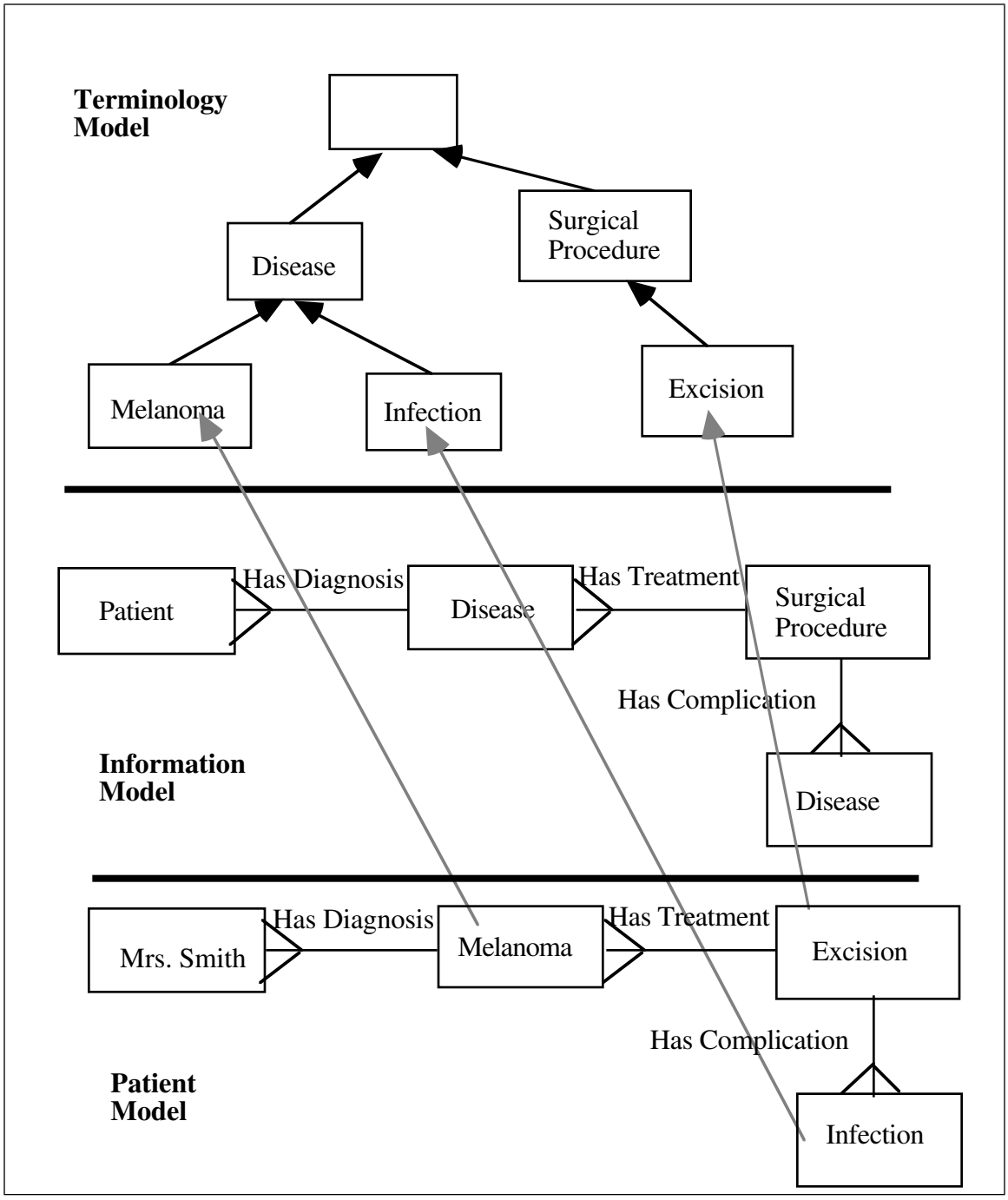


Figure 3-2. Relationships between a terminology model (top) and a patient (data) model (bottom). Grey arrows represent "is-instance-of" relationships. Notice that relationships between the patient model and the terminology model are more specific than those between the information model and the terminology model (compare with Figure 3-1 on page 65). Figure from Rector (1993).

3.3 Representation of Terminological Definitions

The terminology model presented in Figures 3-1 and 3-2 are very simple, requiring only hierarchical relationships. The defining relationships of a robust terminological system, however, are not so simple. Diseases, for example, are often defined with respect to associated morphologic features, known etiologic mechanisms and agents, and affected anatomic locations. Figure 3-3 presents an illustrative expanded set of terminological concepts.[1]

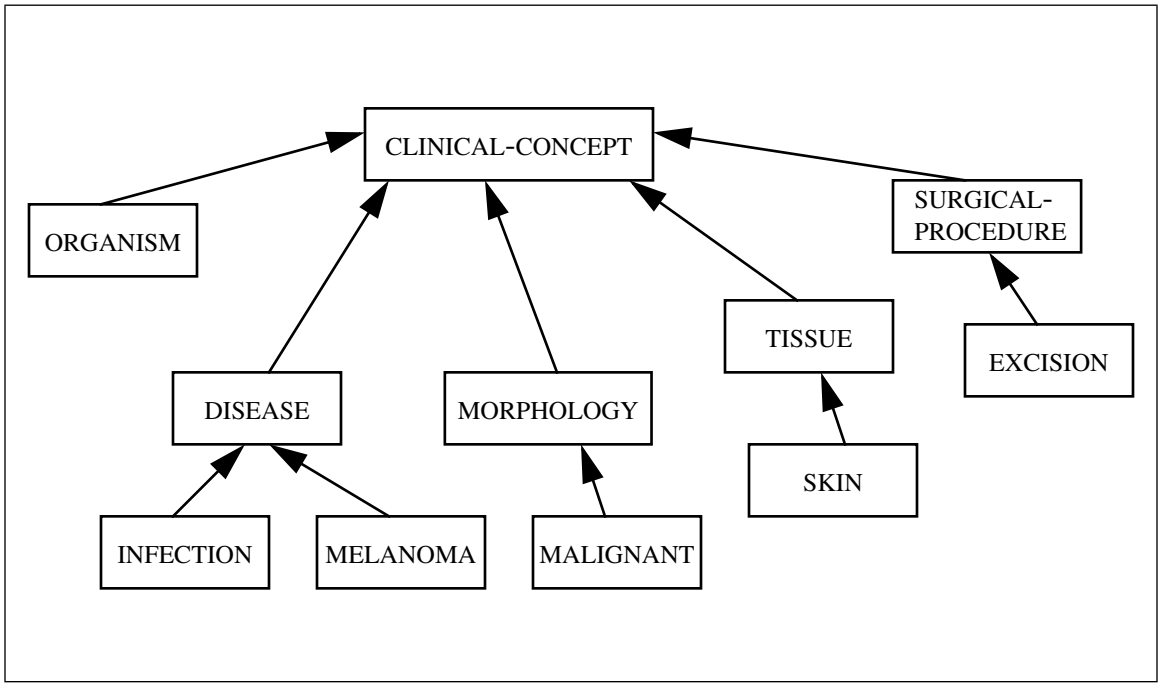


Figure 3-3. An expanded set of terminological concepts.

3.3.1 Concept Definitions

In all concept hierarchies, there must be some starting point from which all other concepts are derived. This is usually called the *top concept*; in Figure 3-3 it is referred to as the

1. I make no claims about the completeness, or correctness of these examples. They are intended to be illustrative only.

CLINICAL-CONCEPT. To create an Aristotelian hierarchy, differentia for every term must be specified. Description logic is used to specify these differentia by defining relations and selection constraints that are appropriate for each term, but that differentiate these terms from their immediate parents. These differentia are contained within concept definitions, statements that incorporate both the genus and differentia of each term. Table 3-1 provides such concept definitions for the expanded set of concepts presented above in Figure 3-3.

Table 3-1. Terms with corresponding definitions

Term	**Definition**
SURGICAL-PROCEDURE	(define-primitive-concept SURGICAL-PROCEDURE)
EXCISION	(define-concept EXCISION (and SURGICAL-PROCEDURE (some REMOVES TISSUE)))
DISEASE	(define-primitive-concept DISEASE)
INFECTION	(define-concept INFECTION (and DISEASE (some CAUSED-BY ORGANISM)))
ORGANISM	(define-primitive-concept ORGANISM)
MORPHOLOGY	(define-primitive-concept MORPHOLOGY)
TISSUE	(define-primitive-concept TISSUE)
SKIN	(define-primitive-concept SKIN TISSUE)
MELANOMA	(define-concept MELANOMA (and DISEASE (some AFFECTS SKIN) (some MORPHOLOGIC-FEATURE MALIGNANT)))
MALIGNANT	(define-primitive-concept MALIGNANT MORPHOLOGY)

The type definition for the term INFECTION found in Table 3-1 is prototypical. The genus of INFECTION is DISEASE. The differentia of an INFECTION is that it is always caused by some type of ORGANISM (bacteria, fungus, or virus). As Table 3-1 shows, the genus and

differentia of INFECTION can be formally defined by specifying the following concept definition:

```
(define-concept INFECTION
        (and DISEASE (some CAUSED-BY ORGANISM)))
```

In some cases, however, a term cannot be adequately defined by relating it to other terms within the terminological system. In these cases, the term is considered *primitive*.[2] Defining a concept as primitive indicates to the system that the concept is different from its parent term in a way that is not expressible within the system. Defining the term SKIN as a type of TISSUE is prototypical of a primitive concept definition. Other examples of such primitive terms include SURGICAL-PROCEDURE and DISEASE, as shown in Table 3-1.

Although medical terminology systems must allow the definition of primitive concepts, they should be avoided whenever possible. Since at least part of a primitive definition is not expressed within the system, the term cannot be classified completely, which limits the extent to which applications can process the terminology.

Often, a particular description logic dialect forces the developer to declare some terms primitive that could otherwise be fully defined using first-order predicate logic. The problem typically is that the particular description-logic dialect does not support the operations necessary to make certain distinctions. Many description logic languages, for instance, do not allow terms to be defined using the "not" operator, in the interest of making classification of the terminology decidable and complete.[3]

2. These concepts are defined in KRSS by using the *define-primitive-concept* operator and then indicating the genus of the concept as with a normal concept definition.

3.4 Derivation of an Information Model

A close coupling of information and terminology models often tempts developers to derive the information model from a terminology. The information model in the lower half of Figure 3-1, for instance, could be derived from the terminology model in the upper half of the figure—provided the terminology model formalized the relationship between patients, diseases, treatments, and complications.

Although tempting, inclusion of such information in the terminology model should be avoided for several reasons. First, the relationship between a disease and its treatment is not definitional.[4] Certainly, "arthritis" cannot be defined as a "disease treated with aspirin." By the same token, melanoma should not be defined as having a treatment of "excision." Second, trying to define all possible relationships in the terminology model comprehensively can lead to representational difficulties. If, for example, the relationships among disease, surgical procedure, and complication found in the information model in Figure 3-1 were formally defined in the terminology model, the result would be a terminological cycle in which term A was defined by reference to B and term B was defined by reference to term A.

---

3. Relatively simple changes in the expressivity of a description-logic dialect can lead to a "computational cliff" for calculating subsumption between terms (Brachman & Levesque, 1984). Inclusion of the "not" operator makes calculating subsumption intractable (the time to compute the subsumption grows exponentially with the number of propositions in the terminology), and is therefore commonly omitted from the supported operations of a particular description-logic dialect.
4. Terminology development focuses exclusively on *defining* relationships. This focus is in contrast to knowledge-base development where interesting—but not necessarily definitional—relationships are often required to provide promised functionality. For a terminology, the functionality is limited to appropriate classification of the terms with respect to one another. A robust, fully-specified terminology may be an ideal foundation for knowledge-base extensions, however.

Although developers should be cautious about using a terminology model to derive an information model, the information model certainly must be consistent with the terminology model, and examination of the terminology model is a necessary part of designing an information model.

3.5 Foundational Models

Deciding which aspects of data modeling should be included in the information model and which should be in the terminology model is a difficult problem for where there are no clear guidelines. This section describes the importance of *foundational models* for resolving this problem. These foundational models represent complex aspects of data such as temporal and anatomical relationships. Some will be embodied in the terminology model, others will be embodied in the information model, and some will be partly embodied in both information and terminology models.

Description logics provide the constructs needed to represent clinical concepts in a foundational model.The simple model presented in Figure 3-3 depends on a small set of defined relations. Existing models contained in standard terminology systems—such as SNOMED—can also be incorporated by using their existing labels and defined relationships to populate a description-logic based terminology system. Although existing systems are a pragmatic starting point for a description-logic based terminology, the set of concepts and relations in existing coding schemes is limited and represents only the beginning of the work needed to develop a comprehensive medical terminology.

Implicit in standard terminologies such as SNOMED are a number of foundational assumptions that taken together constitute a foundational model. SNOMED, for example, includes terms for anatomical concepts, such as *anterior* and *adjacent*; temporal concepts, such as *subacute* and *relapsing*; and measures of probability, such as *possible* and *cannot exclude*. These terms for representing temporal relationships, anatomical relationships, and uncertain relationships reflect underlying models of time, anatomy, and probability that users can apply to almost any concept description created from SNOMED codes. Nonaxial coding schemes, such as ICD-9-CM, also embody such foundational models, but they are reflected only in the distinctions made by the surface-level terms in the scheme. For example, by making a distinction between code 410 (*myocardial infarction, acute*) and code 412 (*myocardial infarction, old*), ICD-9-CM offers a choice of disease codes that reflects its very simple underlying model of time.

SNOMED offers an advance over terminologies such as ICD-9-CM by separating out and making explicit sets of modifiers for anatomical, temporal, and probabilistic concepts. It offers, however, only a list of terms without clarifying the relationships among those terms. With proper support, developers enhance SNOMED by developing reusable models created out of these foundational concepts—models that will allow medical descriptions to be encoded so that the attendant anatomical, temporal, and probabilistic distinctions can be uniformly represented with precise semantics.

SNOMED has a set of concepts and relations that can provide a basis for representing such foundational models. My research seeks to support such development, but has not sought to impose any particular foundational models.

3.6 SNOMED International

SNOMED has a set of concepts and relations that can provide a basis for representing foundational models. This dissertation describes an evolutionary method to formalize SNOMED using a description logic.

SNOMED has properties that are desirable for a comprehensive patient-description terminology. The first is relative domain completeness. SNOMED has over 120,000 terms that represent concepts commonly found in clinical medicine. Because of SNOMED's size, the numerous person-years that have already been invested in its development, and the existing infrastructure and implementations of SNOMED-based systems, enhancing SNOMED makes more sense than starting to develop a new standard from scratch.

The Board of Directors of the American Medical Informatics Association would seem to support this approach, as they have published a position paper in which SNOMED is proposed as a standard to represent diagnoses, symptoms and findings, microbes and etiologies, and anatomic locations (Board of Directors of the American Medical Informatics Association, 1994). SNOMED has also been evaluated for its ability to represent nursing concepts in the patient record, and was found more complete than existing nursing-specific classifications (Henry, Holzemer, Reilly & Campbell, 1994). SNOMED consistently ranks highly when compared to other clinical terminology systems (Campbell et al., 1997; Chute, Cohn, Campbell, Oliver & Campbell, 1996).

Another important property of SNOMED is its *modular structure*. It is divided into a set of independent taxonomies for representing the conceptual categories within medicine.

SNOMED provides 11 modules, each with an independent taxonomy, for representing clinical information:

- **Topography**: terms to describe anatomy
- **Morphology**: terms to describe structural changes
- **Living organisms**: terms to classify the animal kingdom including bacteria and viruses
- **Chemical agents**: terms to describe drugs
- **Function**: terms to describe signs and symptoms
- **Occupation**: terms to describe occupation
- **Diagnosis**: terms to describe diagnosis
- **Procedure**: terms to describe administrative, therapeutic, and diagnostic procedures
- **General**: terms to describe syntactic linkages and qualifiers
- **Physical agents, forces, and activities**: terms to describe devices and activities commonly associated with disease
- **Social context**: terms to describe social conditions and circumstances important to medicine

The structure of SNOMED allows development of independent modules for information that it does not represent directly, allowing for extensions to the SNOMED model without changing its internal structure. In addition, SNOMED provides reserved codes so that institutions can add their own codes within a module.

Despite SNOMED's desirable properties, there are also, as noted in Chapter 2, several problems that limit its usefulness. These include its strict hierarchical coding scheme that

uses implicit, rather than explicit, links between parent and child terms, the lack of a standard syntax to specify explicitly the complex relationships among terms, and the ability to construct redundant statements with no means for determining that they are equivalent.

SNOMED International's strict hierarchical coding scheme is embedded within the alphanumeric code for each term. Trailing zeros are not considered significant. Terms that precede others in the alphanumeric sequence and that have at least one fewer significant digit than these others are considered *ancestors*. The codes for "procedure" (P0-00000) and "radiologic-procedure" (P5-00000) are two examples. "Procedure" is an ancestor of "radiologic-procedure" because it shares the first the digit "P" with "radiologic-procedure," and has one fewer significant digit. There are, however, frequent violations to this general rule of thumb. One such violation is encountered when the number of children (immediate descendents) exceeds the 16 allowed by the hexadecimal term codes.

Because of SNOMED's embedded hierarchy, terms are linked implicitly to one another. These links may be presumed to be IS-A, IS-PART-OF, IS-MADE-OF, or other relations. The only consistent relationship between child and ancestor terms is that a child term is always more specialized than are its ancestors. Moreover, although this coding scheme allows explicit representation of the hierarchy as part of the term code, it does not allow the hierarchical representation of terms with multiple parents (immediate ancestors).

This inability to represent multiple parents within the SNOMED term code has been partially addressed by the crossreferencing of terms (over 34,000) that is provided as part of the SNOMED distribution. Each SNOMED term has an associated crossreference field where one or more related terms have been linked by including the relevant term code in

this field. Figure 3-4 illustrates these crossreferences for the terms “Disease of Respiratory System,” “Disease of Pleura, NOS,” and “Pleural Effusion, NOS.”

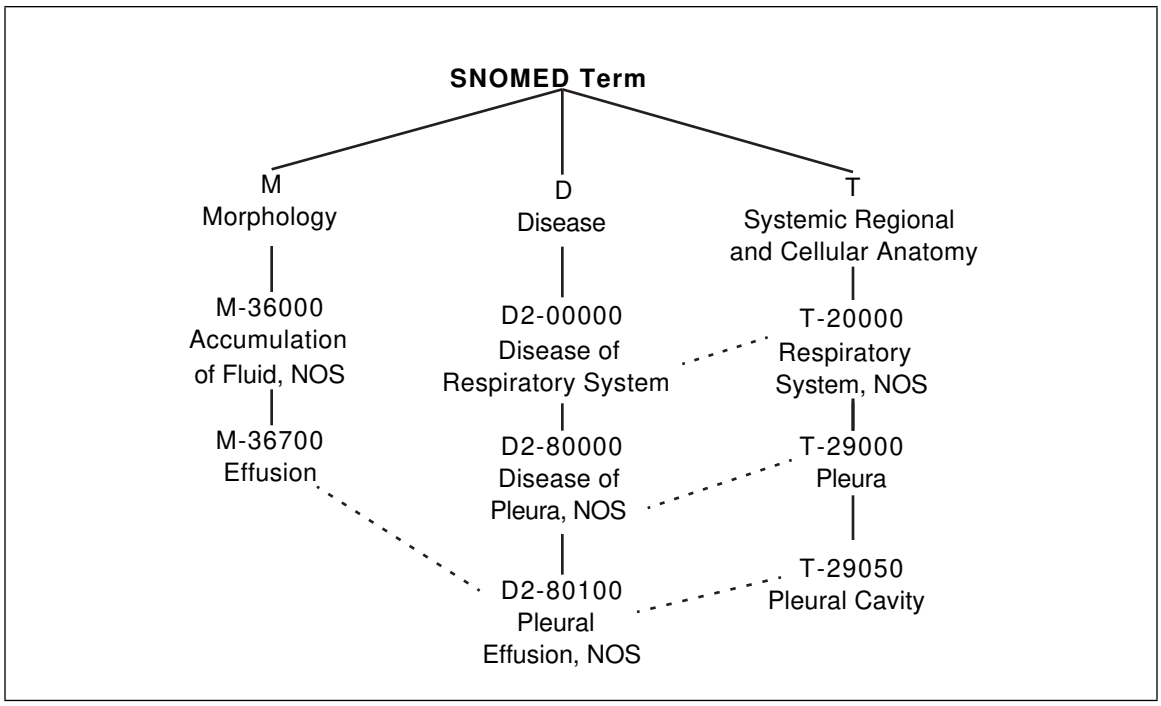


Figure 3-4. SNOMED type hierarchy showing classification of “pleural effusion.” The dashed lines represent cross-reference links provided by SNOMED.

Although this scheme provides a mechanism for linking terms, the types of links are not made explicit. This particular SNOMED shortcoming is not insurmountable, however, since SNOMED’s hierarchy and crossreference links can be transferred easily to other representation schemes that can represent explicitly-typed links. The type of link, for instance, can often be inferred based on the types of terms being crossreferenced. A term from the diagnosis axis, for example, may be crossreferenced to terms in the morphologic axis that represent the morphologic changes associated with a particular disease in the diagnosis axis. In such as case it can be inferred that the linkage between such terms is that

the change represented by the morphology term is CAUSED-BY the disease represented by the disease term.

3.6.1 Migration Path for SNOMED

The first step in recasting SNOMED to fit within a logical framework is to create an Aristotelian type hierarchy from the existing codes, taking advantage of SNOMED's hierarchy and crossreference links. Initially, most of the SNOMED codes will have to be declared as primitive. Once this initial Aristotelian type hierarchy is created, the differentia of each term can be enhanced in an evolutionary way until only a minimum set of terms remain as primitives.

SNOMED's existing crossreference links can be used initially to develop these enhancements, and other links can be inferred as shown above. These crossreference links can be used as a first approximation to create the differentia for each term. A probable relationship between a morphology term and a disease term can be assumed to be that the disease is associated with the morphological change, and a probable relationship between a disease and a location is that the disease is associated with the location. We can use these probable relations, the existing SNOMED crossreferences for the term PLEURAL-EFFUSION, and the hierarchical parent for this term (hierarchical parent and crossreferences are shown in Figure 3-4) to suggest the following definition for pleural effusion:

```
(define-concept PLEURAL-EFFUSION
        (and PLEURAL-DISEASE
             (some ASSOCIATED-MORPHOLOGY EFFUSION)
             (some ASSOCIATED-TOPOGRAPHY PLEURAL-CAVITY)))
```

3.7 Prospects

The foundational models discussed in this chapter will have to evolve over time along with the terminology and information models. Because the underlying terminology model is based in logic, the representation is both general and sufficiently expressive to represent foundational models as they are developed. Since it remains to be seen if such models can be practically implemented on the scale necessary for representing clinical data, it is crucial that actual implementations formally evaluate the practicality of this approach.

Ultimately, the success or failure of a terminology for the representation of clinical data will be based on more than the technical merits of the underlying representation (description logic in this case). Important problems of how to manage the development process, meet the needs of specific applications, coordinate evolutionary enhancements, and develop a political consensus will pose ongoing challenges that must be overcome.

In the remainder of this dissertation, I present an approach to supporting the development process in a distributed fashion, where conflicting changes can be interactively resolved through a consensus process involving the contributing modelers, and demonstrate this approach in a prototype effort to perform concurrent CMT development. Hopefully, through a distributed participatory process, the barriers to political consensus can also be minimized.

Chapter 4
Computer Support for Collaborative Development

CMT development, for the purposes of this dissertation, focuses on a very narrow topic: logical description of terms. CMT development must be distinguished from *knowledge-base* or *expert-system* development, where additional assertional knowledge about a domain is encoded to create a complete application. Here the focus is on the collaborative development of terminological definitions—represented using description logic—that might be included in such an application. Because the focus is narrow, automated support of the development process is a realistic goal.

Given the singular focus on development of terminological definitions, how can computers support this development process? A prerequisite to answering this question is to enumerate the assumptions, actions, and potential conflicts that are part of the process. Section 4.1 describes the assumptions underlying my work. Section 4.2 presents typical enhancements, and conflicts created by those enhancements. Section 4.3 presents a concurrency-control model that will allow multiple developers to work independently, and later to merge their work into a new reference version. An integral part of this concurrency-control

model is a methodology for resolving the inevitable conflicts that are created by independent work.

When developers make local enhancements to a CMT, and a new reference version is released, the developer must expend effort to synchronize the locally enhanced CMT with the new reference version. Developers who expend the most effort creating local enhancements are penalized by this process because they also expend the most effort synchronizing their CMT with the new reference version. Section 4.4 describes how configuration management can minimize this penalty. If local enhancements are used to create new reference versions, custom change-sets can be algorithmically generated to synchronize locally enhanced CMTs.

4.1 CMT Development Assumptions

Gruber (1990) articulates a five level model for sharing knowledge-based technology that may help illustrate what is included in collaborative CMT development, and what is excluded. The five levels he articulates are: Level 1—sharing syntax, Level 2—sharing vocabulary, Level 3—sharing ontology, Level 4—sharing inference methods, and Level 5—sharing heuristic knowledge bases. The collaborative CMT development described in this dissertation includes Levels 1, 2, and, with a caveat, Level 3.

Implications of sharing a syntax (Level 1) are obvious. Sharing a common vocabulary (Level 2), has implications not immediately obvious. For Gruber, these implications include a method for managing the *namespace*. In this dissertation, the method for manag-

ing the namespace is based on identifying conflicts in the Aristotelian concept hierarchy described later in this chapter. For Level 3—sharing ontology, Gruber states that explicit and computationally-enforced semantics are required. Explicit, computationally-enforced semantics are consistent with this dissertation. However, "ontology" is not.

Ontology is intentionally avoided in this dissertation, as it is an unfamiliar word to many, and often is used to imply a more complete accounting of concepts in the world of interest than is intended here. For example, Gruber (1993) cites Enderton (1972) as stating that ontologies are not limited to definitions that only introduce terminology; rather, ontological definitions can add additional knowledge about the world. This dissertation *is* limited exclusively to definitions that introduce terminology; therefore, I do *not* assert that my work is aimed at allowing concurrent development of ontologies.

Given this focus on definitions that introduce terminology, I will now summarize the assumptions made regarding those definitions and the terminology as a whole.

Assumption 1. The CMT consists of terms (T_1... T_m) and defining relationships (R_1... R_n).

Assumption 2. Each term T_i in the CMT represents a unique concept.

Assumption 3. Each term T_i in the CMT has a unique, inviolable, identifier.

Assumption 4. Each term T_i in the CMT has one and only one terminological definition D_i expressed as a description-logic statement.

An example definition for the term CHEST-PAIN can be expressed using the KRSS syntax as:

(define-concept CHEST-PAIN (and PAIN (some LOCATED-IN CHEST)))

This definition can also be expressed in predicate calculus notation as:

$$\text{CHEST-PAIN}(x) \equiv \text{PAIN}(x) \wedge \text{CHEST}(y) \ \wedge \text{LOCATED-IN}(x,y)$$

The methodologies I have developed are all syntax independent.

Assumption 5. Every terminological definition D_i is logically unique.

Assumption 6. Terms may not be deleted; only their definitions may be changed.[1]

Assumption 7. Each relation R_i in the CMT represents a unique binary relationship that is part of a terminological definition. Relationships cannot exist independent from terminological definitions.

An Aristotelian hierarchy contains terms defined by genus (the category of classification for a term) and differentia (element(s), feature(s), or factor(s) that distinguish one term from another). This dissertation requires classification of terminological definitions into an Aristotelian concept hierarchy for identification and classification of concurrent development conflicts. An Aristotelian hierarchy can be formally defined as follows:

1. Although most terminology systems require periodic addition of new concepts and often contain terms that may become obsolete, it is important to permanently maintain the representation of all concepts in a way that remains faithful to the intent of the obsolete terms. Otherwise, evaluation of longitudinal data will be compromised, since terms within the data will be unavailable within the CMT.

Definition 1. A concept hierarchy H is said to be *Aristotelian* if every term T_i is a proper subtype of at least one other term.

Classification of a large terminological system to determine if it is Aristotelian is beyond the cognitive abilities of most individuals. Automated support for classification is assumed.

Assumption 8. An algorithmic classifier will use terminological definitions to create an Aristotelian concept hierarchy, or to determine which definitions violate Aristotelian assumptions.

Given this definition of an Aristotelian concept hierarchy, and assuming a classifier that can take terminological definitions and classify a terminology system, it can be algorithmically determined if every term is classified in one and only one place in the concept hierarchy, and if every location in the concept hierarchy has one and only one term. During distributed development, modelers may inadvertently violate these properties of an Aristotelian concept hierarchy.

Definition 2. If, in the process of development, two modelers modify two different terms so that the terms would be classified in the same place in the concept hierarchy, then the assumptions of the Aristotelian concept hierarchy have been violated. I define this violation as a *non-unique definition conflict*.

Definition 3. If, in the process of development, two modelers modify the same term so that the modified versions would be classified in two differ-

> ent places in the concept hierarchy, then the assumptions of the Aristotelian concept hierarchy have been violated. I call this violation a *multiply-defined term conflict*.

I will now show how computational detection of these two conflicts, the non-unique definition conflict (Section 4.2.1), and the multiply-defined term conflict (Section 4.2.2), can be used to support a semantics-based concurrency-control scheme.

There are no contemporary CMT systems founded upon these assumptions. First, certain systems violate Assumption 5, and allow several terms with the same definition. Usually, such systems consider different terms with equivalent definitions as synonyms. An example would be defining both CHEST-PAIN and ANGINA as "pain located in the chest." For my work, uniqueness of definition is required to allow detection of conflicts that might be created during concurrent development.

Other CMT systems violate Assumption 4, by allowing more than one definition per term. Typically such additional definitions are used when a term may have several sufficient defining criteria. An example would be defining a triangle as a "planar geometric figure with three connected sides" or as a "planar geometric figure with three angles that add up to 180°." My work requires a singular defining form to allow detection of conflicts that might be created during concurrent development.[2]

4.2 CMT Development Examples

This section describes fictional transactions created by independent CMT modelers. These examples produce conflicts. Traditional concurrency control schemes (described in detail within Section 2.2) would force acceptance of either one transaction or the other. These example transactions assume that the modelers were operating on the same CMT base configuration, and that neither had knowledge of the changes being made by the other. Such examples are typical of real-world enhancements to a CMT.

Section 4.3 proposes methods to resolve the conflicts created by examples, such as those in this section, without forcing one modeler's version to be accepted at the expense of the other's. These methods are later illustrated using the same examples.

4.2.1 Nonunique-definition Conflict

The first example illustrates a conflict in which two non-synonymous terms are given identical definitions by two different modelers. Modelers A and B begin with the following primitive definitions of the terms INFECTIOUS-PNEUMONIA and PULMONARY-DISEASE:

(define-primitive-concept INFECTIOUS-PNEUMONIA DISEASE)

(define-primitive-concept PULMONARY-DISEASE DISEASE)

---

2. If the classification system can support multiple "sufficient" conditions (the two example "triangle" definitions could also be viewed as "sufficient" conditions rather than as distinct definitions) as part of a term's definition, then this requirement is of no concern (the system has provided a mechanism for supporting multiple sufficient conditions without requiring more than one definition for the term). If the system can support only one sufficient condition as part of a term's definition (as is the case with the prototype tools presented in Chapter 5), then the importance of this limitation will vary depending upon the domain. For medical terminology, my experience has shown that there are a few concepts where having multiple defining criteria would be convenient, but this limitation is not a serious impediment to developing a useful CMT.

The above two starting definitions are *primitive* definitions for the terms INFECTIOUS-PNEUMONIA and PULMONARY-DISEASE. The "define-primitive-concept" statement in the definitions provides a method to define that a term is different from its parent in a way not described (i.e., that INFECTIOUS-PNEUMONIA is a DISEASE, but that it has unique properties that are not defined within the system). Starting with these primitive definitions, each modeler modifies one of the definitions as follows:[3]

Modeler A: (define-concept infectious-pneumonia
(and disease (some located-in lungs)))

Modeler B: (define-concept PULMONARY-DISEASE
(and DISEASE (some LOCATED-IN LUNGS)))

Note that the modelers also remove the "primitive" distinction from each of the definitions.

Both changes are correct in principal; it is true that INFECTIOUS-PNEUMONIA is a "disease located in the lungs." It is also true that PULMONARY-DISEASE is a "disease located in the lungs." However, the definitions of INFECTIOUS-PNEUMONIA and PULMONARY-DISEASE are in conflict, because they are two different terms with the same definition. Figure 4-1 illustrates this conflict. Note that the conflict is *general*. Such conflicts reflect properties of the Aristotelian concept hierarchy, and are not specific to any particular description-logic syntax. This conflict can be caused by an error or an omission in one of the definitions, or the terms may actually be synonyms. In either case, some action needs to be taken to resolve the conflict, ensuring a consistent CMT. If a CMT is evolutionarily enhanced, there will be

3. The restrictions imposed on the concept definitions are a function of the underlying environment. In this chapter, only simple examples are used for clarity. For limitations on the concept definitions imposed by the prototype environment, see Chapter 5.

many ways to create similar conflicts. These conflicts are termed nonunique-definition conflicts.

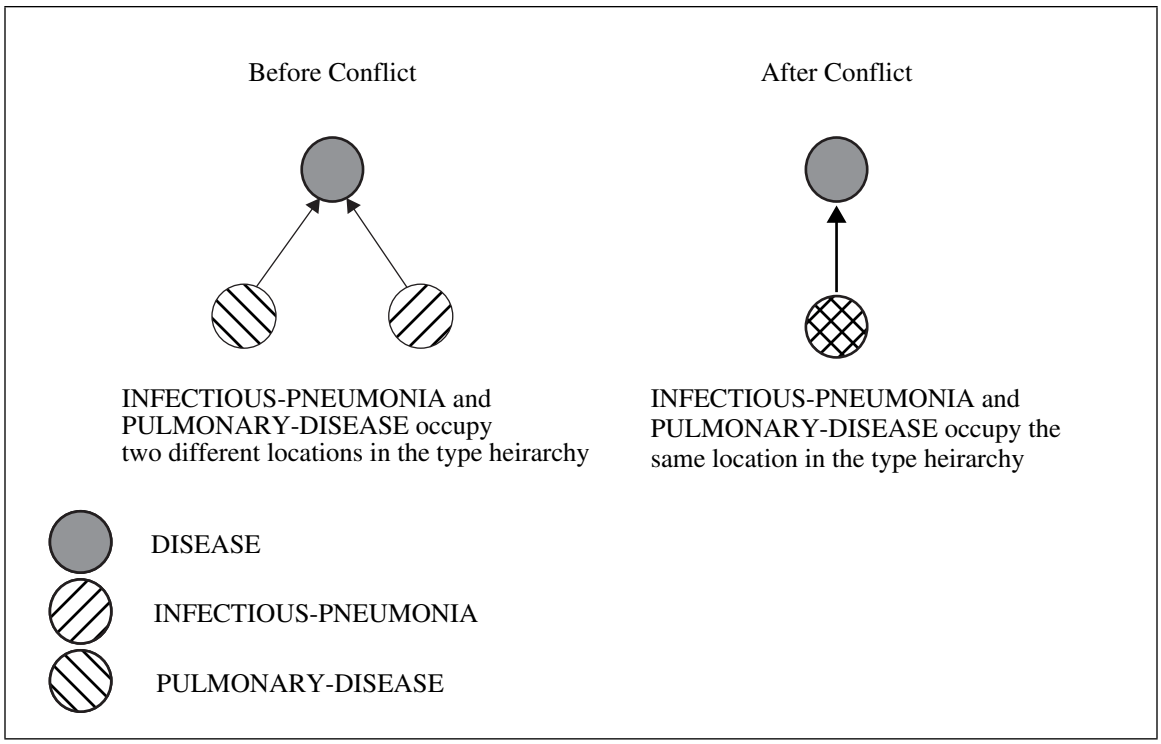


Figure 4-1. Graphical illustration of a nonunique-definition conflict.

If two different terms have the same definition, data retrieval using those definitions is unreliable. For example, a researcher might query a database wanting to know "how many patients with INFECTIOUS-PNEUMONIA were treated as outpatients in the last 12 months?" If the database contained the conflict created by modelers A and B, the query would *over-count* the number of cases of INFECTIOUS-PNEUMONIA by including all PULMONARY-DISEASE (ASTHMA, BRONCHITIS, and others). A development model that provides concurrency control must resolve such nonunique-definition conflicts.

A simple solution is to accept one transaction and reject the other. This solution is not ideal because it forces the acceptance of one modeler's action at the expense of the other

modeler. A system that provides concurrency control should have a strategy that will allow conflict resolution without forcing the acceptance of one modeler's work at the expense of another modeler. The concurrency-control model proposed in Section 4.3 provides such a method.

4.2.2 Multiple-Definition Conflict

The second example occurs when two modelers, C and D, give the same term different definitions. Both modelers begin with the following primitive definition of INFECTIOUS-PNEUMONIA:

(define-primitive-concept INFECTIOUS-PNEUMONIA DISEASE)

Each modeler modifies the definition as follows:

Modeler C: (define-concept INFECTIOUS-PNEUMONIA
(and DISEASE (some LOCATED-IN LUNGS)))

Modeler D: (define-concept INFECTIOUS-PNEUMONIA
(and DISEASE (some CAUSED-BY INFECTIOUS-AGENT)))

Note that the modelers also remove the "primitive" distinction from each of the definitions.

As in the previous example, both changes are correct in principle; it is true that INFECTIOUS-PNEUMONIA is a "disease located in the lungs." It is also true that INFECTIOUS-PNEUMONIA is a "disease caused by an infectious agent." However, the definitions of INFECTIOUS-PNEUMONIA are in conflict, because although they refer to the same term, they do not have the same definition. Figure 4-2 illustrates this conflict. Note that the conflict is

once again *general*, reflecting properties of the Aristotelian concept hierarchy. Such conflicts are termed *multiple-definition conflicts.*

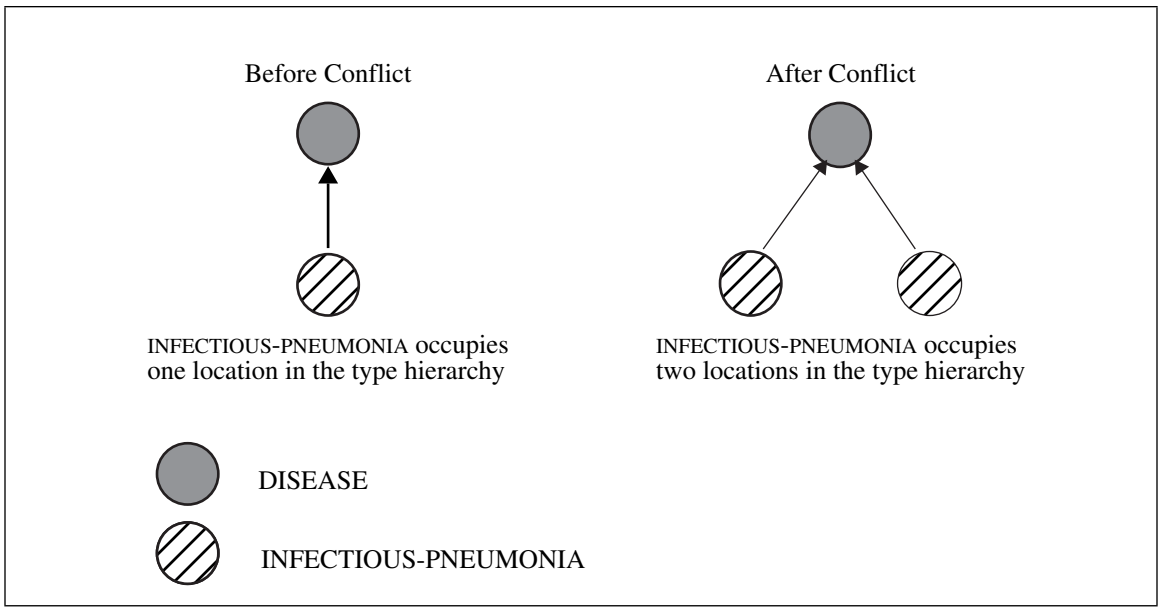


Figure 4-2. Graphical illustration of a multiple-definition conflict.

If one term has two different definitions, data retrieval using those definitions is unreliable. For example, if a researcher again queried the database wanting to know "how many patients with INFECTIOUS-PNEUMONIA were treated as outpatients in the last 12 months?", the query would *undercount* the number of cases of INFECTIOUS-PNEUMONIA (because the query will only use one definition, thus missing the cases of INFECTIOUS-PNEUMONIA encoded using the other definition). A model that provides concurrency control must resolve such multiple-definition conflicts. All terms with equivalent meaning must have identical definitions. If the terms do not have identical definitions, at least one definition is incomplete or incorrect.

