

USING i* MODELS FOR EVALUATION

By

Jennifer Marie Horkoff

A thesis submitted in conformity with the requirement for the degree of

Master of Science, 2006

Department of Computer Sciences

University of Toronto

©Copyright by Jennifer Marie Horkoff (2006)

Using i* Models for Evaluation

Jennifer Horkoff

A thesis submitted in conformity with the requirements
for the degree of Master of Science, 2006

Department of Computer Science, University of Toronto

To cope with the rapidly increasing complexity of socio-technical systems, requirements analysts have explored the practice of modeling stakeholder goals and intentions behind systems using organizational modeling frameworks such as i*. An evaluation procedure can be applied to an i* model to determine whether stakeholder goals are achieved. This work identifies desired qualities of such an evaluation procedure and used these qualities to assess existing goal model evaluation procedures. Based on this assessment, we have chosen the evaluation procedure described with the NFR Framework (Chung, Nixon, Yu, & Mylopoulos, 2000) for adaptation and expansion for use with the i* Framework. We demonstrate the ability of the resulting procedure to improve model quality through a detection of semantic flaws. This procedure is applied to five case studies, exhibiting its ability to provide useful analysis in diverse contexts. Areas of future investigation for i* modeling and evaluation are identified.

Acknowledgements

I am in debt to many individuals for providing guidance and assistance in the completion of this work. First, my supervisor, Eric Yu, for introducing me to the general research area of organizational modeling over the course of two sessions of summer undergraduate employment, and for helping me to define and refine my ideas in the course of my Master's work. My thanks go to John Mylopoulos for reading and providing constructive feedback on this work.

I am grateful for knowledge gained through collaboration with the Kids Help Phone research team, including Eric Yu, Steve Easterbrook, Jorge Aranda, Marcel Leica, Jean (Yuntian) Fan, and Rifat Abdul Qadir. My thanks especially go to Marcel, for defining the model slicing technique, and Jorge for completing all of the work in the prioritization study with me. I am also grateful for the participation of all of the individuals at Kids Help Phone in the study. I feel fortunate to have been given the opportunity to learn so much about such a wonderful organization.

Thank you to Linda (Lin) Liu for her guidance and collaboration on the Trusted Computing Study, and lessons on i^* modeling. Over the course of various presentations, I have received helpful comments on my work from Julio Cesar Sampaio do Prado Leite, Luiz Marcio Cysneiros, Daniel Gross, and the Kids Help Phone Research Team.

I am especially indebted to Yijun Yu for helping me to install and understand the OpenOME application. His assistance in problem solving and debugging was essential in implementing the evaluation algorithm. It would have been very laborious and difficult to figure out the technical details of the application on my own.

During the course of this work I received financial support from an NSERC USRA (Undergraduate Student Research) scholarship and from an Ontario Graduate Scholarship. The Kids Help Phone project is funded by the generosity of Bell University Labs.

On a personal note, thank you to my Dad for his support of my education. Finally, thank you to my Mom for spending an immense amount of time over the past year helping me to proofread and format this document. She should receive official university credit for vicariously completing all of my education.

Table of Contents

Chapter 1: Introduction.....	1
1.1 Background: System Modeling.....	1
1.2 Research Objectives.....	3
1.3 Thesis Organization.....	7
Chapter 2: The i* Framework and the Need for Evaluation.....	8
2.1 Introduction to System Modeling.....	8
2.1.1 Goal Modeling.....	11
2.1.2 The i* Framework: An Introduction.....	14
2.1.3 Contexts of i* Application.....	16
2.2 The i* Framework in Detail.....	18
2.2.1 The Strategic Dependency (SD) Model.....	19
2.2.2 The Strategic Rationale (SR) Model.....	23
2.3 The Need for Evaluation in the i* Framework.....	31
2.3.1 Desired Qualities for an i* Evaluation Procedure.....	38
2.3.2 Desired Qualities: Synergies and Conflicts.....	43
2.4 Conclusion.....	44
Chapter 3: Evaluation in Goal Modeling Frameworks.....	46
3.1 Evaluation for Goal Models vs. Evaluation for i*.....	46
3.2 Goal Model Evaluation Procedures.....	47
3.2.1 The CNYM Method.....	48
3.2.2 The GMNS Method.....	52
3.2.3 The GMNS-# Method.....	56
3.2.4 The SGM-TD Method.....	59
3.2.5 The Jarvis Method.....	60
3.2.6 Evaluation for the KAOS Framework.....	61
3.2.7 Experimental Result Comparison for the NFR Framework Evaluation Methods.....	65
3.3 Choosing a Goal Model Evaluation Procedure as a Basis for i* Evaluation....	69
3.4 Conclusion.....	71
Chapter 4: The i* Evaluation Procedure.....	72
4.1 Introduction.....	72
4.2 The CNYM Method Revisited and Elaborated.....	73
4.2.1 A Review of the CNYM Procedure.....	73
4.2.2 An Elaboration of the CNYM Method.....	76
4.2.2.1 Requiring Human Judgment Multiple Times for one Goal.....	76
4.2.2.2 Initial Labels.....	78
4.2.2.3 Initial Labels for Non-Leaf Nodes.....	80
4.2.2.4 Choosing Alternatives to Evaluate.....	81
4.3 Expanding and Adapting Rules for i* Constructs.....	82
4.3.1 Labels.....	83
4.3.1.1 The Propagation of Unknown vs. Unlabelled.....	84
4.3.2 Elements.....	84
4.3.2.1 Beliefs.....	89

4.3.3	Links	89
4.3.3.1	Contribution Links	89
4.3.3.2	Correlation Links	90
4.3.3.3	Dependency Links	90
4.3.3.4	Decomposition Links	91
4.3.3.5	Means-ends links	92
4.3.3.6	Links to Other Links	93
4.3.3.6.1	The Need for Links to Other Links	93
4.3.3.6.2	Evaluation for Links to Other Links	94
4.3.3.7	Contributions from a Mixture of Link Types	97
4.3.4	Actor boundaries	99
4.3.5	Human Intervention	100
4.4	The i^* Evaluation Algorithm	102
4.4.1	Convergence	103
4.4.2	Termination	105
4.5	Evaluation Example	106
4.6	Domain Analysis	109
4.7	Evaluation Adaptation	114
4.8	Conclusion	115
Chapter 5: Improving Model Quality as a Result of Evaluation		116
5.1	Introduction	116
5.1.1	Model Quality	116
5.2	Expanded i^* Syntax Prompted by Evaluation	117
5.2.1	Circularity	118
5.2.2	Single Dependum with Multiple Dependents or Dependees	119
5.2.3	Repeating Elements across Dependencies	121
5.2.4	Non-Typical i^* Syntax not Hindered by Evaluation	122
5.2.4.1	Non-Dependency Links between Actors as Shorthand	122
5.2.4.2	Non-Dependency Links between Actors in Attack/Defense Models	123
5.2.5	Discussion	124
5.3	Semantic Improvements Prompted by Evaluation	126
5.3.1	Semantic Validation	126
5.3.1.1	Discussion	137
5.3.2	Reduction of Ambiguity	139
5.4	Conclusion	142
Chapter 6: Implementation		144
6.1	Introduction	144
6.2	Implementation in OpenOME	145
6.3	Issues Discovered During Implementation	157
6.4	Testing	160
6.5	Reducing the Need for Human Judgment	165
6.6	Conclusion	168
Chapter 7: i^* Evaluation Case Studies		169
7.1	Introduction	169
7.1.1	Discovering Issues in i^* Modeling and Evaluation	169

7.1.2	Exploratory Case Studies	171
7.1.3	Overview of the Case Studies	171
7.2	Privacy in E-Commerce	172
7.2.1	Privacy in E-Commerce: The Current Situation	172
7.2.2	Potential Solutions to Protect Privacy	174
7.2.3	Discussion: Demonstration of Changes Prompted by Evaluation	174
7.3	Economic Information Security	177
7.3.1	Network Externality	177
7.3.2	Asymmetric Information	179
7.3.3	Adverse Selection	182
7.3.4	Tragedy of the Commons	184
7.3.5	Liability Dumping	185
7.3.6	Discussion	185
7.4	Montreux Jazz Festival	186
7.4.1	MJF Background	186
7.4.2	i* and Evaluation Issues Encountered	186
7.5	Trusted Computing	190
7.5.1	Sources	191
7.5.2	Background	191
7.5.3	Case Study Presentation Style	191
7.5.4	Shared Viewpoints	192
7.5.4.1	How do the Consumers satisfy their Desire for Affordable Products?	194
7.5.4.2	How does the Hacker/Malicious User fit in the Situation?	197
7.5.5	TC Proponent Viewpoint	198
7.5.6	TC Opposition Viewpoint	199
7.5.7	Discussion	201
7.6	Kids Help Phone	201
7.6.1	First Project Stage: General Elicitation, Technology for Counseling, and Viewpoint Modeling	202
7.6.1.1	Model and Evaluation Challenges and Solutions	203
7.6.2	Second Project Stage: Specification for Ask a Counselor Replacement using Prioritization	210
7.6.2.1	Evaluation of Scenarios in i*	210
7.6.2.2	Prioritization of Features using i*	216
7.6.2.3	Model and Evaluation Challenges and Solutions	220
7.6.3	Future Project Directions	223
7.7	Conclusion	223
Chapter 8:	i* Evaluation Procedure Assessment	225
8.1	Introduction	225
8.2	Achieving the Desired Qualities of an i* Evaluation Procedure	225
8.3	Assessing Goal Model Evaluation Procedures Adapted for i*	228
8.3.1	Adaptation of Goal Model Evaluation Procedures for the i* Framework 229	
8.3.2	Comparison of Evaluation Procedures	234
8.4	Conclusion	238

Chapter 9: Contributions, Limitations and Future Directions	239
9.1 Contributions.....	239
9.2 Limitations	242
9.3 Future Work	243
9.3.1 Implementation Issues	243
9.3.2 Examination of i* Syntax.....	243
9.3.3 Scalability Concerns	244
9.3.4 Investigation of Broader Approaches to Evaluation.....	244
9.3.5 The Need for Further Empirical Studies.....	245
9.3.6 Potential Additional Features for the i* Evaluation Procedure.....	245
9.3.6.1 Top Down Evaluation (vi)	246
9.3.6.2 Traceability (vii)	246
9.3.6.3 Conflict Detection (viii).....	248
9.3.6.4 Constraints on Values(ix)	250
9.3.6.5 Facilitating Cost Analysis (x)	251
9.3.6.6 Quantitative Values.....	252
9.3.6.7 Optional Avoidance of Partial Values	255
9.4 Conclusion	256
References.....	257

List of Figures

Figure 1.1: i* SR Model showing interactions in the PC Business Domain	3
Figure 2.1: An Example BPML Model from (White, 2004, p. 5)	10
Figure 2.2: Example NFR Goal Model from (Chung, Nixon, Yu, & Mylopoulos, 2000)	12
Figure 2.3: Example KAOS Goal Model from (Dardenne et al., 1991): Elevator's Goal Structure	13
Figure 2.4: An Example i* Model from the Privacy in E-Commerce Case Study	15
Figure 2.5: Graphical Notation of Actor, Agent, Role, Position, and Associations	19
Figure 2.6: SD Model without Dependencies from a Software Process Domain	20
Figure 2.7: Graphical Notation of Dependency Types	22
Figure 2.8: SD Model from a Software Process Domain	23
Figure 2.9: SR Model Example, a Simplified Version of a Model from the Trusted Computed Case Study	24
Figure 2.10: SD Model Corresponding to Example SR Model, Figure 2.9	25
Figure 2.11: Graphical Notation for Means-Ends and Decomposition Links	26
Figure 2.12: Graphical Notation for Contribution Links	28
Figure 2.13: Graphical Notation for Correlation Links	30
Figure 2.14: Example of a Link to another Link	30
Figure 2.15: Example using Beliefs, adapted from the Trusted Computing Domain, a variation of Figure 2.9	31
Figure 2.16: SR Model Example, a Simplified Version of a Model from the Trusted Computed Case Study (Repeated from Figure 2.9)	33
Figure 2.17: Possible Iteration on Figure 2.16 inspired by Strategic Questions	34
Figure 2.18: Another Possible Iteration on Figure 2.16 inspired by Strategic Questions	34
Figure 2.19: An Example of a Complicated i* Model from the Trusted Computing Case Study	36
Figure 3.1: An example NFR model reproduced from (Chung et al., 2000)	49
Figure 4.1: A partial goal model for GM, taken from (Giorgini et al., 2002) and (Giorgini et al., 2004)	78
Figure 4.2: An example NFR model reproduced from (Chung et al., 2000)	80
Figure 4.3: Mix of Hard and Soft Elements	85
Figure 4.4: Mix of Hard and Soft Elements	86
Figure 4.5: Excerpt from Trusted Computing Case Study	86
Figure 4.6: Excerpt from Montreux Jazz Festival Case Study	87
Figure 4.7 NFR Model of an Office Support System Showing Functional and Non- functional Hierarchy from (Mylopoulos, Chung, & Yu, 1999)	88
Figure 4.8: Example of Propagation through Dependency Links	90
Figure 4.9: Example of Propagation through Decomposition Links	92
Figure 4.10: Example of Propagation through Means-Ends Links	92
Figure 4.11: Links to Links vs. Evaluation Values for Links	94
Figure 4.12: Example of Potential Link Promotion	95
Figure 4.13: Example of Link Demotion	95
Figure 4.14: Description of Columns in Table 4.4	97
Figure 4.15: Common Cases of Mixing Links	98

Figure 4.16: Example from the Montreux Jazz Festival of a Mixture of Links	98
Figure 4.17: A Rearrangement of the Bottom Cluster of Figure 4.13 to Avoid Mixing Links	99
Figure 4.18: A Simple Example of a Loop with Inversing Links.....	105
Figure 4.19: A Simple Example of a Loop without Inversing Links.....	105
Figure 4.20: Propagation Steps in Example Trusted Computing Model	109
Figure 4.21: SR Model Example, a Simplified Version of a Model from the Trusted Computed Case Study.....	110
Figure 4.22: Excerpt from Full TC Opponents Model	111
Figure 4.23: Privacy in E-Commerce Analysis Example	112
Figure 4.24: Excerpt from an Economic Information Security Model showing the Evaluation Results	113
Figure 4.25: Excerpt from an Economic Information Security Model showing the Evaluation Results	114
Figure 5.1: Excerpt from the Trusted Computing Domain showing Model Circularity	118
Figure 5.2: Excerpt from the Trusted Computing Domain redrawn to avoid Circularity	119
Figure 5.3: Excerpt from Privacy in E-Commerce Example showing Multiple Dependency links to Single Elements.....	120
Figure 5.4: Excerpt from Privacy in E-Commerce Example where Multiple Dependency links to Single Elements are partially removed	121
Figure 5.5: Excerpt from the E-Commerce Privacy Case Study Showing Links across Actors.....	123
Figure 5.6: Excerpt from the E-Commerce Privacy Case Study with Links across Actors Removed	123
Figure 5.7: Trusted Computing Example Showing Links Across Actors as Attacks....	124
Figure 5.8: Privacy in E-Commerce Example after Syntax Changes Prompted by Evaluation	128
Figure 5.9: Privacy in E-Commerce Example after Semantic Changes Prompted by Evaluation	129
Figure 5.10: Privacy in E-Commerce Example after Semantic Changes Prompted by Evaluation	130
Figure 5.11: Privacy in E-Commerce Example after Semantic Changes Prompted by Evaluation	132
Figure 5.12: Privacy in E-Commerce Example after Semantic Changes Prompted by Evaluation	135
Figure 5.13: Kids Help Phone Example showing Link Incompleteness	136
Figure 5.14: Kids Help Phone Example Excerpt Showing Large Number of Contribution Links	137
Figure 5.15: Privacy in E-Commerce Example before and after Semantic Changes Prompted by Evaluation.....	139
Figure 5.16: Excerpt from a Trusted Computing Model showing Human Judgment before Refinement.....	140
Figure 5.17: Excerpt from a Trusted Computing Model showing Human Judgment after Refinement.....	141

Figure 5.18: Excerpt from a Montreux Jazz Festival Model showing Human Judgment before Refinement.....	142
Figure 5.19: Excerpt from a Montreux Jazz Festival Model showing Human Judgment after Refinement.....	142
Figure 6.1: Classes Created or Used in the Evaluation Algorithm in OpenOME	146
Figure 6.2: New Classes in the New HumanInterventionReasoning Package	147
Figure 6.3: The HumanInterventionReasoning Class from the Newly Implemented HumanInterventionReasoningPackage	148
Figure 6.4: The EvaluationLabelStack Class from the Newly Implemented HumanInterventionReasoningPackage	149
Figure 6.5: The EvaluationLabel Class from the Newly Implemented HumanInterventionReasoningPackage	149
Figure 6.6: The EvaluationLabelBag Class from the Newly Implemented HumanInterventionReasoningPackage	150
Figure 6.7: The LinkType Class from the Newly Implemented HumanInterventionReasoningPackage	150
Figure 6.8: The HumanInterventionDialog Class from the Newly Implemented HumanInterventionReasoningPackage	152
Figure 6.9: The TelosModel Class from the Preexisting edu.toronto.cs.ome.model Package	153
Figure 6.10: The TelosElement Class from the Preexisting edu.toronto.cs.ome.model Package	154
Figure 6.11: The TelosLink Class from the Preexisting edu.toronto.cs.ome.model Package	155
Figure 6.12: The OMEDefaultPlugin Class from the Preexisting edu.toronto.cs.ome.controller Package	156
Figure 6.13: A Screenshot of the OpenOME Implementation of the i*Evaluation.....	157
Figure 6.14: Example of Contribution Links to “Hard” Element.....	158
Figure 6.15: An Example Cycle which may Result in an Infinite Loop if Previous Human Judgment is Stored	159
Figure 6.16: Example Model used to Test for the Correct Functionality of the Evaluation Algorithm.....	161
Figure 6.17: Example Trusted Computing Model Used to Test the Evaluation Procedure Implementation	162
Figure 6.18: Trusted Computing Example used to Test the Evaluation Implementation	163
Figure 6.19: Example MJF Model used to test the Evaluation Procedure Implementation	165
Figure 7.1: From the Privacy in E-Commerce Case Study Representing the Situation before Potential Solutions to Privacy Problems.....	173
Figure 7.2: Privacy in E-Commerce Example after Semantic Changes Prompted by Evaluation	176
Figure 7.3: Information Security Network Externalities General Model	178
Figure 7.4: Information Security Asymmetric Information for a Car Salesperson	180
Figure 7.5: Information Security Adverse Selection in Product Selection.....	183

Figure 7.6: Tragedy of the Commons General Pattern	184
Figure 7.7: Montreux Jazz Festival Visitor Evaluation	187
Figure 7.8: Montreux Jazz Festival Artist Evaluation	189
Figure 7.9: Montreux Jazz Festival MJF in Relation to Visitor Evaluation	190
Figure 7.10: Trusted Computing Case Study Model Showing the Shared View of the Technology Producer, Technology User, License/Copyright Holder and Content User	194
Figure 7.11: Trusted Computing Case Study Model Showing the Interaction of the Data Pirate with the Actors Considered Previously	195
Figure 7.12: Trusted Computing Case Study Model Showing the Interaction of the Data Pirate while Considering the Multiplicity of Actors	196
Figure 7.13: Trusted Computing Case Study Model Showing the Interaction of the Hacker/Malicious User with All Previously Considered Actors	198
Figure 7.14: Trusted Computing Case Study Model Showing the Big Picture for TC Proponents	199
Figure 7.15: Trusted Computing Case Study Model Showing the Big Picture for TC Opponents	200
Figure 7.16: The Services Provided by Kids Help Phone	204
Figure 7.17: Kids Help Phone Global SD Model	205
Figure 7.18: Bottom-Up Slice Algorithm from (Leica, 2005)	205
Figure 7.19: KHP Global Model Slice of the Figure 7.15 Model to Evaluate the Effects of a Delayed Moderation Bulletin Board	206
Figure 7.20: KHP Counseling Model	208
Figure 7.21: Reduced View of the KHP Counseling Model in Figure 7.19 Showing Repeated Softgoals	209
Figure 7.22: KHP Current System Counseling Model Created for the Second Stage of the KHP Project	211
Figure 7.23: A Section of the KHP Counseling Model Created for the Second Stage of the KHP Project	211
Figure 7.24: Part of the task structure in the i* model for the current system	212
Figure 7.25: Part of the Meta- Scenario corresponding to Figure 7.22	212
Figure 7.26: KHP Future System Counseling Model Created for the Second Stage of the KHP Project	216
Figure 7.27: Prioritization Methodology	219
Figure 7.28: The Tier Two Layer of Figure 7.26	220
Figure 7.29: A Section of the KHP Future System Counseling Model shown in Figure 7.24	221
Figure 8.1: Description of Columns in Table 8.4	232
Figure 8.2: TC Example Model Used to Compare Evaluation Procedures	236
Figure 9.1: Example TC Model Showing a Possible Implementation of Traceability ..	248
Figure 9.2: SR Model Showing Potential Conflict Detection	249
Figure 9.3: An Example Graphical Representation for Constraints	251
Figure 9.4: TC Example Model Used to Test Qualitative Evaluation for i*	255

List of Tables

Table 3.1: CNYM Method Evaluation Labels and Propagation Rules, reproduced from (Chung et al., 2000)	50
Table 3.2: Propagation Rules for the GMNS Method, taken from (Giorgini et al., 2004)	53
Table 3.3: Propagation Rules for the GMNS-# Method, taken from (Giorgini et al., 2004)	57
Table 3.4: Results Comparison of Three Goal Model Evaluation Procedures	66
Table 3.5: Results for Jarvis Method on Model in Figure 3.2	68
Table 3.6: Required Equivalences for Evaluation Method Comparison	68
Table 4.1: CNYM Method Evaluation Labels and Propagation Rules, adapted from (Chung et al., 2000)	73
Table 4.2: Cases where Final Labels for Parent Goals can be Automatically Determined in Step 2	75
Table 4.3: Labels for CNYM and i* Evaluation	84
Table 4.4: Propagation Rules for Links to Links	96
Table 4.5: Cases where Final Labels for Parent Goals can be Automatically Determined in Step 2	102
Table 7.1: Quantitative Values Assigned to Qualitative Evaluation Results	214
Table 7.2: Scenario Evaluation Scores for the Current KHP System	215
Table 7.3: Priorities of future system optional features	217
Table 7.4: Stakeholder Priority Category Adjustment	218
Table 8.1: GMNS and SGM-TD Propagation Rules Adapted for the i* Framework	229
Table 8.2: GMNS-# and SGM-TD-# Propagation Rules Adapted for the i* Framework	230
Table 8.3: Combining Qualitative Evidence in the GMNS Procedure	231
Table 8.4: Propagation Rules for Links to Links in the GMNS and SGM-TD Adaptations	232
Table 8.5: Propagation Rules for Links to Links in the GMNS-# and SGM-TD-# Adaptations	233
Table 8.6: Approximate Equivalencies of Evaluation Labels across Evaluation Procedures	236
Table 9.1: Proposed Propagation Rules for a Quantitative i* Evaluation Procedure	254

List of Appendices

Appendix A:	Evaluation Algorithm Pseudocode	267
Appendix B:	Privacy in E-Commerce Study Additional Models	283
Appendix C:	Economic Information Security Study Additional Models	286
Appendix D:	Trusted Computing Study Additional Models	294
Appendix E:	Comparison of Evaluation Procedure Results	303

Chapter 1: Introduction

1.1 Background: System Modeling

During the last half-century, human society has developed an ever-increasing reliance on technology for solutions to problems. As our personal and professional lives become more tightly coupled with software systems, the systems become more tightly coupled with each other, creating a complex web of human computer interactions and dependencies. In the construction of such systems developers have turned to the use of models as a method of abstraction and understanding. Models help us achieve a shared understanding of software systems. They can depict how systems interact with users, how they interact with other systems, what components they contain, and the relationship between such components. Models have been especially employed in the field of Requirements Engineering in order to facilitate the capture of aspects in the domain, the understanding of which are necessary in order to articulate the required functionality of a system.

The evolution of modeling for system development has seen the introduction of notations which capture various aspects of the domain. The real-world characteristics represented by a modeling language can be described as the ontology of the language. Mylopoulos (1998) describes a four category classification for the ontologies typically employed by modeling languages, namely static, dynamic, intentional and social. Static ontologies capture entities, entity attributes and the relationship between attributes in a static, unchanging abstraction of the domain. In contrast, dynamic ontologies capture aspects of the world which represent change such as processes, states and the transitions between states. Conventional system modeling languages such as the Data-flow diagrams (DFDs), Entity Relationship Modeling (ER), the Unified Modeling Language (UML), and models in the Structured Analysis and Design Technique (SADT), as described by Davis (1993) in the context of software requirements, serve as examples of languages which offer static and dynamic ontologies.

In recent years, work has focused on developing modeling languages which focus on the intentional ontologies. The intentional ontology captures agents in the real world, including the goals, beliefs, and issues of agents, and how these aspects support and deny each other. Intentional ontologies are typically offered by modeling notations known as goal models, used in Goal-Oriented Requirements Engineering (GORE). See (van Lamsweerde, 2000) for an overview of such languages. Goal models are used to capture the wishes or goals of agents in the domain, likely pertaining to the perceived need for a technical system. These goals are *decomposed* or *refined* into more specific *subgoals*, eventually deriving goals specific enough to define a solution to the problems represented by the goals. Often, more than one way to decompose a goal or to solve a problem will be derived. Once the goals have been elicited from the domain, and potential solutions have been derived, one can ask: will a proposed solution achieve these goals? We call the process of determining the answer to this question using the information contained within a model *evaluation*; specifically this is the *evaluation* of a solution in terms of domain goals. A number of procedures to facilitate the evaluation of goal models have been proposed (Chung, Nixon, Yu, & Mylopoulos, 2000), (Giorgini, Mylopoulos, Nicchiarelli, & Sebastiani, 2002), (Giorgini, Mylopoulos, Nicchiarelli, & Sebastiani, 2004), (Jarvis, 1992), (Letier & van Lamsweerde, 2004). These procedures vary in their complexity, representation of goal achievement, and in the level of participation required from the modeler. In this work the term *evaluation*, used in the above sense, is distinguished from the evaluation of the merits of a method proposed in research. When referring to the evaluation of methods in this sense, we shall use the term *assessment* in order to avoid confusion.