A model that provides concurrency control should have a strategy that will resolve multiple-definition conflicts without forcing the acceptance of one modeler's action at the expense of the other modeler. The concurrency-control model proposed in Section 4.3 provides such a method.

4.3 CMT Concurrency Control

Efficient CMT development requires many individuals and groups to work in parallel. A common feature of parallel development schemes is the ability for each site to work concurrently with other sites, and with no real-time control that prevents the sites from creating conflicting transactions. In all cases, conflicting transactions collide long *after* their creation. Optimally, the overall system can resolve conflicts and salvage the semantics of conflicting transactions. This section describes such a system of concurrency control.

4.3.1 CMT Transactions

The first step in controlling a set of actions that may occur at the same time is to define the set of all possible actions. Once these actions are *unambiguously* defined, they can be recorded as transactions. A generic database shell would specify these actions using no application-specific semantics. The generic actions would simply be to *read* a data value and to *write* a data value. Such actions with generic semantics can only be managed with the traditional concurrency control schemes (described in Section 2.2.2). To create a more powerful concurrency control scheme, application-specific actions must be used instead. By using actions that have an understood meaning in the application's domain, a seman-

tics-based concurrency control scheme can be developed specifically for the application (see Section 2.2.3).

Since a CMT is represented by terms, relationships, and definitions,[4] CMT transactions can be represented by changing the definition of a term (in the case of introducing a new term, changing the definition from a null definition to an actual definition) and adding new binary relationships. These two transaction types can represent all the changes that occur in the CMT. Since terminological definitions and relationships have understood meaning in the application's domain, a semantics-based concurrency control scheme can be developed.

4.3.2 CMT Transaction Validity

To ensure that any set of transactions can be validly applied, a method is needed for verifying that the initial conditions required for the desired transactions still exist. One method for verifying these initial conditions is to group the transactions into sets, and to include a copy of the initial definition at the beginning of the set. Before executing any set of transactions, the application can verify that valid initial conditions still exist before allowing the transactions to execute. Figure 4-3 shows a syntax for transaction sets that includes an initial definition, followed by ending definition.

Using the initial definition contained within each set of transactions, the application can verify that a term's initial conditions (a term's starting definition) is the same as the initial conditions for which the transactions were created. If the definitions are *not* the same, the

4. See assumptions in Section 1.4.3

INFECTIOUS-PNEUMONIA

Start State: P DISEASE

End State: D (and DISEASE (some LOCATED-IN LUNGS))

Figure 4-3. A transaction set equivalent to the changes made by Modeler A in Section 4.2.1. The first line contains the identifier of the term being changed. The second line represents the starting state of for which this transaction is valid. The "P" immediately before the starting definition indicates that the definition is primitive. The third line contains the ending definition. The "D" immediately before the ending definition indicates that the definition is defined. This set of transactions forms an entry in a *change set*. A change set consists of one or more entries.

transactions cannot be directly applied. If the definitions *are* the same, they have met initial conditions required to be properly applied; however, this simple test does not guarantee that executing the transactions will result in a semantically consistent CMT. Other types of conflicts may be created (described in Section 4.2).

This set of transactions is a *change set*. In this case, a change set represents the changes made by one modeler, on one definition of the CMT. The transactions of the change set must be executed together: all transactions or no transactions. Given appropriate semantic consistency constraints, this change set can be interleaved with other transactions (or change sets) without requiring a specific serial schedule of transactions for validity.

4.3.3 Terminology Change Sets

Sagas (Garcia-Molina & Salem, 1987) provide an appropriate paradigm for CMT concurrency control.[5] Garcia-Molina's work provides the inspiration for the change sets I

5. See Section 2.2.3 for a more complete presentation of this work

describe in this dissertation. Local enhancements to a CMT can be captured as sagas. These sagas could then be sent to a group responsible for integrating the local enhancements to create a new reference version of the CMT. However, in some circumstances two sagas may conflict. Such conflicts prevent sagas from being shared. The original proposal for sagas provided no method to facilitate the sharing of conflicting sagas. The primary purpose of sagas was to execute long-lived transactions on a central database, without locking large portions of the database. There are important differences between sagas described by Garcia-Molina and Salem and the use of change sets for CMT development described in this dissertation:

- Change sets are not executed only on a central database. They are initially created on an autonomous local database, and are then *secondarily* sent to a central database for integration and execution.
- The execution policies of this central database are significantly different from the policies of a traditional database application. The central database will strive to execute change sets, submitted by developers, in a way that optimizes the databases semantic accuracy and completeness. This policy is very different from traditional database's in which execution of transactions is determined strictly by the order in which the transactions are submitted.
- Finally, the database's semantic accuracy and completeness are determined interactively, with human oversight (as opposed to algorithmically with no human oversight). The proper sequence for resolving conflicts is determined by the human operators, with

computer assistance, rather than being automated, as is typical of concurrency control schemes. Because of these differing policies, new conflict-resolution methods are possible.

Because of the previously described differences between sagas and change sets, I use the term "change sets" when referring to sets of transactions (a term consistent with the configuration-management literature).

Change sets, like sagas, can conflict with one another. When a conflict is identified, some method must be available to resolve the conflict. As discussed previously, a method that can combine the work of the conflicting transaction is preferable to one that requires acceptance of one set of transactions at the expense of another. The following sections describe two conflict resolution methods that can combine the work of the conflicting transactions: The semantically-equivalent change set (SES) and the semantic-addition change set (SAS). These methods are both introduced abstractly; immediately following the abstract introduction, the conflict examples introduced in Section 4.2 are used to illustrate these methods (Section 4.3.4).

Semantically-Equivalent Change Set

The SES is a change set that can be substituted for another change set to resolve a conflict. The SES is equivalent to the original change set, but valid for a different base configuration of the CMT. Consider change set S_3 and change set S_4, both originating from the same base configuration, but created by two different developers at two different sites.

Optimally these change sets could be merged by executing the following sequence of change sets:

S_3, S_4.

However, execution of S_3 alters the base configuration, and this might cause S_3 and S_4 to be in conflict (similarly execution of S_4 first would also alter the base configuration and might similarly cause a conflict if S_3 were then attempted). An equivalent set of transactions, SES_4, could be executed in place of S_4, resulting in the sequence:

S_3, SES_4.

Such a SES would be created by a developer working with a configuration management environment. The environment would present any conflicts identified during import of S_3 and S_4 to the developer. The developer would examine the nature of the conflict and determine what semantic attributes of the two terms should be retained in the merged configuration. The environment would then create SES_4, based on the developer's input, and execute that transition in place of S_4. This alternative execution sequence allows the semantics in change set S_4 to be utilized, even though it conflicts with S_3. SESs can be used to resolve multiple-definition conflicts. Section 4.3.4 provides examples of SESs.

Semantic-Addition Change Set

The SAS is a set of transactions that can be added to a sequence of change sets that are found to conflict. Such a change set must be created when the conflict cannot be resolved by a SES. Addition of the SAS resolves the conflict. Consider change set S_1 and change

set S_2, both originating from the same base configuration. Optimally these change sets could be merged by executing the following sequence of change sets on the base configuration:

S_1, S_2.

However, if S_1 and S_2 are found to be in conflict (again S_1 having altered the base configuration so that S_2 can no longer be applied), a new change set, SAS_1, could be executed after S_1, resulting in the sequence

S_1, SAS_1, S_2.

SAS_1 adds new semantic information to resolve the conflict. As with the SES in the previous section, the SAS is created by the configuration management environment interacting with the developer.

Sometimes it would be better to add the SAS after S_2. If an SAS is added after S_2, however, it will not prevent the semantic conflict, so SAS_2 must be executed before S_2, and S_2 must be replaced with an equivalent change set, SES_2, resulting in the following sequence:

S_1, SAS_2, SES_2.

Section 4.3.4 provides examples of SASs.

4.3.4 Terminology-Specific Conflict Resolution Strategies

The previous section introduced change sets and described how to create change sets to resolve conflicts (using SESs and SASs). This section demonstrates how the SESs and

SASs are applied using the conflict examples from Section 4.2. The reader will recall that two classes of conflicts were created: multiple-definition conflicts and non-unique-definition conflicts. The multiple-definition conflict resolution strategy is discussed first, followed by the non-unique-definition conflict-resolution strategy.

Multiple-Definition Conflict-Resolution Strategy

If a term in the CMT has two different definitions, at least one of the definitions is incomplete or incorrect. Assuming that one definition is incomplete, a strategy for resolving the conflict is to merge the two definitions into one. This merge can be done automatically and the results presented for human verification. This strategy will also present errant definitions, giving the modeler an opportunity to reject them as well.

If the semantic changes of a conflicting change set are captured in a SES that does not conflict,[6] the semantics of the change set can be salvaged. The result of executing the SES is equivalent to merging the definition changes from multiple change sets into a single definition.

The reader will recall from Section 4.2.2 that two modelers (C and D), created conflicting definitions for the same term, INFECTIOUS-PNEUMONIA. The transactions encoding those changes can be captured in change set S_3 for modeler C, and change set S_4 for modeler D. The start and end definitions of these change sets are presented in Table 4-1. The semantic changes in change sets S_3 and S_4 can be put into SESs that are valid for different starting

6. The SES would be created by a developer interacting with a configuration management environment. The developer would indicate the semantic attributes to be retained to resolve the conflict, and the configuration management environment would generate a corresponding SES.

conditions. The basis for creating such change sets is discussed in the remainder of this section. Table 4-1 illustrates two such SESs: *SES3* and *SES4*.

Table 4-1. Change sets *S3* and *S4* with compensating change sets. Valid sequences are Base + *S3*, Base + *S4*, Base + *S3* + *SES4*, Base + *S4* + *SES3*.

Set	Term Label	State	Definition
S3	INFECTIOUS-PNEUMONIA	Start	(define-primitive-concept INFECTIOUS-PNEUMONIA DISEASE)
		End	(define-concept INFECTIOUS-PNEUMONIA (and DISEASE (SOME LOCATED-IN LUNGS)))
S4	INFECTIOUS-PNEUMONIA	Start	(define-primitive-concept INFECTIOUS-PNEUMONIA DISEASE)
		End	(define-concept INFECTIOUS-PNEUMONIA (and DISEASE (some CAUSED-BY INFECTIOUS-AGENT)))
SES4	INFECTIOUS-PNEUMONIA	Start	(define-concept INFECTIOUS-PNEUMONIA (and DISEASE (some LOCATED-IN LUNGS)))
		End	(define-concept INFECTIOUS-PNEUMONIA (and DISEASE (some LOCATED-IN LUNGS) (some CAUSED-BY INFECTIOUS-AGENT)))
SES3	INFECTIOUS-PNEUMONIA	Start	(define-concept INFECTIOUS-PNEUMONIA (and DISEASE (some CAUSED-BY INFECTIOUS-AGENT)))
		End	(define-concept INFECTIOUS-PNEUMONIA (and DISEASE (some LOCATED-IN LUNGS) (some CAUSED-BY INFECTIOUS-AGENT)))

If an application tries to execute the change sets in the order Base + *S3* + *S4*, a conflict will be created because *S4* can only be validly applied to infectious-pneumonia when the starting definition is (define-primitive-concept INFECTIOUS-PNEUMONIA DISEASE). Since change set *S3* has already changed the definition of INFECTIOUS-PNEUMONIA, change set *S4* can not be directly applied. A change set semantically equivalent to *S4* can be created by capturing the semantics of the change set, that INFECTIOUS-PNEUMONIA is "caused by an infectious agent." Change set *SES4* is an example of such a change set that can be validly applied after execution of change set *S3*. Using this change set, an application can

execute the sequence Base + S_3 + SES_4. The result is the creation of a new definition for INFECTIOUS-PNEUMONIA that merges the changes incorporated in change sets S_3 and S_4.

A SES can be created in two ways. First, it is always possible for a human modeler to use personal knowledge and deductive ability to custom craft a SES. Second, it is possible to derive *probable* SESs algorithmically. A tool can examine conflicting change sets and use greatest common subgraph algorithms to determine overlap between the end definitions of the change sets. Any additional or missing nodes or arcs in one definition that are not in the other represents changes unique to that change set. Those changes can then be applied to the ending definition of the other change set. The result is a new change set, SES_j that can be applied after S_i to resolve the conflict. The appropriateness of SES_j is not guaranteed, however. Human oversight, through an on-line editing and review environment, is required to ensure appropriateness of the conflict resolution strategy.

An application may execute the change sets in many different orders, such as Base + S_4 + S_3. A different order of execution may create different conflicts. In this case, a different conflict will be created because S_3 can only be validly applied to INFECTIONS-PNEUMONIA when the starting definitions is (define-primitive-concept INFECTIOUS-PNEUMONIA DISEASE) (as was the case with change set S_4). Another change set, SES_3, can be created (similarly to how SES_4 was created) to execute in place of S_3. Using this change set, an application can execute the sequence Base + S_4 + SES_3. As before, the result is the creation of a new definition for INFECTIOUS-PNEUMONIA that merges the changes incorporated in change sets S_3 and S_4.

Nonunique-Definition Conflict Resolution Strategy

Terms in a CMT may have the same definition because either (1) the terms may actually be synonyms or (2) at least one of the definitions is faulty, either incomplete or incorrect.

If the terms are actually synonyms, their definitions can be unified by adding one term to the synonym list of the other. The definitions no longer conflict because they will become a single definition that will refer to a preferred term and a synonym list. The conflict can be detected algorithmically, but determining that two terms are synonyms will require human evaluation. Additionally, when two synonyms are found, one should be designated the "preferred term"; this will also require human judgment.

If a term's definition is incomplete, resolving the conflict will require modification of the definition. Turning the definition into a primitive is the simplest solution. Such a change is captured in a SAS. This modification will ensure that the two definitions do not conflict, by asserting that a term is different from another term, in a way not described (see Section 3.3.1). This solution can be applied arbitrarily to one of the conflicting definitions, or, using human intervention, can be applied selectively to the most appropriate definition.

Another solution for resolving incomplete definitions is to add additional semantic knowledge to one or both definitions until they are unique. This solution may seem more attractive, since it tries to make the definitions more complete; however, it adds an additional burden to the initial goal of resolving concurrency conflicts: creating logically complete and correct term definitions.

Formal solutions that arbitrarily make one of the terms primitive can be created without human intervention. Formal solutions that selectively make one of the terms primitive or that allow new semantic knowledge to be added to the definition have to be manual. In either case, the methodology is formal, however some are automatic and some are manual. The change sets created during such a conflict-resolution process will be SASs (Section 4.3.3).

The example nonunique definition conflict, described in Section 4.2.1, may be resolved using a SAS. The reader will recall that two modelers, A and B, created identical definitions for two different terms: INFECTIOUS-PNEUMONIA and PULMONARY-DISEASE. The transactions responsible for those changes can be captured in two change sets, S_1 for modeler A and S_2 for modeler B. The starting and ending definitions of these change sets are presented in Table 4-2. The conflicting definitions can be resolved in two ways. The first way utilizes a SAS, SAS_1. The second way utilizes a SES, SES_1. The basis for creating these change sets is discussed in the remainder of this section. These change sets are also presented in Table 4-2.

Table 4-2. Change sets S_1 and S_2 with compensating Change sets. Allowable sequences are Base + S_1; Base + S_2; Base + S_1 + SAS_1 + S_2; and Base + S_2 + SES_1.

Set	Term Label	State	Definition
S_1	INFECTIOUS-PNEUMONIA	Start	(define-primitive-concept INFECTIOUS-PNEUMONIA DISEASE)
		End	(define-concept INFECTIOUS-PNEUMONIA (and DISEASE (some LOCATED-IN LUNGS))))
S_2	PULMONARY-DISEASE	Start	(define-primitive-concept PULMONARY-DISEASE DISEASE)
		End	(define-concept PULMONARY-DISEASE (and DISEASE (some LOCATED-IN LUNGS)))
SAS_1	INFECTIOUS-PNEUMONIA	Start	(define-concept INFECTIOUS-PNEUMONIA (and DISEASE (some LOCATED-IN LUNGS)))
		End	(define-primitive-concept INFECTIOUS-PNEUMONIA (and DISEASE (some LOCATED-IN LUNGS)))
SES_1	INFECTIOUS-PNEUMONIA	Start	(define-primitive-concept INFECTIOUS-PNEUMONIA DISEASE)
		End	(define-primitive-concept INFECTIOUS-PNEUMONIA (and DISEASE (some LOCATED-IN LUNGS)))

If the application tries to execute the change sets in the following order, Base + S_1 + S_2, a conflict will be created because the definition for PULMONARY-DISEASE that results from the actions of S_2 conflicts with the existing definition for INFECTIOUS-PNEUMONIA created by the actions of S_1. One way to resolve this conflict is to create an SAS that will make the existing definition of INFECTIOUS-PNEUMONIA primitive. Using such a change set, labeled SAS_1 in Table 4-2, the following sequence can be executed: Base + S_1 + SAS_1 + S_2. This sequence ensures that the definition of INFECTIOUS-PNEUMONIA will be changed before transaction S_2 is executed, thus resolving the conflict with the definition for PULMONARY-DISEASE.

An application may sometimes execute the change sets in a different order, such as Base + S_2 + S_1. A conflict will also be created when trying to apply S_1 to the already committed actions of Base + S_2. SAS_1 cannot be used to resolve this conflict. SAS_1 was created to

operate after the completion of S_1, and cannot be applied in this circumstance. Rather than creating a change set that adds semantics to already committed transactions, as did SAS_1, a SES can be used in its place. S_1 was trying to add information to the existing definition of INFECTIOUS-PNEUMONIA. This information, that INFECTIOUS-PNEUMONIA is "located in the lungs," can be captured in a SES, labeled SES_1 in Table 4-2. This change set can be executed in place of S_1, with equivalent results. Using this change set, the application can execute the change sets in the order Base + S_2 + SES_1 without creating a conflict.

Although this non-determinism may generate different paths to achieve the same result, the specific path is inconsequential. A configuration management environment will process the change sets in the order they arrive, and when exporting custom change sets for external applications, it will export a set of changes that will be able to update any of the terminology configurations know to the configuration system to any subsequent configuration of the CMT.

4.3.5 Configuration Management Challenges

Section 4.3.3 and Section 4.3.4 have described how SESs and SASs can resolve conflicts created when integrating conflicting change sets. These change sets provide a conflict-resolution solution; however, their existence also introduces new management problems. Each change was initially represented by a single change set. After alternative SESs and SASs are added, each change may be represented by the original change set, the original transaction in combination with a SAS, or a variety of SESs. The additional change sets were developed to apply the changes contained within the original change set to a different

configuration. Keeping track of which change sets can validly be applied to which configurations is a complex task. Methods to manage this complexity are presented in Section 4.4.

4.4 CMT Configuration Management

The development of any complex system requires techniques to coordinate and control its construction. This coordination and control process is called configuration management. Many of the principles of configuration management have been developed for hardware engineering, large building construction, and software systems (Whitgift, 1991). Configuration management is defined as: "the process of identifying and defining the items in the system, controlling the change of these items throughout their lifecycle, recording and reporting the status of items and change requests, and verifying the completeness and correctness of items" (ANSI/IEEE Standard 729, 1983).

Section 4.3 described a concurrency-control method for managing developmental changes that occur to terms in a controlled medical terminology (CMT). Such concurrency-control methods are an important part of configuration management, but are not sufficient to manage all configuration management problems. In addition to concurrency-control methods, CMT configuration management will require naming conventions to identify items and versions of items, methods to verify the completeness and correctness of CMT configurations, and methods to create CMT configurations.

Simple methods of configuration verification, such as comparing a configuration to a reference version, are sufficient for CMT management.[7] Methods to create custom configurations (including naming conventions and configuration generation methods) are the most complex, and are the focus of this section.

4.4.1 Custom Configurations

Creation of custom configurations can be very tedious. Automated tools have been created for generation of custom configurations in other domains, such as Digital's XCON configuration system (Barker & O'Connor, 1989). A formal methodology for creating custom configurations of CMTs will allow the process of creating custom configurations to be automated.

Section 4.3 presented a method, using change sets, for integrating concurrent transactions from multiple sites into a single master configuration. This concurrency-control scheme allows CMT development to be distributed; however, it also creates an additional burden for developers who participate in this process. Although the concurrency-control scheme allows their work to be integrated into the master version, the scheme does not synchronize their local copies with new versions of the master CMT. This failure to automatically synchronize local copies creates a paradoxical situation in which developers who are the most productive (create the most changes) are rewarded with the largest penalty when they

7. Here, configuration verification simply refers to a method of determining that the configuration is equivalent to the reference version in the configuration management system. For such a determination, a binary comparison is sufficient. Configuration verification is distinct from quality-assurance activities where the terminology within a configuration is subject to *semantic* verification rather than to simply binary verification. Assuring the semantic correctness of a CMT configuration is of course more complex, and not the subject of this section.

later manually synchronize their local CMT with the derivative master CMT. This penalty is the "local-update penalty."

Tuttle and colleagues (1991) previously described this paradoxical penalty as it relates to local enhancements of the Unified Medical Language System (Lindberg, Humphreys & McCray, 1993). They have appealed for development of technological solutions and standardized update methods to minimize this penalty. The remainder of Section 4.4 presents such a solution.

4.4.2 Version Naming

To identify each of the revisions, a standard component-naming convention must be used. Figure 4-4 illustrates the convention for naming CMT versions. Each identifier has up to three parts. The first part identifies the reference version from which the component was derived. If a reference version has been altered, creating a branch, it will have a branch identifier and a local revision number. Each field is separated by a period. Reference versions and local revisions are identified by sequential integers. Branch identifiers are not sequential, and are represented alphabetically. This naming convention allows branches to be identified by institution. Figure 4-4 refers to the third local revision of a CMT created by site "A" using reference version 1 as its base CMT.

The individual change sets that generate local revisions, and subsequently are incorporated into new reference version, need a different naming convention. Change sets cannot be modified once they are created; revision identifiers would therefore be meaningless. A change set may also be incorporated into several different versions: the local version, the

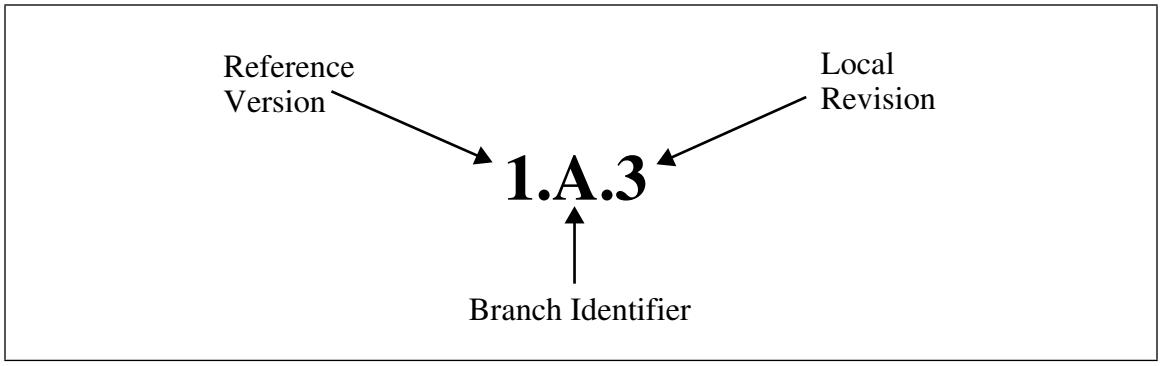


Figure 4-4. CMT version naming convention. Each version identifier can have three parts: the reference version (left), the branch identifier (center), and the local revision number (right).

new reference version, and then, through subsequent distribution, local versions at different sites. Reference version and branch identifiers for change sets are thus meaningless. The only requirement for change set names is that they be unique. Including additional information in the change set names, such as its generation time or its author, may be convenient, but is not required. In this dissertation I identify change sets by their type (S, SES, or SAS), and ensure uniqueness with subscripts (S_i, SES_j, or SAS_k).

Once all of the components are identified, they can be collected, and the process of merging the components to create a new reference CMT can begin. After the merge process is completed, custom configurations can be generated using components that were part of the merge process. Section 4.4.3 describes the merge process. Section 4.4.5 describe the generation of custom configurations.

4.4.3 Version Merging

After local changes are submitted to a central source, they must be merged. During the merge, all local enhancements are combined to create a single configuration. This merge relies on concurrency-control techniques, described in Section 4.3, to identify and resolve

any conflicting transactions created at different sites. After all conflicts have been resolved, a new version has been created that contains all the enhancements created at the local sites.

The reader will recall that four modelers, A, B, C, and D, made changes to the same reference CMT. All modelers began with the following definitions:

(define-primitive-concept INFECTIOUS-PNEUMONIA DISEASE)

(define-primitive-concept PULMONARY-DISEASE DISEASE)

Each modeler then modified one of the definitions as follows:

Modeler A: (define-concept INFECTIOUS-PNEUMONIA
(and DISEASE (some LOCATED-IN LUNGS)))

Modeler B: (define-concept PULMONARY-DISEASE
(and DISEASE (some LOCATED-IN LUNGS)))

Modeler C: (define-concept INFECTIOUS-PNEUMONIA
(and DISEASE (some LOCATED-IN LUNGS)))

Modeler D: (define-concept INFECTIOUS-PNEUMONIA
(and DISEASE (some CAUSED-BY INFECTIOUS-AGENT)))

These changes are captured in four change sets, S_1, S_2, S_3, and S_4, which represent the changes made by modelers A, B, C, and D. These change sets are presented in Table 4-3. Section 4.3 described how these change sets might conflict, and how to create alternative change sets that can be executed instead of the original change set to resolve such conflicts. Two such change sets (originally created in Section 4.3.4), SES_3 and SES_4, are also presented in Table 4-3.

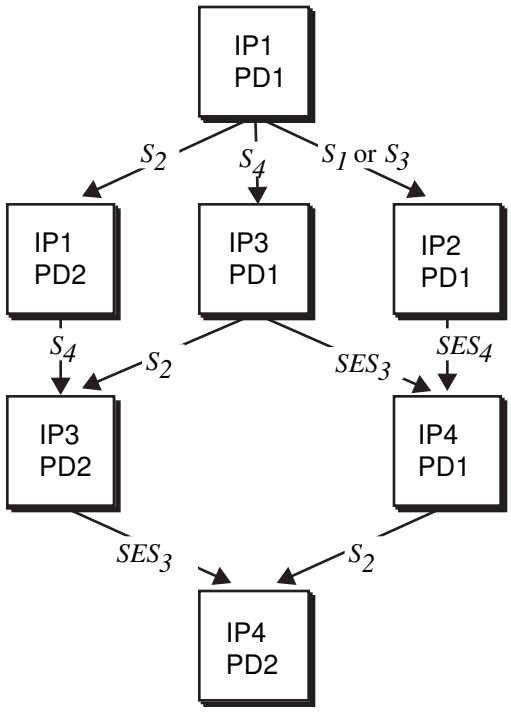
Table 4-3. Change sets to be combined to generate a new CMT reference version. These change sets can be recombined to create custom change sets to synchronize local versions of the CMT with the new reference version (Section 4.4.4).

Set	Term Label	State	Definition
S_1	INFECTIOUS-PNEUMONIA	Start	(define-primitive-concept INFECTIOUS-PNEUMONIA DISEASE)
		End	(define-concept INFECTIOUS-PNEUMONIA (and DISEASE (some LOCATED-IN LUNGS)))
S_2	PULMONARY-DISEASE	Start	(define-primitive-concept PULMONARY-DISEASE DISEASE)
		End	(define-concept PULMONARY-DISEASE (and DISEASE (some LOCATED-IN LUNGS)))
S_3	INFECTIOUS-PNEUMONIA	Start	(define-primitive-concept INFECTIOUS-PNEUMONIA DISEASE)
		End	(define-concept INFECTIOUS-PNEUMONIA (and DISEASE (some LOCATED-IN LUNGS)))
SES_3	INFECTIOUS-PNEUMONIA	Start	(define-concept INFECTIOUS-PNEUMONIA (and DISEASE (some CAUSED-BY INFECTIOUS-AGENT)))
		End	(define-concept INFECTIOUS-PNEUMONIA (and DISEASE (some LOCATED-IN LUNGS) (some CAUSED-BY INFECTIOUS-AGENT)))
S_4	INFECTIOUS-PNEUMONIA	Start	(define-primitive-concept INFECTIOUS-PNEUMONIA DISEASE)
		End	(define-concept INFECTIOUS-PNEUMONIA (and DISEASE (some LOCATED-IN LUNGS)))
SES_4	INFECTIOUS-PNEUMONIA	Start	(define-concept INFECTIOUS-PNEUMONIA (and DISEASE (some LOCATED-IN LUNGS)))
		End	(define-concept INFECTIOUS-PNEUMONIA (and DISEASE (some LOCATED-IN LUNGS) (some CAUSED-BY INFECTIOUS-AGENT)))

The change sets in Table 4-3 can be combined in many different orders, resulting in different CMT states. Figure 4-5 illustrates seven possible states, starting with the initial configuration at the top, and finishing with the final state at the bottom. The bottom state represents a complete merge of all local changes, and will become the new reference version.

4.4.4 Minimization of Local-Update Penalties

The state diagram in Figure 4-5 provides the basis for minimizing local-update penalties, as a by-product of integrating locally developed change sets into a new reference version.



INFECTIOUS-PNEUMONIA States:

IP1: (define-primitive-concept INFECTIOUS-PNEUMONIA DISEASE)

IP2: (define-primitive-concept INFECTIOUS-PNEUMONIA
(and DISEASE (some LOCATED-IN LUNGS)))

IP3: (define-primitive-concept INFECTIOUS-PNEUMONIA
(and DISEASE (some CAUSED-BY INFECTIOUS-AGENT)))

IP4: (define-primitive-concept INFECTIOUS-PNEUMONIA
(and DISEASE (some CAUSED-BY INFECTIOUS-AGENT)
(some LOCATED-IN LUNGS)))

PULMONARY-DISEASE States:

PD1: (define-primitive-concept PULMONARY-DISEASE DISEASE)

PD2: (define-primitive-concept PULMONARY-DISEASE
(and DISEASE (some LOCATED-IN LUNGS)))

Figure 4-5. Possible CMT states. Each box represents a potential CMT state. The initial and final states of sites A, B, C, and D are guaranteed to be on the diagram because the states were created by integrating each of the sites together. Numbers inside each box represent the state of the terms INFECTIOUS-PNEUMONIA and PULMONARY-DISEASE (shown at bottom of figure). Each state was created by executing one of the change sets presented in Section 4.3 (also in Table 4-3). The change set necessary to traverse from one state to another is identified on each arrow.

If all locally developed change sets are merged into a new reference version, and a state diagram is created that represents all the states of the merge process, a node will exist in the state diagram corresponding to the state of each contributing site. From this node, at least one path to the new reference state will exist. By traversing any of these paths, and recording the sequence of change sets along the particular path, a change set sequence is created. If this sequence is executed at the local site, it will synchronize the local CMT with the new reference version.

A configuration management environment that supports CMT development will support this sequence generation process. The change-set configuration management model, described in Section 2.3, provides the appropriate paradigm. The change sets described here represent the long transactions of the change-set configuration management model; a custom *sequence of change sets* created by traversing the state diagram of the reference version is equivalent to a change set of the change-set configuration management model.

The primary difference between what is described here, and what is typically viewed as the change-set configuration management model, is that the CMT is much narrower in scope than are most software systems to which configuration management has typically been applied. Because of this narrow focus, more automation is possible. The ability to resolve conflicts semi-automatically, by algorithmically suggesting alternative change sets as described in Section 4.3.4, is one example of such automation. To fully support CMT development, applications that utilize the change-set configuration management model must incorporate the conflict-resolution strategies developed for integrating change sets. Chapter 5 describes a prototype change-set configuration-management environment. The

next section utilizes the examples developed in this chapter to illustrate further the process of creating change sets capable of synchronizing a locally enhanced CMT with a new reference version.

4.4.5 Custom Configuration Examples

Section 4.3 presented specific examples to illustrate the types of conflicts that may occur when multiple modelers work on the same CMT. This section uses those same examples to demonstrate how these change sets can be recombined into *custom* change sets. These change sets will be custom tailored for each site, and will synchronize the local CMT with the new reference CMT.

The reader will recall from Section 4.4.3 that the terms INFECTIOUS-PNEUMONIA and PULMONARY-DISEASE were modified by four modelers. The locally created change sets were later merged into a new reference version, version 2. Table 4-4 presents these two terms, and their respective definitions in versions 1 and 2. The individual change sets that were used to create version 2 from version 1 were presented previously in Table 4-3.

Table 4-4. Version 1 and version 2 definitions of INFECTIOUS-PNEUMONIA and PULMONARY-DISEASE.

Term Label	Version	Definition
INFECTIOUS-PNEUMONIA	1	(define-primitive-concept INFECTIOUS-PNEUMONIA DISEASE)
	2	(define-concept INFECTIOUS-PNEUMONIA (and DISEASE (some LOCATED-IN LUNGS) (some CAUSED-BY INFECTIOUS-AGENT))
PULMONARY-DISEASE	1	(define-primitive-concept PULMONARY-DISEASE DISEASE)
	2	(define-concept PULMONARY-DISEASE (and DISEASE (some LOCATED-IN LUNGS)))

The change sets in Table 4-3 can be applied to version 1 in several different valid sequences. These sequences are shown in Figure 4-5, along with each resulting intermedi-

ate state. The result of creating this diagram is that every possible state of intermediate CMTs is known (since the diagram was created using all of the change sets), and there is a sequence of change sets that can take the CMT from an intermediate state to the new reference version. The following sections describe how such a state diagram can be used to create custom change sets to synchronize local CMTs with the new reference version.

Sites A and C

Sites A and C coincidentally modified the same term, INFECTIOUS-PNEUMONIA, in the same way, and therefore can utilize the same change set to synchronize their local versions with version 2. Table 4-5 shows the local definitions of site A and C (listed as version 1.A.1), and the definitions in version 2. Version 1.A.1 corresponds to the state IP_2 PD_1 in Figure 4-6. Examining the figure will show that one path exists to traverse from state IP_2 PD_1 to state IP_4 PD_2. This path requires executing two change sets, SES_4 and S_2. After executing these two change sets, the local CMT will be synchronized with the new reference version. As an alternative to looking at the state diagram, change sets can be chosen by simply examining the version 1.A.1 and version 2 definitions, and then finding a set of change sets from Table 4-3 able to update each term to the version 2 definition.

Careful inspection of Table 4-3 reveals one sequence of change sets able to synchronize version 1.A.1 with version 2 that is not listed in Table 4-5: S_2 + SES_4. The sequence S_2 + SES_4 is not on the state diagram because it results in an invalid intermediate state. However, if all the change sets are applied together, the invalid state is only temporary, resulting in a valid final configuration. This additional sequence is of no consequence, since it is

important to find *a* synchronization path, not *all* paths. As a by-product of the merge process, the state diagram will always have at least one path for synchronizing a locally-enhanced CMT, provided the change sets created by the local enhancement were part of the merge.

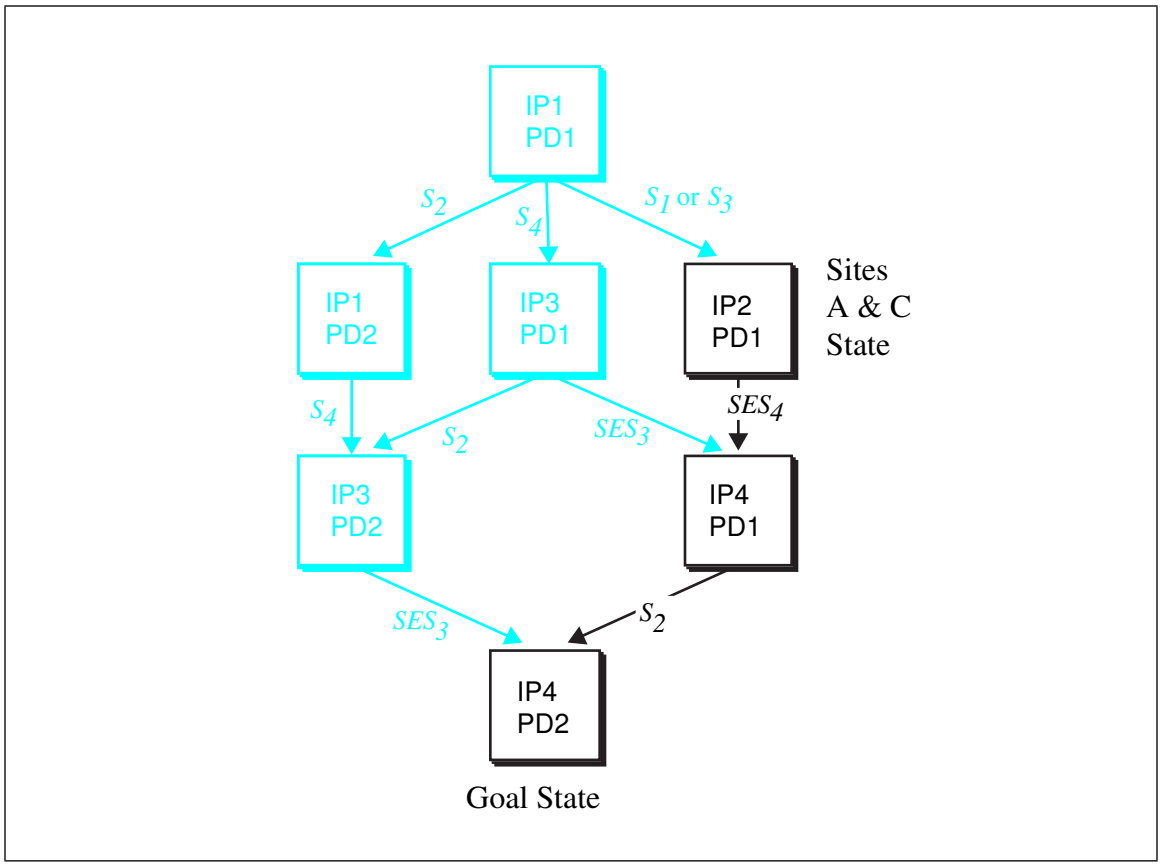


Figure 4-6. Update path for sites A and C.

Table 4-5. Version 1.A.1 definitions, version 2 definitions, and the change sets required to synchronize version 1.A.1 with version 2.

Term Label	Version	Definition	Set
INFECTIOUS-PNEUMONIA	1.A.1	(define-concept INFECTIOUS-PNEUMONIA (AND DISEASE (SOME LOCATED-IN LUNGS)))	*SES4*
	2	(define-concept INFECTIOUS-PNEUMONIA (and DISEASE (some LOCATED-IN LUNGS) (some CAUSED-BY INFECTIOUS-AGENT)))	
PULMONARY-DISEASE	1.A.1	(define-primitive-concept PULMONARY-DISEASE DISEASE)	*S2*
	2	(define-primitive-concept PULMONARY-DISEASE (and DISEASE (some LOCATED-IN LUNGS)))	

Site B

Site B modified the term, PULMONARY-DISEASE, as shown in Table 4-6. This table shows site B's local version (version 1.B.1), and the new reference version (version 2). Version 1.B.1 corresponds to the state IP*1* PD*2* in Figure 4-7. Examining the Figure will show that one path exists to traverse from state IP*1* PD*2* to state IP*4* PD*2*. This path requires executing two change sets, *S4* and *SES3*. After executing these two change sets, the local CMT will be synchronized with the new reference version. As an alternative to looking at the state diagram, change sets can be chosen by simply examining the version 1.B.1 and version 2 definitions. Finding a change set or set of change sets able to update each term to the version 2 definition from the table will have the same result as finding the change sets on a complete state diagram.

Careful inspection of Table 4-3 will again reveal one sequence of change sets able to synchronize version 1.B.1 with version 2 not listed in Table 4-6: *S1* + *SES4*. The sequence *S1* + *SES4* is not on the state diagram because it results in an invalid intermediate state. As described in the previous section, this additional sequence is of no consequence.