In a social ontology, organizational structures and inter-dependencies are captured using notions such as actors, roles, positions, and dependencies. The inclusion of such information allows the representation and analysis of social networks, including the interactions between the roles and positions taken on by agents. The i* Modeling Framework (Yu, 1995) focuses on offering both social and intentional ontologies. The Framework represents the social domain of software systems by including the notions of actors, which can represent concrete agents such as real humans or systems, or abstract entities such as roles and positions. The interactions between these entities are

emphasized with the inclusion of dependencies between actors. Such models depict the intentional or “why” aspects of the domain, via the inclusion of goals and *softgoals*, goals whose achievement are not clearly defined. In addition, the framework includes static and dynamic constructs, via the inclusion of agents and resources, which can be considered entities, and tasks, which can be used to represent processes.

The overall intention of the framework is to ensure that developers obtain an understanding of how systems could be embedded in a social organization, addressing intentional desires, increasing the potential for software systems to solve real problems. See Figure 1.1 for an example i* model in the domain of PC business interactions.

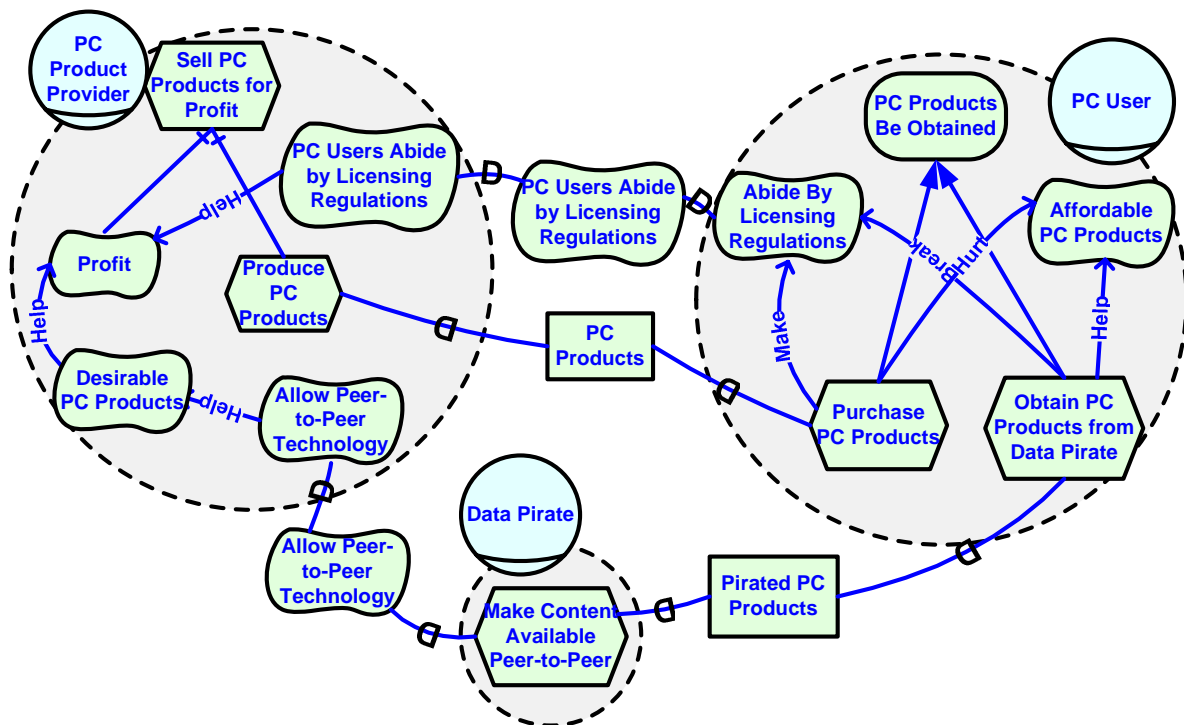


Figure 1.1: i* SR Model showing interactions in the PC Business Domain

1.2 Research Objectives

Similar to goal models, i* constructs can be decomposed in order to produce potential design solutions. However, unlike goal models, the social dimension of i* allows these solutions to more clearly represent variations in dependencies and

commitments. The models allow developers to evaluate different system configurations and potential solutions, determining the effectiveness of each option via their effects on intentional domain criteria such as goals and softgoals. Such analysis is particularly useful in the early stage of system development and design, where high-level design choices are made to determine actor responsibilities and technology choices.

In order to use the i* framework to its full potential for the analysis of possible system configurations and domain interactions, support for model evaluation is required. In this work we aim to define an evaluation procedure for i* models which facilitates the evaluation of potential solutions in terms of intentional domain criteria.

In order to direct the development of such a procedure and assess its effectiveness for facilitating useful domain analysis, we define a set of assessment criteria, or desired properties of an i* evaluation procedure. These properties are derived primarily by observing and assessing case studies of i* usage, examining how i* is used and determining properties which would allow modelers to perform domain analysis in these contexts. We can divide these desired qualities into four general categories: qualities which are essential to facilitate analysis, qualities that are beneficial for analysis, qualities which relate to usability and qualities which result in an improvement in model quality.

(i) Element Evaluation. The facilitation of i* model evaluation requires the ability to evaluate the achievement of model elements, allowing for an analysis of the effectiveness of design alternatives. **(ii) Clear Procedural Guidelines.** In order to allow an evaluation of the satisfaction of each model element, we must clearly define the meaning of model constructs in terms of their effect on the achievement of elements. **(iii) Allowance for Human Intervention.** In the early analysis for which i* is intended, specific qualitative data is often not available. In addition, it is impossible to capture the social contexts of the models completely. To compensate for the information missing in the models we must allow humans to occasionally intervene in the evaluation procedure. **(iv) Accuracy.** In order for the results of an evaluation to assist in beneficial decision making, they must accurately reflect the real-life phenomena of the domain. **(v) Usefulness in Multiple Contexts.** As i* has been applied in various contexts, we want the evaluation procedure to provide useful analysis capabilities in multiple contexts of application.

We are able to outline multiple characteristics of an evaluation procedure which are likely increase the analysis capabilities of the procedure. **(vi) Modes of Analysis.** When posing questions based on the contents of the model, one can ask about the effects of elements on other elements, or one can ask if the achievement of certain elements is possible, and how such achievement can be accomplished. Ideally an evaluation procedure for i^* will facilitate both directions of analysis. **(vii) Traceability.** It would be useful for analysis to be able to clearly see the evaluation values which contributed, directly or indirectly, to the evaluation value of another element. **(viii) Conflict Detection.** The detection of conflicting evidence within the procedure could help to highlight areas of particular interest in the model, where the overall evaluation result of a node may be controversial or uncertain. **(ix) Constraints on Values.** The ability to constrain the evaluation values of certain model elements may help to determine whether or not and under what conditions such constraints can be met. **(x) Facilitating Cost Analysis.** Finally, providing a way to analyze the cost of various design solutions would be useful as a further means to choose between design alternatives.

In terms of the usability of the procedure, we have identified the desired qualities of simplicity and automation. **(xi) Simplicity.** By carefully considering the complexity of the procedure, we can help to ensure that its benefits, in terms of analysis power, outweigh its costs, in terms of application time and effort. **(xii) Automation.** By automating the procedure, we decrease the time it takes to evaluate models and facilitate the evaluation of very large models, where manual evaluation can be difficult.

We are able to identify desired qualities of an evaluation procedure involving the improvement in model quality that such a procedure could prompt. **(xiii) Syntax Checking.** As the application of an evaluation procedure could cause the modeler to examine the constructs of the model in more detail, errors in syntax such as incorrect element or link types could be caught. **(xiv) Semantic Improvement.** The closer examination of a model, including the interpretation of evaluation results, prompted by the application of an evaluation procedure may cause the modeler to notice semantic faults within the model. These faults could be corrected in further iterations of the model, producing a model which better reflects reality. The process of iteration prompted by evaluation could result in interesting domain discoveries, such as the need to more

precisely define concepts or terminology, or the need to clarify relationships between system components. In this process the modeler and domain analysts could acquire a better understanding of the environment in which the system shall be deployed.

As we outline the desired qualities on an evaluation procedure for the i* Framework, the existence of synergies and conflicts between these qualities becomes apparent. For instance, the development of clear procedural guidelines facilitates the determination of an evaluation value, or measure of achievement for model elements. The application of human intervention may help to promote syntactical and semantic improvements, as the modeler is forced to carefully examine the construction of the model when manually intervening. Regarding potential conflicts between these desired qualities; one can see a contradiction between the need for human intervention and the desire for automation, as full automation excludes the use of interactive human input. Furthermore, the incorporation of additional useful analysis capabilities such as traceability, cost analysis, and conflict detection increases the complexity of the procedure, conflicting with the desire for simplicity.

It is our intention to derive an evaluation procedure for i* which makes appropriate tradeoffs among these desired qualities, producing a balance of capabilities. In order to develop such an i* evaluation procedure in this work, we have performed the following research tasks, which are reported in this document:

- Explore the motivations for i* evaluation in more detail
- Provide a description of the constructs and usage of the i* Framework needed for the understanding of i* evaluation
- Further explore and define the desired qualities of evaluation for i*, including the tradeoffs between these qualities
- Assess the existing goal model evaluation procedures in terms of our desired qualities to determine which aspects of these procedures can be included in i* evaluation
- Adapt the goal model evaluation procedure which best satisfies these desired qualities for use with i* models
- Investigate the effects and benefits of i* evaluation, including changes in syntax, semantic improvement, and the capability for domain analysis

- Implement the proposed evaluation procedure in OpenOME, a conceptual modeling tool
- Describe the use of the proposed evaluation procedure in multiple case studies covering the domains of Trusted Computing, Privacy in E-Commerce, Economic Information Security, a children's counselling service (Kids Help Phone), and the business aspects of the Montreux Jazz Festival
- Compare the proposed evaluation procedure to the adaptation of goal model evaluation procedures to i^* , to validate the relative effectiveness of our procedure in meeting our desired qualities
- Describe future extensions, based on features of goal model evaluation, which would expand the functionality of our procedure

By executing these steps we hope to demonstrate the effectiveness and viability of not only our proposed evaluation procedure, but evaluation for i^* models in general.

1.3 Thesis Organization

Chapter 1 has provided an introduction to the motivations for an i^* evaluation procedure. Chapter 2 describes the precise constructs of the i^* Framework, and explores the motivation and desired properties of evaluation in more detail. In Chapter 3 existing goal model evaluation procedures are assessed in terms of these desired properties. The adaptation of a goal model evaluation procedure for i^* is performed in Chapter 4. In Chapter 5 we describe affects and benefits of the proposed i^* evaluation procedure.

Chapter 6 focuses on the implementation of the evaluation procedure. Chapter 7 includes the Case Studies of i^* application. Chapter 8 includes a comparison of the proposed evaluation procedure and the adaptations of other goal model evaluation procedures for i^* . Finally, in Chapter 9, suggestions are made for the incorporation of additional features into the i^* evaluation procedure, the limitations of the procedure are described, and the progress of previous chapters is summarized.

Chapter 2: The i* Framework and the Need for Evaluation

2.1 Introduction to System Modeling

The pervasiveness and complexity of computerized systems has seen a dramatic increase over the last decades. This has prompted an expansion in technology research: building faster and more reliable hardware, creating new and more efficient algorithms, trying to find the most effective software development processes; all in a continual attempt to increase the capabilities of technical systems. In tandem with the complexity growth of technical systems is the complexity growth of the social systems that use them.

Often research in the field of technology focuses solely on technical intricacy, overlooking the corresponding increase in social complexity. However, the field of requirements engineering asserts that an adequate understanding of social systems is necessary for a technical system to be successfully deployed into the social world. Often, technically sound systems fail because they inadequately address the problems present in the human system (<http://www.standishgroup.html>). They fail to take into account social complexities which prevent the system from solving the problem, or which prevent the system analysts from discovering the appropriate problem in the first place. In fact, when we think of the “system”, we must include not only technical components, but human components as well, in order to have the full picture of functionality, mapping to usefulness for some purpose.

If technical elements, such as millions of lines of code, thousands of data structures, and hundreds of user interfaces, make the complexity of technical systems too vast for one person to understand, then social elements, including thousands of end users, millions of stakeholders, myriad viewpoints, and conflicting motivations, cause social systems to likewise be outside the realm of one person’s understanding. Despite this, systems are obviously being built, and some of these systems can be described as successful, so how are system engineers coping with complexity?

Essential elements in the battle against system complexity include the concepts of abstraction, the utilization of high-level views, and modularity, the ability to only consider one part of the system at a time. In addition, software analysts, designers, and programmers have turned to the use of models to better express both high and low level system structures. Such models use the ideas of abstraction and modularity to depict multiple system views including system architectures, interactions with stakeholders, and detailed data structures.