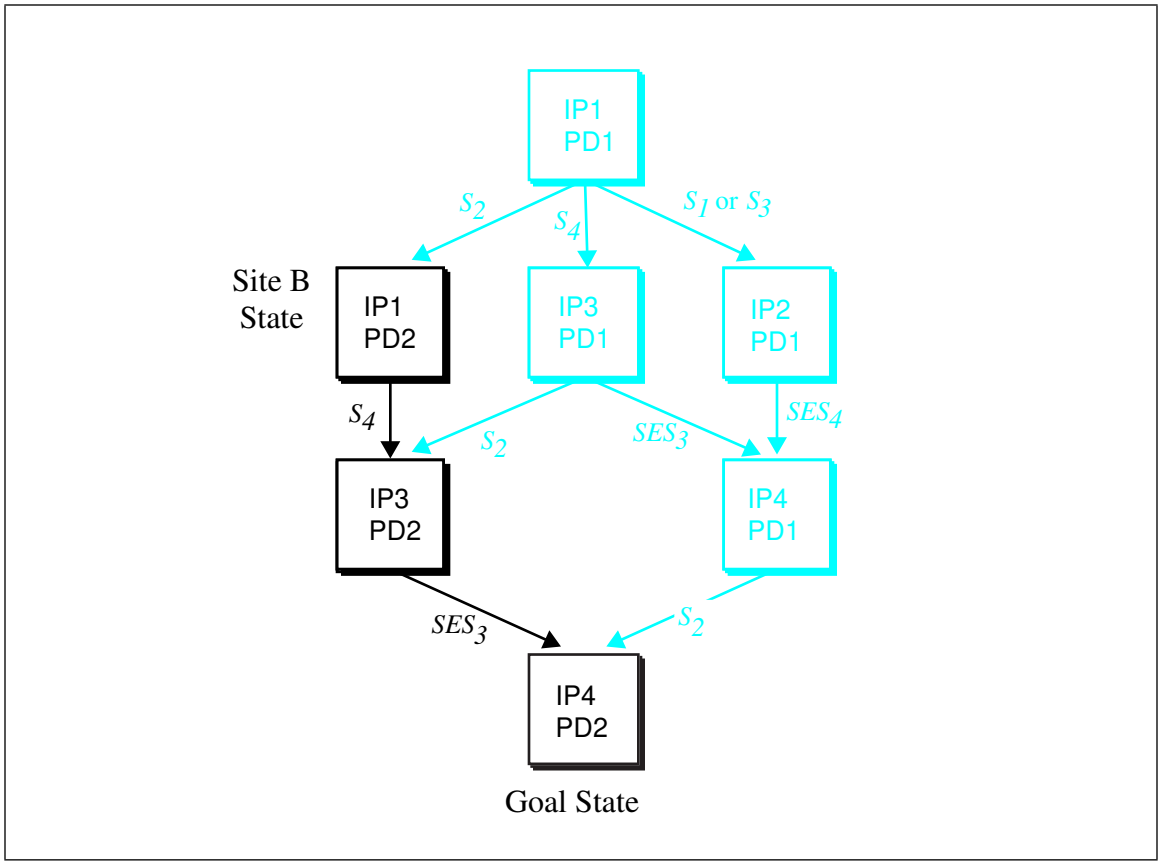


Figure 4-7. Update path for site B.

Table 4-6. Version 1.B.1 definitions, version 2 definitions, and the change sets required to synchronize version 1.B.1 with version 2.

Term Label	Version	Definition	Set
INFECTIOUS-PNEUMONIA	1.B.1	(define-primitive-concept INFECTIOUS-PNEUMONIA DISEASE)	$S4 + SES_3$
	2	(define-concept INFECTIOUS-PNEUMONIA (and DISEASE (some LOCATED-IN LUNGS) (some CAUSED-BY INFECTIOUS-AGENT)))	
PULMONARY-DISEASE	1.B.1	(define-concept PULMONARY-DISEASE (and DISEASE (some LOCATED-IN LUNGS)))	∅
	2	(define-concept PULMONARY-DISEASE (and DISEASE (some LOCATED-IN LUNGS)))	

Site D

Site D modified the term, INFECTIOUS-PNEUMONIA, as shown in Table 4-7. This table shows the intermediate stage, listed as version 1.D.1, and the definition for the terms in

version 2. Version 1.D.1 corresponds to the state IP$_3$ PD$_1$ in Figure 4-8. Examining the figure will show that two path exists to traverse from state IP$_3$ PD$_1$ to state IP$_4$ PD$_2$. Either path requires execution of the same two change sets, S_2 and SES$_3$. After executing these two change sets, the local CMT will be synchronized with the new reference version. As an alternative to looking at the state diagram, change sets can be chosen by simply examining the version 1.D.1 definitions and the version 2 definitions. Unlike the previous two examples, there are no sequences other than those on the state diagram in Figure 4-8.

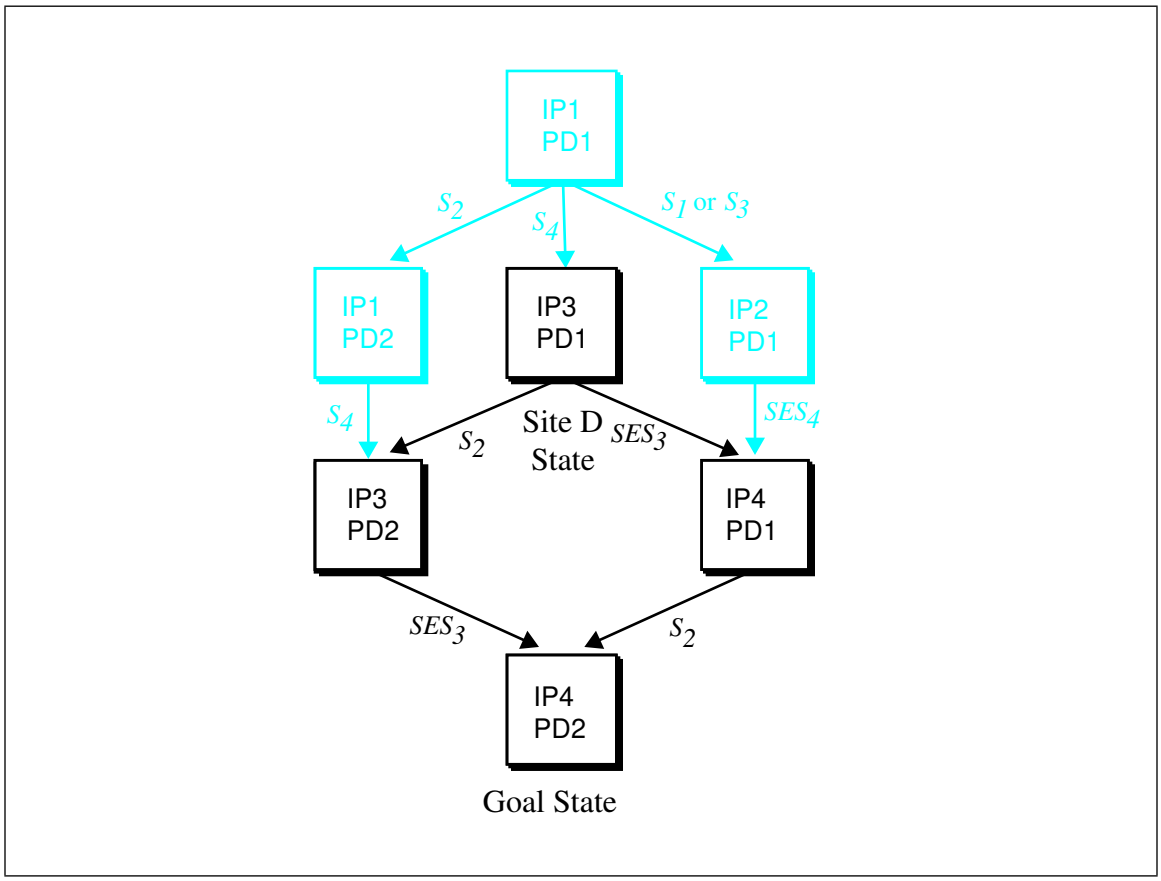


Figure 4-8. Update path for site D.

Table 4-7. Version 1.D.1 definitions, version 2 definitions, and the change sets required to synchronize version 1.D.1 with version 2.

Term Label	Version	Definition	Set
INFECTIOUS-PNEUMONIA	1.D.1	(define-concept INFECTIOUS-PNEUMONIA (and DISEASE (some CAUSED-BY INFECTIOUS-AGENT)))	SES_3
	2	(define-concept INFECTIOUS-PNEUMONIA (and DISEASE (some LOCATED-IN LUNGS) (some CAUSED-BY INFECTIOUS-AGENT)))	
PULMONARY-DISEASE	1.D.1	(define-concept DISEASE (and DISEASE (some LOCATED-IN LUNGS)))	S_2
	2	(define-concept PULMONARY-DISEASE (and DISEASE (some LOCATED-IN LUNGS)))	

4.5 Limitations of Conflict Detection

This chapter defined two classes of conflicts that a classification engine can detect: the nonunique-definition conflict, and the multiply-defined term conflict. Conflicts that are easy to detect are often easy to resolve, but they do not encompass the universe of all errors that might occur during terminology development.

Contemporary terminology management systems identify conflicting terms lexically. That is they look for terms that are lexically equivalent to one another. If lexically equivalent terms represent different concepts, a conflict is created. An example of such a conflict was identified when processing SNOMED for inclusion in the Unified Medical Language System (Lindberg et al., 1993). SNOMED had two terms described as “Mole, NOS.” One term was intended to refer to a specific kind of growth commonly found on the skin, the other term was intended to refer to a living organism that burrows under the ground.

This dissertation expands a terminology management system's ability to identify conflicts beyond these lexical techniques by adding the notion of conflict in an Aristotelian concept hierarchy.

Although using Aristotelian classification expands the repertoire of conflicts that can be used in a concurrency-control scheme, there are still conflicts that fall outside the ability of algorithms to detect. The algorithms described here depend upon the formal properties of the Aristotelian concept hierarchy. It is possible to construct such a concept hierarchy so that it is *internally* consistent, yet it may not reflect the intended coupling with the outside world.

A classic example of such undetectable conflict would the "morning star" and the "evening star." If the "morning star" is defined as the "last star visible in the morning" and the "evening star" is defined as the "first star visible at night," the conflict-detection algorithms described here would not be able to determine that both the "morning star" and the "evening star" are actually the planet Venus if "morning star" and "evening star" are both given unique identifiers (rather than both being called Venus).

Medical examples of such conflicts can be generated by looking at any complex disease with multiple names. One example would be *non-bacterial verrucous endocarditis* also named *Libman-Sacks disease* (Robbins, Cotran & Kumar, 1984). Either term could be defined as "mitral and tricuspid valvulitis due to disseminated lupus erythematous" or alternatively defined as "mitral and tricuspid valvulitis due to autoantibody immune complex activity on the mitral and tricuspid valves." Notice that here the primary difference between the definitions is that one definition defines the disease in terms of another disease (disseminated lupus erythematous) and another definition defines the disease in terms of the disease *process* (autoantibody immune complex activity on the mitral and triscuspid valves).

Detection of conflicts with differing definition and differing concept identifiers are outside the scope of this dissertation. However, I emphasize that human review must be an ongoing part of vocabulary development, thus providing a method by which such terminology defects *may* be identified. The reliability of such identifications will rely on the computational tools and resources allocated for human review and on the training of the individuals involved.

Agreements among concurrent modelers regarding guiding principles, such as "description of the disease process is preferred over simply referring to another disease," may help minimize differing definitions. Enforcement of such agreements will require manual review of the terminology, in addition to review of conflicting terminological definitions. Section 3.5 discussed how implicit as well as explicit foundational models underly any representation scheme. Any change made to the CMT must be consistent with the underlying foundational models, and errors secondary to inconsistent use of such models may be much harder to detect.

4.6 Summary and Discussion

This chapter has presented methodological solutions for many problems that may be encountered during CMT development. The ideas presented here certainly do not solve the universe of all development problems, however. The development process is a social one, and the technological issues described here must fit within such social processes. As noted, the conflicts not resolved by the technological solutions must also be resolved within those social processes.

In this chapter I have described compensating methods that can resolve conflicts by creating new change sets that are semantically equivalent to the original change sets (the SES), or change sets that contain new semantic knowledge (the SAS). These compensation methods present a general notion of how transaction conflicts might be resolved; however, the actual implementation of these methods requires domain knowledge to be supplied during the conflict-resolution process by human oversight. Such transactions may not be appropriate for all domains, or they may be difficult to implement. Fortunately, the semantics of many of the changes that occur to a CMT are simple, and methods can be developed to manage the potential conflicts. Section 4.3.4 described methods for two classes of terminology conflicts, the nonunique-definition conflict (defined by Definition 2 and introduced in 5.1.1), and the multiply-defined term conflict (defined by Definition 3 and introduced in 5.1.2).

As was shown in Section 4.2, some conflicts that are created during CMT development can be easily detected. Conflicts that are easy to detect are often easy to resolve, but they do not encompass the universe of all errors that might occur during CMT development. Section 4.5 discussed potential terminology defects that are beyond the scope of the algorithms of this dissertation.

At the lowest level, these models are built on a system of description logic. At higher levels, additional foundational models may develop new logics, such as Shoham's interval-based logic (Shoham, 1987). When changes are made to the underlying representation, there is a risk that these changes may be inconsistent with the relevant foundational models of the system. The better these foundational models are understood and explicitly

defined, the more easily consistency checks can be developed to see if any changes violate foundational assumptions.

The foundational model used to define the conflict-resolution strategies in this chapter is that of the Aristotelian concept hierarchy. The transactions described in Section 4.2 created conflicts by violating either term-uniqueness constraints (each term in an Aristotelian concept hierarchy, and in the CMT, is therefore unique) or term-consistency constraints (a term only appears once in an Aristotelian concept hierarchy, and therefore must only have one definition). In one case, two different terms were given the same definition; in the other, the same term was given two different definitions. Section 4.3.4 presented strategies for resolving conflicts created by violating the assumptions of the Aristotelian concept hierarchy. As the CMT evolves, and as other foundational models within the CMT solidify, similar methods for detecting—and resolving—conflicts specific to other foundational models can be developed.

Evaluation of the approaches described in this chapter requires the combination of prototype applications that embody the core functionality, as well as a realistic test environment. Chapter 5 describes the prototype applications I developed to demonstrate the feasibility of the concepts presented in this chapter. Chapter 6 presents an evaluation of these prototype applications within a test environment drawn from a real-world effort to perform concurrent CMT development.

Chapter 5
The Galápagos: Applications to Study Evolutionary Terminology Development

The focus of this dissertation —a methodology to support distributed development of logic-based terminologies using semantics-based concurrency control—was realized in a suite of software applications that I refer to as the Galápagos. This chapter describes the applications that either served as test vehicles for the evaluation described in Chapter 6, or that have been enhanced to overcome limitations that were identified during the evaluation (and thus better to illustrate the underlying methodology).

It would be a massive undertaking to develop a description-logic classification system, complete with user interface, persistent database and all the features required to support distributed development. Fortunately, applications were already available that had the core functionality necessary (a description-logic classifier and a persistent database) to serve as a foundation for Galápagos; it was unnecessary to develop these applications from scratch. The prototype applications described herein are based on an existing modeling environment known as K-Rep (Mays et al., 1991).

There are four distinct applications discussed here: the K-Rep Developers Environment (K-Rep DE), Isabella, Cristobal, and Rhabida.[1] Each of these applications relies on a common classification engine and persistent database for processing description-logic statements. The classification engine and database is referred to as the "K-Rep engine." This chapter presents this engine first, followed by discussions of the terminology development cycle and of how the individual application prototypes support this cycle.

5.1 K-Rep: Classification Engine

Chapter 4 presented the methodology for detecting concurrent development conflicts. These conflicts were detected by requiring the modelers to use description logic to represent their terminological definitions and then classifying the changes to determine if one modeler's work conflicted with another's. There are many implementations of description-logic classifiers available (Borgida, Brachman, McGuinness & Resnick, 1989; Brachman et al., 1991; Brachman & Schmolze, 1985; Brill, 1993; Moser, 1983). K-Rep (Mays et al., 1991) was chosen because it was already used within Kaiser Permanente's clinical information system projects, and its classifier functions were sufficient to demonstrate the viability of semantics-based concurrency control. It was necessary, however, to ask K-Rep's development team to make one enhancement specifically for this dissertation project: the addition of a journaling capability consistent with the change-set configuration management model.[2]

1. Applications I created specifically for this dissertation are named after islands of the Galápagos. K-Rep DE was developed by IBM's T.J. Watson Research Center and utilizes the same underlying classifier and persistent database as the Galápagos applications.

K-Rep is a knowledge-representation system based on description logic. As such, it has a well-defined, compositional, set-theoretic semantics, which allows it to generate inferences algorithmically, based on terminological definitions. These inferences are made available to a variety of applications through a C++ application programming interface (API) that provide access to K-Rep's underlying classification engine and persistent object database.

Figure 5-1 illustrates K-Rep's components and its functionality. Specifically, C++ API calls propagate changes through a description-logic classifier into a persistent object database while also appending those changes to a change-set file. The C++ API calls can import or export a definition file. Any application that modifies a concept's definition, or imports or exports a definition file, can only do so through the C++ API, thus ensuring that all modifications will appropriately be recorded in the change set, and that all imported definitions will be processed by the classifier.

For interactive development of terminology, the K-Rep engine's API calls are used by K-Rep DE to show the relationship of a particular term to other terms within the system. Applications use the API to determine if a concept is broader than, narrower than, equivalent to, or disjoint from another concept. Alternatively, the API can be used to modify the definition of a concept and thus cause reclassification of the terminology.

For semantics-based concurrency control, the K-Rep engine's API calls are used to determine if any other term within the system has an equivalent definition to another (the

2. Change sets were previously introduced in Section 2.3.1 and terminology-specific change sets were described in Section 4.3.3.

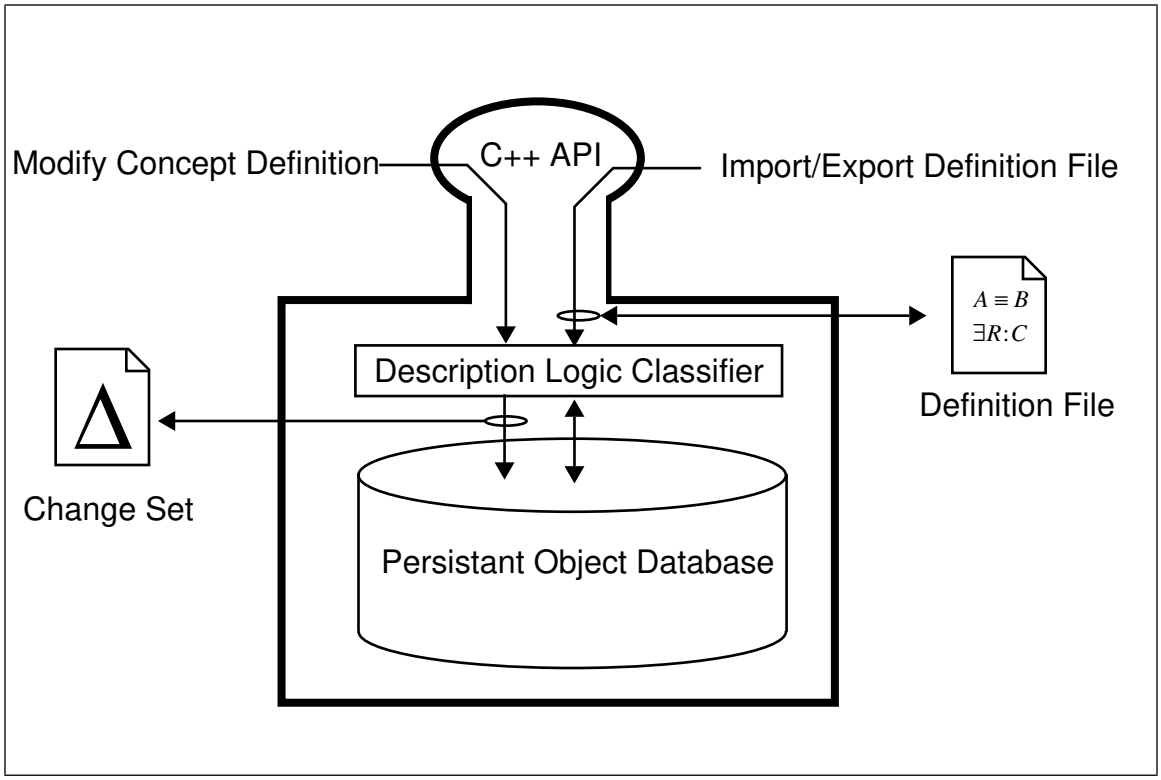


Figure 5-1. K-Rep Engine components and pictographic representation of selected API functions.

non-unique definition conflict—Section 4.2.1), or if a single term within the system was defined by two different modelers in semantically conflicting ways (the multiply-defined term conflict—Section 4.2.2). Chapter 4 described a scenario of changes to terminological definitions, and further described how those conflicts can be identified and resolved. Figure 5-2 summarizes those changes with a presentation of the KRSS definitions corresponding to each state and represents how a description-logic classifier would classify those definitions taxonomically. Ideally (although unattainably), a classifier can properly classify all possible distinctions (a computationally intractable situation), and efficiently process these distinctions (which is only possible if comprehensiveness of the classifier is sacrificed).

Although K-Rep can appropriately classify the KRSS definitions presented in Figure 5-2, the K-Rep engine cannot classify all possible KRSS definitions. The K-Rep engine does

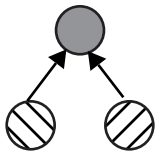
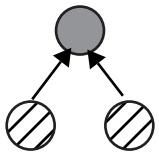
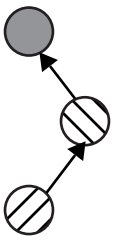
KRSS Concept Definition	Taxonomic Classification
Original Classification	
(defprimconcept DISEASE) (defprimconcept PULMONARY-DISEASE DISEASE) (defprimconcept INFECTIOUS-PNEUMONIA DISEASE)	
Non-unique definition conflict	
(defprimconcept DISEASE) (defprimconcept PULMONARY-DISEASE (and DISEASE (some AFFECTS LUNGS))) (defprimconcept INFECTIOUS-PNEUMONIA (and DISEASE (some AFFECTS LUNGS)))	
Multiply-defined term conflict	
(defprimconcept DISEASE) (defprimconcept INFECTIOUS-PNEUMONIA (and DISEASE (some AFFECTS LUNGS))) (defprimconcept INFECTIOUS-PNEUMONIA (and DISEASE (some CAUSED-BY INFECTIOUS-AGENT)))	
After conflict resolution	
(defprimconcept DISEASE) (defprimconcept PULMONARY-DISEASE (and DISEASE (some AFFECTS LUNGS))) (defprimconcept INFECTIOUS-PNEUMONIA (and DISEASE (some CAUSED-BY INFECTIOUS-AGENT) (some AFFECTS LUNGS)))	

Figure 5-2. Classification of terminological definitions.

not allow definitions that use the complete expressive power of first-order logic. Instead, its developers maintain it retains a “principled semantics” (Mays et al., 1991), using a

computationally tractable subset of first-order logic, and most closely compares to CLASSIC (Borgida et al., 1989). Table 5-1 shows the concept forming operators and the terminological axioms that K-Rep supports.[3]

Table 5-1. Concept forming operators and the terminological axioms of the K-Rep language. Compare with the concept forming operators and terminological axioms of the KRSS standard presented in Appendix B.

Concrete Form	Abstract Form	Description
Concept Forming Operators		
top	$\top$	The "top" of the hierarchy from which all concepts descend.
bot	$\bot$	The imaginary concept at the "bottom" of the hierarchy.
(and C_1...C_n)	$C_1 \wedge \ldots \wedge C_n$	A logical conjunction of concepts C_1 through C_n.
(some R C)	$\exists R{:}C$	There exists at least one relationship R constrained to be of type C.
(all R C)	$\forall R{:}C$	All relationships R are constrained to be of type C.
Terminological Axioms		
(define-concept N C)	$N \equiv C$	C defines necessary and sufficient conditions for N.
(define-primitive-concept N C)	$N \sqsubseteq C$	C defines necessary conditions for N.
(disjointprimitives P1 P2)	$P_1 \wedge P_2 = \bot$	Primitives P_1 and P_2 are mutually exclusive.

Although the K-Rep engine performs the necessary algorithms and database management functions for defining and classifying terminological definitions, it does not directly provide a suitable interface for dynamic modeling of concepts, nor does it provide support for

3. Readers interested in learning more about the algorithms used by description-logic classifiers and the trade-offs required to support different logical operators may be interested in the following references: Boolos, 1993; Boolos & Jeffrey, 1989; Brachman & Levesque, 1984; Fitting, 1990; Garey & Johnson, 1979; Nebel, 1988.

a coordinated terminology development cycle. It has been modified, however, to specifications required by this dissertation in order to generate a journal of all committed changes made by modelers.[4] This functionality provides a suitable foundation for applications designed specifically to support the development cycle. By placing the journaling functionality directly into the K-Rep engine, any application developed using the K-Rep API and database is automatically compliant with the requirements necessary to support semantics-based concurrency control and change-set configuration management.

5.2 Application Support for the Development Cycle

Although there are several development paradigms that differ in organizational perspective,[5] the steps in the CMT development cycle from the perspective of an individual modeler are constant. The cycle, presented in Figure 5-3, has four steps: 1) distribution, in which the modeler acquires a baseline terminology from a central coordinator, 2) enhancement, in which the modeler works to modify the CMT to correct defects and add new relationships and terms as necessary to meet local needs, 3) return, in which the modeler sends the change sets produced as a by-product of local enhancement to the coordinator, and 4) conflict resolution, in which the coordinator identifies conflicting changes made by individual modelers and modelers work together to resolve those conflicts.

4. The necessary modifications to the K-Rep engine were made during a two month period while I was an Academic Visitor at IBM's T.J. Watson Research Center.
5. Section 1.4.2 presented a centralized-coordination, local-control development paradigm. Section 7.3.1 will describe alternatives to such centralized-coordination, local-control development.

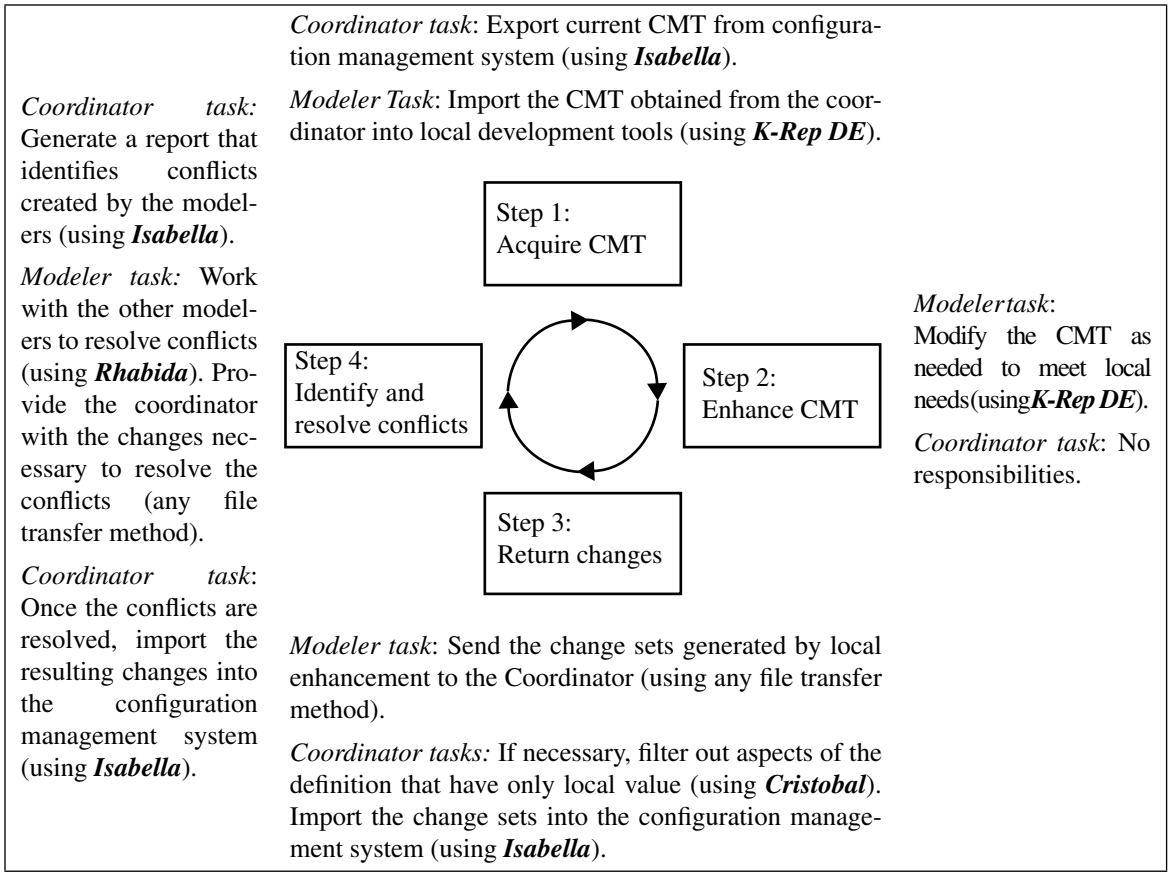


Figure 5-3. Terminology Development Cycle. Tools used to accomplish each step are in bold-face.

To manage these development cycle tasks, four applications are used. Three of them, Isabella, Cristobal, and Rhabida, are used primarily to manage tasks associated with distributed development. Isabella is used by the coordinator to manage the configuration of terminological definitions; Cristobal is used by the coordinator to filter out terminological enhancements that serve only local needs; while Rhabida is used by modelers to resolve conflicts provided to them by the coordinator in such a way that the evolving baseline terminology reflects a consensus of the modelers. The remaining application, (K-Rep DE) is used exclusively by the modelers to enhance the terminology. These applications are discussed in the next four sections in the order encountered within the development cycle.

5.2.1 Isabella: Configuration Management and Conflict Identification

Isabella is a terminology configuration-management application. As such, it forms the foundation for the Galápagos distributed-development support and implements the conflict-identification algorithms described in Chapter 4. Isabella can import change sets submitted by modelers into a configuration-management database, identify changes that conflict with changes made by other modelers, and produce reports of these conflicts that can be used by other applications—such as Rhabida—to resolve conflicts. Isabella supports the first step in the development cycle by allowing export of various configurations of definition files that can be distributed to modelers for enhancement. Although Isabella supports the first step in the development cycle, it also supports other steps in the development cycle, as this section describes.

Isabella is a UNIX character-based application that provides a rudimentary user interface, but with the essential functions necessary to support distributed development.When Isabella is run initially on a terminology database, Isabella transforms it into a *configuration-management* database. It accomplishes this by adding a data structure to represent all states of a terminological definition as well as to represent when, and by whom, a terminological definition was changed from one state to another. This data structure is a standard directed-acyclic graph, in which the states of the terminological definitions are represented by nodes and the transactions between states are represented by arcs. Hereafter, this data structure is referred to as the *definition state graph*.

Once Isabella has initialized the configuration database, users can either work interactively with the database using the character based menu shown in Figure 5-5, or access all func-

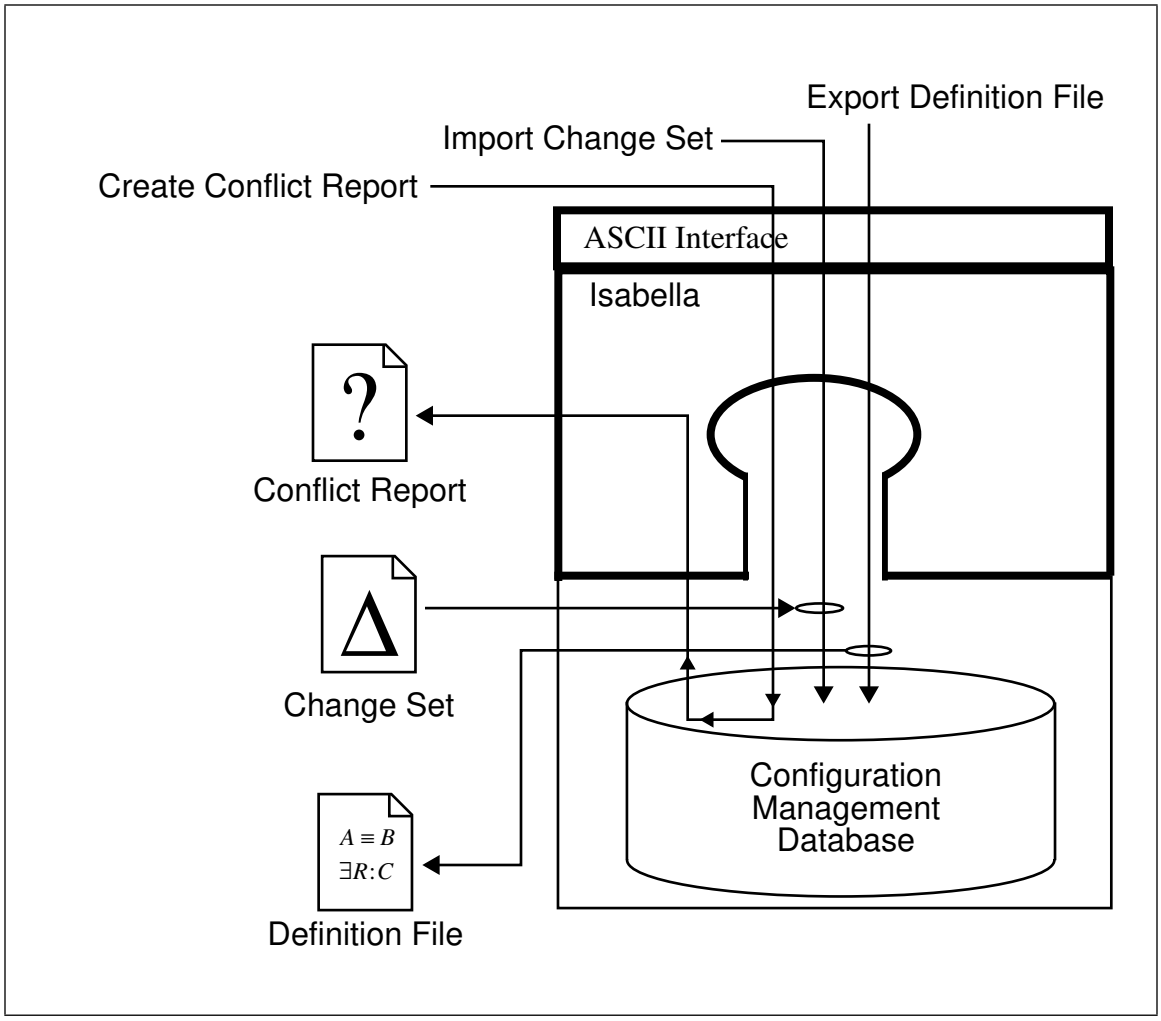


Figure 5-4. Isabella Architecture and representation of selected program functions.

tions using UNIX command line switches, thus allowing functions to be scripted for unattended execution. Isabella's basic functions include: (1) importing a change set into the configuration-management database (an initial step before initiating a development cycle), (2) printing the defining states within the configuration database for a particular concept (complete with all the information about the transitions between states), (3) exporting a definition file (supporting step 1 of the development cycle), (4) reporting all the concepts that are participating in a non-unique definition conflict or a multiply-defined term conflict

(supporting step 4 of the development cycle), and (5) resolving conflicts interactively (also supporting step 4 of the development cycle).

```
$ isabella snomed.kp.cms snomed.kp.cms-2.26.db
Opening persistent KB snomed.kp.cms in file snomed.kp.cms-2.26.db
Open...
================================================================
1. import CS,       2. concept states,    3. export definitions,
4. conflict report, 5. resolve conflicts, q. (quit):
```

Figure 5-5. Isabella's character-based interface, showing the command options available to the coordinator who uses this module.

When a change is imported, the starting and ending states are compared with existing entries in the definition state graph. If the starting state and ending states already exist in the definition state graph, the change is ignored (it has already been encountered in the past). If an ending state is *not* found, a new node is created to represent the new ending state, and an arc is added to the graph connecting that node with its starting state (or against a "null" starting state if a node corresponding to the start state was not found).

If a new entry is created, this entry is classified to determine if it conflicts with another (either a non-unique definition conflict or a multiply-defined term conflict). If it does not, the entry is accepted as final, and the classification of the entire terminology is updated. If the entry does conflict with another, the conflicting definitions are identified and the classification of the terminology—with respect to that concept—is postponed until the conflict is resolved.

When a conflict report is generated, Isabella iterates through the terminology one concept at a time and processes the definition state graph for each concept to see if there are any unresolved conflicts. If any are encountered, it prints out the state graph for each conflict and computes the different ways the conflicting concept might be similar (e.g., is one definition more specific than another? are the defining concepts equivalent? are the defining relationships equivalent?). Figure 5-6 presents an entry from an actual conflict generated when comparing Kaiser Permanente's definition for "congenital syphilitic mucous patches" with the equivalent representation in SNOMED version 3.3.

This conflict report entry shows the name of the concept in question (`Congenital-syphilitic-mucous-patches_DE-14514`) followed by the three states of the definition for this term in the configuration database. The first state is a "null" state representing the definition of this concept before its creation, the second state is the definition created by Kaiser Permanente, and the third state is derived from SNOMED 3.3.

Below these three states are two arcs. The first arc, from 1 to 2, with the comment "KP start" below it, represents the initial creation of a definition for this term by a Kaiser Permanente modeler. The second arc, from 1 to 3, represents the initial created on a definition for this term in an official SNOMED release. Notice that since these definitions were independently created, they start from a "null" definition, rather than having one definition follow from the other.

Since these two definitions are not equivalent, and since neither derived from the other, they represent conflicting definition states for the concept "congenital syphilitic mucous patches." Isabella determined that these concepts are conflicting by performing a

```
Congenital-syphilitic-mucous-patches_DE-14514

1 N

2 P (and Syphilis-NOS_DE-14300
     (some ASSOC-MORPH Patch-NOS_M-04200)
     (:SNOMED-CODE "DE-14514"))

3 P (and Syphilis-NOS_DE-14300
     Congenital-infectious-disease-NOS_DE-01900
     Sexually-transmitted-disease-NOS_DE-01600
     (some ASSOC-ETIOLOGY Treponema-pallidum_L-25901)
     (some ASSOC-TOPO Mucous-membrane-NOS_T-00400)
     (:SNOMED-CODE "DE-14514"))

Arc 1 2
"KP Start"

Arc 1 3
"SNOMED 3.3"

2 3 Facts
N    // --Concepts are NOT equivalent--
AdjB // --Concept 2 is not subset or superset of 3--
DCNE // --Defining concepts are NOT equivalent--
DRNE // --Defining roles are NOT equivalent--
EndFacts
End
```

Figure 5-6. Example entry from a conflict report.

depth-first search over the definition state graph and determining that there was more than one terminal node (both nodes 2 and 3 are terminal). If more than one terminal node is found, Isabella then compares the definitions corresponding to the terminal nodes to determine how they are alike and how they are different. The results of this comparison is then

recorded in the conflict report as a set of facts. In Figure 5-6, these facts are immediately below the line "`2 3 Facts`."

As the example in Figure 5-6 shows, reviewing these reports can be very tedious, since it is hard to determine exactly how the conflicting definitions are different—as well as how they are alike. Although Isabella has a menu option that will allow a modeler to resolve the conflicts interactively with the character-based interface, this interface was found to be even more tedious than the paper-based reports. To overcome this problem, a separate tool, Rhabida, was created to make reviewing and resolving conflicts more intuitive and efficient. Since Rhabida relies upon the conflict reports generated by Isabella to function, Isabella's conflict resolution function will be more fully illustrated when the Rhabida tool is discussed in Section 5.2.4.

5.2.2 K-Rep DE: Terminology Enhancement

The second step in the development cycle is enhancement of the CMT. The K-Rep Development Environment (K-Rep DE) is the principal tool used by modelers to enhance the terminology. Modelers can modify the definition of a term with K-Rep DE and interactively review the classification results after committing each change. K-Rep DE is a UNIX hosted X-Windows application. As such, K-Rep DE links with both the API provided by K-Rep's engine as well as X-Window's API. Figure 5-7 illustrates K-Rep DE's architecture.