A model of a system, or system component, represents a particular view. Models are often incomplete and at least partially inaccurate. However, the purpose of models is not necessarily to capture a complete and accurate view of the system, but to capture enough of the system to facilitate communication and understanding, improving the ease of system analysis, design, and implementation.

In order to facilitate system modeling, numerous modeling languages defining the syntax and semantics of model constructs, have been introduced. One of the most widely used languages is UML (Unified Modeling Language) which includes a standard syntax for a set of models which allow the modeler to express aspects such as: user interaction, action or tasks sequences, system components, the relationships among components, and their states (Booch, Rumbaugh, & Jacobson, 1999). Other common examples of modeling frameworks include data flow diagrams (DFDs), which allow modelers to describe the flow of data throughout a system (DeMarco, 1978); SADT (Structured Analysis and Design Techniques) models, which allow modelers to model the input, output, and controls of systems (Ross, 1977); and business process models, created using the Business Process Modeling Language (BPML) (White, 2004) . See Figure 2.1 for an example BPML model.

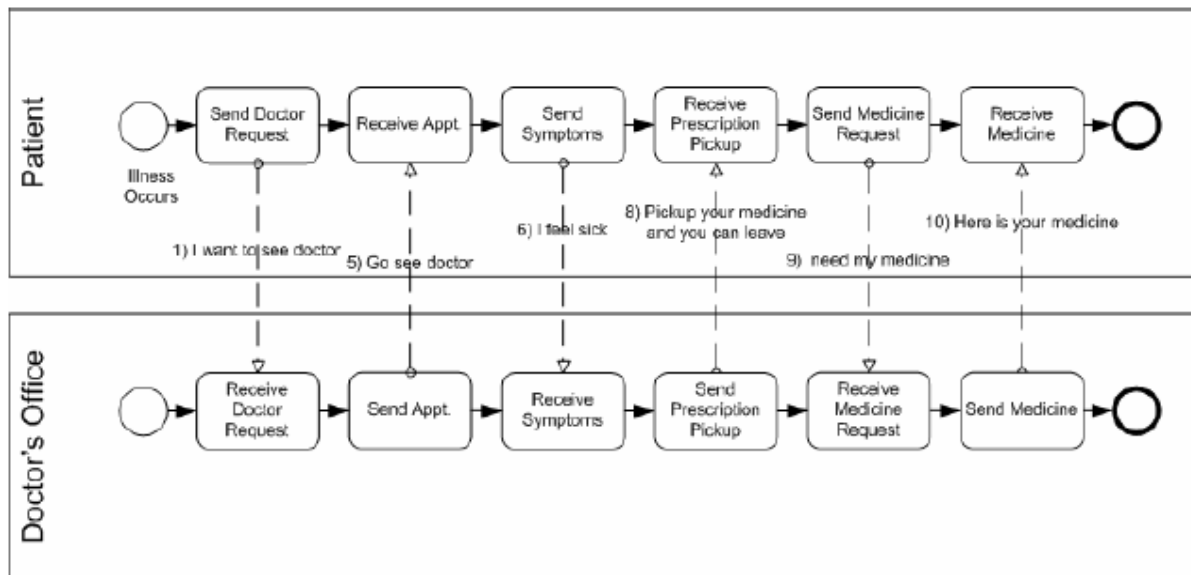


Figure 2.1: An Example BPML Model from (White, 2004, p. 5)

All of these modeling languages, as well as numerous others, have the shared characteristic of describing the “what” and “how” of systems (Yu, 97). As mentioned in Chapter 1, these languages provide ontologies which are static and/or dynamic. UML helps to show what actions are performed, how the system is structured, what states are present, and how entities relate to each other. DFDs show what data is present and how the data flows through certain components of the system. SADT models show how system components are controlled, and what information passes through them, while BPML shows what steps are present in a process, in what order these steps are taken, and possibly how these steps are accomplished. What is missing is the “why”, the intentional motivations behind the actions, processes, relationships, and structures present in socio-technical systems. Why do actions or processes need to be performed? What do they accomplish? Why do the relationships between system components, both electronic and human, exist as they do? Why is certain information needed, and why does it arrive via a particular flow? What happens if it doesn’t arrive? How can we assure that this situation does not occur? Questions such as these are difficult to answer using common functional or procedural modeling languages. Yet it seems that sufficient understanding of these motivations and intentions is required in order to perform effective analysis, design, and implementation of a system.

2.1.1 Goal Modeling

Motivations for systems arise not from the technical aspects of the system, but from the social, human components. It is the people or organizations who provide the need, the want, and the why. This corresponds to Jackson's (1997) notion that requirements most often come from the environment domain, and are fulfilled by the technical components of the system, (called the machine), triggered by phenomena which are shared by the machine and its environment. Therefore a modeling language which models the intentional aspects of systems should be suited not only for the technical, but for the social aspects in the environment.

Explicitly addressing human intentions in technical systems was the motivation for goal modeling, such as the NFR modeling framework (Mylopoulos, Chung, & Nixon, 1992). This framework introduces the goals of the stakeholders explicitly using a graphical notation, and uses these goals to drive system requirements and design activities. The NFR framework (see Figure 2.2 for an example), uses the notion of *softgoals* and contribution links. *Softgoals* are goals that are not satisfied via clear-cut criteria, and contribution links represent potentially partial negative and positive contributions to such goals. These constructs produced a qualitative framework, able to represent non-functional requirements which are more difficult to define rigorously, such as usability.

In the NFR Framework, sufficient evidence for the satisfaction of a softgoal, given the satisfaction of a contributing softgoal, can be indicated with the use of the *make* link. Partial evidence for satisfaction can be indicated with a *help* link. As these contributions can be inherently incomplete, the modeler must determine whether the contribution to a softgoal results in a sufficient satisfaction of the softgoal. Conversely, the *break* and *hurt* link can be used to represent sufficient and partial evidence for the denial of a softgoal. In the case of hurt, intervention is required to determine whether the contribution is sufficient to judge the softgoal as denied. Such qualitative measures requiring human intervention are necessary when involved in the early, exploratory stages of domain analysis, when quantitative measures are not typically available.

In addition to judgment involving single contributions of partial evidence, softgoals will often receive contributions from multiple softgoals, in some cases requiring human intervention to combine and resolve this evidence into a final judgment of satisfaction or denial. These judgments are recorded using graphical notations such as check marks for satisfied and exes for denied. The need for modeler intervention to determine satisfaction of softgoals motivated the creation of an evaluation procedure, for systematically determining the degree of satisfaction and denial for each softgoal in a model. This evaluation procedure is used to ensure that top-level goals are sufficiently met by design choices. The process of choosing between alternatives using goal models becomes a process of trade-off and negotiation. The objective is to find the design which most effectively achieves the softgoals in the model, where the overall judgment of effectiveness is based on the modeler's knowledge of the domain.

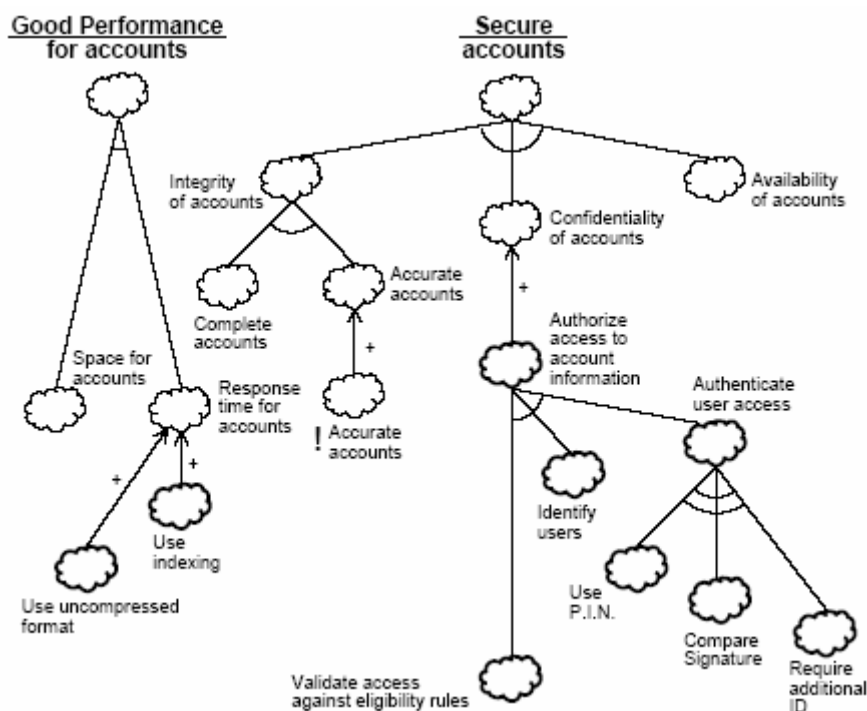


Figure 2.2: Example NFR Goal Model from (Chung, Nixon, Yu, & Mylopoulos, 2000)

An additional goal modeling framework, contained in the KAOS Methodology, introduced a formal goal framework applying binary *And* and *Or* relationships between goals, including actions and wishes which are assigned to agents (Dardenne, Fickas, &

van Lamsweerde, 1991). See Figure 2.3 for an example KAOS model. In contrast to the NFR procedure, the need for an explicit evaluation procedure does not arise, as models are complete in the logic sense, with the achievement of goals specified clearly by the *And* and *Or* relationships.

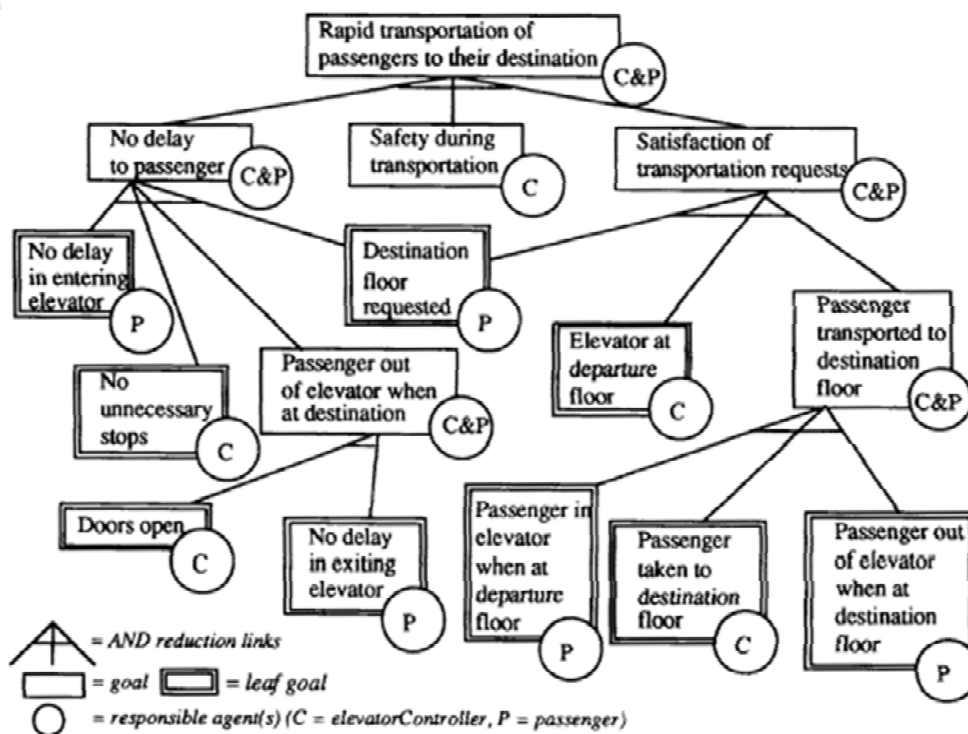


Figure 2.3: Example KAOS Goal Model from (Dardenne et al., 1991): Elevator's Goal Structure

The general intention of both frameworks is to decompose upper level goals until arriving at low-level functional pieces that can be converted into a specific list of requirements. Thus, the focus of these goal modeling frameworks is on the goals themselves, without an emphasis on the origin of the goals, or the agent who wishes the goal to be accomplished. The addition of agents often occurs near the end of the model development process, in order to assign responsibilities for specific tasks and discover conflicts among the wishes and responsibilities of agents.

The goal decompositions in these frameworks must be initiated with a set of system goals, but how are these goals discovered? Which goals are included and which are not? Where do they come from and whom do they belong to? The NFR framework is designed to aid in choosing between alternatives, but how are these alternatives

derived? Once a goal is assigned to an agent, how can the analyst ensure that the agent will be motivated to fulfill this goal? How can we analyze agent and goal vulnerability in order to see where potential for failure lies? How do we create the means to avoid such vulnerability?

2.1.2 The i* Framework: An Introduction

The i* Framework ("i*" representing distributed intentionality), first introduced by Yu (1993), employs elements included in goal models such as softgoals, contribution links, *And* and *Or* links, and softgoal decompositions. In addition, the Framework utilizes further constructs including *goals*, *tasks*, *resources*, and additional link types. In this case goals are elements which have precisely definable criteria for satisfaction.