Using X-Windows, K-Rep provides a graphical user interface that allows the concepts in the knowledge base to be displayed in hierarchical browser and allows modelers to create

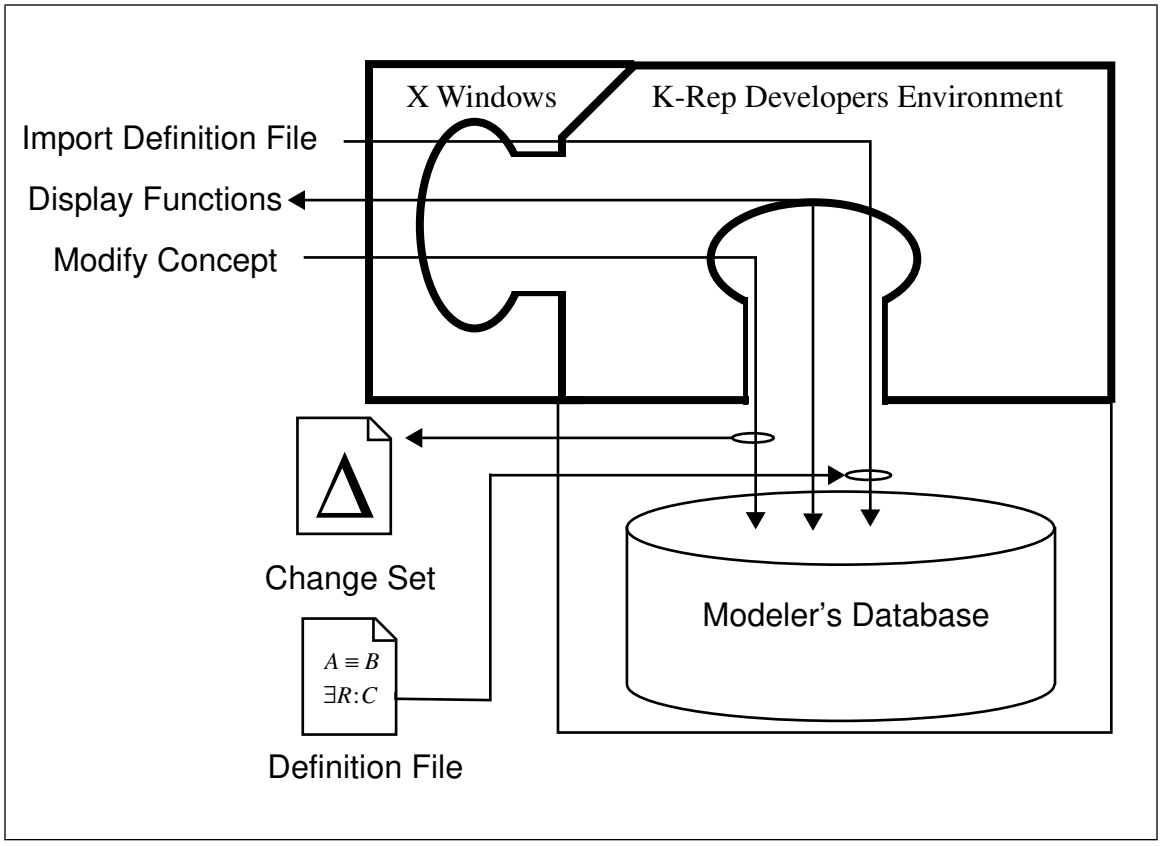


Figure 5-7. K-Rep DE Architecture. K-Rep DE uses the K-Rep engine and provides an interactive X-Windows interface.

new defining relationships between terms through "add role" and "add parent" functions. Figure 5-8 shows K-Rep DE's taxonomic view. This view displays the concepts in a hierarchical browser in which the location of individual concepts is based upon the classification of individual terms, rather than strictly upon the defining relationships of those terms. There are also other tabs within this window that can be selected to perform other operations on the terminology as a whole, such as searching for concepts whose names may match an expression, or viewing all the roles that are defined for this terminology.

In order to view individual concepts in more detail, users can double click on them in the taxonomy viewer or select concepts retrieved from the search function, and then be presented with a "concept viewer." Figure 5-9 shows such a concept viewer for the term

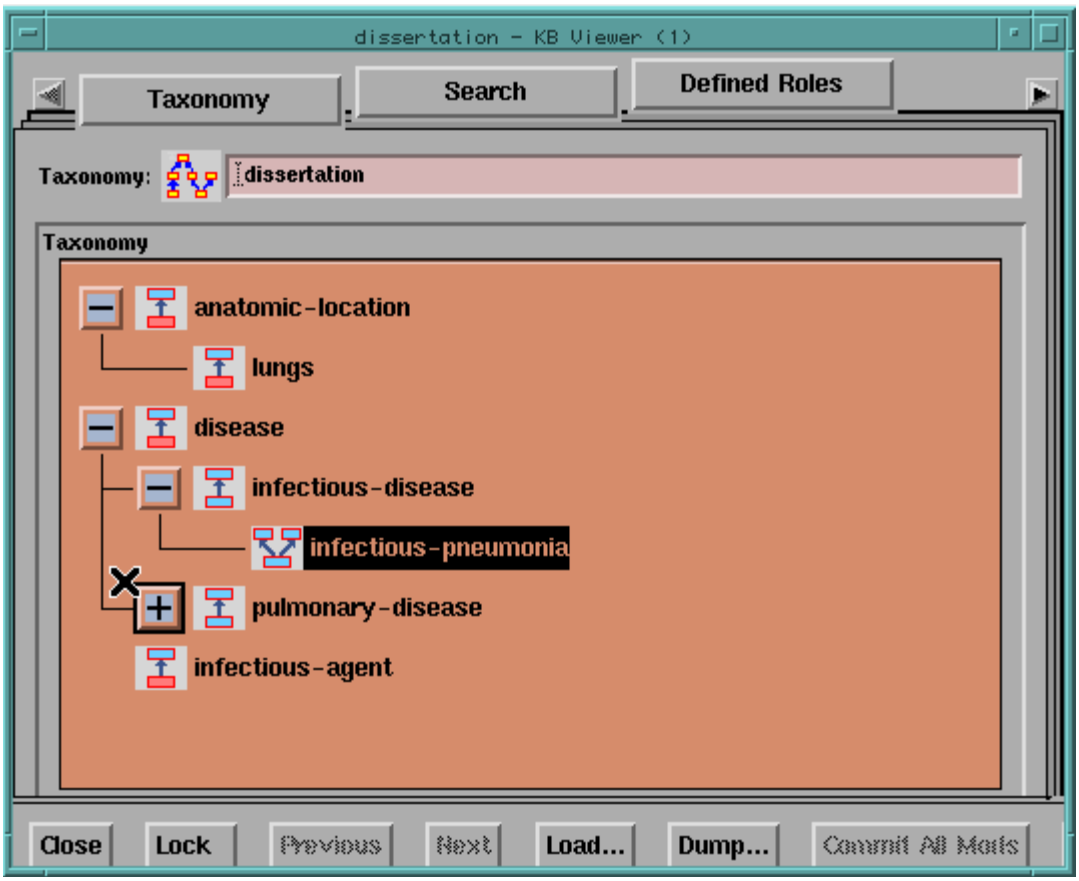


Figure 5-8. K-Rep DE's taxonomy view. This view is used to browse the terminologies classification hierarchically. Terms with children are indicated by buttons with either a "-" (in the open state) or a "+" (in the closed state). The user may toggle between the open and closed state by clicking on the button with the mouse. Terms that have more than one immediate parent are indicated by the icon with two arrows. Terms that have only one parent are shown by icons that have only one arrow. In addition, this view differentiates terms that are fully defined (with a blue box at the bottom of the icon) from those terms that are primitive (with an orange box at the bottom of the icon).

"infectious pneumonia." This figure shows a graphical representation of the definition for infectious pneumonia equivalent to this KRSS definition:

```
(define-concept INFECTIOUS-PNEUMONIA
    (and DISEASE (some AFFECTS LUNGS)
                 (some CAUSED-BY INFECTIOUS-AGENT)))
```

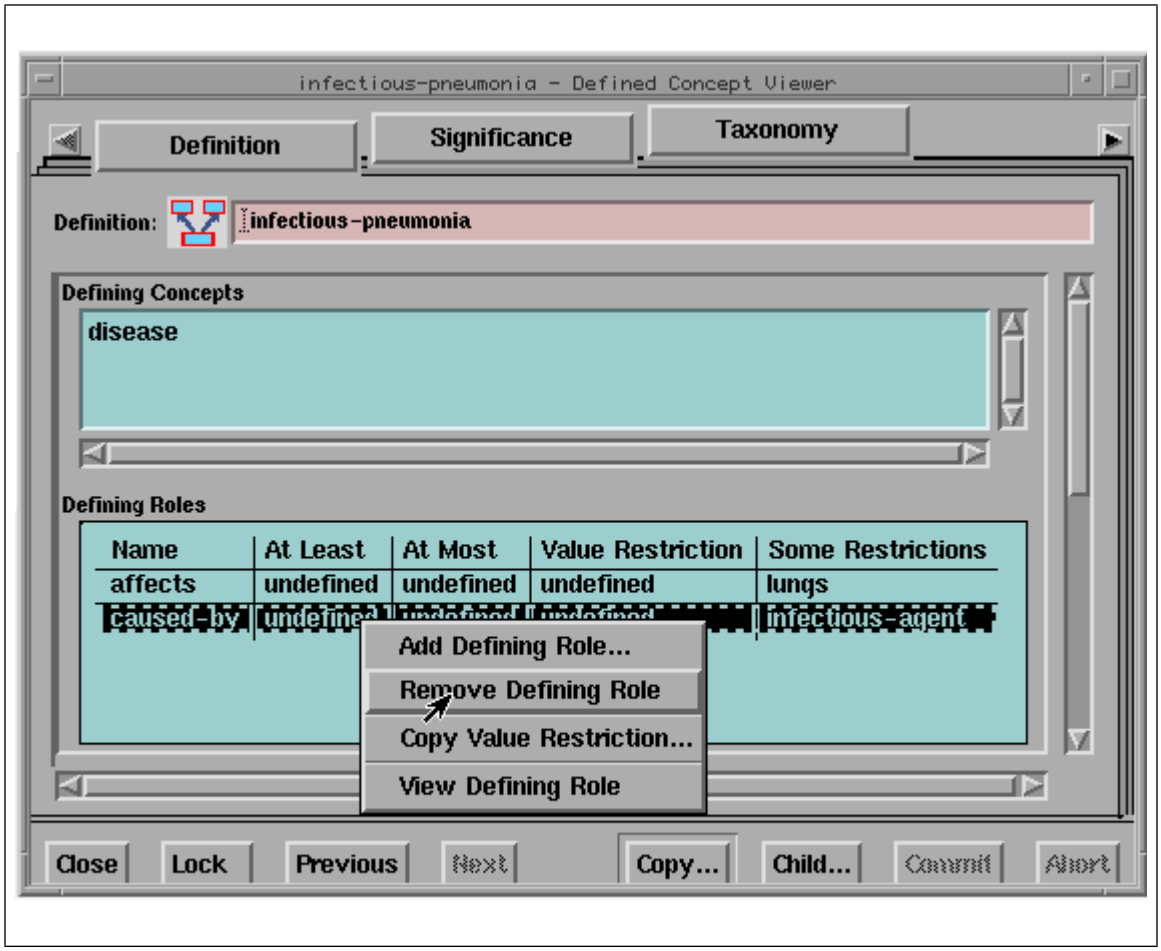


Figure 5-9. Concept viewer displaying the definition of “infectious pneumonia”.

Modelers can refine the definition of a term by selecting from among the appropriate functions from pop-up menus which allow them to: (1) add or remove defining relationships (roles), (2) add or remove value restrictions to a defining role, or (3) add or remove defining concepts to or from a concept’s definition.[6]

Although K-Rep DE was developed independent of this dissertation, changes made by developers within this environment can be imported into the Isabella application because it

6. Representation of these defining terminological relationships was previously presented in Section 3.3.

relies upon the K-Rep engine (which has been modified for this dissertation) to create change sets.

When the developer commits the changes made to an individual term, the editing environment (1) records the changes in a change set that can be sent to the central coordinator, and (2) verifies the internal consistency of the modeler's CMT by ensuring that the new type definition does not conflict with other type definitions within the CMT. The editing environment looks only for *local* conflicts since it has no access to the changes of other developers.

5.2.3 Cristobal: Filter Changes

The third step in the development cycle is to collect enhancements of the CMT and to filter out changes that are strictly of isolated (local) interest. There are many circumstances when such a filter step is necessary, one of which is presented below in a scenario.

Suppose a modeler experiments with a new model for anatomical relationships. She chooses to represent more detail in her model than is allowed using a simple IS-PART-OF relationship that the other modelers have agreed to use. She defines two specializations of these relationships: IS-A-FUNCTIONAL-PART-OF (for defining relationships between a functional component—such as the knee—and the physical parts that together constitute that functional component—such as the patella) and IS-A-PHYSICAL-PART-OF (for defining relationships between structural components—such as the femur—and other structural components that are physically integral parts of the femur—such as the femoral head). Since these two roles are proper subtypes of the IS-A-PART-OF role, anytime an IS-A-FUNC-

TIONAL-PART-OF or IS-A-PHYSICAL-PART-OF role is defined, it is appropriate to apply the IS-A-PART-OF relationship. The modeler also defines one new role that has no equivalent in the convergent terminology: IS-A-TRIBUTARY-OF (for defining relationships between larger vessels—such as the aorta—and connected smaller vessels—such as the subclavian artery).

A simple filtering application can allow modelers to do these kinds of experiments while providing a transformation function that can convert their change sets into the format agreed upon by the other modelers. If the relationships that are to be filtered out are always more detailed than the relationships in the reference model, a simple parsing application is adequate for this purpose, provided that it can identify all the tokens in a change set, use a memory-resident mapping table to determine which tokens should be substituted, and write the filtered change set to disk.

Although this filtering functionality was not anticipated as part of the original proposal, circumstances within the Kaiser Permanente project made such an application essential to allow continued collaboration between groups who, while agreeing to certain general principles of modeling, wanted to include additional application-specific enhancements they did not want incorporated into the reference terminology. Cristobal is an example of such a simple one-way filter. Figure 5-10 is a representation of Cristobal's filtering functionality.

Cristobal currently does not support situations in which the local model is in some cases more general and in other cases more specific than the reference version. Such support is planned, however, for future versions (see Section 7.3.2).

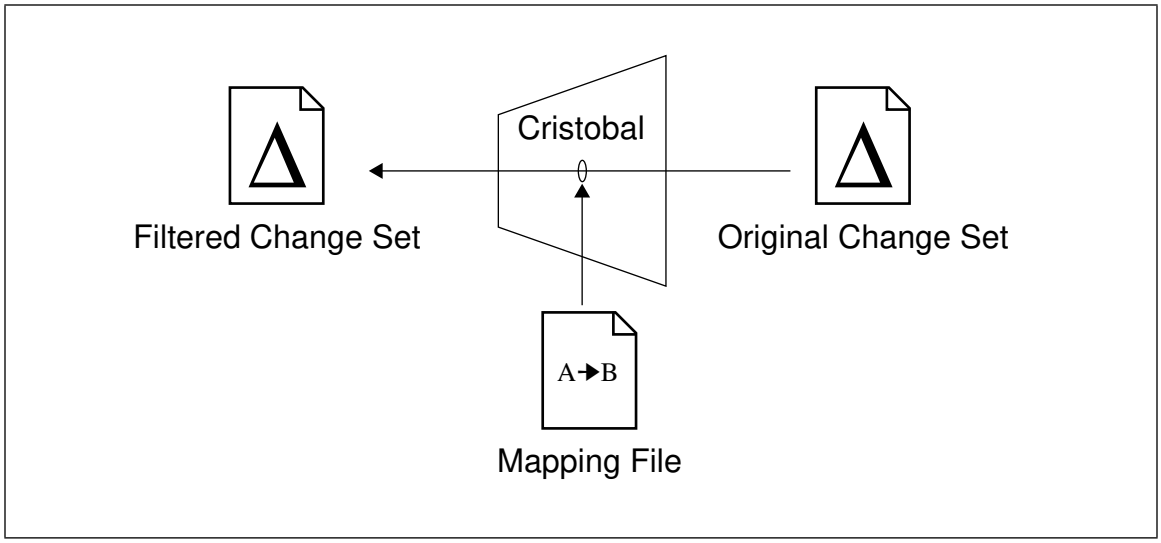


Figure 5-10. Representation of Cristobal's filtering functionality.

5.2.4 Rhabida: Conflict Resolution

The final step in the development cycle is to identify and resolve conflicts. Isabella is the application that identifies the conflicts and generates a report enumerating them. Rhabida is designed to parse the conflict reports and to present the conflicts to the user in an intuitive and efficient manner. Figure 5-11 is a representation of Rhabida's architecture and functionality.

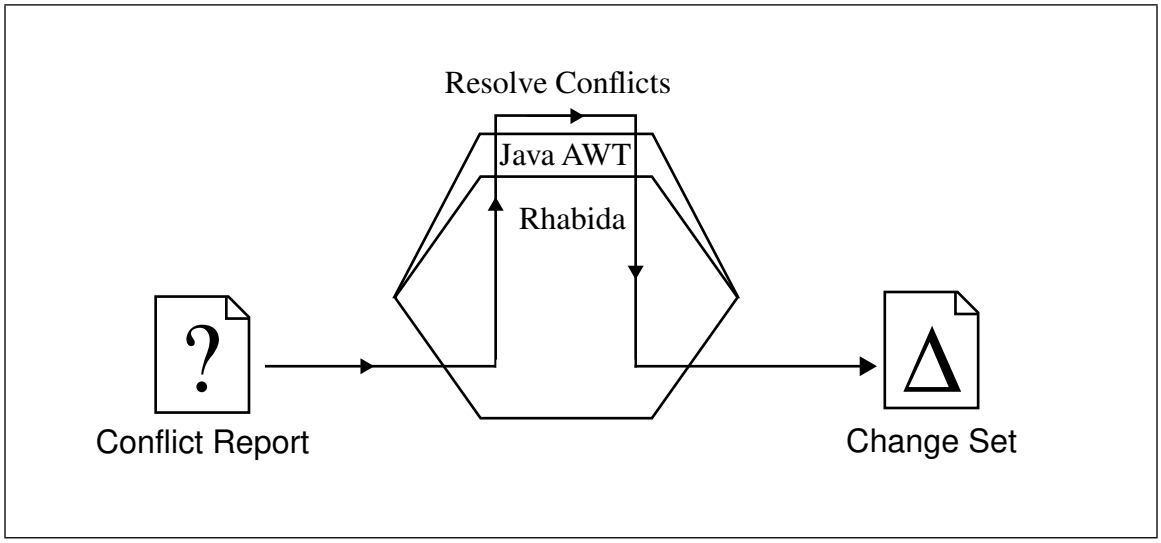


Figure 5-11. Rhabida architecture and representation of its functionality. Rhabida interacts with the user through the Java Abstract Window Toolkit (AWT).

Rhabida has no real-time connection with the K-Rep engine. Instead, it relies entirely upon information contained within the conflict report generated by Isabella. As users resolve conflicts with Rhabida, their actions are captured in change sets (the same format as change sets generated by K-Rep DE). These change sets are then imported into the configuration database by Isabella to resolve the conflicts.

Rhabida was designed to present conflicts to users in a visually intuitive and efficient interface. Figure 5-12 shows the Rhabida conflict resolution interface. The example presented in this figure is the same example presented in a textual conflict report in Figure 5-6. There are several features that make the Rhabida interface easier to use than a strictly character-based interface.

The first of these is the computation of the maximal common definition and the conflicting parts. The conflicting definitions, presented in the lower panel of Figure 5-12, are processed by an efficient maximal common subgraph algorithm to determine what defining features the definitions share.[7] Common features are colored white, and initially presented in the upper left pane of the Rhabida window. Conflicting defining features (or parts of a definition) are color coded in sequential colors and presented both within the definitions from which they are derived as well as collectively in the upper right pane of the Rhabida

7. Computation of the maximal common subgraph is of course intractable (it is NP complete) (Garey & Johnson, 1979). However a backtrack search algorithm combined with methods for ordering the search and for limiting the search by refuting intermediate invalid states can provide acceptable performance in selected domains (McGregor, 1982). Rhabida orders the search using the term labels to limit the domain values for the backtrack search (nodes are only considered valid candidates if their labels lexically match), and by using heuristics to test validity of intermediate states (e.g. a node can only be validly mapped to another if the parents of those nodes are also mapped to each other; and a search should only be continued if the sum of the nodes currently matching and the nodes unassigned is greater than or equal to the previous “best” match). Using these three methods for limiting the search space, example graphs that take 48,000 iterations in the worst case to compare have been reduced to 70 iterations.

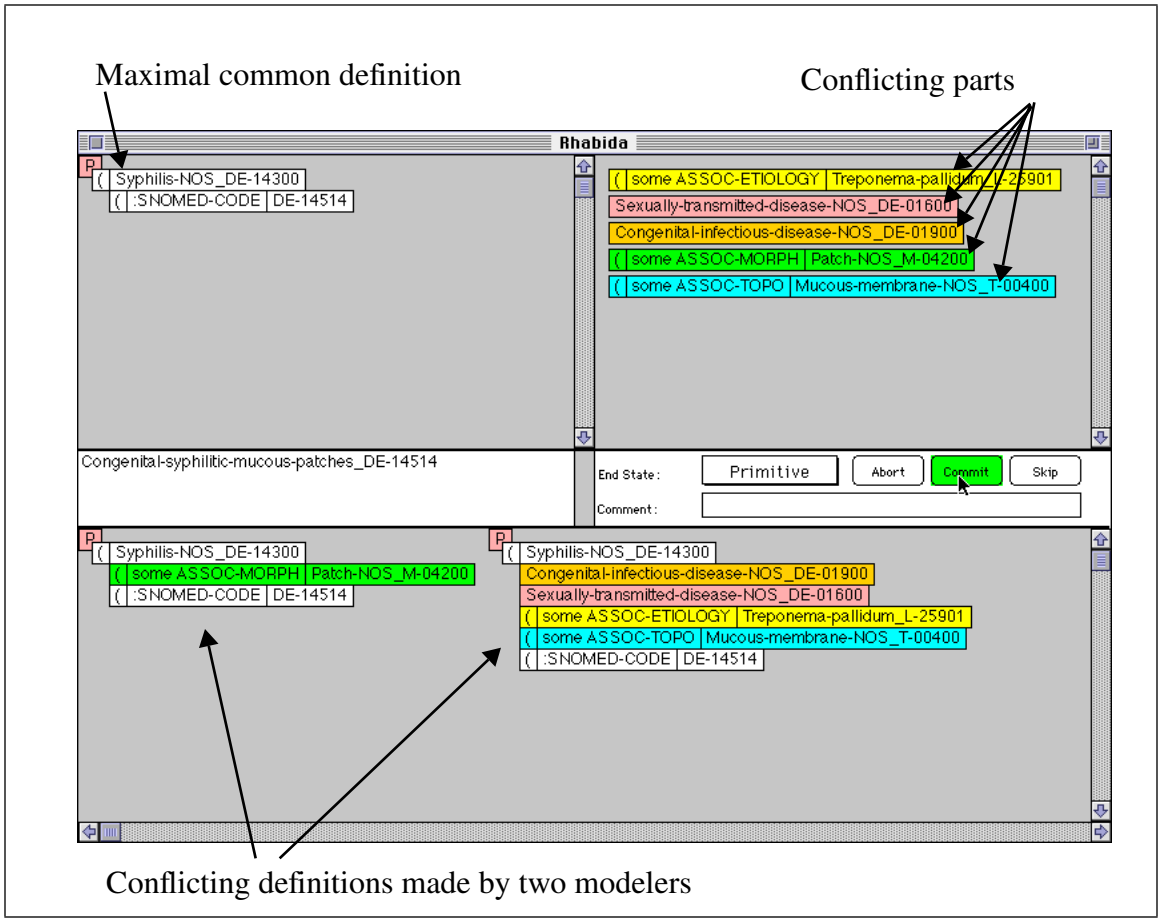


Figure 5-12. Rhabida tool demonstrating conflicting definitions of CONGENITAL-SYPHILITIC-MUCOUS-PATCHES.

window. The sequential colors allow users to quickly track the conflicting parts between the part panel and the panel displaying the conflicting definitions.

To resolve conflicts, the users review the conflicting parts, and then determine which—if any—parts they wish to include in the "resolved" definitions. The parts may be included simply by double clicking on the desired part in the upper right panel. Those parts are then added to the maximal common definition in the upper left panel. Since they maintain their respective colors, they remain easy to track. Once users are satisfied with resolved definition, they click on the "commit" button. Rhabida will then write the changes to the commit journal, and then process the next conflicting definition in the conflict report.

Once the conflicts have been resolved, the change set generated by Rhabida is imported by Isabella into the configuration database.

5.3 Prototype Application Summary

Rather than looking at the development process strictly through the eyes of a modeler, the prototype applications described in this chapter have been designed to support important steps in this cycle that are ignored by traditional description-logic development applications such as the conflict identification, conflict resolution, and change-set configuration management. Figure 5-13 brings these steps into higher relief. It is based on the same development cycle originally presented in Figure 5-3, but focuses on the flow of information among the Galápagos applications. It uses the previously presented representations of Isabella, Rhabida, Cristobal, and K-Rep DE to clarify the relationships among these applications, change sets, definition files, and conflict reports, as well as to reinforce the order in which these applications are encountered within the development cycle.

In step one, Figure 5-13 shows the information flow as a coordinator exports a definition file from the configuration database using Isabella. A modeler subsequently imports that definition file into the local database using K-Rep DE. In step 2, the modeler enhances the CMT using the K-Rep DE application. All changes made using K-Rep DE are captured as change sets. In step 3, the change set created by terminology enhancement is filtered by the Cristobal application to remove any relationships that are strictly of local interest. Once filtered, these changes are then imported into the configuration database by Isabella. Finally, in step 4, Isabella generates a conflict report identifying any conflicting changes.

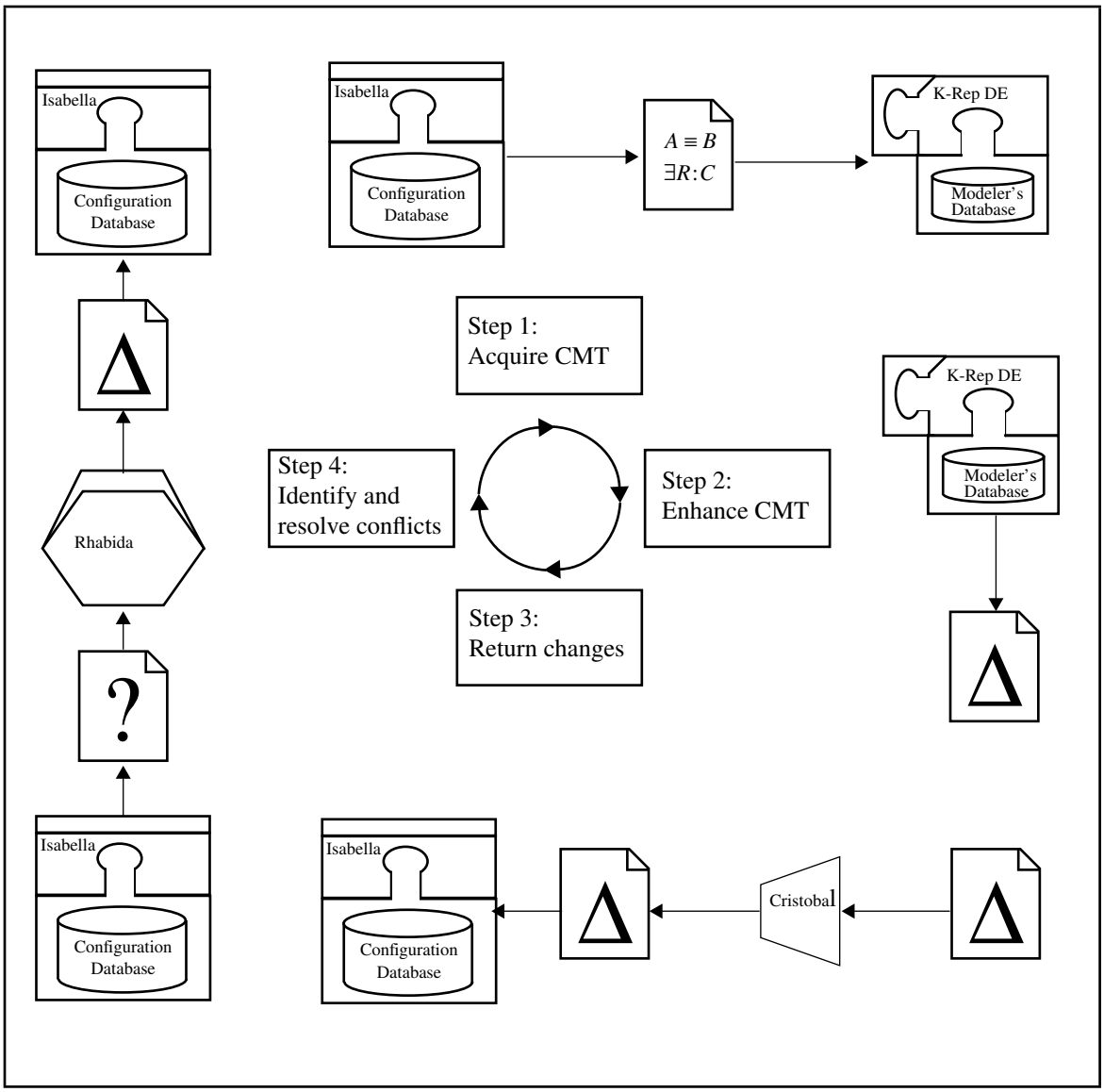


Figure 5-13. Representation of the terminology development cycle showing information flow between the Galápagos applications.

This conflict report is processed by Rhabida and presented to a modeler to resolve the conflicts. The change set produced by Rhabida is then imported into the configuration database by Isabella, and the conflicts are resolved. The cycle then repeats.

5.4 Future Needs for CMT Development Applications

Although the Galápagos applications provide support for each step in the development cycle, there are certainly ample opportunities for improvement. Examination of Figure 5-3 makes the number of steps readily apparent. The number of file transformations involved in the development cycle is an obvious area where the process can be improved. Although it might not reduce the actual number of steps in the process, providing an environment in which the functions of Isabella, Rhabida, Cristobal, and K-Rep DE are all integrated would significantly simplify management of the development cycle by eliminating the need to manage files that are constantly being exported from one application only to be imported into another.

Finally, although Cristobal filled an important need within the Kaiser Permanente project, the exchange of information is unidirectional. This kind of exchange is only a temporizing solution. The group benefits through application of the filtered change sets to the convergent terminology, but individuals are unable to apply the work of others to their locally enhanced terminology,[8] thus limiting their benefit from the collaboration. This filtering

8. Because of the impedance mismatch created by using relationships that are more specific than those in the convergent terminology model, the filtering of changes is only unidirectional. Consider the example presented in Section 5.2.3 where a modeler chose to use IS-A-FUNCTIONAL-PART-OF and IS-A-PHYSICAL-PART-OF rather than the agreed upon IS-PART-OF relationship. Although the modeler can work locally with the IS-A-FUNCTIONAL-PART-OF and IS-A-PHYSICAL-PART-OF relationships, and have those relationships generalized by Cristobal to the IS-PART-OF relationship for the convergent terminology, the reverse is not true. When modelers at other sites make changes to the IS-PART-OF relationship, Cristobal is unable to determine which of the more specific relationships (IS-A-FUNCTIONAL-PART-OF or IS-A-PHYSICAL-PART-OF) should be applied when filtering changes destined for the site with the more specific relationships, therefore, any change made to the IS-PART-OF relationship at other sites cannot be directly applied, thus limiting that sites benefit from the collaboration. Exchange of relationships *other* than the IS-PART-OF, IS-A-FUNCTIONAL-PART-OF or IS-A-PHYSICAL-PART-OF remains unhindered (thus the benefits of the collaboration are not eliminated).

functionality must be improved before meaningful *bidirectional* exchange of work is practical.

Chapter 6
Evaluation

To demonstrate the validity of the approach outlined in this dissertation, I have adopted an evaluation framework outlined in a recent report from the National Research Council (1994). That framework includes three components: 1) proof-of-existence, 2) proof-of-concept, and 3) proof-of-performance. The first of these, proof-of-existence for semantics-based concurrency control, has already been demonstrated by researchers in other fields (Barghouti & Kaiser, 1991; Garcia-Molina, 1983). My evaluation of Galápagos has thus been designed first of all to demonstrate as a proof-of-concept that semantics-based concurrency control methods are transferable to the task of developing logic-based terminology, and, second, to demonstrate as a proof-of-performance that semantics-based concurrency control methods can meet performance expectations for routine use.

My thesis (Section 1.4.5) was evaluated using the applications described in Chapter 5. As I describe in the remainder of this chapter, I have shown that:

- Distributed development of a logic-based CMT generates conflicts detectable by Aristotelian classification.

- Once identified, a coordinating modeler can resolve these conflicts by interacting with the prototype configuration management environment described in Chapter 5. The results of these resolutions can be stored in a configuration management environment that is capable of exporting and importing change sets.
- The configuration management environment can utilize the change sets to coordinate CMT configurations among multiple sites.

6.1 Proof-of-Concept[1]

For the proof-of-concept of semantics-based concurrency control, existing data were imported into Galápagos to verify the environment's ability to identify conflicts. Conflicting definitions were then presented to a group of terminology modelers to introduce them to Galápagos' methods and to obtain their initial reactions.

Kaiser Permanente and the Mayo Clinic have been working to enhance SNOMED for use in their electronic medical record projects. As a pragmatic first step in formalizing SNOMED, Lexical Technology, Inc. (LTI) generated reports by using lexical inferences from a term's names and synonyms to suggest relationships between terms. The term DIARRHEA-DUE-TO-ESCHERICHIA-COLI, for example, is classified as a diarrheal illness in SNOMED, but it may not be linked to the living organism ESCHERICHIA-COLI with a formally defined relationship. LTI processed the SNOMED nomenclature to generate reports with many suggested relationships of this sort (Lipow et al., 1996).

1. This section is adapted from a paper presented at the 1996 Fall Symposium of the American Medical Informatics Association (Campbell et al., 1996).

The LTI analysis proposed approximately 250,000 IS-A, DUE-TO, HAS-MORPHOLOGY, HAS-FUNCTION, and AFFECTS relationships for SNOMED. These relationships were then split into several hundred smaller files which were an appropriate size to review in about an hour. These files were distributed to Kaiser Permanente and Mayo Clinic for reviewers to accept or reject the proposed relationships. In most cases, only one reviewer evaluated each file. In a small number of cases, however, the files were reviewed by more than one individual; these multi-reviewer cases formed the basis of the Galápagos proof-of-concept experiment.

6.1.1 Conflict Detection

The files reviewed by multiple individuals were processed into description-logic concept definitions, and then imported into the Galápagos environment. Galápagos generated a conflict report that identified all terms with multiple definitions and it classified each pair of definitions for a single term as either semantically equivalent or semantically conflicting.

Mayo and Kaiser Permanente each modified the definitions of 1843 SNOMED disease terms by creating new DUE-TO relationships between the SNOMED disease terms and other SNOMED terms representing the etiology of the disease. These DUE-TO relationships were created by either accepting or rejecting candidate relationships. Of the 1843 terms modified, just 82 definitions were defined differently by the two sites for an overall conflict rate of 4.4%. Of the 82 conflicts, 14 were judged to be semantically equivalent (see Figure 6-1) and 68 were semantically conflicting (see Figure 6-2).

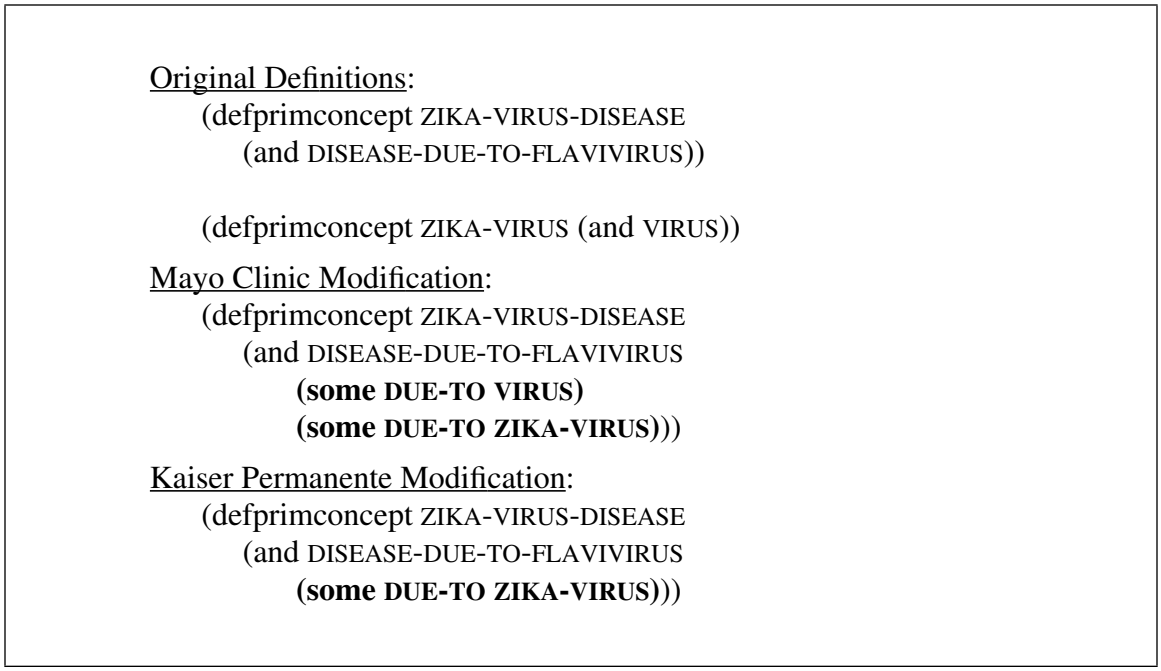


Figure 6-1. Semantically equivalent changes.

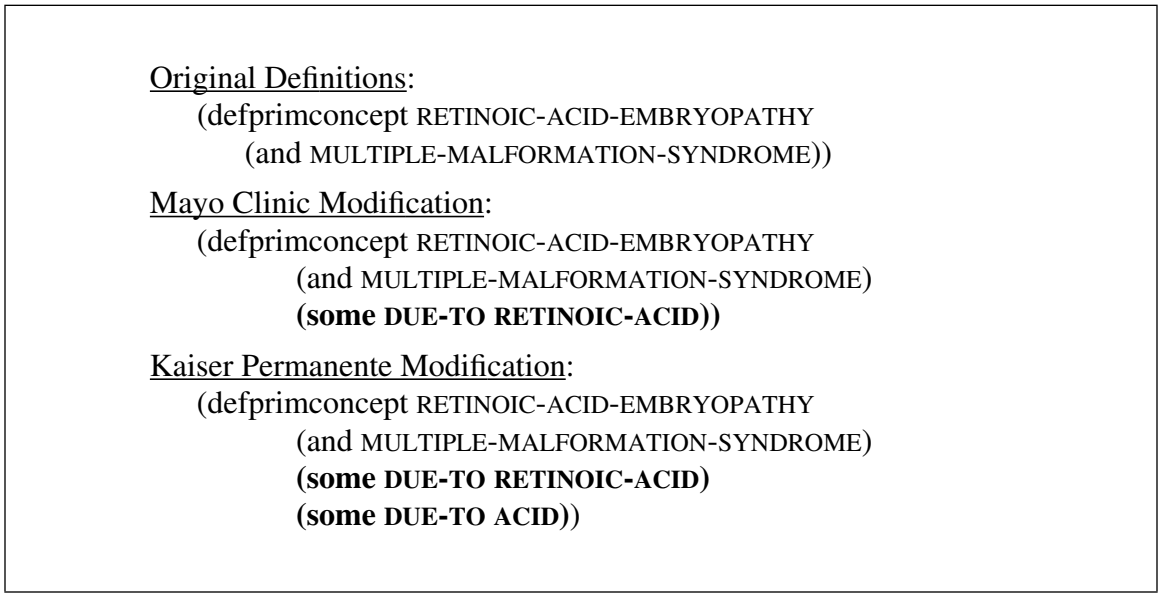


Figure 6-2. Semantically conflicting changes.

6.1.2 Conflict Review

The conflict report generated after Galápagos imported and classified each of the definitions was reviewed by modelers at Mayo Clinic and at Kaiser Permanente. Although there

was no formal study of the response to the conflict report, modelers at both sites noted that:

- The overall low conflict rate was reassuring, although its validity for tasks other than the review of lexically-generated reports was uncertain. Since the lexically-generated reports constrained the modeler to answering only yes or no to the proposed relationships, the modelers were not allowed to propose any new relationships. Therefore, the number of conflicts encountered may not generalize to tasks where the modeler is allowed to make unconstrained changes to any number of concepts. Fortunately, the next section will show that even when users are allowed to make unconstrained changes, the conflict rate has continued to be manageable.
- The concurrent work provided a mechanism for identifying many mistakes and improving the quality of the work since it was unlikely that two modelers would make identical errors.
- Some of the conflicts identified different approaches to the modeling task, and discussion of such conflicts at an early stage was helpful for clarifying the design task. For example, reviewing the conflicts revealed that the DUE-TO relationship was being applied inconsistently. Figure 6-2 shows one example of such inconsistent use (one modeler was considering ACID to be any "strong" or "weak" acid, another modeler was

only considering ACID to be a "strong" acid[2]). Identifying such conflicting application of terms or relationships through semantics-based concurrency control—and resolving those conflicts—provides a structured framework for ensuring consistent modeling.

- The classification of conflicts into semantically equivalent and semantically conflicting categories provided a means to review the quality of the hierarchy that was related to, although not directly defined by, the conflicting definitions. For example, if a concept was defined in two different—but redundant—ways (as is the case with the definitions presented in Figure 6-1), and, the classifier failed to recognize that the concepts were equivalent, (as would be the case for the example in Figure 6-1 if another relationship had failed to represent that ZIKA-VIRUS is-a VIRUS), then a defect in the hierarchical classification related to the concept of interest can be identified.

Although this demonstration of the Galápagos semantics-based concurrency control methods was a major project milestone, and laid the foundation for the subsequent evaluation work described in the remainder of this chapter, retrospective analysis of this demonstration—with the clarity provided by subsequent project experience—reveals that a significant opportunity was lost. At the time, the project focus was on preparing the tools for distributed, *interactive*, terminology development (using the K-Rep DE application described in Section 5.2.2), and the lexically-generated reports were viewed as an efficient approach to "bootstrap" the process. Although a few reports were edited redundantly by more than one modeler, the vast majority of reports had no such redundant review in the

2. A chemical substance that yields hydrogen ions upon ionization is defined as an acid. A *strong* acid undergoes complete ionization (hydrochloric acid and sulfuric acid are examples of strong acids). A *weak* acid undergoes partial ionization (acetic acid (household vinegar) and amino acids (building blocks for proteins) are examples of weak acids) (Bohinski, 1973).

interest of developing a baseline terminology as quickly as possible. The few cases where there was duplication of effort were seen as an opportunity to validate the behavior of the conflict-detection and configuration-management software intended to support the interactive tools, not as an opportunity to review the actual conflicts in a group setting and thus provide an opportunity for group convergence on modeling principles.

This oversight was made apparent during a presentation of the findings of this experiment to the modeling group. The presentation focused on the assertion that the conflict-identification and configuration-management tools had been successfully validated, and presented five representative examples of the 82 conflicts identified. Although the group found the examples presented interesting, they were not satisfied with those limited examples, and as a group, they wanted to go through *all* the conflicts and converge upon terminology modeling principles. The project's need to move to the next phase as quickly as possible prevented further review of the lexically proposed relationships. Had these relationships been further reviewed, and conflicting design decisions identified at this early phase of the terminology project, subsequent phases of the project may have been less problematic.[3]

3. Section 7.2.2 presents a subsequent problem introduced in part by spending inadequate time reviewing the lexically generated reports and identifying conflicting relationships accepted by different modelers. As of this writing, these relationships are being re-reviewed by the modelers. Each relationship is reviewed by at least two modelers, conflicts are identified using the same methodology described here, and conflicts are being resolved through a consensus process.

6.2 Proof-of-Performance[4]

Shortly after the Galápagos applications demonstrated correct conflict-identification and configuration-management capabilities, the Kaiser Permanente CMT project moved into its next phase: interactive, distributed terminology development using the K-Rep DE application. Prior to this phase, all terminology work was coordinated by allowing only one modeler to work on the terminology at a time. The productivity demands of the CMT project quickly made such coordination untenable. One of the Kaiser Permanente regions (Colorado) had hired and was in the process of training two additional modelers (for a total of three), and they wanted these modelers to work concurrently on the terminology. They were faced with significant pressure to deliver a terminology suitable for use within their electronic medical record application. Additionally, this region wanted to enhance a portion of the terminology (the disease axis of SNOMED) that was being concurrently enhanced by two modelers in another of the Kaiser Permanente regions (Northern California), and also wanted to enhance a third portion of the terminology (the procedure axis of SNOMED) that was being concurrently enhanced by a modeler in a third Kaiser Permanente region (Southern California).

These three Kaiser Permanente regions provided the setting for evaluating whether the Galápagos applications would perform up to the standards necessary for routine use. For this evaluation, data were collected prospectively over a six month period (5/96-11/96). As this section will show, the typical number of conflicts identified did not impose an undue

4. This section was presented to the IMIA WG6 conference on Natural Language and Medical Concept Representation. Jacksonville, FL. January, 1997. A revised version of that presentation will appear in a special issue of *Methods of Information in Medicine*.

burden on the development process and discussions of the conflicting definitions actually helped the modelers to focus productively on modeling tasks. Finally, the evaluation showed the need to prevent a large number of trivial conflicts.

Five development cycles, beginning with distribution of a baseline terminology and culminating with the merging of changes from multiple site to generate a new baseline terminology, were completed during the data collection period. Each cycle had a merge process in which individual modelers' changes were imported into the configuration-management system, the resulting conflicts were identified and resolved, and a new baseline was generated. Table 6-1 presents the statistics for the individual merges that took place during the evaluation period. Merge 1 was particularly critical, and was closely scrutinized by Kaiser Permanente's CMT project management.

Table 6-1. Statistics for the 5 merges included in the 6 month evaluation period.

	Modelers	**Changes**	**MDT[a] Conflicts**	**Conflict Rate**
Merge 1	3	1,816	39	2.2%
Merge 2	2	1,174	17	1.5%
Merge 3	4	45,079	249	0.55%
Merge 4	4	4,647	895	19%
Merge 5	4	1,610	16	0.99%
Total		54,326	1,216	2.2%

a. MDT is an abbreviation for "multiply defined term." Multiply-defined-term conflicts are illustrated in Figure 4-2 on page 91.

Although the validation step for the Galápagos applications demonstrated that they could appropriately identify and allow modelers to resolve conflicts, some of the CMT managers were uncertain how the applications would perform in a setting where the modelers could make unconstrained changes (this limitation of the semantics-based concurrency control

proof-of-concept was discussed in the previous section). They required further convincing before they would allow complete reliance on the Galápagos applications. As an initial test, three of the Colorado modelers were instructed to work for three weeks on overlapping portions of the terminology.

Once the modelers had submitted their work developed using K-Rep DE, and Isabella[5] had imported those changes and identified the conflicts they concurrently created, the modelers met to discuss ways to resolve their conflicts. This first session was audio-taped. I moderated this first session, and IBM's K-Rep development manager also participated in the session.[6] When this conflict-resolution session was completed, and the conflicts were resolved, the modelers reported their experiences to their management.

The modelers had concurrently worked for three weeks on terminology enhancement, and during those three weeks created only 39 conflicts (2.2% of all transactions). They completed their editing on a Friday, and submitted their change-sets for importation into the configuration management database. The change-sets were imported over the weekend, and a conflict report was available for them Monday morning. On Monday morning, they met to discuss proper resolution of the conflicts, and that afternoon, the conflicts were resolved and a new baseline terminology was generated. Essentially, the modelers incurred one day of overhead in return for being able to work concurrently for three weeks. Kaiser Permanente's CMT Management found that this overhead was acceptable,

5. Isabella was previously described in Section 5.2.1.
6. A transcript from this session is presented in Appendix A, and an analysis of a portion of that session is also presented in Section 6.2.1.

and thus provided approval for reliance upon Galápagos for concurrency-control and configuration management.

As Table 6-1 indicates, there were 54,326 observed transactions during the entire 6 month evaluation. Overall, these transactions resulted in a multiply-defined-term conflict rate of 2.2%, which was again found to be acceptable by the Kaiser Permanente CMT project managers (coincidentally the same overall rate as the rate for the first merge, upon which reliance on the Galápagos applications was contingent). Merge 4, however, was an outlier with a 19% conflict rate. Examination of the transactions involved in this merge revealed that one of the modelers had imported a set of changes that was larger than is ordinarily made possible by manual editing of the terminology. Moreover, these changes affected a portion of the terminology where other modelers at other sites were actively making changes.

Following this merge, a policy was adopted of doing bulk imports only on baseline terminologies (before distribution), thus assuring that such large scale conflict rates would not be encountered in the future.[7]

In the following pages I present representative examples of multiply-defined-term conflicts encountered during the six month evaluation. These conflicts illustrate that multiply-defined-term conflicts can be further classified into semantically-conflicting changes (Figure 6-3) and semantically equivalent changes (Figure 6-4).

7. Section 6.2.2 discusses alternative methods for resolving such outliers efficiently.

Semantically-conflicting changes are the most interesting from the perspective of terminology modeling because such conflicts presage a lively debate surrounding the "true nature" of the world and the "proper" way it should be modeled. Figure 6-3 presents such an example. In this case, both modelers felt that the existing definition of FLEXION was incorrect (they both removed the defining concept MUSCULOSKELETAL-SYMPTOM), but they disagreed on the proper way to define FLEXION. Their debate concerning the "proper" way this concept should be modeled is presented in Section 6.2.1. Semantically-conflicting changes, however, may also be reflective of more mundane problems—either a simple mistake by one of the modelers or an incomplete definition by one or all of the terminology modelers involved in the conflict.

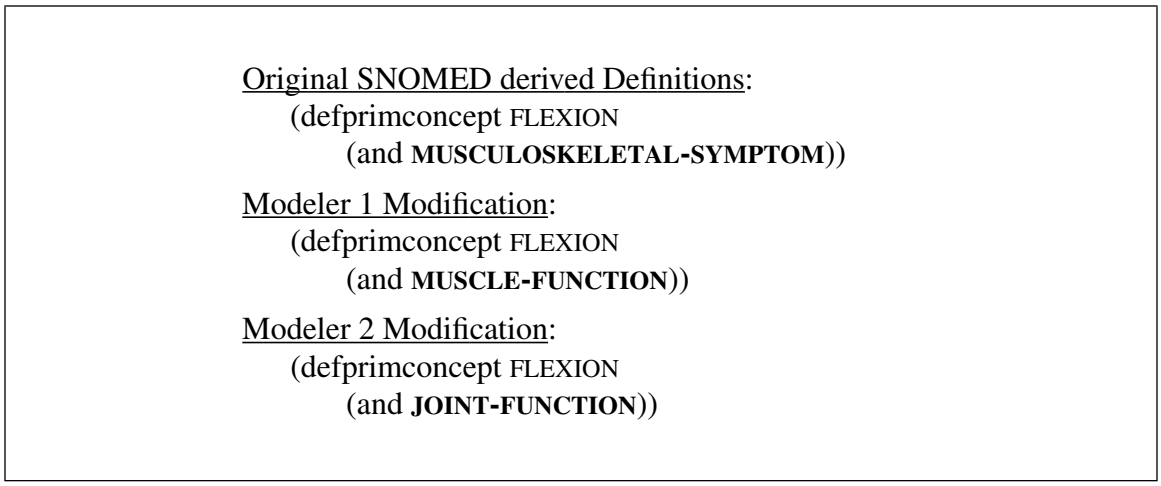


Figure 6-3. Semantically-conflicting changes from Kaiser Permanente development.

The semantically-equivalent conflict (Figure 6-4) illustrates the potential power of the underlying classifier to resolve some classes of conflicts on its own. When conflicting concepts are semantically equivalent, the underlying environment can either chose the simplest or the most comprehensive definition and thereby resolve the conflict with no human intervention. For this early work, however, in which the underlying terminology is imma-

ture and classification is often unpredictable due to mistakes in distant portions of the hierarchy, I and my collaborators chose to continue to resolve such conflicts manually.

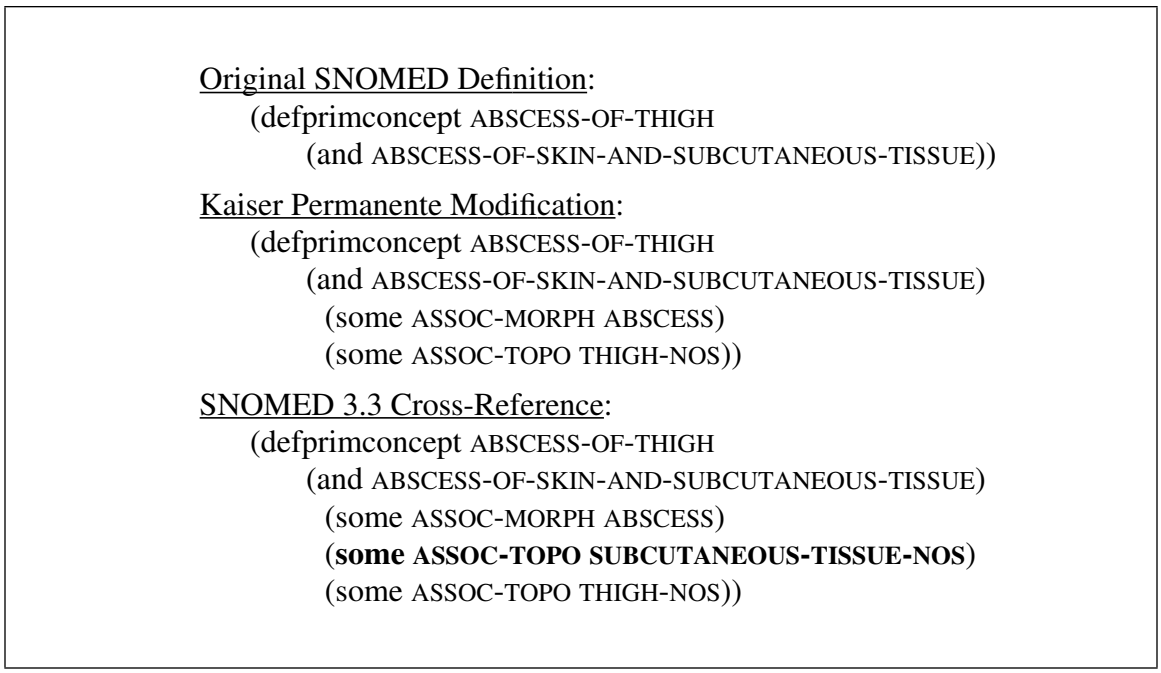


Figure 6-4. Semantically-equivalent changes. Although the definitions are different, the K-Rep classifier used inheritance to determine that the concepts are semantically equivalent. Such conflicts are candidates for automated conflict resolution.

The final class of conflict, the non-unique definition conflict, is illustrated in Figure 6-5. These are usually the result of incomplete definition of terms. In the case illustrated, the body site and root operation were imported from information encoded within the SNOMED procedure code. During the term of the evaluation, 12,944 such conflicts were identified. Nearly all of these (12,876) resulted from importing the body site and root operation from the SNOMED procedure codes, while a small number (68) were created by the modelers during their iterative refinement of existing definitions.

After Galápagos imported and classified each modelers' change sets, a conflict report was generated that listed all of the terms with multiple definitions and classified each pair of

```
(defconcept ARTHROSCOPY-OF-SHOULDER
    (and SHOULDER-AND-ARM-ENDOSCOPY)
        (some HAS-BODY-SITE UPPER-EXTREMITY)
        (some HAS-ROOT-OPERATION ENDOSCOPY))
(defconcept ARTHROSCOPY-OF-ELBOW
    (and SHOULDER-AND-ARM-ENDOSCOPY)
        (some HAS-BODY-SITE UPPER-EXTREMITY)
        (some HAS-ROOT-OPERATION ENDOSCOPY))
```

Figure 6-5. Non-unique definition conflicts. Two concepts with identical definitions. The relations for body site and root operation were imported directly from SNOMED.

definitions as semantically equivalent or as semantically conflicting. This report was then reviewed at a group meeting involving modelers who participated in the development process. Here the conflicts were discussed, and a consensus was sought to guide the resolution of the conflicts.

Typically, resolution of the conflicts identified after each merge cycle required an afternoon—a reasonable price to pay for the ability of the modelers to work in parallel. The conflict resolution process was not limited to the clerical work of identifying obvious solutions to conflicting definitions. In some cases, group discussions of the conflicting definitions led to an improved common understanding of the modeling task which was an overall benefit to the development process.

6.2.1 Conflict Resolution and Evolutionary Design

As was noted earlier, support for an evolutionary design by periodic reconciliation of the conflicts created through parallel local enhancement has been a fundamental principle of Galápagos. Although group discussions of conflicts do not always yield improved under-

standing of the design process, there are frequent cases where meaningful discussion is prompted by evaluation. Figure 6-3 represents such an example. It was the focus of the following discussion transcribed from the first conflict-resolution session whose participants included three of the CMT modelers, K-Rep's development manager, and myself. For this session, I acted as a moderator, trying to promote discussion among the modelers, yet also being careful not to impose my own vision of the modeling activity.

Two of the three CMT modelers had independently reviewed the original definition[8] for flexion, and thought they could improve it with their respective changes. They both agreed that the existing definition was incorrect as evidenced by their removal of the musculoskeletal symptom term. The "correct" solution was not immediately apparent, but through the process of discussing their modeling of flexion, they developed a new shared understanding of their modeling task which they subsequently applied to the similar conflicts created for extension, abduction, and adduction. A transcript of their discussion follows:[9]

> ***Modeler 2****: I think it is a musculoskeletal function, as I wrote, so really it is both.*
> ***Moderator****: So you would go back to a more general term [musculoskeletal function rather than either of the more specific terms joint function or muscle function].*
> ***Modeler 2****: If joint function is a musculoskeletal function and muscle function is a musculoskeletal function then I would categorize flexion separately under each.*
> ***Moderator****: Under both [muscle function and joint function]?*
> ***Modeler 1****: Separately under both?*
> ***Modeler 2****: Yes.*
> ***Modeler 3****: That's fine. Really it has very different meaning. Flexion is a muscle function.*
> ***Modeler 1****: Really the joint has the movement.*
> ***Moderator****: Isometric exercise involves muscle function with no joint movement.*

8. The original definition in this case was derived from the hierarchy embedded within the SNOMED termcode.
9. Notice that the discussion involves the entire group, not just the two modelers involved in creation of the conflict.

Modeler 2*: Flexion requires a joint and activity of a flexor.*
Modeler 1: *Now wait. It does not require [a flexor]. And remember this is just general flexion. It does not distinguish between active and passive. Active flexion requires the flexor.*
Modeler 3*: That's what I said. It could be intrinsic or extrinsic.*
Modeler 2: *The flexor could be your arm moving my arm.*
Modeler 1*: Exactly.*
Modeler 3: *Flexion is a joint function.*
Modeler 2*: But as far as the concept flexion, you'd agree that it is a musculoskeletal function?*
Modeler 3: *Well I think that you should make it...*
Modeler 2*: Most generally?*
Modeler 3: *Well most generally it's a body function, but more specifically it is a joint function.*
Modeler 2*: OK.*
Modeler 3: *I think it really is a joint function. The muscle function for instance... The muscle function of the extensor muscle in flexion is to relax. That's true, it is a muscle function, but its function is to relax in that case.*
Moderator*: But I would think it is a complex process involving both. If you have a frozen joint, it won't flex. If you have no motor control it won't...*
Modeler 1*: ACTIVELY flex.*
Modeler 3: *Well, but you are talking about active flexion.*
Modeler 1*: But we haven't represented with further specificity active and passive flexion.*

At this point in their discussion, the modelers reached a turning point in their collective understanding of the modeling problem. One of the modelers discovered that they were trying to assign properties to flexion that cannot be done properly without "representing with further specificity active and passive flexion." One way to represent the current collective understanding of the modeling problem is with an argumentation diagram (Streitz, Hannemann & Thüring, 1989). Figure 6-6 contains an argumentation diagram that presents the conflicting hypotheses under consideration and the evidenciary arguments supporting and refuting those hypotheses along with the two proposed classifications for flexion.[10]

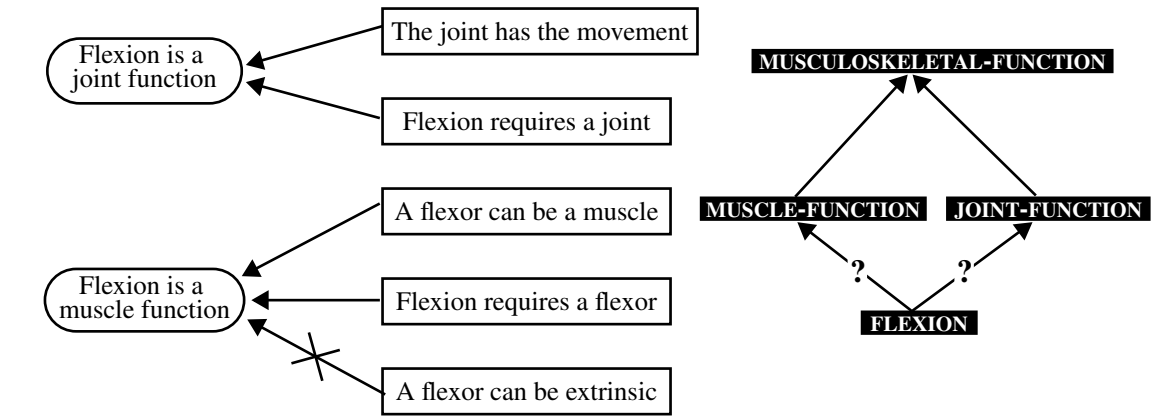


Figure 6-6. Argumentation diagram of arguments and conflicting hypotheses and the conflicting classifications for flexion. Arrows indicate arguments supporting a hypothesis. The arrow with an X indicates an argument refuting a hypothesis.

As the diagram clearly shows, the evidenciary arguments cannot be reconciled with the existing hypothesis that flexion is either a muscle function, a joint function, or both. As the discussion continued, the modelers revised their conflicting hypotheses to satisfy all the evidenciary arguments:

Modeler 2*: So you [Modeler 3], would categorize active flexion under muscle function*
Modeler 1*: And joint function*
Modeler 2*: And passive flexion under flexion and nothing else.*
Modeler 3*: I would say that flexion is something done to a joint by an actor. In active flexion the actor is the muscle that crosses that joint. In passive flexion the actor is extrinsic. Is environmental.*
Modeler 2: *OK. So, Flexion is a joint function?*
Modeler 3*: Flexion and extension and rotation...*
Modeler 2: *So we take the 3rd option here.We delete the muscle function as a parent.*
Modeler 1*: But what if we add a concept called active flexion?*

10. Some researchers (Cavalli-Sforza, Weiner & Lesgold, 1994; Suthers, Weiner, Connelly & Paolucci, 1995) are exploring how collaborative software can help to develop scientific thinking skills by allowing students to develop argumentation diagrams collaboratively, similar to the ones presented here, that illustrate their collective understanding of assigned scientific questions, as well as provide a structured approach to examining the validity of a hypothesis. Such approaches may be adaptable for collaborative conflict resolution.

Modeler 3: *Yes active flexion has a joint that is flexed but it also has a muscle group that is the performer of the flexion.*
Modeler 1*: Well [the muscle group] is used as an effector.*
Modeler 2*: So it will have two parents. It will have a parent that is a muscle function and a parent that is a joint function.*
Moderator*: So for flexion we will preserve it as a joint function but it would be a good idea to add active flexion... Well maybe this should be Flexion, NOS and we should add Active Flexion, NOS and Passive Flexion, NOS.*
Modeler 2*: Right, so we are going to have to add some concepts.*
Modeler 1*: Well, they may already be there. We just haven't looked.*
Modeler 2*: Yes, we need to look.*
Modeler 1: *But we still have to decide if the relationships should be is-a relationships vs. a role relationship has-effector flexion.*
Modeler 2*: I think based on our model it's an is-a. Flexion is-a joint function.*
Modeler 1*: Yes. But we are talking about active and passive.*
Moderator*: Well, you would say active flexion is-a joint function and is-a muscle function.*
Modeler 2*: Or active flexion is-a flexion.*
Moderator*: Yes it is-a flexion.*
Modeler 2*: And it is-a muscle-function.*
Moderator*: So then it would inherit the joint function.*
Modeler 1: *Right.*
Modeler 2*: Uh, OK.*

The modelers finally reached consensus. Figure 6-7 contains an argumentation diagram that presents the new hypotheses under consideration and the evidenciary arguments supporting them. It also contains the new classifications for flexion as well as the new concepts (active flexion and passive flexion) introduced by the revised hypotheses.

The conflict created by the different modeling of flexion was arguably one of incomplete understanding on the part of the modelers, and the discussion of the conflict became a group learning session in which the modelers reached an improved, shared understanding. In some cases, however, conflicts are created by differing perspectives rather than incomplete understanding of the modeling task. Consider the example in Figure 6-8, in which a conflict was created by differences between the Kaiser Permanente and SNOMED defini-

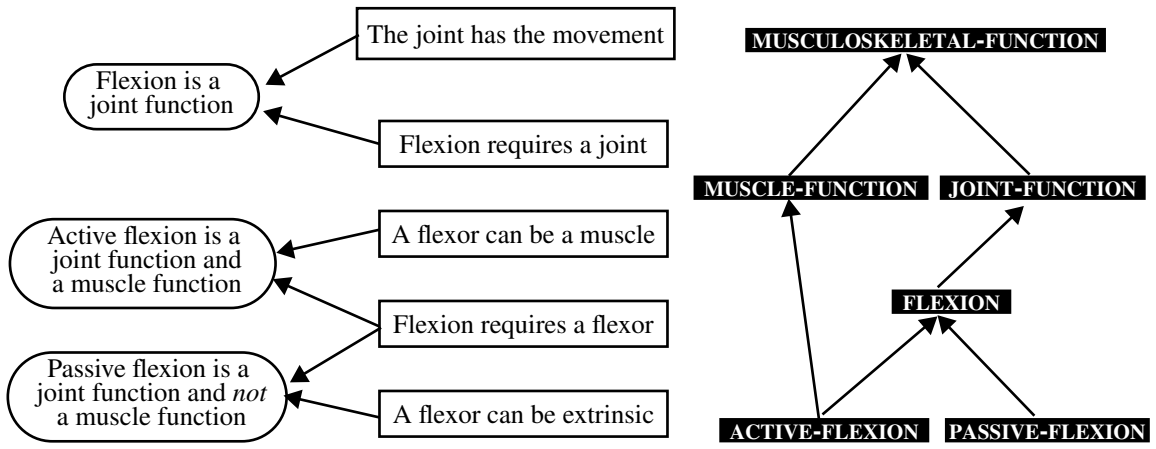


Figure 6-7. Diagram of arguments and revised hypothesis and newly agreed classification.

tions of the term Cellulitis of skin with lymphangitis, NOS. The Kaiser Permanente modelers were all ambulatory care physicians whereas the SNOMED crossreferences were created by pathologists.

On initial examination of the conflicts, it seemed that a simple merging of the definitions would be appropriate. Conversations with a few pathologists, however, indicated that they focused on the morphologic changes of cellulitis that take place completely below the dermis, whereas from an ambulatory care perspective, cellulitis is diagnosable by surface features of the skin such as erythema and induration.

When groups with fundamentally different perspectives are involved in the modeling tasks, there is a risk of an irreconcilable difference of perspectives. Hopefully the perspectives common to health care will be sufficiently congruent so that consensus can be readily achieved. Within the Kaiser Permanente project, such consensus is not always immediate, but we have never found an irreconcilable conflict.

Original Definition:

(defprimconcept CELLULITIS-OF-SKIN-WITH-LYMPHANGITIS-NOS
(and INFECTION-OF-THE-SKIN-AND-SUBCUTANEOUS-TISSUE-NOS))

Kaiser Permanente Modification:

(defprimconcept CELLULITIS-OF-SKIN-WITH-LYMPHANGITIS
(and INFECTION-OF-THE-SKIN-AND-SUBCUTANEOUS-TISSUE-NOS
LYMPHANGITIS
(some ASSOC-MORPH CELLULITIS-NOS)
(some **ASSOC-TOPO SKIN-NOS**))

SNOMED 3.3 Cross-Reference:

(defprimconcept CELLULITIS-OF-SKIN-WITH-LYMPHANGITIS
(and INFECTION-OF-THE-SKIN-AND-SUBCUTANEOUS-TISSUE-NOS
(some ASSOC-MORPH CELLULITIS-NOS)
(some **ASSOC-TOPO SUBCUTANEOUS-TISSUE-NOS**)
(some **ASSOC-TOPO LYMPHATIC-VESSEL-NOS**))

Figure 6-8. Semantically conflicting changes between Kaiser Permanente and SNOMED 3.3.

Allowing all the modelers to participate in some form in the resolution of conflicts proved to be essential for maintaining group cohesion. There was a short period during the evaluation when one of the regions did not communicate the reasoning behind the conflict-resolution decisions to all of the participating modelers, and one of them noticed that his changes were not being incorporated into the new baseline terminology.[11] This modeler chose to withhold his work until he could be assured that his changes would not be discarded without involving him in the process.

11. Although conflict-resolution sessions were intended to involve all modelers, geographic separation (the modelers were located in California and Colorado) and local delivery pressures conspired to allow a few merges where only the modelers at one of the sites determined how the conflicts would be resolved. Subsequently, improved communication has ensured appropriate input from all modelers during the merge cycles.

This reluctance of modelers to participate in a process in which their work could be arbitrarily discarded when it conflicted with the work of others has been anticipated (Campbell, 1994), and it was identified as one of the reasons that traditional concurrency-control methods that used procedural rather than semantic criteria to resolve conflicts would prove unacceptable for the development of distributed terminology.

6.2.2 Future Research on Conflict-Resolution Methods

As was previously noted, one outlier merge during the evaluation period had a conflict rate of 19%. Resolving these conflicts created a significant cost, offsetting much of the benefit of the distributed development for that merge.

Analysis of actions that led up to this merge revealed that one of the modelers had imported an ICD-9-CM code into virtually every definition for a SNOMED diagnosis term. This bulk importation assured that a conflict would be created for any change to a diagnosis term until the ICD-9-CM changes were incorporated into a new baseline and disseminated to all modelers. Unfortunately the conflict resolution environment had no method for distinguishing these kinds of conflicts from potentially more interesting ones and sorting out the latter for discussion, thereby compounding the costs of resolution.

In order to improve the efficiency of the development process, methods are needed to allow the conflict resolution environment to distinguish reliably between what might be called “uninteresting” and “interesting” conflicts, as well as further definition of methods that can dependably resolve the former while reserving the latter for group discussion.

Experience gained from dealing with conflicts created by the bulk importation of ICD-9-CM codes indicates one promising direction for future research on this problem. The solution might involve an editing environment in which each editor or process can be declared "authoritative" with respect to a particular component of a definition or a specific taxonomy within the terminology. In cases like the bulk import of the ICD-9-CM codes, the import process could be declared authoritative with respect to these codes, and unless the other modelers who participated in the conflict made a change *within the authoritative domain of another*, the conflict would be considered "uninteresting" and subject to automatic resolution by merging the definitions with conflicting changes from each of the modelers.[12] If a modeler in this proposed editing environment made a change that overlapped with the authoritative domain of another modeler, however, such a conflict would be classified as "interesting," and put up for group discussion.

6.3 Conclusion

The Galápagos environment has demonstrated the viability of semantics-based concurrency control for terminology development. It continues to provide ongoing support at Kaiser Permanente for managing the inevitable conflicts created by concurrent development of enhancements to its terminology, and it is being utilized in increasing portions of Kaiser Permanente's CMT development.[13] Evaluation of Gálapagos has also identified some limitations of the approach, along with potential solutions. Chapter 7 discusses further implications, limitations, and generalizability of the Galápagos approach.

12. Had this been done with the ICD-9-CM codes, most of the conflicts would have been automatically resolved, since none of the changes made by the individual modelers involved a change to a code.

13. The Kaiser Permanente CMT project has evolved from a collaborative project between three regions (Colorado, Northern California, and Southern California) seeking a common solution to their terminology problems, to a enterprise-wide requirement for all clinical information systems. When the CMT project is fully deployed, all 12 Kaiser Permanente regions (the three regions previously mentioned as well as Hawaii, Ohio, Oregon/Washington, Texas, Connecticut/New York/Massachusetts, North Carolina, Georgia, Kansas/Missouri and the Washington DC/Virginia/Maryland regions) will participate either by continuing development of the terminology, or by deployment of the terminology within regional systems.

Chapter 7
Conclusion

In this dissertation, I have discussed the principal challenges facing the representation of clinical data, from the formalisms required to define the terminology to the principles of configuration management needed to support the development process. I have also described and evaluated potential solutions for many of these challenges. Section 7.1 presents a review of the components of this dissertation and discusses the generalizability and limitations of the solutions that have been proposed.

My work has also been situated within an ongoing terminology development project, and as such, it has afforded me with an opportunity to learn many pragmatic lessons relating to terminology development. Section 7.2 presents some ancillary lessons that were learned from this project, but that were not directly related to the thesis of this dissertation.

Although the Galápagos experiment demonstrated the viability of semantics-based concurrency control for terminology development, it represents only a beginning. There are many extensions that can make distributed development more productive and reliable.

Section 7.3 describes opportunities for future work that can expand upon and strengthen the underlying framework described here.

Sections 7.4 and 7.5 summarize the contributions of this dissertation and provide some concluding remarks.

7.1 Generalizability and Limitations

This dissertation has been grounded in a framework for formalizing the representation of medical concepts using description logic. It has described and evaluated a process to make evolutionary enhancements of SNOMED International, an existing medical terminology system, within a description logic model. The combination of SNOMED and description logic provide a declarative representation for sharing medical knowledge, the need for which has been discussed by Shwe and colleagues (1992), and by Musen (1992).

This description-logic formalism provided a unifying framework that can be broadly applied to several different practical applications and research areas. This approach, however, has important limitations. Costs are associated with the expressive power provided by complete first-order logic, and the foundational models upon which the logical representations are built also have limitations. Moreover, a formal representation for clinical data is just a first step. Somehow clinical data must be collected from real-world situations and transformed into the formalisms required by automated processing. This is a challenging undertaking.

Any comprehensive attempt to provide an infrastructure for developing standardized clinical terminologies must also facilitate collaborative activity among terminology modelers who may be working at a variety of geographically dispersed institutions. This dissertation has described concurrency-control and configuration management-strategies that can support distributed development of a CMT. During the development process, individual modelers may make incompatible changes to both the content and the structure of the CMT. Chapter 4 has presented strategies for managing concurrent development, including mechanisms for resolving conflicts that may arise during concurrent development. The limitations and general applicability of these strategies for detection and resolution of conflicts must be understood, as must be the limitations of the configuration-management strategy.

7.1.1 Logic-Based Approach

Chapter 3 recommended the use of description logic for representation of the logical relationships among the terms used to represent medical concepts. It also described a particular syntax for this logic: KRSS (Patel-Schneider et al., 1993). Although use of some form of logic is essential for representing medical concept descriptions, the KRSS notation is only one of several equivalent formalisms that are available. Medical concepts can also be represented using either the common algebraic notation for logical propositions or any of the popular knowledge-representation languages, although representational diversity is inherently problematic.

Fortunately, researchers in the computer-science community are working to overcome the problems of multiple syntaxes competing for knowledge representation. Members of the Defense Advanced Projects Research Agency (DARPA)-sponsored knowledge-sharing

project (Neches et al., 1991) have created the Knowledge Interchange Format (KIF) (Genesereth & Fikes, 1992) for interconverting knowledge bases created in a variety of representations. KIF thus promises to allow medical concepts described using a combination of SNOMED and description logic to be translated directly into a knowledge-representation system such as KL-ONE, although a proof-of-performance is not yet available.

The flexibility to translate among the members of a family of related syntaxes is critical. Whereas description logic captures the full semantics of first-order logic—and thus offers enormous expressive power—proving theorems in first-order logic is, in the worst case, computationally intractable. Consequently, developers of knowledge-representation systems must decide which elements of first-order logic they may safely omit to improve runtime performance (Levesque & Brachman, 1985). Common object-oriented languages, for example, do not allow the expression of concepts such as negation or disjunction. No practical knowledge-representation system is perfect; each embodies a particular set of tradeoffs. The approach adopted under this thesis has been to use complete first-order logic (as reflected in the syntax of description logic) for *canonical* representation of clinical concepts and to translate those representations into the knowledge-representation systems that are most suitable for individual applications.

7.1.2 Foundational Models

Because logic is topic neutral, the choice of description logic as a method for medical-concept representation does not imply any particular model of the medical domain. Implicit in any logic-based representation is a number of *foundational assumptions* detailing how logical relationships are to be applied. To use a representation consistently, these assump-

tions must be made explicit as *foundational models* upon which the representation depends. Medicine provides some useful foundational models (i.e. pharmacology, anatomy, and physiology) for capturing the rich semantics of clinical data and formalizing clinical concepts and their relationships. Although an existing medical nomenclature system (such as SNOMED) can provide a set of labels and defined relationships that can serve as the basis for a comprehensive terminology, the foundational models embodied in such nomenclatures are limited either in scope or by being implicitly encoded. Developers of logic-based conceptual models, therefore, face the challenge of formalizing such foundational models.

Determining whether a foundational model is sufficiently expressive for encoding clinical findings requires empirical evaluation. Some foundational models, such as temporal data models for clinical databases (Das, Tu, Purcell & Musen, 1992), may already be implemented in data-management systems. In such cases, the limitations of the foundational models may be well understood. In other cases (such as the model of anatomy in GALEN (Rector, Nowlan & Glowinski, 1993)), the appropriateness of such models cannot be empirically evaluated until the models are used for data acquisition or encoding. To establish the suite of foundational models necessary for a logic-based clinical representation, modelers will therefore need to (1) incorporate current or developing formal models of specific topics whenever possible, (2) extend or combine these models as necessary on an evolutionary basis, and (3) instantiate the concepts and relations in these models with standardized codes from existing medical nomenclatures.

Galápagos has met the proof-of-performance criteria necessary to support such evolutionary enhancement of standard reference terminologies on a national scale, and promises to provide a pragmatic solution to the need to enhance reference terminologies locally while synchronizing these changes with an evolving standard of reference.

7.1.3 Evolutionary Enhancement

The evolution of a standard representation for clinical data produces a challenging management task. The work must be guided by empirical studies of real-world applications that may require local enhancements if they are to function properly. If these enhancements are useful, they should eventually be incorporated into the standard so that other applications can use them. The technical and managerial aspects of how this process should be managed are complex.

Tuttle and colleagues (Tuttle et al., 1991) have described how developers incur a penalty for creating local enhancements. This occurs when developers try to synchronize their enhancements with similar changes that may be incorporated into the Unified Medical Language System (Lindberg et al., 1993). If participation by more than one institution in the development of the standard is desired, a way must be found to incorporate local enhancements from multiple sites without penalizing institutions for their efforts.

Concurrency-control models of advanced database applications (Barghouti & Kaiser, 1991) can be used to manage local enhancements so that the penalty created by local enhancements is reduced. The solution proposed in this dissertation is based on *change-sets* (see Chapter 4). The dissertation has presented an empirical proof-of-performance

that validates the Galápagos methodologies, showing that change-sets are an appropriate paradigm for managing CMT development.

There is a potentially large benefit for using tools that implement a proper concurrency-control model. If tools for managing evolutionary enhancement are developed, national participation in the evolution of a standard for representation of clinical data may become a reality. In the absence of such a concurrency-control model, however, the problems of managing local enhancements will discourage participation by all but the most determined developers.

7.1.4 Domain-Specific Conflict Detection and Resolution

The dissertation has presented methods for detecting and resolving conflicts created during CMT development. The proposed methods for conflict resolution are domain-specific, but detecting conflicts and finding mechanisms to resolve them is a general problem. Any application that allows distributed development, such as software engineering tools, computer-aided-design (CAD) tools, or distributed-document-creation tools, must have a way of handling conflict detection and resolution.

Hall (1991) discussed how current CAD environments do not support conflict detection and resolution. He proposed that next-generation CAD environments have a *conflict record* in which developers would manually enter conflicts they discover. This record would be used to notify the affected parties who would try to resolve the conflict by altering the affected designs. The resolution process is manual, and Hall did not propose building an understanding of design semantics into next-generation CAD environments.

Other researchers have developed applications to detect conflicts that occur when integrating several versions of a computer program. Most integration techniques are variants of text-based differential file comparators such as provided by the UNIX tool "diff." Horwitz and colleagues (1989) have discussed the limitations of this approach and have proposed a semantics-based tool for automatically integrating program versions. The tool they described integrates noninterference versions of programs using domain-specific knowledge, either to determine a proper merge or to show the existence of a conflict. The tool would not try to resolve the conflicts it identifies. Horwitz and colleagues did, however, describe the need for semiautomatic, interactive integration tools to make their work applicable on a practical basis. Reps and Bricker (1989) have proposed methods that an interactive tool could use to illustrate these conflicts for the user. Rhabida (see Section 5.2.4) is an example of such a tool.

Westfechtel (1991) tried an alternative approach to that of Horwitz and colleagues. He described a structure-oriented merge tool that can be applied to software documentation written in arbitrary languages. The language-independence of this approach is an advantage; however, Westfechtel readily admitted that language-specific approaches (such as (Horwitz et al., 1989)) would allow more intelligent merge decisions. He also correctly pointed out that practical tools relying on the semantically based approaches (of which the prototype tools developed for this dissertation are examples) are not yet generally available.