In contrast to goal modeling frameworks, i* provides a means of reasoning about the involvement of agents earlier in the modeling process, introducing agents as a modeling construct which having a graphical representation. The underlying agent-oriented paradigm of the Framework, as described in (Yu, 2001), allows it to capture agent properties such as intentionality, autonomy, sociality, contingent identity and boundaries, strategic reflectivity and rational self-interest. The early reasoning involving such agents facilitates the discovery of goals and further agents, the assigning of goals to agents, the determination of alternatives. Such agents in i*, called actors, can represent individual stakeholders, more abstract roles or positions, or components within a software system. Each goal is then assigned to a specific actor by means of a graphical boundary. Key to the utility of this framework is the notion of a dependency, where one actor depends on another actor for the furnishing of some element, which could take the form of a goal, softgoal, task, or resource. This construct promotes the analysis of actor vulnerability and opportunity. See Figure 2.4 for an example i* model.

The i* framework, like the NFR framework, allows for the analysis of alternatives by examining the effects of each option on the goals and softgoals of the actor, making necessary trade-offs. However, the social ontology of i* allows for the explicit configuration and reconfiguration of system boundaries, dependencies, and responsibilities. The exploration of these alternatives in the construction of social networks allow for the discovery of new actor desires which can be captured as

intentional elements, previously undiscovered actor vulnerabilities and opportunities, and additional system design alternatives. These design alternatives can be compared using their effects on the intentional elements (goals, softgoals, tasks, resources) depicted in the models. This type of high-level design alternative, often concerning the interactions and responsibilities of a social network including the technical system, are ideal for analysis and design early in the system development process, as described in (Yu, 1997). The context behind the "why" dimensions of systems can be better understood by creating models which are "embedded in organizational context" (Yu, 1997, p. 3). The output of the framework, a high-level design which makes appropriate tradeoffs between the satisfaction of the intentional wishes of stakeholders, can be further analyzed and decomposed, facilitating a more detailed lower level design and the production of a detailed software specification.

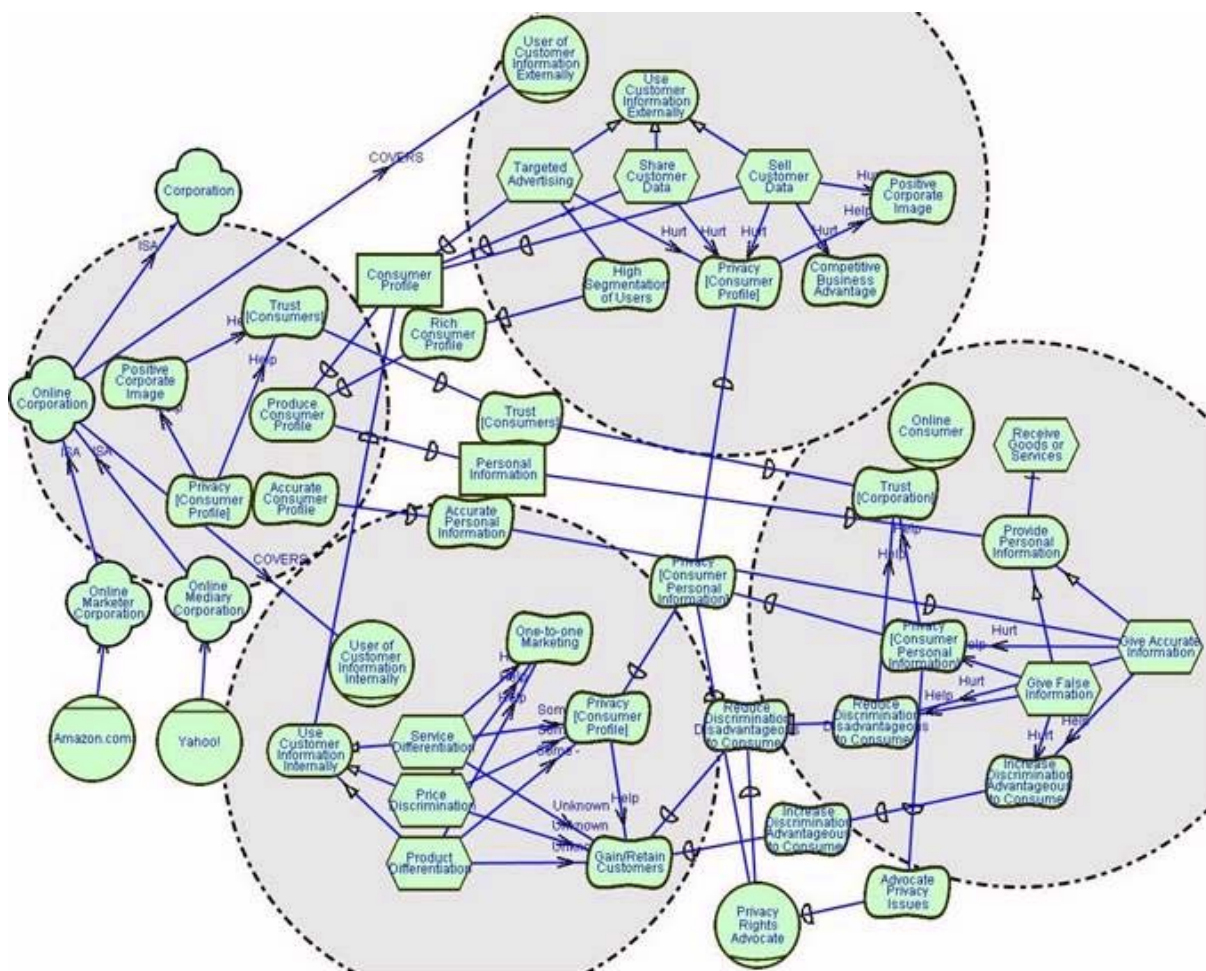


Figure 2.4: An Example i* Model from the Privacy in E-Commerce Case Study

2.1.3 Contexts of i* Application

As the i* Framework is suitable for representing situations involving relationships and interactions between multiple actors, the Framework can be utilized in multiple contexts. In i* evaluation contextual domain knowledge is often essential, as we have indicated in our criteria of the Allowance for Human Intervention (iii), and as we shall explore further in Chapter 4. Therefore, providing an overview of some of the contexts in which i* is applied is useful.

(i) Requirements Engineering. Likely the most widely known application of i* is to the field of Requirements Engineering (RE), specifically focusing on the phase of Early Requirements Engineering, as introduced in (Yu, 1997). As a part of Requirements Engineering, i* provides the ability to model the socio-technical domain, helping to facilitate communication, perform high level design, explore alternatives, discover conflicts, and make domain assumptions explicit. The properties of i* developed in an agent-oriented mode of reasoning can be incorporated into the Requirements Engineering domain in order to perform more effective analysis of distributed and networked systems, to understand the increasing interdependency and vulnerability of systems, to understand the limits of knowledge and control, to create successful cooperation, and to aid in the modeling of boundaries.

Specifically, i* has been combined with cost and workflow analysis to analyze socio-technical system requirements (Sutcliffe, & Minocha, 1999). A methodology has been proposed for deriving use cases, used in RE to capture high-level requirements encompassing multiple usage scenarios, from i* models (Santander, & Castro 2002). Comparably, a methodology has been developed to transform constructs of an i* model into formalized requirement specification statements by converting them into formal structures in the KAOS framework (Martinez, Pastor, & Estrada, 2004). In the RESCUE method, streams of modeling, including activity diagrams, i*, and use cases/scenarios, are created in parallel with stages of synchronization. These artifacts are ultimately used to create specific requirements (Maiden et al., 2004).

(ii) Software Development. The i* framework has been incorporated into software design processes. In Tropos, described in (Mylopoulos, & Castro, 2000), the

social system context is first considered via i^* , then the new system is added to the model as an agent, helping to assign system responsibilities. The design decisions are then decomposed, adding detail through various steps until a detailed agent-oriented design is created.

(iii) Software Process Analysis. As well as being incorporated into software production processes i^* can be used to analyze and design the processes themselves, as described in (Yu, & Mylopoulos, 1994). The creation of software usually includes multiple parties, each having their own local goals as well as the global goals of software production. The i^* framework can promote the consideration of alternatives in order to ensure that both production and individual goals are satisfied, helping to produce a successful product. In an example of this sort of application, Briand et al. (1995) employed i^* to assess software development, using i^* Strategic Dependency (then called Actor Dependency) models in an analysis of a large software organization.

(iv) Business Processes. Similar to the analysis of software processes, i^* can be used to explicitly view the intentionality behind business processes, in order to facilitate the discovery of a workable division of labor (Yu, & Mylopoulos, 1996). These ideas are applied in the work of Katzenstein & Lerch (2001) where components of the i^* Framework are incorporated into a Framework used to represent business processes, facilitating their redesign.

(v) Trust, Privacy and Security. The ability of i^* to model agents makes it appropriate for understanding and analyzing trust, privacy and security, as these ideas involve the conflicting intentions of different social entities. Such issues are addressed using i^* in (Liu, Yu, & Mylopoulos, 2003), (Yu, & Cysneiros, 2002), and (Yu, & Liu, 2001). Similarly, the Tropos Methodology, which uses the i^* Framework, has been extended in order to consider trust and security requirements, testing the feasibility of the augmentation via application to a case study (Giorgini, Massacci, & Mylopoulos, 2003), (Giorgini, Massacci, Mylopoulos, & Zannone, 2004), (Mouratidis, Giorgini, & Manson 2003). In addition, a Framework using goal modeling with i^* constructs such as dependencies and softgoals, has been used to capture risk-based security requirements (Mayer, Rifaut, & Dubois, 2005).

(vi) Intellectual Property Management. The framework can also be used in the field of Intellectual Property Management, described in (Yu, Liu, & Li, 2001), to analyze the agents involved in the protection and acquisition of vital business knowledge.

(vii) Identity Management. The capability of i* to model actors, agents, roles, and positions, including the relationships between these entities, can be used to model identity management, an increasingly important aspect of businesses providing internet services, as described in (Liu, & Yu, 2004).

(viii) Knowledge Management. In the field of Knowledge Management, systems are often designed using a traditional centralized approach, where knowledge is collected and accessed from a central location. Systems such as these can be disruptive to the normal routines of work as they ignore the social constructs of communities in an organization, and do not take full advantage of tacit knowledge. As described in (Molani, Perini, Yu, & Bresciani, 2003), i* can be used in this field to consider and support the knowledge context of communities, including mediation, boundary objects, and boundary encounters; helping to facilitate the sharing of knowledge. In (Bertolini, Busetta, Nori, & Perini, 2002) the Tropos Methodology is extended to support Knowledge Management, specifically for systems involving peer-to-peer and multi-agent systems technologies.

2.2 The i* Framework in Detail

The i* Framework provides two different, but related, types of models: the Strategic Dependency (SD) Model and the Strategic Rationale (SR) Model. The SD model focuses on the external relationships among actors, providing a more abstract view, while the SR model focuses on the internal motivations of actors which rationalize the dependencies.

Concerning the fonts used in the description of i* constructs and domain specific components, we use the Arial Narrow font to distinguish domain concepts which occur within a model, an *Italic Font* is used to introduce and emphasize technical terms in the i* modeling language, and *Italicized Arial Narrow* is used occasionally to distinguish link types from their equivalent natural language words.

2.2.1 The Strategic Dependency (SD) Model

The fundamental constructs of the i* SD model can be divided into two categories: *actors* and *dependencies*. The i* *actor* represents a general entity which can be human, abstract, or electronic. It is represented graphically as a circle containing the actor name. The actor notion can be expanded into the more specific constructs of an *agent*, *role*, or *position*. In addition, i* provides the capability to analyze relationships between actors via *association links*. See Figure 2.5 for the graphical representation of the actor and association constructs.

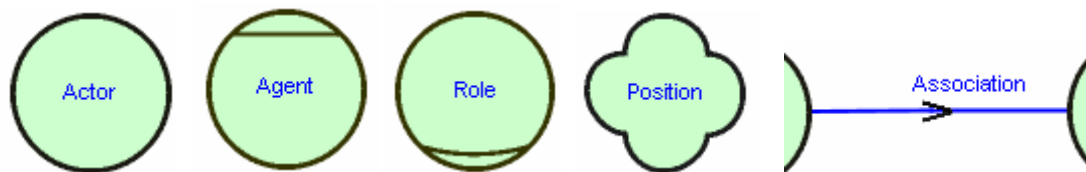


Figure 2.5: Graphical Notation of Actor, Agent, Role, Position, and Associations

An *agent* represents a real entity such as a software agent or a system, as well as specific people or organizations. In Figure 2.6, example agents include “real” people like Jill, Jeff, Judy, and Jack, as well as the abstract entities like a Design Specialist, Software Professional and QA Specialist. A *role* is an abstract notion representing responsibility for related tasks. Concrete agents take on these responsibilities by playing a role. Example roles in Figure 2.6 include Modifying Design, Modifying Code, Reviewing Design, and a Technical Task Role. Looking in more detail at Modifying Code role, for example, this would represent the higher level task of modifying code based on new design decisions, and would encompass all of the tasks and responsibilities which that higher level task would entail. The *plays* association is used between an agent and a role, with an agent playing a role. For example, the Team Member agent *plays* a Technical Task role, meaning that each specific team member is assigned a Technical Task role. The identity of the agent who plays a role should have no effect on the responsibilities of that role, and similarly, aspects of an agent should be unaffected by the roles it plays.

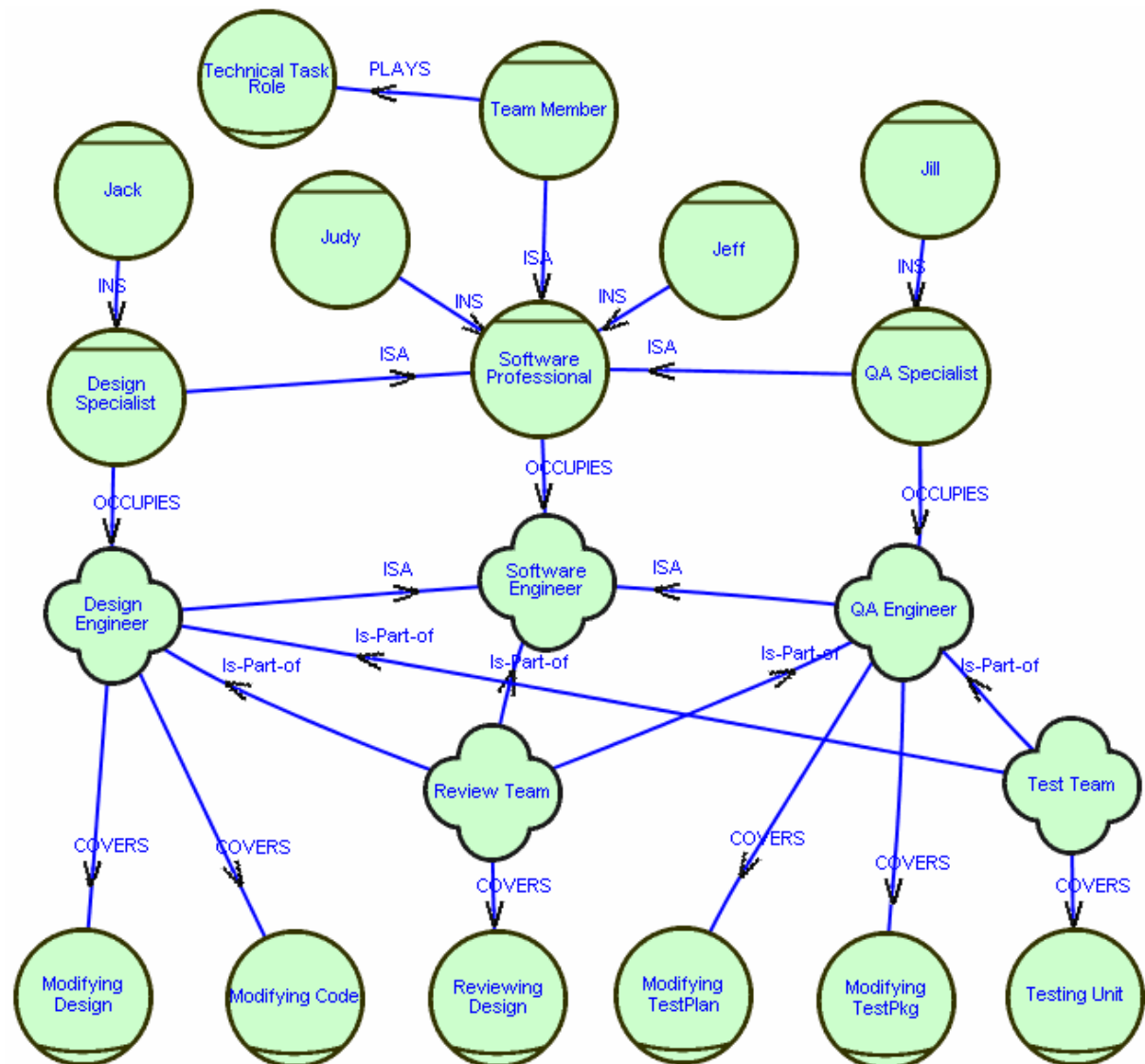


Figure 2.6: SD Model without Dependencies from a Software Process Domain
 This is a simplified version of a model appearing in (Yu & Mylopoulos, 1994).

A *position* represents a collection of roles whose combination creates another abstract entity. As the same agent often plays many roles, grouping these roles into a position provides useful descriptive capabilities. The association link *covers* is used to describe the relationship between a position and the roles that it covers. The *occupies* link is used to show that an agent occupies a role, meaning that it plays all of the roles that are covered by the position. Figure 2.6 shows example positions such as Design Engineer, Software Engineer, QA Engineer, Review Team, and Test Team. If we examine the Design Engineer Position more carefully we can see that this position covers the roles of

Modifying Design and Modifying Code, and is occupied by a Design Specialist. It is important to note that although positions are composed of roles, positions are themselves strategic and intentional entities, and may have goals and responsibilities that are separate from the roles they cover, or the agents who occupy them.

Further associations are used to describe the relationship between actors. The *is_a* association represents a generalization, such as a Design Engineer being a specialization of a Software Engineer, or a Design Specialist being a specialization of a Software Professional. The *is_part_of* or *part* association is used to represent aggregation, or subparts, with each subpart possessing independent intentionality. For example the Review Team and Test Team Positions are composed of roles that are *part of* the positions of Design Engineer and QA Engineer. Both *is_a* and *is_part_of* can be applied between any two instances of the same type of actor. The *ins* association, representing instantiation, is used to represent a specific instance of a more general entity. This is used between agents, as agents represent real entities. In the example model, all of the “real” people are shown as instantiations of various general agents; for example Jeff is an instantiation of a Software Professional. The actor notations, including associations, are described in detail in (Yu, & Mylopoulos, 1994) and (Liu, & Yu, 2004).

A critical construct for the i* SD model is that of a *dependency*. An actor, agent, role or position depends on another actor, agent, role, or position, for the provision of some element. The word *element* is used as a general term in i* to represent a *goal*, *softgoal*, *task*, *resource*, or *dependency*. Therefore, dependencies in i* can take the form of a *goal dependency*, *softgoal dependency*, *task dependency*, or *resource dependency*. The graphical representation of each type of dependency is shown in Figure 2.7. The actor who depends on another actor is called the *depender*, the actor who is depended upon is called a *dependee*, and the element that is depended on is called the *dependum*.

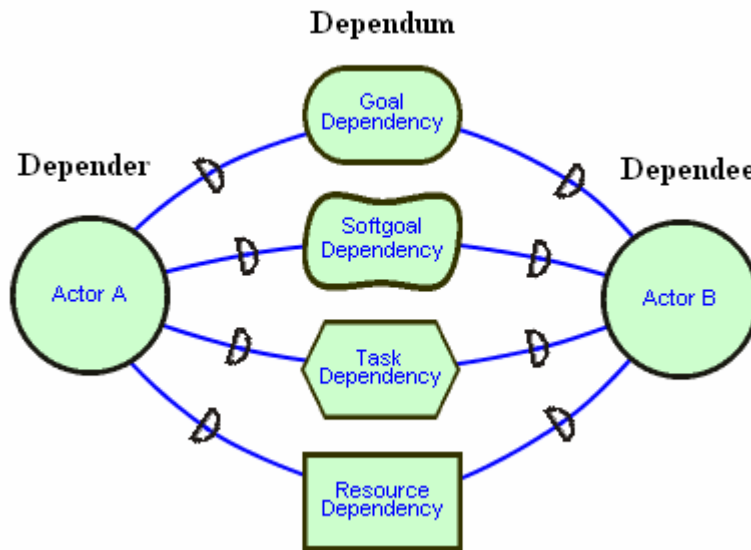


Figure 2.7: Graphical Notation of Dependency Types

A *goal dependency* represents the depender's desire for a goal to be met by another actor, the dependee. The specifics as to how this goal is to be accomplished are left to the dependee; the depender does not care by what means the goal is satisfied. For example, in Figure 2.8, the Modifying Design role depends on the Reviewing Design role for the goal, Design Be Approved, but does not care how the dependee goes about approving the design. A *softgoal dependency* is similar to a goal dependency, except that the criteria for the dependum's satisfaction are not clear-cut, it is judged to be sufficiently satisfied by the depender. In Figure 2.8, the Reviewing Design role depends on the Modifying Design role for an improved design, something that cannot be measured precisely.

In a *task dependency* the depender depends on the dependee to accomplish some specific task. For example, the QA Specialist depends on the Testing Unit to Test Software Guidelines, a task that would be performed in a particular pre-arranged way. In a *resource dependency*, the depender is depending on the provision of some entity, physical or informational. This type of dependency assumes no open issues or questions between the depender and the dependee. For example, the Reviewing Design and Modifying Code roles depend on the Modifying Design role for the resource of a Modified Design. This particular dependum is not expressed as a goal, Design be Modified, or a task, Modify Design, because the dependers actually require the entity of the design itself. Dependencies are described

in numerous sources, including (Yu, & Mylopoulos, 1994), (Yu, & Cysneiros, 2002), and (Yu, Liu, & Li, 2001).

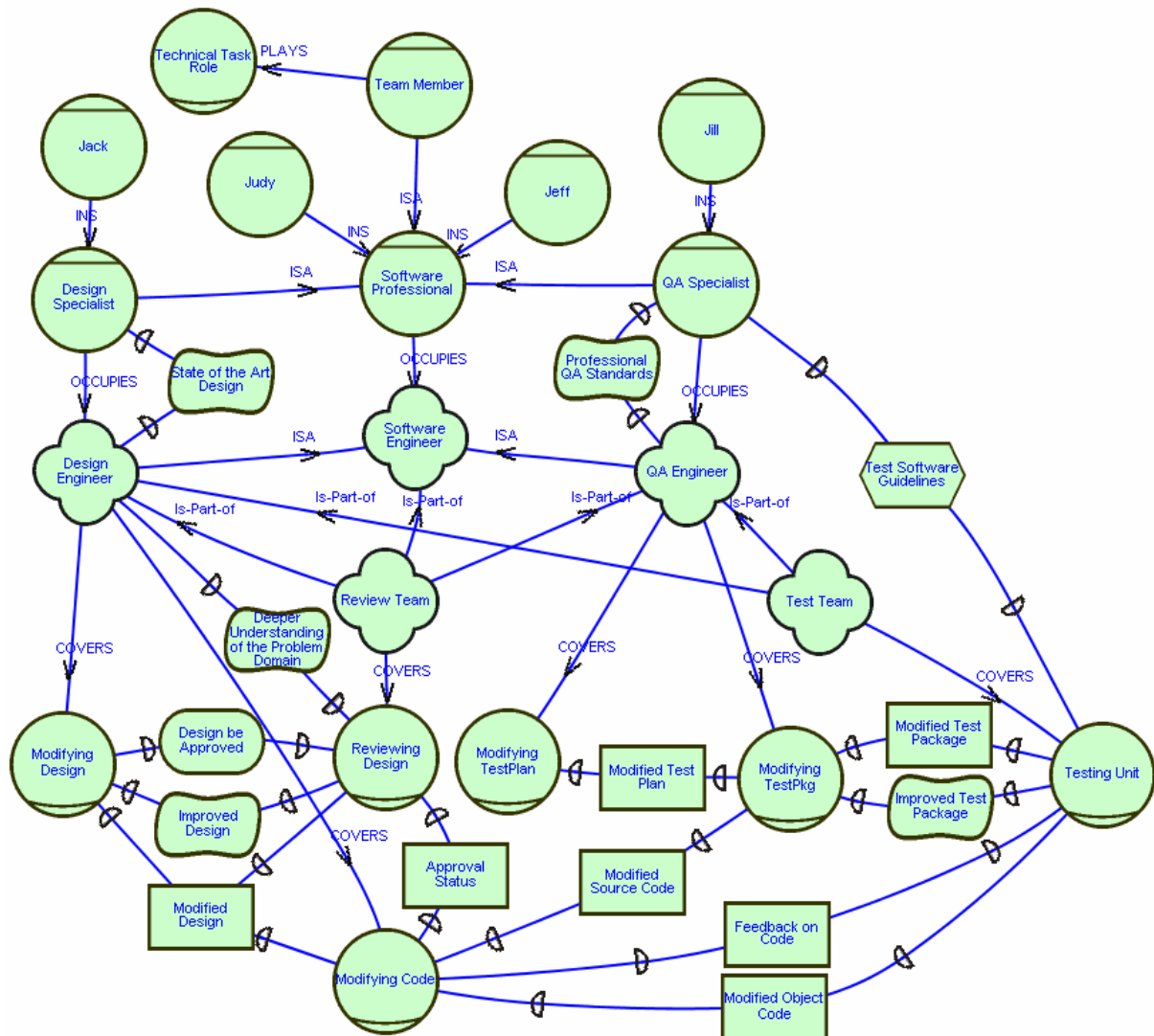


Figure 2.8: SD Model from a Software Process Domain

This is a simplified version of a model appearing in (Yu, & Mylopoulos, 1994), and is the same model as in Figure 2.7, but with dependencies added.