Galápagos' tools for conflict identification and resolution are based upon description logic but are not dependent upon methods specific to any particular dialect or syntax of descrip-

tion logic. That is to say, the conflict-identification methodology in Galápagos is *syntax* independent and, therefore, general for description-logic applications. Moreover, instead of taking a language-specific approach to allow more intelligent merge decisions, Galápagos uses a domain-specific strategy for conflict identification and resolution. It focuses on potential conflicts detected during the *classification* of terminological definitions.

These kinds of domain-specific strategies for conflict identification and resolution are being developed for a variety of applications that require distributed development. As techniques in each domain improve, related domains are bound to benefit. The three fields discussed here (CAD, software development, and CMT development) are closely related because of the kinds of problems they encounter and their pursuit of automated tools to solve them. Configuration management, which will be discussed next, is another problem common to these three fields.

7.1.5 Configuration Management: No Free Lunch

Chapter 4 described how custom change sets can minimize the local-update penalty. These change sets can be generated automatically as a by-product of merging the local work of multiple developers. They are not, however, entirely cost free. A merge process at a central location, including manual review of any conflicts, is required to make use of the local enhancements. Only after conflicts have been resolved can the automatically generated change sets be created.

In addition to the cost of resolving conflicts during the merge process, there are also local costs. Although the local sites are given a set of changes to update their representation

with the new reference terminology, these changes may result in unwanted side effects. The changes must be reviewed to determine what impact, if any, they will have on local applications. If the changes do affect local applications, modifying these applications to perform properly with the updated terminology may result in considerable costs.

Finally, all parties have to agree on a common representation scheme that may not be optimal for their individual applications. The costs associated with trying to use terminology developed for one purpose in an application with a different purpose may be overwhelming.

7.1.6 Evaluation

I have consistently sought to provide a better characterization of the development process of CMTs and to determine if the methodologies embodied within the Galápagos applications perform reliably in a real world setting. Although the evaluation successfully demonstrated the viability of semantics-based concurrency control for distributed terminology development in a real-world setting that provided real stresses to the computer applications and to the development process and product, it did not take place in a controlled setting where only a single variable could be altered and the results studied as an isolated issue separate from the total process.

Just as there is a continuum of expressivity–tractability for description-logic implementations, there is also a real world–controlled environment continuum for evaluations. My system evaluation clearly lies on the real world side of the continuum. I anticipated that a controlled, randomized, statistical methodology would not lend itself to a comparison of

productivity before and after the implementation of distributed development tools, and the applied project setting further made such an approach unrealistic. Therefore, I adopted a proof-of-performance methodology (National Research Council, 1994).

Finally, although the proof-of-performance demonstrated that the Galápagos tools could successfully meet the current needs of Kaiser Permanente's convergent medical terminology project, further information is needed on the limitations of the approach. How will the system scale from a high volume use of a focused development tool used by a few modelers up to a low volume, distributed maintenance tool used by many modelers? What is the maximum number of configurations that can be practically supported? When does conflict-resolution stop being a productive part of model development and turn instead into a developmental burden? Answers to these and other questions must be sought as part of future evaluations of extensions to the Galápagos environment.

7.2 Ancillary Lessons

A proof-of-performance test validates the underlying methodology and implementation of an idea, but often important (but not immediately obvious) lessons are learned in large projects that are critical to successful implementation of an idea. Here are a few of those lessons I have drawn from my work.

7.2.1 Clearly Define the Boundaries of Collaboration

The general notion of "collaboration to solve common problems" is an idea that can derive universal support. However, translating consensus around a common theme to an actual

project with real commitment of resources requires more than best wishes. Collaborations with commercial implications are often difficult to foster. In such collaborations, concerns regarding giving a competitive advantage to another party, developing a dependency upon a specific commercial product, or provoking potential competition between the work products of the collaboration and related commercial products of a collaborator, may prove overwhelming.

Within Kaiser Permanente, "enterprise-wide aggregation of clinical data" has been an explicit goal for its electronic medical record projects. Thus, almost without exception, projects readily agreed to the principle of collaboration on a CMT that will be common to Kaiser Permanente's electronic medical record projects.

Although projects may agree in principle to collaborate, the same projects are also competing with one another in several settings: the vendors associated with the projects (which vary among the Kaiser Permanente sites) have a commercial interest in maximizing their return (and no interest in giving proprietary or competitive advantage to a competing vendor), and the Kaiser Permanente project sponsors have a professional interest in demonstrating that their recommendation to invest in a particular product is sound. The Kaiser Permanente CMT project has struggled with these issues, and has fortunately been able to manage them appropriately so that the collaboration has been able to continue, while commercial implications are being dealt with on an ongoing basis.

This eclectic mix of Kaiser Permanente regional participants and competing electronic medical record projects has provided a challenging environment for fostering collaboration. Despite the desire for a common CMT standard (the ability to aggregate data was a

feature claimed by all the medical record projects), there was considerable distress over the potential flow of confidential and proprietary information from one vendor to another as an inadvertent byproduct of the collaboration.

To further confound efforts at developing consensus surrounding the CMT project, all the participants recognized the close intertwining between the functionality of specific applications (e.g., decision support, quality assurance, medical research, and data acquisition applications as discussed in Section 1.1), and the functionality of the developing CMT. It become clear that the collaborative needed to determine where the CMT project should end and where the proprietary enhancing aspects of application development should begin.

It proved helpful to focus on the original foundation for the Kaiser Permanente CMT project: enterprise-wide aggregation of clinical data. Project leaders and team members began to refer to a terminology optimized for data aggregation and retrieval as a *reference terminology*, and agreed that this terminology would form the basis of the collaboration. Terminology features beyond the scope of relationships needed to define and aggregate terms were described as *application-specific enhancements* to the reference terminology and were excluded from the collaboration. Examples of application-specific enhancements from Kaiser Permanente perspective included: structured-data-entry constraints (CHEST-PAIN may be described as MILD, MODERATE, or SEVERE), drug-disease interactions (beta-blockers are contraindicated in patients with asthma), recommended treatment protocols (asthmatic patients should be treated with an inhaled steroid), and many others.

Although the separation of the reference terminology from the application specific enhancements is now a foundational principle of the CMT project, such was not always

the case. In the beginning, the CMT project was often criticized for not having many of the features that are now explicitly excluded, the most troublesome feature being the structure-data-entry constraints. Many project critics could not understand how the CMT project could be of use if it was not ready for implementation in a medical record system "right out of the box." However, some vendors have invested significant resources in developing their own structured-data-entry constraints which they believe provide them with competitive advantages, and so they would be reluctant to participate in a collaborative project that they felt would undermine their very livelihood. Eliminating the structured-data-entry constraints from the CMT project was an essential component to survival of the collaboration.

This finding contrasts with the experience of the Galen project, in which a primary focus of the collaboration is allowing the system to generate "sensible combinations" of things to say; the resulting terminology thus can be directly used as an engine to drive user interfaces (Rector et al., 1993).

Discovering the principles on which the boundaries of the collaboration could be described was difficult, but once found, served as a reorienting principle that was frequently revisited both when trying to ascertain the appropriateness of modeling decisions, as well as when trying to describe the boundaries of the project to ourselves, to Kaiser Permanente's EMR vendor/partners, and to interested third parties.

7.2.2 Limit Distribution of Errors of Commission

Any sizeable terminological system will have defects that can be categorized into two general classes: errors of commission and errors of omission.

An error of commission is created by making an incorrect statement about a concept. For example, stating that the NOSE is attached to the FOOT would be an error of commission. Such errors may be obvious (apparent by examining a concept's definition), and if found may become the focus of attention until they are resolved.

An error of omission is created by failing to make a correct statement about a concept that would have related that concept to another concept within the terminology according to agreed-upon principles of representation. Failing to state that VASCULITIS-OF-THE-SKIN is a type of VASCULITIS, for example, is an error of omission. Such errors may be subtle and, therefore, easily missed. They cannot be found just by examining a concept's definition. The definitions of intermediate concepts that may inherit the properties in question need to be examined before one can definitively conclude that a necessary relationship is missing.

Both errors of omission and errors of commission will prevent the terminology from functioning as intended. A query using a concept that is defined with a definition that includes an error of commission will retrieve records other than those intended by the user. Alternatively, a query using a concept that is defined with a definition that includes an error of omission will fail to retrieve all the records of interest. Similar failures will be encountered by decision-support and user-input systems.

Because the SNOMED terminology, which formed the foundation for this project, had significant errors of omission, Kaiser Permanente chose to use hierarchical relationships based upon SNOMED's termcodes combined with lexically-generated relationships as a pragmatic first step toward improving the terminology (Campbell, Cohn, Chute, Rennels & Shortliffe, 1996). After they were generated, they were reviewed by the modelers and then imported into the baseline terminology. Despite the review for the lexically derived relationships, many errors of commission were introduced. Some were introduced secondary to using synonyms that actually had slightly different meanings to derive lexical relationships. Most were introduced secondary to problems encountered algorithmically in deciphering the hierarchies built into the SNOMED termcodes.

Although this process eliminated many of the errors of omission, the errors of commission became the focus of the initial users of the system, and resulted in one group abandoning the baseline terminology in favor of their own baseline. This new baseline arguably had a greater number of errors of omission than the original baseline's errors of commission. Such a diversion of resources and effort could have been prevented if the additional relationships were introduced slowly over time with additional quality assurance steps. Such a process would allow an opportunity for individuals committed to removing the errors of commission—yet who also understand the importance of preventing errors of omission—to review the terminology. It would also shield the errors of commission from individuals who are perhaps less tolerant of these errors and who might not understand the importance of avoiding errors of omission.

7.2.3 Independently Edit Top-Level Hierarchy

The number of errors in concepts at the bottom levels of the hierarchy was magnified as an indirect result of the inheritance of these errors by all of the dependent children and the increased complexity of the classification algorithms. It was found that committing a single change to a top level concept that had perhaps 10,000 dependencies could take up to an hour to process—hardly what could be called "interactive."

Many of the early problems of productivity could have been avoided if the Kaiser Permanente project had focused first on editing a smaller hierarchy (containing only the top level concepts) and had introduced additional dependent children only after the group agreed upon the base hierarchy's appropriateness.

Initially, this focused refinement was not a practical option since the K-Rep tool provided no mechanism to extract a small portion of the hierarchy for focused editing. Kaiser Permanente subsequently developed such extraction tools, but developers of similar terminology management environments would be well advised that they should provide tools to extract portions of the terminology together with all of the dependencies of that portion so that they can be independently loaded into the system and modified. These tools will also need to provide a mechanism to synchronize changes with the original terminology source.

7.2.4 Meaningless Identifiers

The use of meaningless identifiers is a foundational principle of well-designed systems such as the UMLS (Lindberg et al., 1993). Unfortunately, many terminological systems

insist on putting meaning in their termcodes (Côté et al., 1993; National Center for Health Statistics, 1995).[1] Although the Kaiser Permanente project sought to represent all meanings through explicit, description logic statements about each term in the terminology, it was still necessary to place meaning into concept identifiers used by the system due to the limitations of the K-Rep application (Mays, Dionne & Weida, 1991). K-Rep had no ability to chose a secondary field as the "display name" that would be presented to the user for manipulating the concept. To compensate for this limitation, Kaiser Permanente developed a naming scheme that used the initial preferred name of a concept with dashes replacing spaces as a mnemonic for the modelers and then appended an underscore followed by the SNOMED termcode to ensure uniqueness of the concept identifier:

Chest-pain-NOS_F-37000

Such changes were frequent and repeated, despite explicit instructions not to change a concept identifier after its initial creation because its mnemonic portion had a spelling error or because a modeler desired a different capitalization. Examples of such changes, if they had been applied to the "chest pain" term, would typically include:

Chest-Pain-NOS_F-37000

Chest-Pain_F-37000

Chest-pain_F-37000

The inability to force compliance with naming principles was definitely a shortcoming of the paradigm of central coordination *local* control. Modelers insisted that they were

1. Hierarchical relationships are examples of common meaning represented in termcodes, however many systems also represent additional meaning by reserving certain digits to represent anatomic sites affected by a disease, to represent the instrument used in an operation, and to represent many other facts about a concept.

required to make the changes to meet the quality standards of their local EMR projects, despite the fact that these concept identifiers would never be seen by end users of the applications.[2] These unnecessary changes compounded the difficulty of coordinating work between the regions because they invalidated mapping tables that were keyed off of what were supposed to be unchanging concept identifiers.

The need for such unnecessary changes can be eliminated if CMT development applications would support *indirection*—in which the system would use an internal symbol for manipulation (a unchangeable meaningless identifier), while externally displaying an alternative representation to the user (a display name). The ability to support such indirection is a requirement of all source-code debuggers (they display the source code of an application instead of the binary machine code which was derived from the source code). Such support for indirection should be part of all description-logic development systems intended to scale beyond a small terminology and a single developer.

7.2.5 Spelling Correction

Although the problems created by changing the names of misspelled mnemonic identifiers can be eliminated if the CMT system supports indirection, similar spelling problems were found in the display names and synonyms associated with given terms. Spelling correctors are found in virtually all word processor systems with good reason: spelling mistakes are common. Support for spelling correction should be built into CMT development systems, so that mistakes can be prevented.

---

2. End users would only see appropriate “display-name” or “synonym” facets associated with the concept identifier, and no restrictions had ever been placed upon the spelling and capitalizing of these concepts.

7.3 Future Work

Development of a sound methodology for distributed terminology development is a first step toward achievement of a convergent terminology. Assuring that a methodology is sound and that it can perform as intended in a real-world deployment, however, does not ensure optimality. The existing Galápagos tools also provide a supportive foundation for alternative paradigms of development. Additional extensions to these tools can support bidirectional transfer of change sets between terminology modelers who are working on overlapping—but non-identical—terminological models. This would be possible while allowing the modelers to retain the distinctness of their own models in configuration management databases.

7.3.1 Alternative Development Paradigms

Kaiser Permanente used Galápagos within a developmental paradigm of *centralized coordination local control* although other paradigms are also possible. Two of the alternatives are *centralized coordination and control* and *local coordination and control*. Each of these scenarios provides for very different roles for the central coordination body, ranging from absolute control to just observance, and for the local development sites, ranging from being completely autonomous to only doing as the central body directs. The same processes of managing configuration of change sets and conflict resolution evaluated for the *centralized coordination local contro*l paradigm can be directly applied to these alternatives.

Centralized Coordination and Control

As its name implies, this paradigm requires strong central coordination and control. There are many ways such a central body could be formed. A governmental agency, such as the National Library of Medicine or the Department of Health and Human Services, might receive a congressional mandate to develop and maintain a national CMT intended to meet the needs of anyone requiring encoded clinical data, including hospitals, commercial information system vendors, and governmental agencies. The funding for this mandate might consist of intramural funding for coordination and control of the development process and extramural funding for subcontractors.

Under this paradigm, the central organization forms a steering committee that includes representatives from the potential users of the CMT such as physicians, nurses, hospital administrators, insurance administrators, and health policy analysts. This steering committee is responsible for defining the objectives of the CMT, and for setting priorities for developmental tasks. After the committee defines priorities, a dedicated team within the central organization assumes responsibility for implementing those recommendations. This team breaks the tasks into logical units and assigns them to subcontractors. Periodically, the team collects the subcontractors' work, merges it into a new reference version, and then starts the cycle over again.

Figure 7-1 illustrates this development process. In this illustration, A, B, C, and D, are sites managed by subcontractors. Each site starts with the reference version (version 1), and creates *version branches* by making local changes. Within each branch, there may be sequential *revisions*, reflecting the local changes to the CMT. Finally, all local changes are

merged by the central organization to create a new reference version (version 2). Once the new reference version is created, it becomes the new base set for local work, new tasks are assigned to the subcontractors, and the cycle is repeated.

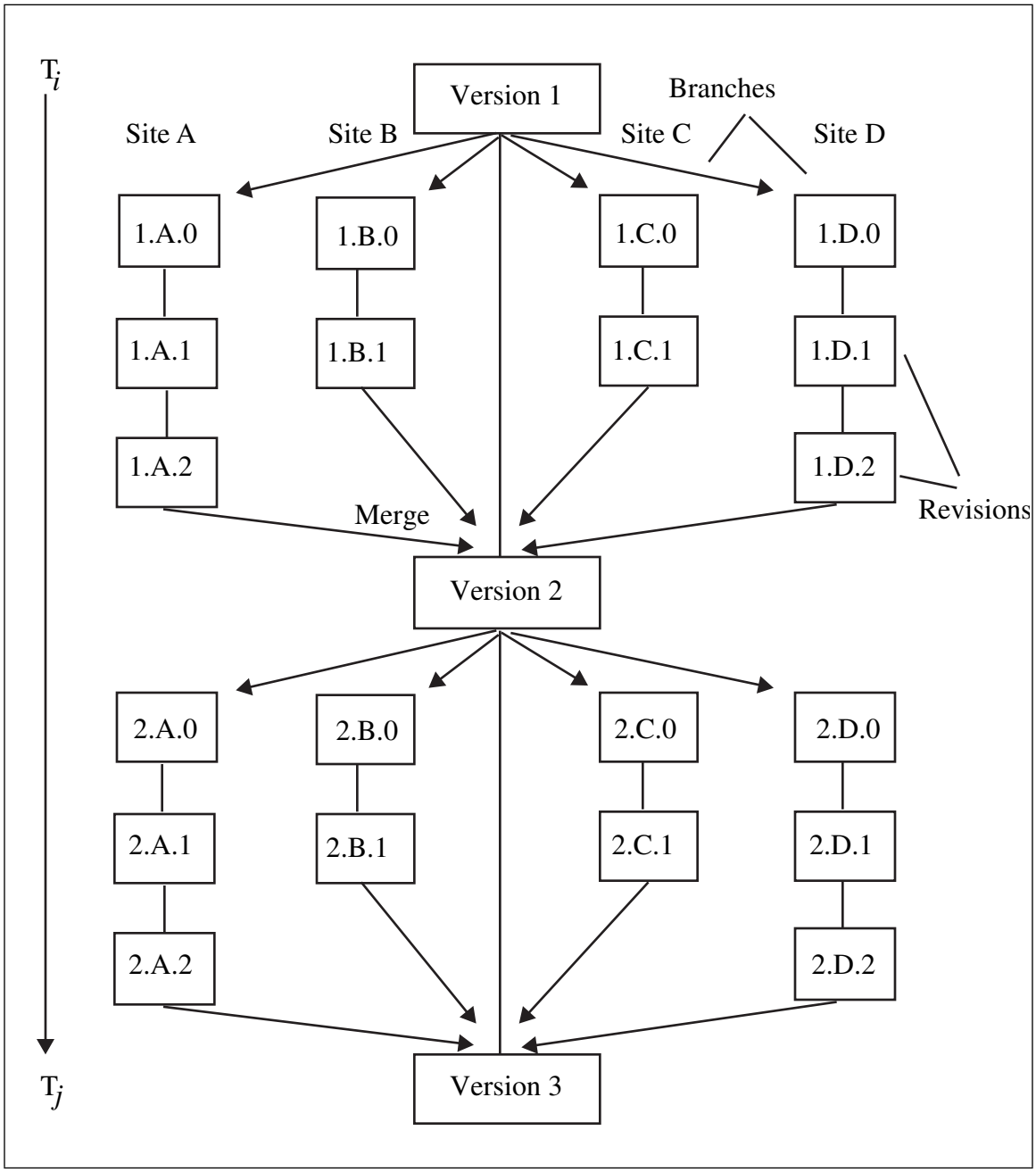


Figure 7-1. A development scenario where a central body directs and coordinates all changes to the CMT. Sites A, B, C, and D each begin working with Version 1 of some terminology. Boxes below each site indicate serial revisions of the terminology.

According to this scenario, the steering committee is responsible for making commitments and determines the direction of the project. Planning is done by the central organization's team responsible for implementing the recommendations of the steering committee. The central organization also provides configuration management. Quality assurance is a shared responsibility. The subcontractors are responsible for the quality of their work, but the central organization will confirm the quality of this work as part of integrating the individual components into the new reference version.

Although the change-set configuration-management model and the conflict-resolution methodologies and tools can all be applied to the central coordination/control paradigm, they may not be optimal for this model. Unlike the central coordination and local control paradigm, central coordination/control may allow strict partitioning of work into non-overlapping pieces. In such a case, centralized databases with locking mechanisms for concurrency control may prove more efficient than an evolutionary process with semantics-based concurrency control. Further quantification of the benefits of pluralistic evolutionary design must be compared with the benefit of automated concurrency-control methods.

Despite the need for further evaluation, one benefit of the semantics-based concurrency control seems evident. If somehow one of the centrally controlled sites falls out of synchronization with the central process, there would be an established mechanism for resynchronizing that site. This is not the case with traditional concurrency control mechanisms.

In addition to further evaluation of the costs and benefits of the different approaches for central coordination and control paradigms, it may be worthwhile to consider a hybrid

methodology. A system that supports traditional, lock-based concurrency control for most tasks but allows a user the *option* of violating these safeguards for specific tasks might provide an optimal combination of methodologies, provided there were built in methods for resolving any potential conflicts.

Local Coordination and Control

The local control and coordination paradigm gives local sites complete autonomy and the responsibility for coordination. Two commercial companies, one a pharmacy system vendor, the other a laboratory system vendor, might form a strategic alliance. The companies agree that their systems will use a common CMT so each can develop applications that can use data stored on the other. When a renal-toxic medication is prescribed, for example, the pharmacy system may query the laboratory system to access any renal-function tests. Neither company obtains new funding as part of this alliance but they expect that the alliance will pay off with future sales.

The companies agree that each will concentrate on its area of expertise, one on pharmacy-related terminology, the other on laboratory related terminology. The two companies agree to share their work on a periodic basis. The specific development tasks of each company are locally controlled, and each company is responsible for incorporating all terminology changes submitted by the other company.

Although this scenario postulates a harmonious alliance, there might well be differing priorities in each company that could lead to conflict. The pharmacy system company, for example, might want its partner to focus on laboratory terminology useful for pharmacy-

related applications, while the laboratory system vendor might expect its partner to focus on other areas that similarly benefit laboratory-related applications. There are no mediators for this effort. It either succeeds or fails.

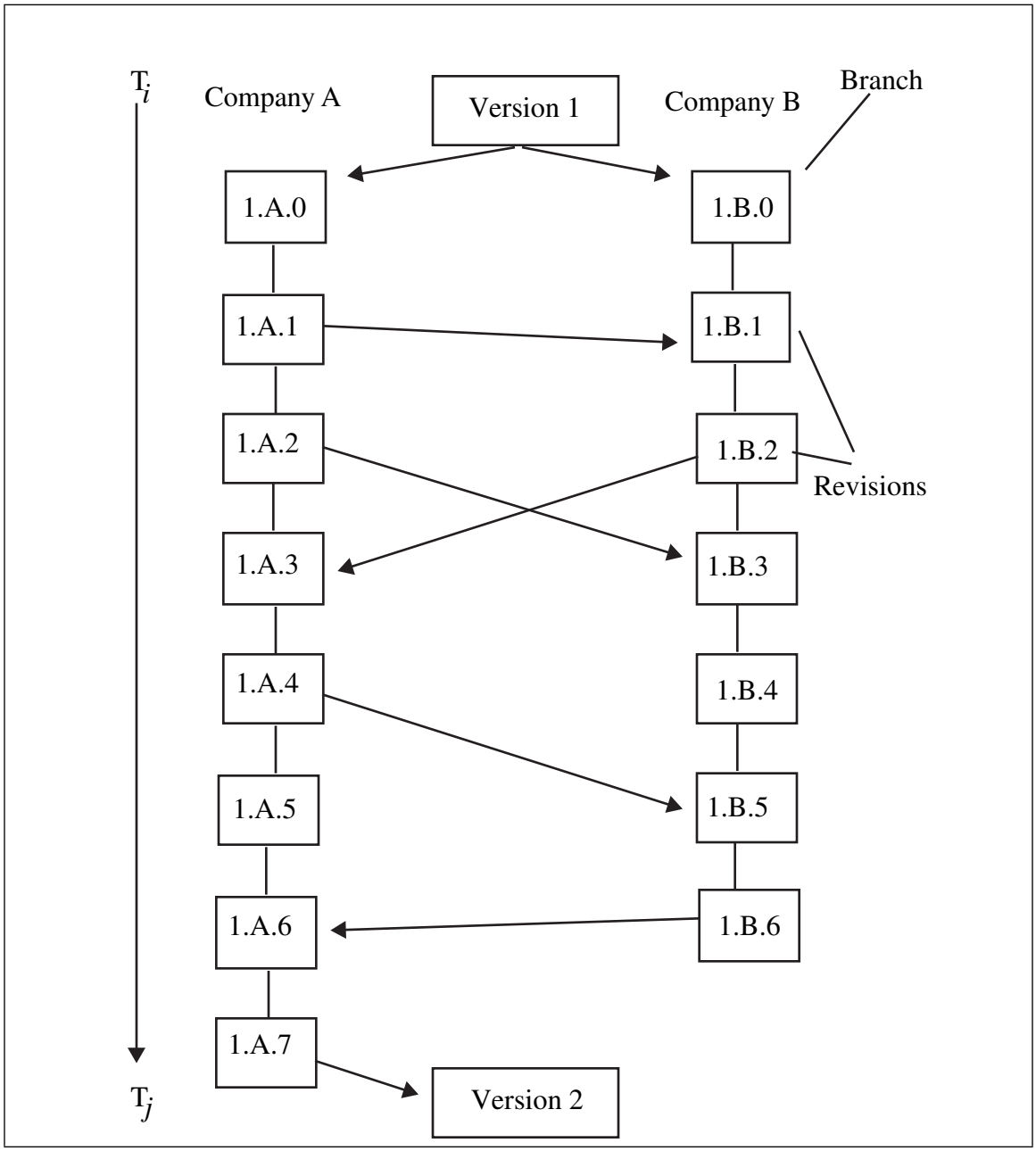


Figure 7-2. A development scenario with no central coordination. Companies A and B each begin working with Version 1 of some terminology. Left-right arrows indicate sharing changes between the companies. Boxes below each company indicate serial revisions of the terminology.

Figure 7-2 illustrates such a development process. This figure refers to the companies simply as “A” and “B.” These two companies begin with a common version of the CMT, version 1. Both companies periodically share their work with the other, and each company expends local effort to incorporate the shared work. At the end of the development cycle, the latest version at one of the two sites will become the new reference version.Although it is technically possible for more that two companies to collaborate in this way, the coordination process would be very complex. Nevertheless, the semantics-based concurrency control, together with the change-set model of configuration management, could provide an enabling methodology for such a collaboration to occur.

7.3.2 Support for Locally Maintained Enhancements

In Chapter 5 of this dissertation there was a description of a scenario in which a modeler decides to experimentally represent more detail in her model—by using IS-A-FUNCTIONAL-PART-OF or IS-A-PHYSICAL-PART-OF relationships—than is allowed in the convergent model through use of a simple IS-PART-OF relationships. The dissertation also described a prototype application—Cristobal—that supported a one-way filter of changes to the terminological definitions. By applying the filter, change sets created using a more detailed model can be applied directly to the convergent model with the appropriate level of detail and role names although the exchange of information is unidirectional. A group benefits through application of the filtered change sets to the convergent terminology, but an individual modeler is unable to directly apply the work of others to her locally enhanced terminology (see Section 5.4), thus limiting her benefit from the collaboration.

If the reason for the divergent models is strictly experimental (to test a model that is intended for future integration into the convergent terminology) and short term (validation and incorporation of the model or invalidation and abandonment), then the existing prototype is adequate for the task. If, however, the divergent model is developmental (intended to meet the needs of a specific application that has terminological requirements excluded from the convergent model), and ongoing, then a mutually beneficial methodology for merging overlapping terminology models is desirable.

Cristobal can be extended to support bidirectional coordination by recognizing when changes to a role in the more general convergent model—such as IS-PART-OF—potentially affect one or more of the roles in the more detailed local model—such as IS-A-FUNCTIONAL-PART-OF or IS-A-PHYSICAL-PART-OF. In such cases, Cristobal can put out conflicting change sets where the value restriction from the IS-PART-OF relationship is applied to both the IS-A-FUNCTIONAL-PART-OF and IS-A-PHYSICAL-PART-OF relationships. These change sets can then be loaded into a locally maintained configuration-management database using Isabella, and the resulting conflict can be resolved using Rhabida. Such an extension is the immediate next step for the Galápagos tools.

7.4 Contributions

This dissertation is an interdisciplinary work. It draws upon techniques of advanced database applications and modern configuration management to make possible the development of a large-scale controlled medical terminology. In turn, this terminology can make possible the informatics applications that are needed to support higher quality and perhaps

also less expensive health care. As such, this work has made contributions to the fields of medical informatics, medicine, and experimental computer science.

7.4.1 Medical Informatics

Sittig cites development of a unified controlled medical terminology as one of the "grand challenges" for the medical informatics field to solve within the next decade (Sittig, 1994). My work provides foundational methodologies for meeting the challenge of developing a unified controlled medical terminology by evaluating scalable methodologies within an actual distributed-development environment.

Although many researchers have proposed using logic-based formalisms as standards for representing terminology (Bernauer, 1991; Campbell & Musen, 1992b; Cimino, Hripcsak, Johnson & Clayton, 1989; Masarie et al., 1991; Rector et al., 1991), skeptics have often pointed out what they believe to be insurmountable obstacles such as the difficulty of achieving computational tractability with logic-based approaches and the considerable amount of work required to generate appropriate definitions for each of the terms within a unified terminology.

I have demonstrated that computational-tractability difficulties can be managed by relying on an appropriately designed description-logic classifier (Mays et al., 1996), and that sizeable terminologies can be produced through parallel development by utilizing semantics-based-concurrency-control and change-set-configuration-management methods.

7.4.2 Medicine

Health-care enterprises are seeking to integrate their existing applications and to develop many new ones that require standardized representations of medical terminology. These applications include large clinical database applications for retrospective and prospective research, monitoring of quality and cost of care, and applications that provide decision support through information management, focusing attention, and patient-specific consultation. Although my work does not directly provide end users with such applications, the development methodology described here is an enabling technology for the achievement of terminologies of the requisite size and quality for successful deployment of robust systems dependent upon standardized representations of medical terminology. Through deployment of such systems, improvements in the delivery of medical care may be realized.

7.4.3 Experimental Computer Science

Researchers in theoretical computer science have described the notion of semantics-based concurrency control (Garcia-Molina, 1983; Garcia-Molina & Salem, 1987), although actual implementations have been lacking. Moreover, strict adherence to the notion of semantics-based concurrency control prescribed in their work required acceptance of one modeler's conflicting changes at the cost of forcing a rollback of the work of another modeler based on essentially arbitrary criteria such as who made the change first. These criteria serve to undermine the goal of supporting cooperative work. Therefore, I have chosen to extend the notion to allow for interactive conflict resolution in an experimental setting. The experimental evaluation has provided a proof-of-concept and a proof-of-performance

for semantics-based concurrency control techniques to cooperative, distributed development of a controlled medical terminology. Prior to this work, it is my belief that no such proof-of-concept or proof-of-performance existed.

Another established technique, the change-set configuration-management model, was integrated with the semantics-based concurrency control to provide a comprehensive set of prototype applications capable of performing in a real-world development environment. Only through this deployment, in a setting with real world development pressures, is it possible to evaluate the practicality of the evolutionary design approach made possible by the combination of the semantics-based concurrency control methods and the change-set model of configuration management. The evaluation in Chapter 6 also provide insights into the thought process inherent in development of controlled terminologies, thus providing important background material for future improvements in both the design process and design tools.

7.5 Final Remarks

I have described a development methodology that enables distributed development of terminological systems. Significant challenges remain, however, before a robust terminological standard suitable for a wide variety of applications can be achieved. These include development of sound foundational models for reasoning about the notion of uncertainty and about anatomy, pathology, pharmacology, and other important aspects of medical care. These models are needed for decision support and for epidemiological tools that are able to give advice and to draw useful conclusions about complex patient situations.

These foundational models will have to evolve over time. Because the underlying model for my work is based on logic, the representation is both general and sufficiently expressive to represent foundational models as they are developed. Because the development methodology supports evolutionary enhancement and development by consensus, it also supports diversity of input and a sense of ownership through participation in the process.

Finally, the ultimate success or failure of a standard for representation of clinical data such as the one proposed here will depend on more than the technical merits of the underlying representation. Important problems such as (1) how to manage the development process, (2) how to meet the needs of specific applications, and (3) how to develop a political consensus, have yet to be solved. The methods and experiences described in this dissertation are steps toward the development of solutions to these problems.

Appendix A
Transcript of Conflict Resolution Session

This transcript was generated from the first conflict-resolution session that used the Galápagos applications. To prepare for this session, three developers worked on enhancing a terminology concurrently using the K-Rep DE application as described in Section 6.2 on page 158.

These transcripts demonstrate a focused discussion of terminology modeling issues. As you will see within the transcripts, often an individual would reach a specific recommendation only to agree later that an alternative modeling strategy is preferable. Clearly the group negotiated resolution of conflicts, and thus truly collaborated.

The participants in this session were myself (Moderator), IBM's K-Rep development manager (K-Rep developer), and the three physicians who performed the terminology modeling (Modeler 1, Modeler 2, Modeler 3). The three modelers are practicing physicians in Internal Medicine and Emergency Medicine, Family Practice, and Nephrology.

Moderator: *OK. I believe that this tape recorder is on and working. So just to reiterate, I'm taping this for the purpose of just being able to go back and verify what happened during the processing of these two conflict reports and discussing them. And I would just like to have each of you verbally acknowledge that your willing to have this session recorded. K-Rep Developer, is that OK with you?*

K-Rep Developer: *Sure.*

Moderator: *And Modeler 3?*

Modeler 3: *Fine.*

Modeler 1: *Fine.*

Modeler 2: *No problem.*

Moderator: *And we may generate transcripts from this, but any transcripts that would identify people by name and would be published would be verified with you before they would be published for your approval. And likewise for any video clips that would ever come out of this. OK? And, that's all the formalities I have.*

So I have two reports here. One was generated on Friday, and the report that was generated on Friday had brought together all the change sets that you had been working on, integrated them, and then identified all the conflicts in them. The other page that you have is part of a report that has about three more additions. And that was because one of your commit journals got corrupted and the .krep file was brought in directly, and if you look at the very first concept on that page,[see Figure A-1] that's one of the ones that had a change that was not reflected in the commit journal but was part of the .krep file, so we have brought that in as a branch. If you look you will see a line that says "ARC 1 5," so that means that concept number 5 came in from the .krep file. Now prior, the concept had gone from 1 to 2, from 2 to 3, and from 3 to 4. So concept 4 was the latest one in the commit journal.

I selected to bring them in that way so you at least verify that you wanted to continue to keep the differences. And we would just bring them in through the conflict resolution process.

What we need to do is go through each of the conflicts and decide how we want to resolve them. And eventually some gnome has to make it over to that other machine to go through what we agreed to and then dump a new version of pilot so that people can start working on a new

```
Tremor-NOS_F-A4600 has multiple end states!

1 P (and Nervous-system-function-NOS_F-A0000
    (:Display_Name "Tremor-NOS")
    (:Displayable "T")
    (:SNOMED_Code "F-A4600")
    (:Synonym "Tremor-NOS")
    (:Synonym "Involuntary trembling")
    (:Synonym "Involuntary quiver")
    (:Synonym "Quivered")
    (:Synonym "Trembled"))

2 P (and Nervous-system-function-NOS_F-A0000
    Motor-exam_F-3RR44
    (:Display_Name "Tremor-NOS")
    (:Displayable "T")
    (:SNOMED_Code "F-A4600")
    (:Synonym "Tremor-NOS")
    (:Synonym "Involuntary trembling")
    (:Synonym "Involuntary quiver")
    (:Synonym "Quivered")
    (:Synonym "Trembled"))

3 P (and Nervous-system-function-NOS_F-A0000
    General-motor-activity_F-3RR47
    (:Display_Name "Tremor-NOS")
    (:Displayable "T")
    (:SNOMED_Code "F-A4600")
    (:Synonym "Tremor-NOS")
    (:Synonym "Involuntary trembling")
    (:Synonym "Involuntary quiver")
    (:Synonym "Quivered")
    (:Synonym "Trembled"))

4 P (and Motor-exam_F-3RR44
    (:Display_Name "Tremor-NOS")
    (:Displayable "T")
    (:SNOMED_Code "F-A4600")
    (:Synonym "Tremor-NOS")
    (:Synonym "Involuntary trembling")
    (:Synonym "Involuntary quiver")
    (:Synonym "Quivered")
    (:Synonym "Trembled"))

5 P (and Nervous-system-function-NOS_F-A0000
    Neurologic-symptom_F-AR000
    (:Display_Name "Tremor-NOS")
    (:Displayable "T")
    (:SNOMED_Code "F-A4600")
    (:Synonym "Tremor-NOS")
    (:Synonym "Involuntary trembling")
    (:Synonym "Involuntary quiver")
    (:Synonym "Quivered")
    (:Synonym "Trembled"))

Arc 1 2
##Modeler 2|From K-Rep DE

Arc 1 5
##|From .krep file Apr 22 96 4:3:29 GMT

Arc 2 3
##Modeler 2|From K-Rep DE

Arc 3 4
##Modeler 2|From K-Rep DE

--Concepts are NOT equivalent--
```

Figure A-1. Conflict report for "tremor."

baseline.

So do you just want to go through this report one by one?

Modeler 2: *Yeah, but I got to the report first this morning and I already circled what I thought was the correct resolution for the conflict except for this first one.*

Moderator: *OK.*

Modeler 2: *Uh. Flexion. And I wrote the comments there which are definitions [from the dictionary].*

Moderator: *OK. Well lets just go to the first one. Actually "procedure order form." [see Figure A-2] Now it really wasn't a conflict but potentially what*

procedure-order-form has multiple end states! 1 N 2 D Order-form 3 P Order-form	Arc 1 2 ##Modeler 3|From K-Rep DE Arc 1 3 ##Modeler 3|From K-Rep DE **Concepts are equivalent**

Figure A-2. Conflict report for "procedure order form."

was happening, I think, is you were redefining a concept with the same name. It did not actually get a journaled history of the concept, it just got "defprimconcept, defprimconcept, defprimconcept..."

Modeler 3: *Yes. Yes, I was in the situation where I wanted to remove a facet and replace it with a different facet and I knew [the krep DE tool would not allow me] to remove a facet. I had tried that before. So I tried overwriting the facet which did not work.*

Moderator: *So you just overwrite the whole concept? OK. So I assume that the version you want is the "order-form" should be primitive.*

Developer 3: *Yes.*

Moderator: *And there is really no other defining characteristics of it. So this one was easy.*

Now the next one is flexion [see Figure A-3], and Modeler 2 has looked at this already.

Modeler 2: *And it goes along with extension and abduction and several other conflicts that are going to come up.*

Moderator: *Yes, actually I noticed these and I thought they were quite interesting. So what comments do people have about these? I guess the difference between the two is that one person said flexion was a "muscle function" and another person said it was a "joint function." Does anybody have any comments about what they think is the right thing to do here?*

Modeler 2: *I think it is a musculoskeletal function, as I wrote, so really it is both.*

Moderator: *So you would go back to a more general term [musculoskeletal function*

Flexion-NOS_F-10110 has multiple end states! 1 P (and Musculoskeletal-symptom-NOS_F-10050 (:Display_Name "Flexion-NOS") (:Displayable "T") (:SNOMED_Code "F-10110") (:Synonym "Flexion-NOS") (:Synonym "Flexed")) 2 P (and Muscle-function-NOS_F-11000 (:Display_Name "Flexion-NOS") (:Displayable "T") (:SNOMED_Code "F-10110") (:Synonym "Flexion-NOS") (:Synonym "Flexed"))	3 P (and Joint-function-NOS_F-13000 (:Display_Name "Flexion-NOS") (:Displayable "T") (:SNOMED_Code "F-10110") (:Synonym "Flexion-NOS") (:Synonym "Flexed")) Arc 1 2 ##Modeler 2|From K-Rep DE Arc 1 3 ##Modeler 1|From K-Rep DE --Concepts are NOT equivalent--

Figure A-3. Conflict report for "flexion."

rather than either of the more specific terms joint function or muscle function].

Modeler 2: *If joint function is a musculoskeletal function and muscle function is a musculoskeletal function then I would categorize flexion separately under each.*

Moderator: *Under both [muscle function and joint function]?*

Modeler 1: *Separately under both?*

Modeler 2: *Yes.*

Modeler 3: *That's fine. Really it has very different meaning. Flexion is a muscle function.*

Modeler 1: *Really the joint has the movement.*

Moderator: *Isometric exercise involves muscle function with no joint movement.*

Modeler 2: *Flexion requires a joint and activity of a flexor.*

Modeler 1: *Now wait. It does not require [a flexor]. And remember this is just general flexion. It does not distinguish between active and passive. Active flexion requires the flexor.*

Modeler 3: *That's what I said. It could be intrinsic or extrinsic.*

Modeler 2: *The flexor could be your arm moving my arm.*

Modeler 1: *Exactly.*

Modeler 3: *Flexion is a joint function.*

Modeler 2: *But as far as the concept flexion, you'd agree that it is a musculoskeletal function?*

Modeler 3: *Well I think that you should make it…*

Modeler 2: *Most generally?*

Modeler 3: *Well most generally it's a body function, but more specifically it is a joint function.*

Modeler 2: *OK.*

Modeler 3: *I think it really is a joint function. The muscle function for instance, the muscle function of the extensor muscle in flexion is to relax. That's true, it is a muscle function, but its function is to relax in that case.*

Moderator: *But I would think it is a complex process involving both. If you have a frozen joint, it won't flex. If you have no motor control it won't...*

Modeler 1: *ACTIVELY flex.*

Modeler 3: *Well, but you are talking about active flexion.*

Modeler 1: *But we haven't represented with further specificity active and passive flexion.*

Modeler 2: *So you [Modeler 3], would categorize active flexion under muscle function?*

Modeler 1: *And joint function.*

Modeler 2: *And passive flexion under flexion and nothing else.*

Modeler 3: *I would say that flexion is something done to a joint by an actor. In active flexion the actor is the muscle that crosses that joint. In passive flexion the actor is extrinsic. Is environmental.*

Modeler 2: *OK. So, Flexion is a joint function?*

Modeler 3: *Flexion and extension and rotation...*

Modeler 2: *So we take the 3rd option here. We delete the muscle function as a parent.*

Modeler 1: *But what if we add a concept called active flexion?*

Modeler 3: *Yes, active flexion has a joint that is flexed by it also has a muscle group that is the performer of the flexion.*

Modeler 1: *Well is used as an effector.*

Modeler 2: *So it will have two parents. It will have a parent that is a muscle function and a parent that is a joint function.*

Moderator: *So for flexion we will preserve it as a joint function but that it would be a good idea to add active flexion... Well maybe this should be Flexion, NOS and we should add Active Flexion, NOS and Passive Flexion, NOS.*

Modeler 2: *Right, so we are going to have to add some concepts.*

Modeler 1: *Well, they may already be there. We just haven't looked.*

Modeler 2: *Yes, we need to look.*

Modeler 1: *But we still have to decide if the relationships should be is-a relationships vs. a role relationship has-effector flexion.*

Modeler 2: *I think based on our model it's an is-a. Flexion is-a joint function.*

Modeler 1: *Yes, but we are talking about active and passive.*

Moderator: *Well, you would say active flexion is-a joint function and is-a muscle function.*

Modeler 2: *Or active flexion is-a flexion.*

Moderator: *Yes, it is-a flexion.*

Modeler 2: *And it is-a muscle-function.*

Moderator: *So then it would inherit the joint function.*

Modeler 1: *Right.*

Modeler 2: *Uh, OK.*

Moderator: *OK. The next one is the "prone body position." [see Figure A-4]*

Prone-body-position_F-10310 has multiple end states! 1 P (and Musculoskeletal-symptom-NOS_F-10050 (:Display_Name "Prone body position") (:Displayable "T") (:SNOMED_Code "F-10310") (:Synonym "Prone body position") (:Synonym "Prone position")) 2 P (and Body-position-NOS_F-R3412 (:Display_Name "Prone body position") (:Displayable "T") (:SNOMED_Code "F-10310") (:Synonym "Prone body position") (:Synonym "Prone position"))	3 P (and Posture-NOS_F-10300 (:Display_Name "Prone body position") (:Displayable "T") (:SNOMED_Code "F-10310") (:Synonym "Prone body position") (:Synonym "Prone position")) Arc 1 2 ##Modeler 2|From K-Rep DE Arc 1 3 ##Modeler 1|From K-Rep DE --Concepts are NOT equivalent--

Figure A-4. Conflict report for "prone body position."

Modeler 2: *Which I categorized as a body position using is-a. And Modeler 1 said this is a posture, and I disagree with that.*

Modeler 1: *Part of that was probably based upon the [SNOMED termcode].*

Modeler 3: *What is a posture?*

Modeler 2: *There are postures, I can't list them for you.*

Moderator: *Decorticite posture, decerebrate posture…*

Modeler 3: *Bad posture, good posture.*

Modeler 2: *Kyphotic posture, lorodotic posture, hyperlorodotic posture, bent over hunch back posture.*

Moderator: *Actually, can't you have a posture in a position? Like can't you have decorticite posture in the prone position?*

Modeler 3: *We've just had a clean run around posture and I'm still not sure what it is.*

Modeler 1: *Body position was created by you [Modeler 2], correct?*

Modeler 2: *Body position is a created term, yes. And it is because of the area of lexicon that I was modeling...*

Modeler 1: *Not only that, the posture itself was overlapping because it did not trap for body position in different posture.*

Modeler 3: *Are positions postures?*

Modeler 2: *No. Body positions are positions like "frog position," and "prone position," and "supine position."*

Modeler 3: *Why are they not postures?*

Modeler 2: *They are usually exclusive of posture. You may have a posture and a position at the same time.*

Modeler 3: *Well, I mean, what I would take... I don't see a clean definition. You could say that position is a matter of the attitude of the axis of the body. It's like this, or it's like this, or something.*

Modeler 2: *That's correct. Attitude and facing.*

Modeler 3: *That's all you need. You have a direction it faces and an axis. Prone, supine, correct?*

Modeler 2: *All positions.*

Modeler 3: *What are other positions?*

Modeler 2: *Frog position.*

Modeler 3: *What's that?*

Modeler 2: *It's that position for pelvic exam.*

Moderator: *The lithotomy position?*

Modeler 2: *The lithotomy position.*

Modeler 3: *How is lithotomy position not a posture? How is it fundamentally different from decorticate position?*

Modeler 2: *Maybe it's not. But what I would say is that for these particular concepts, they contain the word position in them or they contain the word posture in them. So that's kind of where I came from.*

Modeler 3: *Will you have synonyms for these positions that are postures?*

Modeler 2: *Yes.*

Moderator: *You could make the lithotomy position and give it a synonym that is a posture. And make the posture thing the preferred form.*

Modeler 3: *Then lithotomy position would be available lexically. Dorsal lithotomy position [garbled]... But if positions are postures, then how would you determine the axis? Maybe that is not appropriate.*

Moderator: *So the position that we are taking for prone-body-position is the number 2 state*

Modeler 2: *The Body-position*

Moderator: *And we don't want the number 3 [state].*

All right. The next one is Flaccidity. [see Figure A-5]

Flaccidity-NOS_F-11300 has multiple end states! 1 P (and Muscle-property-NOS_F-11100 (:Display_Name "Flaccidity-NOS") (:Displayable "T") (:SNOMED_Code "F-11300") (:Synonym "Flaccidity-NOS") (:Synonym "Muscular flaccidity") (:Synonym "Flaccid")) 2 P (and Muscle-tonus_F-11190 (:Display_Name "Flaccidity-NOS") (:Displayable "T") (:SNOMED_Code "F-11300") (:Synonym "Flaccidity-NOS") (:Synonym "Muscular flaccidity") (:Synonym "Flaccid"))	3 P (and Muscle-property-NOS_F-11100 Musculoskeletal-symptom-NOS_F-10050 (:Display_Name "Flaccidity-NOS") (:Displayable "T") (:SNOMED_Code "F-11300") (:Synonym "Flaccidity-NOS") (:Synonym "Muscular flaccidity") (:Synonym "Flaccid")) Arc 1 2 ##Modeler 2|From K-Rep DE Arc 1 3 ##Modeler 1|From K-Rep DE

Figure A-5. Conflict report for "flaccidity."

Modeler 2: *Modeler 1 categorized this as a musculoskeletal-symptom, and I categorized this as a type of muscle-tonus. The synonym for muscle tonus is muscle tone.*

Modeler 3: *What would the parents of that be?*

Modeler 2: *Muscle property. Basically what I did was take flaccidity out of muscle property and moved it down to muscle tonus. So that the reason for that was when you are describing your neuro exam or your motor exam, you could say whether the muscle is rigid or flaccid or normal tone. And*

then, Modeler 1 said that flaccidity is a muscular symptom.

Modeler 1: *There is a problem with a lot of the symptoms. They can be used as symptoms and are also often findings as well.*

Modeler 2: *Well the way that I have been using things like this are with this associated findings is that the proper place for flacidity is that it is truly a physical finding, and that it should be available from the symptoms taxonomy through an associated finding facet.*

Modeler 3: *Shouldn't we say that symptoms are purely things that are not discernible by an outside observer?*

Modeler 2: *That's kind of the rules that I've been using.*

Modeler 3: *And most things reported as symptoms actually are findings.*

Modeler 2: *Things like pain are clearly symptoms.*

Modeler 3: *Nausea is a symptom, vertigo is a symptom.*

Modeler 2: *So what do you think about associating [flaccidity] to musculoskeletal symptom as an associated finding.*

Modeler 1: *Are you saying that flaccidity has an associated finding that is a musculoskeletal symptom?*

Modeler 2: *No I would say that muscle symptom would have an associated finding of flaccidity.*

Modeler 3: *Remember that this is not a lexical connection that you are talking about.*

Modeler 2: *No, this is not a semantic connection.*

Modeler 3: *No.*

Modeler 2: *This is a facet connection.*

Modeler 3: *Signs are signs. They can be recorded by the party, and they can show up in the symptoms column, but they still are signs.*

Modeler 1: *Exactly.*

[Garbled]

Modeler 1: *What, I didn't hear a lot of that.*

Modeler 3: *Given that kind of thinking, it's a sign. It's a physical sign. And it's a muscle... what did you call it? A muscle property?*

Modeler 2: *A muscle tonus.*

Modeler 3: *A muscle tonus. [Garbled].*

Modeler 2: *And if Modeler 1 wants that there in the Kruiser under muscle symptom, then it would have to be associated under muscle symptom using the associated finding.*

Moderator: *And that's a facet?*

Modeler 2: *The associated finding is a facet. Like if you wanted to say nausea and vomiting, they don't have a semantic relationship.*

K-Rep Developer: *Is there associated findings and associated symptoms?*

Modeler 2: *No. Symptoms and signs are both findings.*

Modeler 1: *[garbled]. Both signs and symptoms are considered in the lexicon to by kinds of clinical findings.*

Modeler 2: *John's not thinking in terms of the Kruiser, but the reason this facet was invented was because of the Kruiser.*

Modeler 3: *What was invented?*

Modeler 2: *The associated findings facet.*

Modeler 3: *It does work for reporting history because you report all kinds of stuff, but if you go through what are called symptoms, most of them are available as signs. And really people put... all kinds of stuff in the HPI. And most of what is up there is really [garbled].*

Modeler 2: *So we agree that number 2...*

Moderator: *Number 2 is the way to go.*

Who knows what Intermalleolar-straddle is? [see Figure A-6] I have no clue...

Modeler 3: *Well Modeler 1 classified it as a posture, and Modeler 2 classified it as*

Intermalleolar-straddle_F-10430 has multiple end states! 1 P (and Musculoskeletal-symptom-NOS_F-10050 (:Display_Name "Intermalleolar straddle") (:Displayable "T") (:SNOMED_Code "F-10430") (:Synonym "Intermalleolar straddle")) 2 P (and Muscle-function-NOS_F-11000 (:Display_Name "Intermalleolar straddle") (:Displayable "T") (:SNOMED_Code "F-10430") (:Synonym "Intermalleolar straddle"))	3 P (and Posture-NOS_F-10300 (:Display_Name "Intermalleolar straddle") (:Displayable "T") (:SNOMED_Code "F-10430") (:Synonym "Intermalleolar straddle")) Arc 1 2 ##Modeler 2|From K-Rep DE Arc 1 3 ##Modeler 1|From K-Rep DE

Figure A-6. Conflict report for "intermalleolar straddle."

a muscle function.

Modeler 2: *Is that where you like straddle a horse?*

Modeler 1: *Well the horse is between your two malleoli.*

Modeler 3: *Maybe it's when you walk with your hips on the inside of your malleoli.*

Modeler 1: *Well that would be a gait.*

Modeler 3: *Well how did that get in there? I'm delighted that it did...*

K-Rep Developer: *Is this a veterinary term?*

Modeler 2: *I think its a muscular function and Modeler 1 thinks its a posture.*

Modeler 3: *Neither of you know what the hell this thing is.*

Modeler 2: *And I could not find it in any of my references. Based upon it's code, it's most general parent would be...*

Modeler 1: *Closest to posture but, I'm not sure what to do with this one. What do we do with these questions? Do we have to go back and research them?*

Modeler 3: *We need some experts here.*

Modeler 2: *We are content experts.*

Modeler 1: *This is an uncertainty because of ignorance on both of our parts.*

Moderator: *Well, one of the things you can do in the editing environment is to change the parent to be something more general like "weird findings" [chuckle].*

Modeler 1: *Right. Garbage pail.*

Moderator: *For the purposes here though, the assumption is that one or the other of these is "right" because somebody did it. So you have to chose the best from what is here.*

Modeler 1: *We should go with the most general at this time since we don't know.*

Modeler 2: *Which is muscle function.*

Moderator: *Well musculoskeletal symptom is more general I think.*

Modeler 3: *Well, if we make it the most general, then we won't look at it again, and we won't know anything until the person who needs it do describe something can't find it.*

Moderator: *Well, if you put it in the most general place, it will be highest and annoying.*

Modeler 3: *But your idea of putting it under weird finding category…*

Modeler 1: *But he's saying for conflict resolution.*

Moderator: *For conflict resolution, you have to pick one of these. So if you were to pick the most general, it would be musculoskeletal symptom.*

Modeler 2: *Yea, that's fine. So what we want to do is to stick with the current version.*

Moderator: *Really what you want to do is to roll back to a prior version. Hmm. That's a little harder to do, because you are creating a cycle in your path. And computers have a hard time dealing with cycles. So what I think I'll do is arbitrarily pick one, and if that's a problem, you can edit the concept in the next baseline using K-Rep DE.*

Modeler 2: *The next one is a little cycle that Developer 1 did. [see Figure A-7] I don't know what. Pick the latest?*

Moderator: *Transient paralysis of the limb is...*

Transient-paralysis-of-limb_F-11750 has multiple end states! 1 P (and Muscle-property-NOS_F-11100 (:Display_Name "Transient paralysis of limb") (:Displayable "T") (:SNOMED_Code "F-11750") (:Synonym "Transient paralysis of limb")) 2 P (and Musculoskeletal-symptom-NOS_F-10050 (:Display_Name "Transient paralysis of limb") (:Displayable "T") (:SNOMED_Code "F-11750") (:Synonym "Transient paralysis of limb"))	3 P (and Muscle-property-NOS_F-11100 Musculoskeletal-symptom-NOS_F-10050 (:Display_Name "Transient paralysis of limb") (:Displayable "T") (:SNOMED_Code "F-11750") (:Synonym "Transient paralysis of limb")) Arc 1 2 ##Modeler 1|From K-Rep DE Arc 1 3 ##Modeler 1|From K-Rep DE --Concepts are NOT equivalent--

Figure A-7. Conflict report for "transient paralysis of limb."

Modeler 2: *He redefined this a couple of times.*

Moderator: *He went through a cycle actually. The reason this concept was identified was because he went to 2, and then undid what he did and went back to 1, and then changed it to be 3.*

Modeler 1: *Well actually I merged these.*

Moderator: *But the process by which you did this is you undid it, then made a commit, which went back to where you were before, and then you added to it.*

Moderator: *So how would you... Well for this, it's OK to go from 2 to 3?*

Modeler 1: *Right.*

Moderator: *So for this, we'll make that the winner.*

OK. Chemoreceptor function. [see Figure A-8]

Modeler 2: *This one I think our changes should be combined. chemoreceptor-function is a nervous-system function and is also an autonomic-cardiovascular function.*

Modeler 1: *Correct.*

Modeler 2: *So, it should get those two parents. And sensation should not be it's parent.*

Chemoreceptor-function_F-A2180 has multiple end states! 1 P (and Sensation-NOS_F-A2000 (:Display_Name "Chemoreceptor function") (:Displayable "T") (:SNOMED_Code "F-A2180") (:Synonym "Chemoreceptor function")) 2 P (and Nervous-system-function-NOS_F-A0000 (:Display_Name "Chemoreceptor function") (:Displayable "T") (:SNOMED_Code "F-A2180") (:Synonym "Chemoreceptor function"))	3 P (and Autonomic-cardiovascular-function_F-A8570 (:Display_Name "Chemoreceptor function") (:Displayable "T") (:SNOMED_Code "F-A2180") (:Synonym "Chemoreceptor function")) Arc 1 2 ##Modeler 2|From K-Rep DE Arc 1 3 ##Modeler 1|From K-Rep DE

Figure A-8. Conflict report for "Chemoreceptor function."

Moderator: *OK. So we'll merge them.*

Modeler 2: *And then we have extension which is the same deal which we already talked about. [see Figure A-9]*

Extension-NOS_F-10100 has multiple end states! 1 P (and Musculoskeletal-symptom-NOS_F-10050 (:Display_Name "Extension-NOS") (:Displayable "T") (:SNOMED_Code "F-10100") (:Synonym "Extension-NOS") (:Synonym "Extended")) 2 P (and Muscle-function-NOS_F-11000 (:Display_Name "Extension-NOS") (:Displayable "T") (:SNOMED_Code "F-10100") (:Synonym "Extension-NOS") (:Synonym "Extended")) 3 P (and Musculoskeletal-mobility-NOS_F-10030 (:Display_Name "Extension-NOS") (:Displayable "T") (:SNOMED_Code "F-10100") (:Synonym "Extension-NOS") (:Synonym "Extended"))	4 P (and Joint-function-NOS_F-13000 (:Display_Name "Extension-NOS") (:Displayable "T") (:SNOMED_Code "F-10100") (:Synonym "Extension-NOS") (:Synonym "Extended")) Arc 1 2 ##Modeler 2|From K-Rep DE Arc 1 3 ##Modeler 1|From K-Rep DE Arc 3 4 ##Modeler 1|From K-Rep DE

Figure A-9. Conflict report for "extension."

Moderator: *So we agreed to make it a joint function?*

Modeler 2: *Yes.*

Moderator: *Cranial Nerve Exam... [see Figure A-10]*

Cranial-nerve-XI-exam_F-3RR04 has multiple end states! 1 N 2 P (and Physical-exam (some has_Symmetry Symmetry) (some in_System Vagus-nerve-NOS_T-A8640) (:Design_Note "") (:Display_Name "Cranial nerve X exam") (:Displayable "T") (:SNOMED_Code "F-3RR03") (:Synonym "Cranial nerve X exam")) 3 P (and Physical-exam (some has_Symmetry Symmetry) (some in_System Accessory-nerve-NOS_T-A8780) (:Design_Note "") (:Display_Name "Cranial nerve XI exam") (:Displayable "T") (:SNOMED_Code "F-3RR04") (:Synonym "Cranial nerve XI exam")) 4 P (and Physical-exam (some has_Symmetry Symmetry) (some in_System Accessory-nerve-NOS_T-A8780) (:Associated_Findings Fasciculation_F-11630) (:Design_Note "") (:Display_Name "Cranial nerve XI exam") (:Displayable "T") (:SNOMED_Code "F-3RR04") (:Synonym "Cranial nerve XI exam")) 5 P (and Physical-exam (some has_Symmetry Symmetry) (some in_System Accessory-nerve-NOS_T-A8780) (:Associated_Findings Fasciculation_F-11630) (:Associated_Findings Muscle-tonus_F-11190) (:Design_Note "") (:Display_Name "Cranial nerve XI exam") (:Displayable "T") (:SNOMED_Code "F-3RR04") (:Synonym "Cranial nerve XI exam"))	6 P (and Physical-exam (some has_Symmetry Symmetry) (some in_System Accessory-nerve-NOS_T-A8780) (:Associated_Findings Fasciculation_F-11630) (:Associated_Findings Muscle-tonus_F-11190) (:Associated_Findings Atrophy-NOS_M-58000) (:Design_Note "") (:Display_Name "Cranial nerve XI exam") (:Displayable "T") (:SNOMED_Code "F-3RR04") (:Synonym "Cranial nerve XI exam")) 7 P (and Physical-exam (some has_Symmetry Symmetry) (some in_System Accessory-nerve-NOS_T-A8780) (:Associated_Findings Fasciculation_F-11630) (:Associated_Findings Muscle-tonus_F-11190) (:Associated_Findings Muscle-size_F-111R5) (:Design_Note "") (:Display_Name "Cranial nerve XI exam") (:Displayable "T") (:SNOMED_Code "F-3RR04") (:Synonym "Cranial nerve XI exam")) Arc 1 2 ##Modeler 2|From K-Rep DE Arc 2 3 ##Modeler 2|From K-Rep DE Arc 3 4 ##Modeler 2|From K-Rep DE Arc 4 5 ##Modeler 2|From K-Rep DE Arc 5 6 ##Modeler 2|From K-Rep DE Arc 5 7 ##Modeler 2|From K-Rep DE **Concepts are equivalent**

Figure A-10. Conflict report for "Cranial nerve XI exam."

Modeler 2: *Cranial Nerve XI Exam. This is one that I changed like 15 times. I don't know why it did this. All I was doing was adding facets, it looks like.*

Moderator: *Right. Well you know you don't have to hit the commit button every time you change a facet.*

Modeler 2: *I know. It depends upon how recent the last crash was.*

Modeler 1: *Now wait a minute. Are you talking about commit transaction or commit concept?*

Moderator: *Commit concept. You can make multiple commits to a concept before you have to commit a transaction.*

Modeler 1: *But this writes every commit to a concept?*

Moderator: *Every time you hit the commit button, it's a state.*

Modeler 2: *And it doesn't just automatically take the latest state from my CJ file?*

Moderator: *It takes all the states. For example, one of your states may have been consistent with something that Developer 1 did, and it forms a path to merge his work with your work.*

Modeler 1: *And not only that, you could have been working on this concept, and within the browser cruised to another concept, making changes that would have percolated between, and the CJ file wouldn't know the difference.*

Modeler 2: *In any event, 7 is the appropriate choice.*

Modeler 1: *So the lesson is, If you know all the changes, make them all at once before you hit the commit button. We're all guilty of that.*

Moderator: *Ok. Posture. Well here we go again. You can't have posture without muscle function. [see Figure A-11]*

Modeler 2: *Or without skeletal function. So I think it is a musculoskeletal function.*

Moderator: *So could we merge those then?*

Modeler 2: *I think 3... Muscle function is contained within musculoskeletal function.*

Modeler 1: *Joint function, muscle function, skeletal function. It's really a musculoskeletal function in general.*

Posture-NOS_F-10300 has multiple end states! 1 P (and Musculoskeletal-symptom-NOS_F-10050 (:Display_Name "Posture-NOS") (:Displayable "T") (:SNOMED_Code "F-10300") (:Synonym "Posture-NOS") (:Synonym "Body position-NOS")) 2 P (and Muscle-function-NOS_F-11000 (:Display_Name "Posture-NOS") (:Displayable "T") (:SNOMED_Code "F-10300") (:Synonym "Posture-NOS") (:Synonym "Body position-NOS"))	3 P (and Musculoskeletal-function-NOS_F-10000 (:Display_Name "Posture-NOS") (:Displayable "T") (:SNOMED_Code "F-10300") (:Synonym "Posture-NOS") (:Synonym "Body position-NOS")) Arc 1 2 ##Modeler 2|From K-Rep DE Arc 1 3 ##Modeler 1|From K-Rep DE --Concepts are NOT equivalent--

Figure A-11. Conflict report for "posture."

Modeler 2: *Right. We could probably change within our KB, we could put muscle function underneath musculoskeletal function. Correct?*

Modeler 1: *It should [already] be there.*

K-Rep Developer: *Is musculoskeletal function here the superconcept of muscle function?*

Modeler 2: *It should be.*

Modeler 1: *In the knowledge base it should be at the bottom.*

K-Rep Developer: *OK, but there is kind of two distinct concepts here. One of which is sort of the union of all of the things that are one or the other. And the other that is something that is the intersection of the two.*

Modeler 2: *Musculoskeletal function encompasses all functions of the musculoskeletal system. Muscles are encompassed within the musculoskeletal system. Therefore logically I would consider that muscle function is-a musculoskeletal function.*

Moderator: *Will I think if you merge the two, it would still classify the same way.*

Modeler 2: *It would classify under muscle.*

Modeler 3: *But it isn't right.*

[Garbled]

Modeler 2: *You know what I just saw? One synonym of posture is body position! Isn't that cool?*

Modeler 1: *That was the problem. That's what I think I saw. That's why I put all the body positions under posture.*

Moderator: *So like the lithotomy position is a posture.*

Modeler 2: *Is considered a synonym of posture.*

Modeler 3: *Prone position is a posture. I just don't see a real fruitful... I see a lot of confusion coming from trying to separate the two. I don't see the benefit.*

Moderator: *Well, there are subtle distinctions.*

Modeler 2: *It's for an examination position.*

Moderator: *I guess there are sort of two things I see. One is describing the patients general state of ability...*

Moderator: *So you have sort of spontaneous posture. And then you can force them into a posture to do a procedure. And that's more you know the lithotomy position. Sort of the difference is the intent.*

Modeler 3: *So one is a clinical maneuver, and posture is spontaneous. So if someone is found on the ground face down. And their nose is necrotic because they were in prone posture for 2 hours?*

Modeler 2: *Prone position.*

Modeler 3: *Ah Ha! but that's the thing [position] you put them into clinically.*

Modeler 2: *The position.*

Modeler 3: *That's the thing that there is more than one definitional axis between the two. One is that a general position is of the body axis and that position is [garbled].*

Modeler 1: *The problem is that in common sorts of usage they are interchangeable, and not well delineated.*

Modeler 2: *Lets keep them under one taxonomy and we will re-evaluate this.*

Modeler 1: *OK. So we'll go back a couple of pages.*

Moderator: *So for prone body position, it's both a posture and a body position?*

Modeler 1: *It's on page one and two.*

Modeler 2: *OK. The problem is that body position needs to be... Body position NOS I created. I created a concept that didn't need to be created.*

Modeler 3: *Body position?*

Modeler 2: *Yes. Oh maybe not. This is why we need the synonym search. If I had been able to do a synonym search. I would have found that body position was already defined as a posture.*

Modeler 1: *See, I found it that way. I looked [garbled]*

Moderator: *So this would be a good concept to try to expunge somehow.*

Modeler 2: *Body position?*

Modeler 3: *Give it the status of synonym to posture.*

Modeler 2: *See, it already does.*

Moderator: *But the thing is that we now have this unique concept identifier, something that is a synonym of something else.*

Modeler 1: *Right.*

Moderator: *And that's not a good thing.*

Modeler 1: *Right. We need to expunge this somehow.*

Moderator: *So on one of the versions, we need to dump the data file, strip it out, and reload.*

Modeler 1: *Well, we either change SNOMED, get rid of the synonym, or we...*

Modeler 3: *It's OK as a synonym.*

Modeler 1: *Get rid of the concept that Modeler 2 created.*

Moderator: *Well I think we have a hard time articulating a repeatable distinction.*

Modeler 1: *That's right.*

Moderator: *And if we can't do it, then formalizing that distinction is really hard.*

Modeler 2: *I understand. So I should get rid of the concept...*

Modeler 3: *We should get rid of the concept.*

Modeler 2: *And reassign it's children to a parent called posture.*

Moderator: *So prone body position doesn't go under body position, it goes under posture. And somebody is going to have to go through and remove all links to body position.*

Modeler 2: *OK. Because it is used as a role restriction for certain maneuvers I believe.*

Moderator: *And krep 2.24 has a special function to do that... [2.24 is not released. This was a jab at the developer to try to get this feature in 2.24].*

Modeler 1: *K-Rep Developer, it does?*

K-Rep Developer: *Maybe 2.25.*

Moderator: *So posture. We would just leave it under 3?*

Modeler 2: *Yes.*

Modeler 2: *Morning Stiffness. [see Figure A-12]*

Modeler 2: *I think Modeler 1 was right on this one. I think it's a musculoskeletal symptom.*

Moderator: *OK. But there is another issue here, that is the rigidity. Would you say that morning stiffness meets rigidity criteria?*

Modeler 2: *No. I think that they may be related in an associated fashion.*

Modeler 1: *I think on the original SNOMED they might have been siblings.*

Modeler 2: *Rigidity to me is the physical exam finding that can be cogwheel rigidity or decorticate rigidity.*

Moderator: *So actually we want to go from state 4 to state 3.*

Morning-stiffness_F-11330 has multiple end states! 1 P (and Rigidity-NOS_F-11320 (:Display_Name "Morning stiffness") (:Displayable "T") (:SNOMED_Code "F-11330") (:Synonym "Morning stiffness")) 2 P (and Muscle-property-NOS_F-11100 (:Display_Name "Morning stiffness") (:Displayable "T") (:SNOMED_Code "F-11330") (:Synonym "Morning stiffness")) 3 P (and Musculoskeletal-symptom-NOS_F-10050 (:Display_Name "Morning stiffness") (:Displayable "T") (:SNOMED_Code "F-11330") (:Synonym "Morning stiffness"))	4 P (and Rigidity-NOS_F-11320 Musculoskeletal-symptom-NOS_F-10050 (:Display_Name "Morning stiffness") (:Displayable "T") (:SNOMED_Code "F-11330") (:Synonym "Morning stiffness")) Arc 1 2 ##Modeler 2|From K-Rep DE Arc 1 4 ##Modeler 1|From K-Rep DE Arc 2 3 ##Modeler 2|From K-Rep DE --Concepts are NOT equivalent--

Figure A-12. Conflict report for "morning stiffness."

Modeler 2: *Visual acuity testing... Looks like this is between me and Modeler 3. [see Figure A-13]*

Visual-acuity-testing-NOS_P2-A0012 has multiple end states! 1 P (and Ophthalmic-exam-and-evaluation-NOS_P2-A0010 (:Display_Name "Visual acuity testing-NOS") (:Displayable "T") (:SNOMED_Code "P2-A0012") (:Synonym "Visual acuity testing-NOS")) 2 P (and Ophthalmic-exam-and-evaluation-NOS_P2-A0010 (:Display_Name "Visual acuity testing-NOS") (:Displayable "T") (:Int_Ext "I") (:SNOMED_Code "P2-A0012") (:Synonym "Visual acuity testing-NOS"))	3 P (and Ophthalmic-exam-and-evaluation-NOS_P2-A0010 Cranial-nerve-II-exam_F-3R3R4 (:Display_Name "Visual acuity testing-NOS") (:Displayable "T") (:SNOMED_Code "P2-A0012") (:Synonym "Visual acuity testing-NOS")) Arc 1 2 ##Modeler 3|From K-Rep DE Arc 1 3 ##Modeler 2|From K-Rep DE --Concepts are NOT equivalent--

Figure A-13. Conflict report for "visual acuity testing."

Moderator: *That's right. You two can duke it out.*

Modeler 2: *What's the difference between 2 and 3?*

Moderator: *Well cranial nerve exam. There is visual field testing which certainly tests cranial nerve II, And you sort of assume that CN II is working. Your visual acuity is testing your refraction.*

Modeler 3: *Visual acuity is a very high level test in the sense that it tests a lot of things. Like "Gait"*

Modeler 2: *I think 2 is the proper...*

Modeler 3: *And it needs to have the more general parent. Just like gait is a musculoskeletal function even though it is a "foot function."*

Modeler 2: *Oh I see, Modeler 3 added internal/external code. That's what the added feature is. The change I made to state 3 should be revised back to state 2.*

Moderator: *OK. So we are happy with state 2?*

Modeler 2: *State 2 is the correct state.*

Modeler 2: *Myalgia. Pretty straight forward I think. [see Figure A-14]*

Myalgia_F-11550 has multiple end states! 1 P (and Muscle-property-NOS_F-11100 (:Display_Name "Myalgia") (:Displayable "T") (:SNOMED_Code "F-11550") (:Synonym "Myalgia") (:Synonym "Muscle pain") (:Synonym "Myalgic")) 2 P (and Musculoskeletal-symptom-NOS_F-10050 (:Display_Name "Myalgia") (:Displayable "T") (:SNOMED_Code "F-11550") (:Synonym "Myalgia") (:Synonym "Muscle pain") (:Synonym "Myalgic"))	3 P (and Muscle-property-NOS_F-11100 Musculoskeletal-symptom-NOS_F-10050 (:Display_Name "Myalgia") (:Displayable "T") (:SNOMED_Code "F-11550") (:Synonym "Myalgia") (:Synonym "Muscle pain") (:Synonym "Myalgic")) Arc 1 2 ##Modeler 2|From K-Rep DE Arc 1 3 ##Modeler 1|From K-Rep DE --Concepts are NOT equivalent--

Figure A-14. Conflict report for "myalgia."

Moderator: *Is it a muscle property or a symptom?*

Modeler 2: *A symptom. So state 2.*

Modeler 2: *Eye and eyelid symptom goes under state 4 I believe. These are all my changes. [see Figure A-15]*

Modeler 3: *No they are Modeler 1's actually.*

Eye-and-eyelid-symptom-NOS_F-F1700 has multiple end states! 1 P (and Biological-function-NOS_F-00000 (:Display_Name "Eye and eyelid symptom-NOS") (:Displayable "T") (:SNOMED_Code "F-F1700") (:Synonym "Eye and eyelid symptom-NOS")) 2 P (and Biological-function-NOS_F-00000 Symptom-NOS_F-01250 (:Display_Name "Eye and eyelid symptom-NOS") (:Displayable "T") (:SNOMED_Code "F-F1700") (:Synonym "Eye and eyelid symptom-NOS")) 3 P (and Vision-NOS_F-F0000 Symptom-NOS_F-01250 (:Display_Name "Eye and eyelid symptom-NOS") (:Displayable "T") (:SNOMED_Code "F-F1700") (:Synonym "Eye and eyelid symptom-NOS"))	4 D (and Vision-NOS_F-F0000 Symptom-NOS_F-01250 (some in_Region Eyes-and-eye-appendages_T-AA000) (atleast 1 in_Region) (:Display_Name "Eye and eyelid symptom-NOS") (:Displayable "T") (:SNOMED_Code "F-F1700") (:Synonym "Eye and eyelid symptom-NOS")) Arc 1 2 ##Modeler 2|From K-Rep DE Arc 1 3 ##Modeler 1|From K-Rep DE Arc 3 4 ##Modeler 1|From K-Rep DE --Concepts are NOT equivalent--

Figure A-15. Conflict report for "eye and eyelid symptom."

Modeler 1: *You did from 1 to 2 Modeler 2.*

Moderator: *So the conflict is between state 2 and state 4.*

Modeler 1: *Right. It's not a kind of vision.*

Modeler 2: *It's not a kind of vision.*

Moderator: *Eye and eyelid symptoms, I would just put it under symptoms.*

Modeler 2: *The problem is that there are some extra facets.*

Moderator: *Well we'll be able to merge those to. So the first thing we want is symptom. And we want to delete vision, and we want do delete biological function.*

Modeler 2: *The difference is the some restriction "in region". We need to maintain that.*

Moderator: *OK. So we can keep some in_region... So here we are actually combining a couple from both people. Were their any other facets...*

Modeler 1: *The facets are the same.*

Modeler 2: *Peritoneal dialysis is Modeler 3's debacle. [see Figure A-16]*

Peritoneal-dialysis-NOS_P2-77500 has multiple end states! 1 P (and Dialysis-NOS_P2-77r00 (:Display_Name "Peritoneal dialysis-NOS") (:Displayable "T") (:SNOMED_Code "P2-77500") (:Synonym "Peritoneal dialysis-NOS")) 2 P (and Dialysis-NOS_P2-77r00 Patient-education-urological-or-renal_PA-60r06 (:Display_Name "Peritoneal dialysis-NOS") (:Displayable "T") (:SNOMED_Code "P2-77500") (:Synonym "Peritoneal dialysis-NOS")) 3 P (and Dialysis-NOS_P2-77r00 (:Display_Name "Peritoneal dialysis-NOS") (:Displayable "T") (:Int_Ext "E") (:SNOMED_Code "P2-77500") (:Synonym "Peritoneal dialysis-NOS"))	4 P (and Dialysis-NOS_P2-77r00 (:Display_Name "Peritoneal dialysis-NOS") (:Displayable "T") (:Int_Ext "E") (:Referral_Department Facility-business-office_5111) (:SNOMED_Code "P2-77500") (:Synonym "Peritoneal dialysis-NOS")) Arc 1 2 ##Modeler 3|From K-Rep DE Arc 1 3 ##Modeler 3|From K-Rep DE Arc 3 4 ##Modeler 3|From K-Rep DE --Concepts are NOT equivalent--

Figure A-16. Conflict report for "peritoneal dialysis."

Moderator: *Oh. You did a backtrack. So do you really want to say 4? The choice is between 2 and 4.*

Modeler 3: *Yes...*

Moderator: *Peritoneal dialysis is not a patient education thing.*

Modeler 3: *No.*

Moderator: *But there is education associated with it. So I think you want to say 4, which is what you wanted to say in the first place. This is just kind of a side effect of the program, in that if you back track, it doesn't know what to do with it.*

Modeler 3: *This was right next to pertional dialysis training which is both a peritoneal dialysis and training.*

Moderator: *Alright. Tetany... [see Figure A-17]*

Modeler 2: *I think its a symptom, but I also think it is a motor exam finding.*

Modeler 3: *Wait a second. Now we are in the situation where the same finding seems to be in a very different nature as a symptom. That is that*

Tetany-NOS_F-11370 has multiple end states! 1 P (and Muscle-property-NOS_F-11100 (:Display_Name "Tetany-NOS") (:Displayable "T") (:SNOMED_Code "F-11370") (:Synonym "Tetany-NOS")) 2 P (and Motor-exam_F-3RR44 (:Display_Name "Tetany-NOS") (:Displayable "T") (:SNOMED_Code "F-11370") (:Synonym "Tetany-NOS"))	3 P (and Muscle-property-NOS_F-11100 Musculoskeletal-symptom-NOS_F-10050 (:Display_Name "Tetany-NOS") (:Displayable "T") (:SNOMED_Code "F-11370") (:Synonym "Tetany-NOS")) Arc 1 2 ##Modeler 2|From K-Rep DE Arc 1 3 ##Modeler 1|From K-Rep DE --Concepts are NOT equivalent--

Figure A-17. Conflict report for "tetany."

experiencing tetany is so different from finding tetany that we want to make it both? Are you sure we want to do that?

Modeler 2: *That's my thinking. As opposed to the associated finding relationship.*

Modeler 3: *Now for cough you could say, for instance, painful cough is a symptom, barking cough is a sign.*

Modeler 2: *The problem is that "cough" has to go somewhere. And it will have both of it's children with this.*

Modeler 3: *The only reason I see to make a clinical finding both a symptom and a sign is when it has children which are clearly differentiated. But I would not do it just because it feels different to have tetany than to observe tetany. I mean... What do you think? We should have a sense of why we should have something be both, because many things could be both. Which is another option. Everything could be both.*

Modeler 2: *There are, as I go through these physical exam things, a lot of them are both. I'm leaving them as symptoms, and I'm associating them or I'm pulling them out of symptoms completely. It's a case by case thing.*

Modeler 1: *Tetany is really a muscle function that is continuous muscle contraction. Actually continuous muscle stimulation.*

Modeler 2: *That's right.*

Modeler 3: *So to me it seems like a sign.*

Modeler 2: *Well, it is a physiologic function.*

Modeler 1: *In it's truest definition.*

Modeler 3: *Well, make it a function if you want.*

Modeler 1: *Well it was a muscle property.*

Modeler 2: *Well, it's a waste though to just leave it as a function. We have to put it somewhere that is useful. Either a symptom or a sign.*

Modeler 3: *Well all I'm asking is why is it a symptom. What are the reasons we make something that is already a sign a symptom.*

Modeler 2: *Because of what you said. It's a presenting complaint for a specific set of diseases.*

Modeler 3: *Is that a complaint?*

Modeler 1: *We are not saying that all symptoms are complaints. They are limited to just lay terminology. Now people present with epistonic posturing. Which is what you commonly see in advanced tetanus.*

Modeler 2: *Correct.*

Modeler 1: *That's a disease. Or they present with trismus or lock-jaw, which really is the common terminology symptoms.*

Moderator: *Well I had a patient that came in to the ER with their jaw in spasm stating "I have low calcium." That patient had previously had a parathyroidectomy, and so they knew what they had.*

..... Story.....

Modeler 2: *Well in this case, I think that probably sign or motor exam finding is the proper place.*

Moderator: *OK. So tetany goes under motor exam, not muscle property or musculoskeletal symptom?*

(agreement).