2.2.2 The Strategic Rationale (SR) Model

The second type of i^* model introduces a rationale for the dependencies appearing in SD models via linked internal elements. SR models contain all actors and dependencies from the corresponding SD models; however in SR models the actor is

“opened up”. This can be seen in Figure 2.9 and 2.10, which depict the SR and SD view of the same model, respectively. The *boundary of an actor* is represented as a dotted circle, with the actor circle located near the top. The elements within this boundary “belong” to the actor, in that the actor wishes them to be accomplished, often in light of a higher-level goal, and is responsible for their accomplishment, often through delegation via dependency links. The inside of an actor boundary appears somewhat like a typical goal model in the NFR framework, but with the addition of new constructs, described in this section.

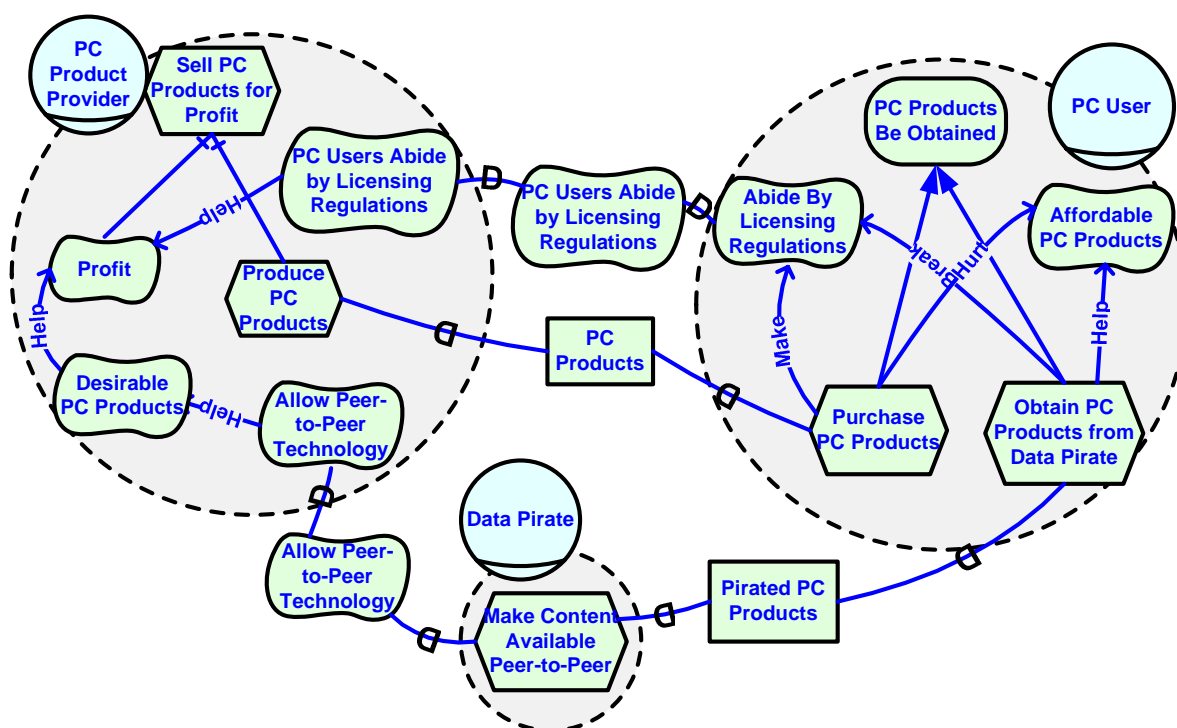


Figure 2.9: SR Model Example, a Simplified Version of a Model from the Trusted Computed Case Study

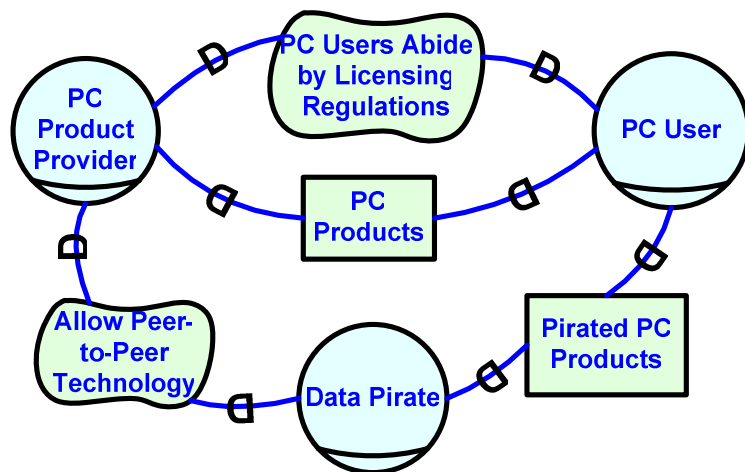


Figure 2.10: SD Model Corresponding to Example SR Model, Figure 2.9.

In the SR model we can now see the specific depender and dependee elements which are attached to dependums via dependency links, helping to answer the question of “why” and “how”. In the SD model in Figure 2.10, we can see that the PC Product Provider depends on the PC User to Abide by Licensing Regulations. With the SR model we can now see the “how” for this dependency: the PC User will Abide by Licensing Regulations by Purchasing PC Products and not Obtaining PC Products from the Data Pirate. We can also see the “why”: The PC Product Provider wants the PC Users to Abide by Licensing Regulations to help its softgoal of Profit, required in order to Sell PC Products for Profit.

The elements contained within actor boundaries are identical to those found in dependencies; goals, softgoals, tasks, and resources. The meanings of these elements are the same as found in dependencies, with the exception that the satisfaction of elements may be accomplished internally. For example, in Figure 2.9, the softgoal Allow Peer-to-Peer Technology in the PC Product Provider does not depend on another agent for its satisfaction, and is not satisfied by any other element within the PC Product Provider; therefore its satisfaction is dependent completely on the PC Product Provider. In fact, this element can be called a *leaf node*, or an *operationalization*, concepts that will be later explored in further detail.

The relationships between elements within an actor are described using three types of links: *means-ends links*, *decomposition links*, and *contribution links*. The graphical representations for means-ends and decomposition links are shown in Figure

2.11. Means-ends links are used to represent alternatives and link different “means”, in the form of tasks, to their “end” in the form of a goal. For example, in Figure 2.9, the PC Product Provider has two ways to satisfy its goal of PC Products be Obtained. It can either Purchase PC Products, or Obtain PC Products from a Data Pirate, represented by the means-ends links between these tasks and the goal. As this link corresponds closely to a standard *Or* link, the question of whether or not the *Or* is exclusive arises. Can the PC User only Purchase Products, or only Obtain them from a Data Pirate, or can it do both at once? The original constructs of *i** do not provide a way to indicate whether or not a means-ends is meant to be exclusive or inclusive. This serves as an example of *i** evaluation forcing a more precise definition of *i** syntax, a subject which is explored in more detail in Chapter 5.

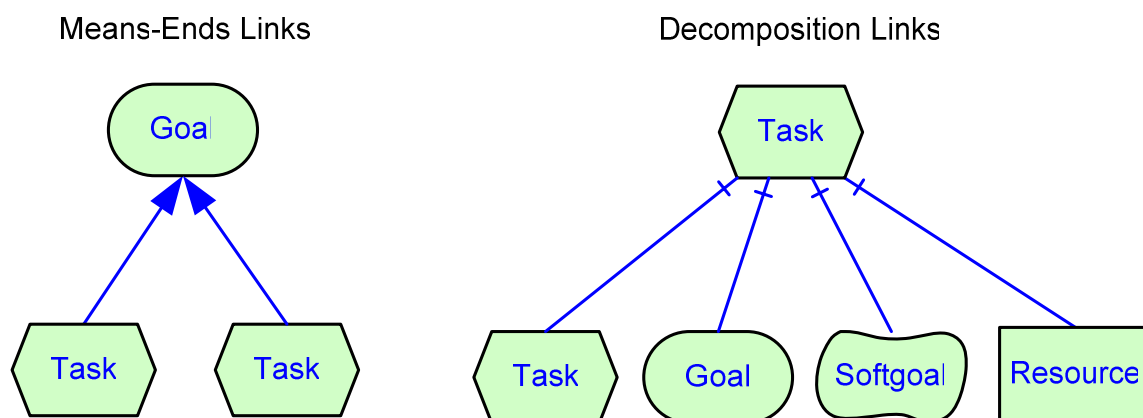


Figure 2.11: Graphical Notation for Means-Ends and Decomposition Links

The decomposition link in *i** represents sub-elements that must be accomplishable in order for a task to be accomplishable. The sub-elements for a task are not restricted to other tasks; they can be any of the other *i** elements. In the decomposition example in Figure 2.9, in order for the task PC Products be Sold for Profit to be satisfied, the softgoal Profit and the task Produce PC Products must be accomplishable. This is of course a grossly simplified version of the reality of creating and running a profitable business in the field of technology. However, this serves as an example of the incompleteness of *i** models. If all elements of running a profitable business were added, the size of this model would increase substantially, making it more difficult to understand and analyze.

Therefore only those elements that are deemed relevant and useful, Profit and Produce PC Products are included.

Similar to the process in other goal models, a higher-level element can be decomposed via means-ends and decomposition links by continually asking “how” questions, until the level of granularity is sufficient for analysis. The means-ends and decomposition links often work together to form a rough tree-like structure within an i^* actor. The term *operationalization* is borrowed from the NFR Framework to describe the results of such decompositions, the bottom leaf nodes of trees. Operationalizations are nodes that can be performed by some agent, either directly or through delegation. As i^* is meant for a higher level of analysis, these types of elements often remain at a high level, although the constructs could facilitate a detailed decomposition if necessary. In Figure 2.9 the high level tasks Purchase PC Products and Obtain PC Products from Data Pirate, as well as the softgoal Allow Peer-to-Peer technology, could be considered operationalizations.

Contribution links in i^* are used to show the contributions of elements to softgoals. In order to understand the idea of softgoal contribution, a basic understanding of the ideas behind softgoal satisfaction is required. As softgoals represent non-binary, qualitative elements, their level of satisfaction can be full, meaning the softgoal is sufficiently satisfied, or partial, meaning the element is partially, but not sufficiently satisfied. In fact, in order to differentiate the satisfaction of softgoals from more clear-cut elements, i^* borrows the term *satisficed* from the NFR framework, meaning that a softgoal is satisficed when it is sufficiently satisfied. As well as reasoning about the level of satisfaction of a softgoal, we can also reason about the level of denial, derived from negative evidence. As with the level of satisfaction, the denial of a softgoal can be partial or full, with the term “denied” indicating sufficiently full denial. Here, we use the term *full* to designate elements which are judged sufficiently satisfied (satisficed) or denied, as opposed to elements which are only partially satisficed or denied and. The ideas involved in the partial or full satisficing or denial of elements is central to this work, and will be extensively addressed in Chapter 4 of this work.

The framework contains nine types of contribution links, representing different levels of contribution in either a negative or positive way. The graphical representations

for all links are similar, and represented in Figure 2.12. The *Make* and *Break* links represent full positive and partial contributions, respectively; they alone would be sufficient to judge a softgoal as satisfied or denied. In Figure 2.9 Purchasing PC Products will fully satisfy the softgoal of Abiding by Licensing Regulations, while Obtaining PC Products from a Data Pirate will fully deny the same element. The *Help* and *Hurt* links indicate partial positive or negative contributions, not in itself sufficient to satisfy or deny a softgoal. In Figure 2.9 Purchasing PC Products will partially deny the softgoal of Affordable PC Products, while Obtaining PC Products from a Data Pirate will only partially satisfy this softgoal.