Modeler 2: *Now this next one, Skin_Rash has been deleted. The correct one is Skin-rash. [see Figure A-18]*

```
Skin_Rash has multiple end states!

1 P (and Skin-exam
  (some has_Color Skin-finding-color_M-04R10)
  (some has_Morphologic_Features Rash,_NOS)
  (:Design_Note "")
  (:Diagnostic_Specificity Eczema-NOS_D0-10100)
  (:Diagnostic_Specificity Psoriasis-NOS_D0-22100)
  (:Diagnostic_Specificity Impetigo-NOS_DE-12200)
  (:Diagnostic_Specificity Varicella-NOS_DE-32300)
  (:Diagnostic_Specificity Measles_DE-33910)
  (:Diagnostic_Specificity Rubeola_scarlatinosa_DE-39100)
  (:Diagnostic_Specificity Wart)
  (:Diagnostic_Specificity Pityriasis-NOS_D0-22160)
  (:Diagnostic_Specificity Seborrheic-keratosis_M-72750)
  (:Diagnostic_Specificity Actinic-keratosis_M-72850)
  (:Diagnostic_Specificity Pityriasis_alba_D0-22163)
  (:Display_Name "Skin Rash/Erruption")
  (:Displayable "T")
  (:SNOMED_Code "")
  (:Synonym "skin rash"))

2 P (and Skin-exam
  (some has_Color Skin-finding-color_M-04R10)
  (some has_Morphologic_Features Rash,_NOS)
  (:Design_Note "")
  (:Diagnostic_Specificity Eczema-NOS_D0-10100)
  (:Diagnostic_Specificity Psoriasis-NOS_D0-22100)
  (:Diagnostic_Specificity Impetigo-NOS_DE-12200)
  (:Diagnostic_Specificity Varicella-NOS_DE-32300)
  (:Diagnostic_Specificity Measles_DE-33910)
  (:Diagnostic_Specificity Rubeola_scarlatinosa_DE-39100)
  (:Diagnostic_Specificity Wart)
  (:Diagnostic_Specificity Pityriasis-NOS_D0-22160)
  (:Diagnostic_Specificity Seborrheic-keratosis_M-72750)
  (:Diagnostic_Specificity Actinic-keratosis_M-72850)
  (:Diagnostic_Specificity Pityriasis_alba_D0-22163)
  (:Display_Name "Skin Rash/Erruption")
  (:Displayable "T")
  (:SNOMED_Code "F-9R765")
  (:Synonym "skin rash"))

3 P (and Skin-exam
  (some has_Color Color_M-04000)
  (some has_Morphologic_Features Rash,_NOS)
  (:Design_Note "")
  (:Diagnostic_Specificity Eczema-NOS_D0-10100)
  (:Diagnostic_Specificity Psoriasis-NOS_D0-22100)
  (:Diagnostic_Specificity Impetigo-NOS_DE-12200)
  (:Diagnostic_Specificity Varicella-NOS_DE-32300)
  (:Diagnostic_Specificity Measles_DE-33910)
  (:Diagnostic_Specificity Rubeola_scarlatinosa_DE-39100)
  (:Diagnostic_Specificity Wart)
  (:Diagnostic_Specificity Pityriasis-NOS_D0-22160)
  (:Diagnostic_Specificity Seborrheic-keratosis_M-72750)
  (:Diagnostic_Specificity Actinic-keratosis_M-72850)
  (:Diagnostic_Specificity Pityriasis_alba_D0-22163)
  (:Display_Name "Skin rash")
  (:Displayable "T")
  (:SNOMED_Code "F-9R765")
  (:Synonym "Skin rash")
  (:Synonym "Skin erruption"))

4 P (and Skin-exam
  (:Design_Note "")
  (:Display_Name "Skin rash")
  (:Displayable "T"))

5 P (and Skin-exam
  Skin-Symptoms_F-42R00
  (some has_Color Skin-finding-color_M-04R10)
  (some has_Morphologic_Features Rash,_NOS)
  (:Design_Note "")
  (:Diagnostic_Specificity Eczema-NOS_D0-10100)
  (:Diagnostic_Specificity Psoriasis-NOS_D0-22100)
  (:Diagnostic_Specificity Impetigo-NOS_DE-12200)
  (:Diagnostic_Specificity Varicella-NOS_DE-32300)
  (:Diagnostic_Specificity Measles_DE-33910)
  (:Diagnostic_Specificity Rubeola_scarlatinosa_DE-39100)
  (:Diagnostic_Specificity Wart)
  (:Diagnostic_Specificity Pityriasis-NOS_D0-22160)
  (:Diagnostic_Specificity Seborrheic-keratosis_M-72750)
  (:Diagnostic_Specificity Actinic-keratosis_M-72850)
  (:Diagnostic_Specificity Pityriasis_alba_D0-22163)
  (:Display_Name "Skin Rash/Erruption")
  (:Displayable "T")
  (:SNOMED_Code "")
  (:Synonym "skin rash"))

Arc 1 2
##Modeler 2|From K-Rep DE

Arc 1 5
##Modeler 1|From K-Rep DE

Arc 2 3
##Modeler 2|From K-Rep DE

Arc 3 4
##Modeler 2|From K-Rep DE

--Concepts are NOT equivalent--
```

Figure A-18. Conflict report for "skin rash."

Modeler 1: *So you did this yesterday?*

Modeler 2: *Yes.*

Modeler 1: *You weren't supposed to make any changes until these were resolved you bone-head.*

Modeler 2: *No, I thought I was supposed to continue working.*

Moderator: *We can resolve these. We can import his change set and then go through it. Except, some of his changes. If they overlap with the conflicts here, you will have to do some redundant work.*

Modeler 1: *We have to resolve these conflicts*

Moderator: *And then bring his stuff in, and then resolve them again.*

Modeler 1: *Sure.*

Modeler 2: *Yes, because I'm sure I've done another thousand at least (joke).*

Modeler 1: *I'm sure you have.*

.....

Moderator: *How did you delete this concept? Does K-Rep have a delete button?*

K-Rep Developer: *Yes, you can delete a concept as long as there aren't any other concepts that are dependent upon it.*

Moderator: *So what get's written in the commit journal when that happens?*

K-Rep Developer: *Delete concept*

Moderator: *So that's it?*

Modeler 2: *You'll also see some activity in the commit journal before. You'll see the concept that I modified related to the one I deleted.*

Moderator: *It's just that my program will probably break or barf when it finds delete concept.*

(Discussion of how to delete a concept in K-Rep.)

At the interregional meeting, I talked about why delete concept was a really bad idea. I think we need to work on deleting concepts in a different way. Because if I'm running a configuration management

environment, and I have this history, and then all of a sudden you delete the concept, then I delete its history. What I would like to propose is that when you want to delete a concept, you can write a perl script that can go through and strip the concepts from the datafile, and so we can tag them for deletion, i.e. go through this configuration management process, resolve the conflicts, dump a new version, and then strip the concepts from there, and then go on.

Modeler 1: *That's fine. If that's necessary, then we can do that.*

Moderator: *It still would be a good idea if you did all that dependency removal prior to stripping it out.*

...

Appendix B
The Knowledge Representation System Specification

The Knowledge Representation System Specification (KRSS) defines a standard *syntax* for description-logic-based knowledge-representation systems. Interested readers may download the complete draft of KRSS from the Defense Advanced Projects Research Agency Knowledge Sharing web page (Patel-Schneider et al., 1993).[1] KRSS itself does not require compliant systems to reason over all the distinctions defined by the KRSS syntax. Rather, compliant implementation are only required to *parse* the entire language, and to perform non-trivial inference over a subset of first-order predicate logic with model-sets (Donini, Lenzerini, Nardi, Schaerf & Nutt, 1992).[2] This appendix does not provide a comprehensive discussion of all the representational requirements of KRSS, but rather gives an overview of the KRSS syntax, and simple examples of how terminological definitions

1. http://www-ksl.stanford.edu/knowledge-sharing/papers/index.html#dl-spec
2. The draft specification requires that such inference be logically correct and complete over the subset of logic supported by any particular implementation. Therefore, if the system determines that "A is a B," then "A" must truly be a "B" in all possible interpretations of statements within the knowledge-representation system (correctness). Further, the system must answer "A is a B" if there are statements within the system that would allow one to properly conclude that "A is a B" (completeness).

are constructed. Table B-1 presents the principle concept-forming operators and terminological axioms of KRSS.

Table B-1. Concept forming operators and the terminological axioms of the Knowledge Representation System Specification.

Concrete Form	Abstract Form	Description
Concept-Forming Operators		
top	$\top$	The "top" of the hierarchy from which all concepts descend.
bot	$\bot$	The imaginary concept at the "bottom" of the hierarchy.
(and C_1...C_n)	$C_1 \wedge \ldots \wedge C_n$	A logical conjunction of concepts C_1 through C_n.
(or C_1...C_n)	$C_1 \vee \ldots \vee C_n$	At least one and possibly more of concepts C_1 through C_n.
(not C_n)	$\neg C$	A logical negation of the concept C.
(some R C)	$\exists R{:}C$	There exists at least one relationship R constrained to be of type C.
(all R C)	$\forall R{:}C$	All relationships R are constrained to be of type C.
Terminological Axioms		
(define-concept N C)	$N \equiv C$	C defines necessary and sufficient conditions for N.
(define-primitive-concept N C)	$N \sqsubseteq C$	C defines necessary conditions for N.
(disjointprimitives P1 P2)	$P_1 \wedge P_2 = \bot$	Primitives P_1 and P_2 are mutually exclusive.

Each concept in a knowledge base must be identified with a unique name. New primitive[3] concepts can be introduced into the terminology with the "define-primitive-concept" operator. For example, the concept of "sexual genotype" and "human" can be introduced into the knowledge-representation system with the following statements:[4]

(define-primitive-concept SEXUAL-GENOTYPE) **(Statement 1)**

(define-primitive-concept HUMAN) **(Statement 2)**

Once introduced, the concepts of "male" and "female" sexual genotypes can be introduced into the knowledge base as subconcepts of sexual genotype with the following statements:

(define-primitive-concept MALE-GENOTYPE SEXUAL-GENOTYPE) **(Statement 3)**

(define-primitive-concept FEMALE-GENOTYPE SEXUAL-GENOTYPE) **(Statement 4)**

Now that the knowledge base has the concepts necessary to define a male and female human, they can be defined using a defining relationship known as a "role." Roles can be introduced into a KRSS knowledge base using a "define-primitive-role" operator. A "has genotype" role would be introduced into the knowledge base with the following statement:

(define-primitive-role HAS-GENOTYPE) **(Statement 5)**

This role can now be used together with the concepts HUMAN, MALE-GENOTYPE, and FEMALE-GENOTYPE to define "male human" and "female human:"

---

3. A concept is considered primitive if it is defined with necessary, but not sufficient conditions. For example, it is necessary for a human to be a mammal, but to be a human, there are additional requirements (such as the number of chromosomes and other features shared by all humans). If a human concept is defined simply as a mammal, then it should be designated as a primitive concept. A concept may necessarily be primitive if the underlying description-logic dialect lacks the expressive power to completely represent necessary and sufficient conditions. Alternatively, a concept may be primitive simply because the modeler chooses not to represent certain necessary conditions.
4. Whenever a concept does not specify a parent concept from which it inherits, it is assumed to descend from the "top" concept presented in Table B-1.

```
(define-concept MALE-HUMAN
        (and HUMAN
            (some HAS-GENOTYPE MALE-GENOTYPE))          (Statement 6)

(define-concept FEMALE-HUMAN
        (and HUMAN
            (some HAS-GENOTYPE FEMALE-GENOTYPE))        (Statement 7)
```

There are several things to note about these last two statements. First, they use the define-concept operator rather than the define-primitive-concept operator. Unlike the previous *primitive* concepts (see statements 1-4), these statements have specified *necessary and sufficient conditions* to be either a male or a female human. Second, previous defining statements only have one or fewer necessary conditions (the statements immediately to the right of the concept label), these statements have two necessary conditions (the HUMAN and the HAS-GENOTYPE statements) and are therefore joined with an "and" clause, which is represented in prefix notation. KRSS represents both "and" and "or" clauses in prefix notation as specified in Table B-1 and demonstrated in Statements 6 and 7). Third, the HAS-GENOTYPE role is introduced with a *restriction* ("*some*" in this case), that specifies the scope of the role (the abstract form and description of some and all role restrictions are presented in Table B-1).

Modelers can continue to introducing concepts and roles until the desired terminology is completed. A complete terminology system may range from 20 to 200,000 concepts depending upon the intended domain and purpose of the system.

Bibliography

American Medical Association. (1995) Physicians' Current Procedural Terminology: CPT 1996. Chicago: American Medical Association.

American Psychiatric Association Task Force on Nomenclature and Statistics. (1980) Diagnostic and Statistical Manual of Mental Disorders, Third Edition, (DSM-III). Washington, D.C.: American Psychiatric Association.

ANSI/IEEE Standard 729. (1983) Glossary of software engineering terminology. New York: IEEE Press.

Audet A. M. and Scott H. D. (1993) The Uniform Clinical Data Set: An evaluation of the proposed national database for Medicare's quality review program. Annals of Internal Medicine, 119(12):1209-13.

Barghouti N. S. and Kaiser G. E. (1991) Concurrency control in advanced database applications. ACM Computing Surveys, 23(3):269-317.

Barker V. E. and O'Connor D. E. (1989) Expert systems for configuration at Digital: XCON and beyond. Communications of the ACM, 32(3):298-318.

Baud R., Lovis C., Alpay L., *et al*. (1993) Modeling for natural language understanding. In: Safran C., ed. Seventeenth Annual Symposium on Computer Applications in Medical Care. Washington, D.C.: McGraw-Hill, 289-93.

Bell D. S., Greenes R. A. and Doubilet P. (1992) Form-based clinical input from a structured vocabulary: Initial application in ultrasound reporting. In: Frisse M. E., ed. Sixteenth Annual Symposium on Computer Applications in Medical Care. Baltimore, MD: McGraw-Hill, 789-90.

Benoit R. G., Canfield K., Teitelbaum S., *et al*. (1992) Frame-based clinical data entry: Design issues for physician acceptance. In: Lun K. C., Degoulet P., Piemme T. E. and Rienhoff O., eds. MEDINFO 92. Geneva, Switzerland: North Holland: Elsevier Science Publishers, 1280-85.

Bergeron B. (1991) Iliad: a diagnostic consultant and patient simulator. MD Computing, 30:187-93.

Bernauer J. (1991) Conceptual graphs as an operational model for descriptive findings. In: Clayton P. D., ed. Fifteenth Annual Symposium on Computer Applications in Medical Care. Washington, D.C.: McGraw-Hill, 214–18.

Berwick D. M., Godfrey A. B. and Roessner J. (1990) Curing Health Care: New Strategies for Quality Improvement. San Francisco: Jossey-Bass.

Blumberg M. S. (1991) Biased Estimates of Expected Acute Myocardial Infarction Mortality using MedisGroups Admission Severity Groups. Journal of the American Medical Association, 265(22):2965-70.

Board of Directors of the American Medical Informatics Association. (1994) Standards for Medical Identifiers, Codes, and Messages Needed to Create an Efficient Computer-stored Medical Record. Journal of the American Medical Informatics Association, 1(1):1-7.

Bohinski R. C. (1973) Modern Concepts in Biochemistry. (3rd ed.) Boston: Allyn and Bacon, Inc.

Boolos G. (1993) The Logic of Provability. Cambridge: Cambridge University Press.

Boolos G. S. and Jeffrey R. C. (1989) Computability and Logic. (3rd ed.) Cambridge: Cambridge University Press.

Borgida A. R., Brachman R. J., McGuinness D. L. and Resnick L. A. (1989) CLASSIC: A Structured Data Model for Objects. ACM SIGMOD International Conference on Management of Data. Association for Computing Machinery, 59-67.

Bowen O. R. and Roper W. L. (1987) Medicare Hospital Mortality Information, 1986. U.S. Dept. of Health and Human Services Publication 00744. Health Care Financing Administration.

Brachman R. J., Fikes R. E. and Levesque H. J. (1983) Krypton: a functional approach to knowledge representation. COMPUTER, 16(10):67-73.

Brachman R. J. and Levesque H. J. (1984) The Tractability of Subsumption in Frame-Based Description Languages. Proceedings of AAAI-84, 34-37.

Brachman R. J., McGuinness D. L., Patel-Schneider P. F., Resnick L. A. and Borgida A. (1991) Living With Classic: When and how to use a KL-One-like language. In: Sowa J. F., ed. Principles of Semantic Networks: Explorations in the representation of knowledge. San Mateo, California: Maurgan Kaufman Publishers, Inc., 401-56.

Brachman R. J. and Schmolze J. G. (1985) An Overview of the KL-One Knowledge Representation System. Cognitive Science, 9(2):171-202.

Brill D. (1993) Loom Reference Manual: Version 2.0. University of Southern California.

California Legislature. (1991) Assembly Bill no. 524. State of California.

Campbell J. R., Carpenter P., Sneiderman C., Cohn S., Chute C. G. and Warren J. (1997) Phase II Evaluation of Clinical Coding Schemes: Completeness, Taxonomy, Mapping, Definitions, and Clarity. Journal of the American Medical Informatics Association, 4(3):238-51.

Campbell K. E. (1994) Distributed Development of a Logic-Based Controlled Medical Terminology [Dissertation Proposal (unpublished)]. Stanford University.

Campbell K. E., Cohn S. P., Chute C. G., Rennels G. and Shortliffe E. H. (1996) Gálapagos: Computer-Based Support for Evolution of a Convergent Medical Terminology. In: Cimino J. J., ed. AMIA Annual Fall Symposium. Washington, D.C.: Hanley & Belfus, Inc., 269-73.

Campbell K. E., Das A. K. and Musen M. A. (1994) A Logical Foundation for Representation of Clinical Data. Journal of the American Medical Informatics Association, 1(3):218-32.

Campbell K. E. and Musen M. A. (1992a) Creation of a Systematic Domain for Medical Care: The need for a comprehensive patient-description vocabulary. In: Lun K. C., Degoulet P., Piemme T. E. and Rienhoff O., eds. Proceedings of MEDINFO 92. Geneva, Switzerland: North Holland: Elsevier Science Publishers, 1437-42.

Campbell K. E. and Musen M. A. (1992b) Representation of Clinical Data using SNOMED III and Conceptual Graphs. In: Frisse M. E., ed. Sixteenth Annual Symposium on Computer Applications in Medical Care. Baltimore, MD: McGraw-Hill, 354-58.

Campbell K. E., Wieckert K., Fagan L. M. and Musen M. A. (1993) A Computer-based Tool for Generation of Progress Notes. In: Safran C., ed. Seventeenth Annual Symposium on Computer Applications in Medical Care. Washington, D.C.: McGraw-Hill, 284-88.

Cavalli-Sforza V., Weiner A. W. and Lesgold A. (1994) Software Support for Students Engaging in Scientific Activity and Scientific Controversy. Science Education, 78:577-99.

Chute C. G., Cohn S., Campbell K. E., Oliver D. and Campbell J. R. (1996) The Content Coverage of Clinical Classifications. Journal of the American Medical Informatics Association, 3(3):224-33.

Cimino J. J., Clayton P. D., Hriscsak G. and Johnson S. B. (1994) Knowledge-based Approaches to the Maintenance of a Large Controlled Medical Terminology. Journal of the American Medical Informatics Association, 1(1):35-50.

Cimino J. J., Hripcsak G., Johnson S. B. and Clayton P. D. (1989) Designing an Introspective, Controlled Medical Vocabulary. In: Kingsland L. W., ed. Thirteenth Annual Symposium on Computer Applications in Medical Care. Washington, D.C.: IEEE Computer Society Press, 513-18.

Côté R. A., Rothwell D. J., Palotay J. L., Beckett R. S. and Brochu L., eds. (1993) The Systematized Nomenclature of Medicine: SNOMED International. Northfield, Illinois: College of American Pathologists.

Dart S. A. (1992) Parallels in Computer-aided Design Framework and Software Development Environment Efforts. Technical Report CMU/SEI-92-TR-9, ESC-TR-92-009. Software Engineering Institute, Carnegie Mellon University.

Das A. K., Tu S. W., Purcell G. P. and Musen M. A. (1992) An Extended SQL for Temporal Data Management in Clinical Decision-support Systems. In: Frisse M. E., ed. Sixteenth Annual Symposium on Computer Applications in Medical Care. Baltimore, MD: McGraw-Hill, 128-32.

Dennett D. C. (1995) Darwin's Dangerous Idea: Evolution and the Meanings of Life. New York: Simon & Schuster.

Dick R. S. and Steen E. B., eds. (1991) The Computer-based Patient Record: An essential technology for health care. Washington, D.C.: National Academy Press.

Donini F. M., Lenzerini M., Nardi D., Schaerf A. and Nutt W. (1992) Adding Epistemic Operators to Concept Languages. Third International Conference on Principles of Knowledge Representation and Reasoning. Cambridge, MA: Morgan Kaufmann, 342-53.

Enderton H. B. (1972) A Mathematical Introduction to Logic. San Diego, CA: Academic Press.

Eswaran K., Gray J., Lorie R. and Traiger I. (1976) The Notions of Consistency and Predicate Locks in a Database System. Communications of the ACM, 19(11):624-32.

Evans D. A., Cimino J. J., Hersh W. R., Huff S. M. and Bell D. S. (1994) Toward a Medical-concept Representation Language. Journal of the American Medical Informatics Association, 1(3):207-17.

Feiler P. H. (1991) Configuration Management Models in Commercial Environments. Technical Report CMU/SEI-91-TR-7, ESD-91-TR-7. Software Engineering Institute, Carnegie Mellon University.

Feldman S. I. (1979) Make: A program for maintaining computer programs. Software—Practice & Experience, 9(4):255-65.

Fischer P. J., Stratmann W. C., Lundsgarrde H. P. and Steele D. J. (1980) User reaction to PROMIS: Issues related to acceptability of medical innovations. In: Fourth Annual Sym-

posium on Computer Applications in Medical Care. Washington, D.C.: IEEE Press, 1722–30.

Fitting M. (1990) First-Order Logic and Automated Theorem Proving. New York: Springer-Verlag. (Gries D., ed. Texts and Monographs in Computer Science; vol 242).

Flood A. B. (1990) Peaks and Pits of using Large Databases to Measure Quality of Care. International Journal of Technology Assessment in Health Care, 6:253-62.

Flores M. M. (1993) Insurers Rating Doctors by What They Charge. The Seattle Times November 21:A1.

Friedman C., Cimino J. J. and Johnson S. B. (1993) A Conceptual Model for Clinical Radiology Reports. In: Safran C., ed. Seventeenth Annual Symposium on Computer Applications in Medical Care. Washington, D.C.: McGraw-Hill, 829-33.

Garcia-Molina H. (1983) Using Semantic Knowledge for Transaction Processing in a Distributed Database. ACM Transactions on Database Systems, 8(2):186-213.

Garcia-Molina H. and Salem K. (1987) Sagas. ACM SIGMOD 1987 Annual Conference. ACM Press, 249-59.

Garey M. R. and Johnson D. S. (1979) Computers and Intractability: A guide to the theory of NP-Completeness. New York: W. H. Freeman and Company.

Genesereth M. R. and Fikes R. E. (1992) Knowledge Interchange Format, version 3.0 reference manual. Technical Report Logic-92-1. Computer Science Department, Stanford University.

Gruber T. (1990) The Role of Standard Knowledge Representation for Sharing Knowledge-Based Technology. Technical Report KSL-90-53. Stanford University.

Gruber T. R. (1993) Toward Principles for the Design of Ontologies Used for Knowledge Sharing. Technical Report KSL 93-04. Stanford University.

Hall K. (1991) A Framework for Change Management in a Design Database [Doctoral Dissertation]. Stanford University.

Hannan E. L., Kilburn H., Jr., Lindsey M. L. and Lewis R. (1992) Clinical versus Administrative Databases for CABG Surgery. Does it matter? Medical Care, 30(10):892-907.

Hayes P. J. (1977) In Defense of Logic. In: Reddy R., ed. International Joint Conference on Artificial Intelligence–1977. Cambridge, Massachusetts: Carnegie-Mellon University, 559-65.

Henry S. B., Holzemer W. L., Reilly C. A. and Campbell K. E. (1994) Terms used by Nurses to Describe Patient Problems: Can SNOMED III represent nursing concepts in the patient record? Journal of the American Medical Informatics Association, 1(1):61-74.

Horwitz S., Prins J. and Reps T. (1989) Integrating Noninterfering Versions of Programs. ACM Transactions on Programming Languages and Systems, 11(3):345-87.

Hsia D. C., Ahern C. A., Ritchie B. P., Moscoe L. M. and Krushat W. M. (1992) Medicare Reimbursement Accuracy under the Prospective Payment System, 1985 to 1988. Journal of the American Medical Association, 268(7):896-99.

Hutchins R. M., ed. (1952) The Works of Aristotle: Volume I. Chicago: Encyclopedia Britannica. (Hutchins R. M., ed. Great Books of the Western World; vol 8).

International Standards Organization. (1990) International Standard ISO 1087: Terminology—Vocabulary. Geneva, Switzerland: International Standards Organization.

Johnson S. B., Aguirre A., Peng P. and Cimino J. (1993) Interpreting Natural Language Queries using the UMLS. In: Safran C., ed. Seventeenth Annual Symposium on Computer Applications in Medical Care. Washington, D.C.: McGraw-Hill, 294-98.

Jollis J. G., Ancukiewicz M., DeLong E., Pryor D. B., Muhlbaier L. H. and Mark D. B. (1993) Discordance of Databases Designed for Claims Payment versus Clinical Information Systems. Annals of Internal Medicine, 119:844-50.

Kahn M. G. (1993) A Reference Model for Health Information Systems: A White Paper. The Reference Model Group, Health Information Trial Systems (HITS) Project, Microelectronics and Computer Technology Corporation.

Katz R. H. (1990) Toward a Unified Framework for Version Modeling in Engineering Databases. ACM Computing Surveys, 22(4).

Kent D. L., Shortliffe E. H., Carlson R. W., Bischoff M. B. and Jacobs C. D. (1985) Improvements in Data Collection through Physician use of a Computer-based Chemotherapy Treatment Consultant. Journal of Clinical Oncology, 3(10):1409-17.

Kuhn K., Zemmler M. R. and Heinlein D. R. (1993) Structured Data Collection and Knowledge-based User Guidance for Abdominal Ultrasound Reporting. In: Safran C., ed. Seventeenth Annual Symposium on Computer Applications in Medical Care. Washington, D.C.: McGraw-Hill, 311-15.

Kung H. and Robinson J. (1981) On Optimistic Methods for Concurrency Control. ACM Transactions on Database Systems, 6(2):213-26.

Kuperman G. J., Gardner R. M. and Pryor T. A. (1991) HELP: A dynamic hospital information system. New York: Springer Verlag.

Lamiell J. M., Zbigniew M. W. and Isaacks J. (1993) Computer Auditing of Surgical Operative Reports written in English. In: Safran C., ed. Seventeenth Annual Symposium on Computer Applications in Medical Care. Washington, D.C.: McGraw-Hill, 269-73.

Lau L. M. and Warner H. R. (1992) Performance of a Diagnostic System (Iliad) as a Tool for Quality Assurance. Computers and Biomedical Research, 25:314-23.

Lenert L. A. and Tovar M. (1993) Automated Linkage of Free-text Descriptions of Patients with a Practice Guideline. In: Safran C., ed. Seventeenth Annual Symposium on Computer Applications in Medical Care. Washington, D.C.: McGraw-Hill, 274-78.

Levesque H. J. and Brachman R. J. (1985) A Fundamental Tradeoff in Knowledge Representation and Reasoning (revised version). In: Brachman R. J. and Levesque H. J., eds. Readings in Knowledge Representation. San Francisco: Morgan Kaufman, 42-70.

Lindberg D. A. B., Humphreys B. L. and McCray A. T. (1993) The Unified Medical Language System. Methods of Information in Medicine, 32:281-91.

Lipow S. S., Fuller L. F., Keck K. D., *et al*. (1996) Suggesting Structural Enhancements to SNOMED International. In: Cimino J. J., ed. AMIA Annual Fall Symposium. Washington, D.C.: Hanley & Belfus, Inc., 901.

Luft H. S. and Romano P. S. (1993) Chance, Continuity, and Change in Hospital Mortality Rates: Coronary artery bypass graft patients in California hospitals, 1983 to 1989. Journal of the American Medical Association, 270(3):331-37.

Lynch N. A. (1983) Multilevel Atomicity: A new correctness criterion for database concurrency control. ACM Transactions on Database Systems, 8(4):484-502.

Masarie F. E., Miller R. A., Bouhaddou O., Nunzia B. G. and Warner H. R. (1991) An Interlingua for Electronic Interchange of Medical Information: Using frames to map between clinical vocabularies. Computers and Biomedical Research, 24(4):379-400.

Mays E., Dionne R. and Weida R. (1991) K-Rep System Overview. SIGART Bulletin, 2(3):93-97.

Mays E., Weida R., Dionne R., *et al*. (1996) Scalable and Expressive Medical Terminologies. In: Cimino J. J., ed. AMIA Fall Symposium. Washington, D.C.: Hanley & Belfus, Inc., 259-63.

McDonald C. J., Blevins L., Tierney W. M. and Martin D. K. (1988) The Regenstrief Medical Records. MD Computing, 5(5):34-47.

McDonald C. J., Hui S. L., Smith D. M., *et al*. (1984) Reminders to Physicians from an Introspective Computer Medical Record: A two-year randomized trial. Annals of Internal Medicine, 100(1):130-38.

McGregor J. J. (1982) Backtrack Search Algorithms and the Maximal Common Subgraph Problem. Software Practice and Experience, 12:23-34.

Miller R. A. (1994) Medical Diagnostic Decision Support Systems—Past, Present and Future: A threaded bibliography and brief commentary. Journal of the American Medical Informatics Association, 1(1):8-27.

Miller R. A., Masarie F. E. and Myers J. D. (1986) "Quick Medical Reference" for Diagnostic Assistance. MD Computing, 3:34-38.

Moser M. G. (1983) An Overview of NIKL: The new implementation of KL-ONE. In: Sidner C., Bates M., Bobrow R., eds. Research in knowledge representation for natural language understanding, annual report (BBN Report No. 5421). Cambridge, MA: Bolt, Bernek and Newman,

Musen M. A. (1992) Dimensions of Knowledge Sharing and Reuse. Computers and Biomedical Research, 25:435-67.

Musen M. A., Carlson R. W., Fagan L. M., Deresinski S. C. and Shortliffe E. H. (1992) T-Helper: Automated support for community-based clinical research. In: Frisse M. E., ed. Sixteenth Annual Symposium on Computer Applications in Medical Care. Baltimore, MD: McGraw-Hill, 719-723.

Musen M. A., Weickert K. E., Miller E. T., Campbell K. E. and Fagan L. M. (1995) Development of a Controlled Medical Terminology: Knowledge acquisition and knowledge representation. Methods of Information in Medicine, 34(1):85-95.

Naeymi-Rad F., Almeida F. and Trace D. (1992) IMR-Entry (Intelligent medical record-entry). In: Frisse M. E., ed. Sixteenth Annual Symposium on Computer Applications in Medical Care. Baltimore, MD: McGraw-Hill, 783-84.

National Center for Health Statistics. (1995) The International Classification of Diseases, 9th revision, Clinical Modification (ICD-9-CM). DHHS Publication No. (PHS) 80-1260. U.S. Department of Health and Human Services.

National Library of Medicine. (1992) Medical Subject Headings. NTIS NLM-MED-92-01. NLM.

National Research Council. (1994) Academic Careers for Experimental Computer Scientists and Engineers. Washington, D.C.: National Academy Press.

Nebel B. (1988) Computational Complexity of Terminological Reasoning in BACK. Artificial Intelligence, 34(3):371-83.

Neches R., Fikes R. E., Finin T., *et al*. (1991) Enabling Technology for Knowledge Sharing. AI Magazine, 12(3):16-36.

Norman D. A. and Draper S. W. (1986) User Centered System Design: New perspectives on human-computer interaction. Hillsdale, NJ: Lawrence Erlbaum Associates.

Patel-Schneider P. F., Swartout B. and KRSS working group of the DARPA Knowledge Sharing Effort. (1993) Working Version (Draft): Description Logic Specification from the KRSS Effort. http://www-ksl.stanford.edu/knowledge-sharing/papers/index.html#dl-spec.

Perez S. and Sarris A. (1993) Information Resource Dictionary System (IRDS) Conceptual Schema (CS). ANSI X3H4/92-003, ISO/IEC JTC1/SC21 N7486. American National Standards Institute X3H4 IRDS and the International Organization for Standardization.

Poon A., Fagan L. M. and Shortliffe E. H. (1996) The Pen Ivory Project: Exploring User-interface Design for the Selection of Items from Large Controlled Vocabularies of Medicine. Journal of the American Medical Informatics Association, 3(2):168-83.

Rassinoux A. M., Baud R. H. and Scherrer J. R. (1992) Conceptual Graphs Model Extension for Knowledge Representation of Medical Texts. In: Lun K. C., Degoulet P., Piemme T. E. and Rienhoff O., eds. MEDINFO 92. Geneva, Switzerland: North Holland: Elsevier Science Publishers, 1368-74.

Rector A. L. (1993) Medical Concepts and Medical Records. AIM-CEN European Workshop on the Medical Record. Brussels, 8-18.

Rector A. L., Nowlan W. A. and Glowinski A. (1993a) Goals for Concept Representation in the GALEN Project. In: Safran C., ed. Seventeenth Annual Symposium on Computer Applications in Medical Care. Washington, D.C.: McGraw-Hill, 414-18.

Rector A. L., Nowlan W. A. and Kay S. (1991) Foundations for an Electronic Medical Record. Methods of Information in Medicine, 30(3):179-86.

Rector A. L., Nowlan W. A., Kay S., Goble C. A. and Howkins T. J. (1993b) A Framework for Modeling the Electronic Medical Record. Methods of Information in Medicine, 32(2):109-19.

Reps T. and Bricker T. (1989) Illustrating Interference in Interfering Versions of Programs. Second International Workshop on Configuration Management. Princeton, NJ: ACM Press, 46-55.

Robbins S. L., Cotran R. S. and Kumar V. (1984) Pathologic Basis of Disease. (3rd ed.) Philadelphia, PA: W. B. Saunders.

Rochkind M. J. (1975) The Source Code Control System. IEEE Transactions on Software Engineering, SE-1:364-70.

Sager N., Lynman M., Tick L. J., Ngô T. N. and Bucknall C. E. (1993) Natural Language Processing of Asthma Discharge Summaries for the Monitoring of Patient Care. In: Safran

C., ed. Seventeenth Annual Symposium on Computer Applications in Medical Care. Washington, D.C.: McGraw-Hill, 265-68.

Schröder M. (1992) Knowledge-based Analysis of Radiological Reports using Conceptual Graphs. In: Pfeiffer H. D., ed. Seventh Annual Workshop on Conceptual Graphs. Las Cruces, New Mexico, 213-22.

Schubert L. K. (1991) Semantic Nets are in the Eye of the Beholder. In: Sowa J. F., ed. Principles of Semantic Networks: Explorations in the Representation of Knowledge. San Mateo: Morgan Kaufmann, 95-107. (Brachman R. J., ed. Morgan Kaufmann Series in Representaiton and Reasoning)

Shewhart W. (1931) Economic Control of Manufactured Product. New York: D. Van Nostrand Company, Inc.

Shoham Y. (1987) Temporal logic in AI: Semantical and ontological considerations. Artificial Intelligence, 33(1):89-104.

Shortliffe E. H. (1990) Clinical Decision-Support Systems. In: Shortliffe E. H., Perreault L. E., Wiederhold G. and Fagan L. M., eds. Medical Informatics: Computer applications in health care. Reading, Massachusetts: Addison-Wesley, 466-502.

Shortliffe E. H. and Hubbard S. M. (1989) Information Systems for Oncology. In: DeVita V. T., Hellman S. and Roseberg S. A., eds. Cancer: Principles and Practice of Oncology. Philadelphia: J. B. Lippincott, 2403-12.

Shwe M., Sujansky W. and Middleton B. (1992) Reuse of Knowledge Represented in the Arden Syntax. In: Frisse M., ed. Sixteenth Annual Symposium on Computer Applications in Medical Care. Washington, D.C.: McGraw-Hill, 47-51.

Siegel E. R., Cummings M. M. and Woodsmall R. M. (1990) Bibliographic-Retrieval Systems. In: Shortliffe E. H., Perreault L. E., Wiederhold G. and Fagan L. M., eds. Medical Informatics: Computer applications in health care. Reading, Massachusetts: Addison-Wesley, 434-65.

Sittig D. F. (1994) Grand Challenges in Medical Informatics? Journal of the American Medical Informatics Association, 1(5):412-13.

Smith J. W. and Svirbely J. R. (1990) Laboratory Information Systems. In: Shortliffe E. H., Perreault L. E., Wiederhold G. and Fagan L. M., eds. Medical Informatics: Computer applications in health care. Reading, Massachusetts: Addison-Wesley, 273-97.

Smith M. W. (1989) Hospital Discharge Diagnoses: How accurate are they and their international classification of diseases (ICD) codes? New Zealand Medical Journal, 102(876):507-08.

Sowa J. F. (1984) Conceptual Structures. Reading, Massachusetts: Addison-Wesley. (Aron J. D., Bohl M., Chase R. P., eds. The Systems Programing Series)

Spencer-Smith R. (1991) Logic and Prolog. Hertfordshire, Great Britain: Harvester Wheatsheaf.

Streitz N. A., Hannemann J. and Thüring M. (1989) From Ideas and Arguments to Hyperdocuments: Travelling through activity spaces. In: Akseyn R., ed. Hypertext '89. Pittsburgh: Association for Computing Machinery, 343-64.

Suthers D., Weiner A., Connelly J. and Paolucci M. (1995) Belvedere: Engaging students in critical discussion of science and public policy issues. AI-Ed 95, the 7th World Conference on Artificial Intelligence in Education. Washington D.C., 266-73.

Tatro D., Briggs R., Chavez-Pardo R., *et al*. (1975) Online Drug Interaction Surveillance. American Journal of Hospital Pharmacy, 32(4):417.

Tichy W. F. (1985) RCS: A system for version control. Software: Practice and Experience, 15(7):637-54.

Tuttle M. S., Sherertz D. D., Erlbaum M. S., *et al*. (1991) Adding your Terms and Relationships to the UMLS Metathesaurus. In: Clayton P. D., ed. Fifteenth Annual Symposium on Computer Applications in Medical Care. Washington, D.C.: McGraw-Hill, 219-23.

Ullman J. D. (1988) Principles of Database and Knowledge-Base Systems. Rockville, Maryland: Computer Science Press, Inc. (Aho A. V. and Ullman J. D., eds. Principles of Computer Science Series; vol 1).

United States Congress. (1996) Health Insurance Portability and Accountability Act of 1996. Public Law 104-191.

United States General Accounting Office. (1993) Standards for Automated Medical Records. Report to Congress GAO/IMTEC-93-17.

van Walraven C., Wang B., Ugnat A. M. and Naylor C. D. (1990) False-positive Coding for Acute Myocardial Infarction on Hospital Discharge Records: Chart audit results from a tertiary centre. Canadian Journal of Cardiology, 6(9):383-86.

Weed L. (1969) Medical Records, Medical Education and Patient Care: The Problem-Oriented Record as a basic tool. Chicago: Year Book Medical Publishers.

Wells A. (1965) SNOP: The Systematized Nomenclature of Pathology. Chicago, Illinois: College of American Pathologists.

Westfechtel B. (1991) Structure-oriented Merging of Revisions of Software Documents. In: Third International Workshop on Software Configuration Management. Trondheim, Norway: ACM Press, 68-79.

Whitgift D. (1991) Methods and Tools for Software Configuration Management. West Sussex, England: John Wiley & Sons Ltd.

Wiederhold G. and Perreault L. E. (1990) Hospital Information Systems. In: Shortliffe E. H., Perreault L. E., Wiederhold G. and Fagan L. M., eds. Medical Informatics: Computer applications in health care. Reading, Massachusetts: Addison-Wesley, 219-43.

Willems J. L., Abreu-Lima C., Arnaud P., *et al*. (1991) The Diagnostic Performance of Computer Programs for the Interpretation of Electrocardiograms. New England Journal of Medicine, 325(25):1767-73.

Winograd T. and Flores F. (1986) Understanding Computers and Cognition: a new foundation for design. Reading, Massachusetts: Addison-Wesley.

Yeh S., Ellis C., Ege A. and Korth H. (1987) Performance Analysis of Two Concurrency Control Schemas for Design Environments. Technical Report STP-036-87. MCC, Austin, Texas.