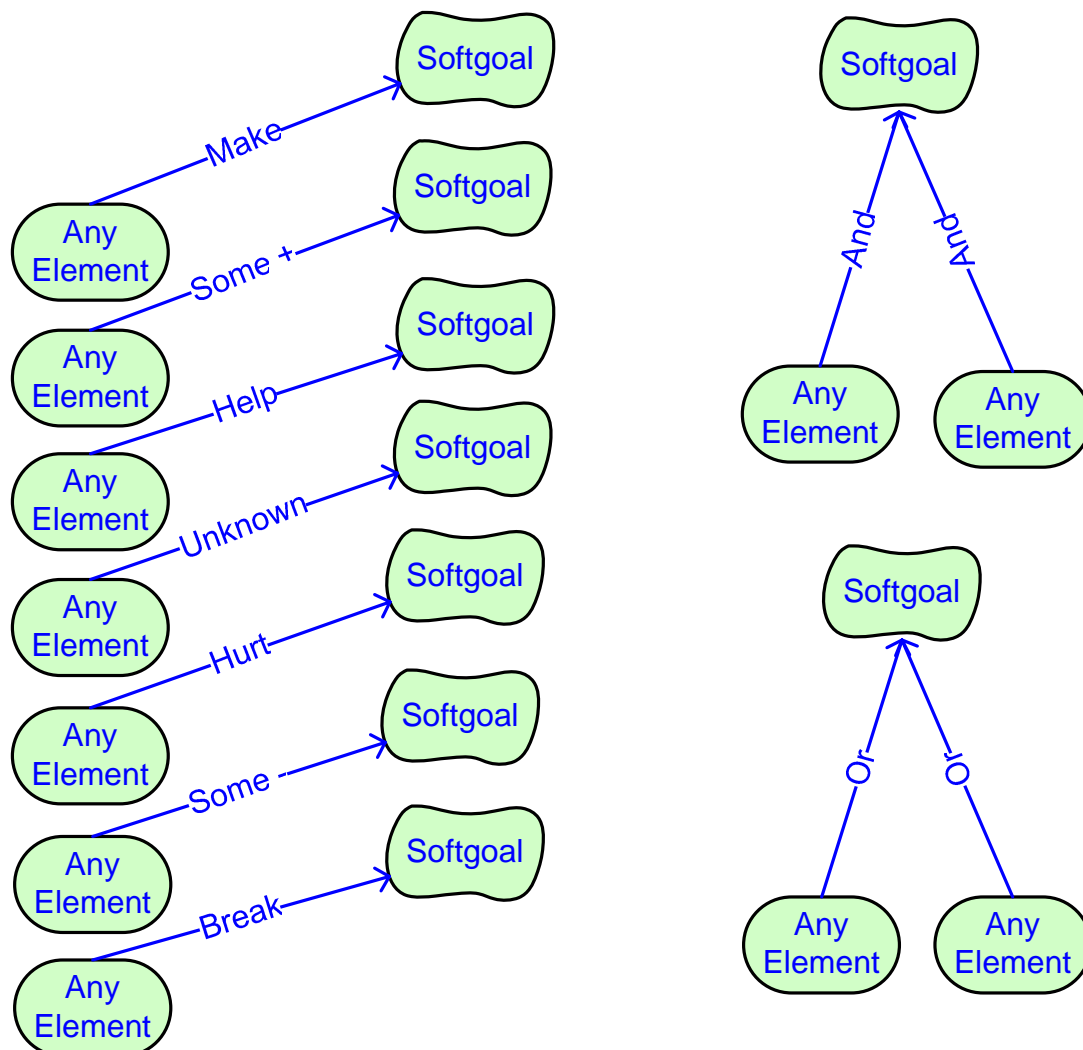


Figure 2.12: Graphical Notation for Contribution Links

This example, in conjunction with the previous discussion on the exclusivity of the means-ends link may bring to mind an important question: if the PC User both Purchases PC Products and Obtains Products from a Data Pirate, then the Abide by Licensing Regulations and Affordable PC Products softgoals will have both negative and positive contributions. What does this mean? As a softgoal may receive many contributions from different nodes, situations involving conflicting contributions such as this are common in i^* . Providing a systematic way to understand and evaluate these types of conflicts and what their meaning is a central focus of this work, and will be explored in Chapter 4.

The *Some+* and *Some-* links indicate positive and negative contributions, respectively, where the strength of the contribution is unknown. A *Some+* could be a help or a make link, and a *Some-* could be a hurt or a break link. The unknown link represents the situation where a contribution is known to exist, but the polarity of the contribution, as well as the strength, is unknown. Finally the *And* and *Or* links have a meaning typical to conventional goal models. All softgoals linked by an *And* contribution, and at least one softgoal linked by an *Or* contribution needs to be satisfied in order for the parent softgoal to be satisfied.

Softgoals and the contributions to softgoals made by elements, particularly elements in a means-ends relationship, are significant constructs in facilitating design tradeoffs via model analysis. In the Figure 2.9 example the PC User's choice in obtaining PC Products will be influenced by the effect of these choices on its softgoals. It can either Purchase PC Products legally and hurt its goal of Affordability, or it can Obtain Products via a Data Pirate, breaking Licensing Regulations, but saving money. Neither situation is ideal. Further analysis and iteration on the model would likely be pursued in order to try and find a better solution, such as finding a PC Product Provider with lower prices.

The i^* framework also provides the means to represent a contribution that is not deliberate or intentional, a contribution which is a side effect. These links, called correlation links, are considered to have the same effect as a standard contribution link, and are given the same labels as regular contributions links. Graphically they are differentiated from contribution links by using a dotted line, see Figure 2.13. Often this extra notation is not used, and both deliberate and side effect contributions are represented using regular contribution links.

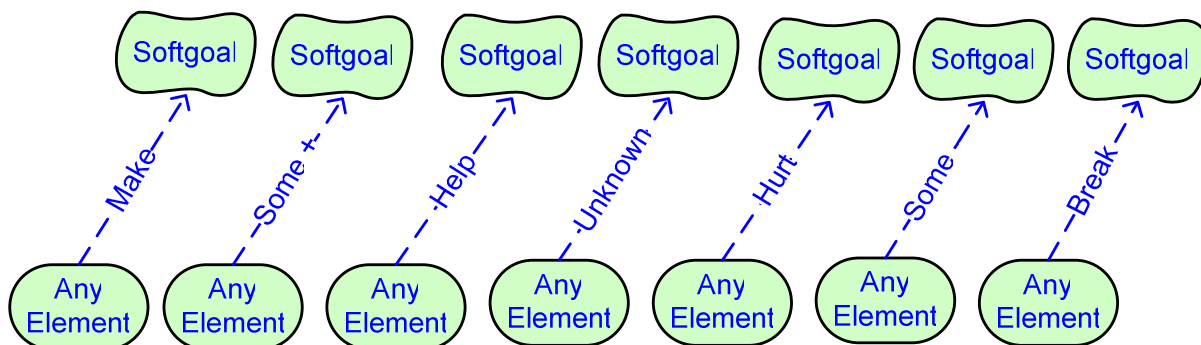


Figure 2.13: Graphical Notation for Correlation Links

Just as a contribution or correlation link can have an effect on a node, a link can also have an effect on another link. This is an i* construct which is less frequently used, but which can provide useful expression, especially when the effects of a link are not symmetric, or the same for both negative and positive evidence. The precise meaning of a link making, helping, hurting, or breaking another link is explored in more detail in Chapter 4 and a graphical example of this effect is shown in Figure 2.14.

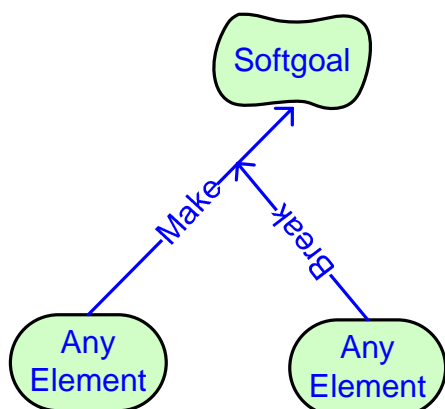


Figure 2.14: Example of a Link to another Link

Elements in i* are designed to express wants. An actor wants a goal to be satisfied, a task to be accomplished, or a resource to be furnished. However, i* also provides a notation to indicate knowledge which influences relevant intentional elements. This construct is called a belief, and is represented by a white cloud that contains a phrase describing a belief that belongs to an actor. The belief's effect on softgoals can be described via contribution or correlation links. Sometimes if the modeler wants to include any extra information critical to the model in a graphical way, the belief construct is used. Figure 2.15 contains examples of belief usage in a variation of the model in

Figure 2.9. In this example, the belief that PC Users may have to purchase products that they do not desire because they are locked into them breaks the decomposition link from Desirable PC Products to Purchase PC Products, meaning that it is no longer necessary for PC Products to be Desirable in order for the PC User to Purchase them.

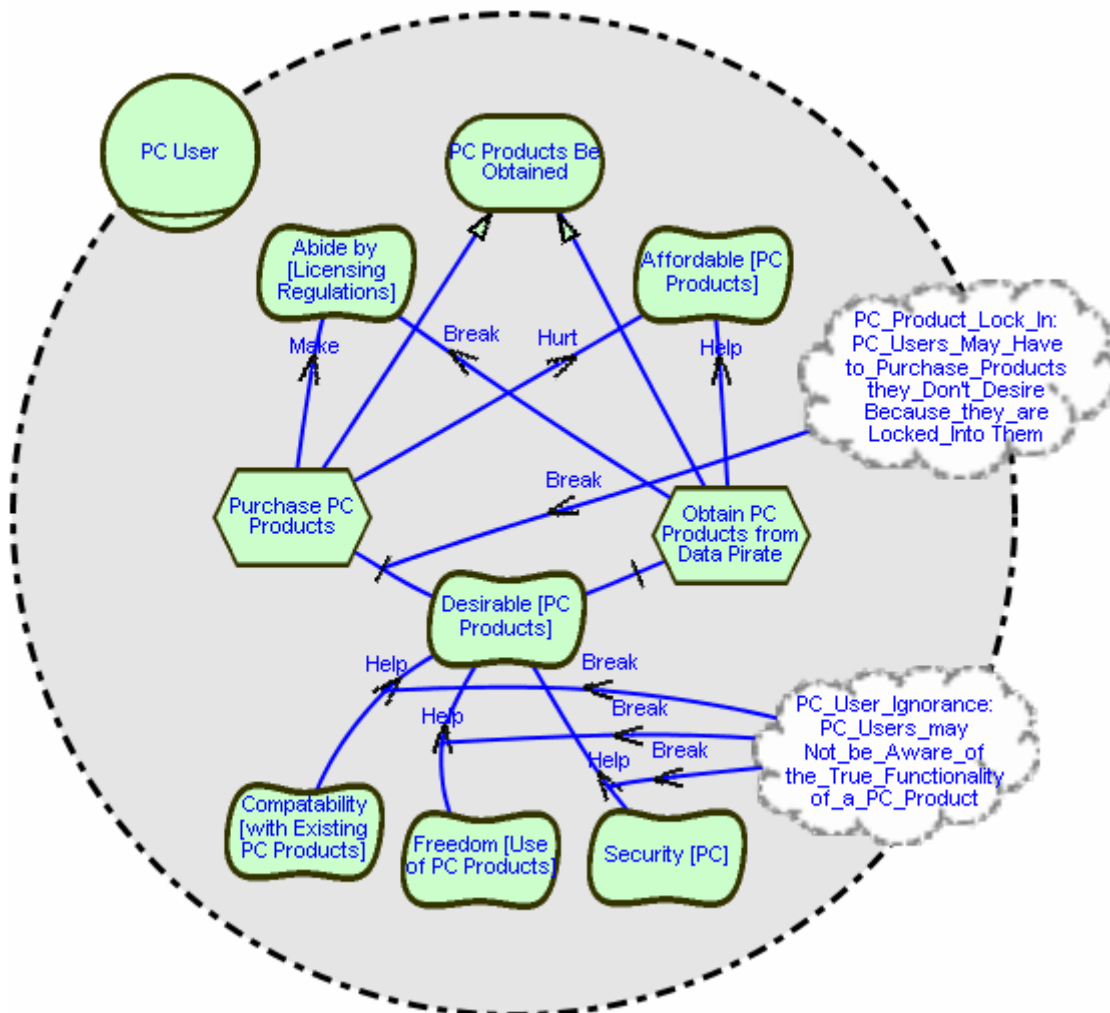


Figure 2.15: Example using Beliefs, adapted from the Trusted Computing Domain, a variation of Figure 2.9

2.3 The Need for Evaluation in the *i Framework**

In Section 2.1 we described the motivation behind the *i** Framework, facilitating early reasoning on socio-technical systems. We have also provided a thorough description of the framework itself. What remains is a detailed explanation of how to

use the framework to meet early reasoning goals; how to perform the analysis which i^* facilitates.

Previous usage of i^* has shown that the process of building an i^* model is a valuable exercise, due to the in-depth domain exploration and greater understanding it provides. However, as we attempt to emphasize in this work, i^* models provide benefits beyond their initial creation, namely when the model is iterated upon, questioning knowledge and assumptions about the domain at each stage. This iteration is fueled by analysis, asking strategic and interesting questions, and searching for the answers to these questions in the domain knowledge captured by the model. When the model is unable to answer such questions adequately, or the answer appears contrary to the modeler's view of reality, this can indicate that the model is insufficient in some way or can reveal qualities in the domain which were not previously apparent. If model deficiencies are revealed, this prompts model correction and expansion, which improves the quality of the model in terms of how accurately it captures a subset of reality. However, without an evaluation procedure which facilitates this cycle of question and refinement in a systematic way, this type of useful analysis is seldom performed, as we can see in the i^* literature, see section 2.1.3 for examples.

We can best demonstrate the benefits of interactive analysis for i^* models, motivating the definition of a systematic evaluation procedure, with a detailed example. Examining Figure 2.9, repeated below in Figure 2.16, prompts numerous strategic domain questions. For example, how does Purchasing PC Products affect the PC User's desire for Affordable Products? We can see that there is a negative effect, shown by the hurt contribution link. How does the PC Product Provider's decision to Allow Peer-to-Peer Technology affect the Data Pirate's ability to Make Content Available through a Peer-to-Peer network? We can trace through the dependency link and see that Allowing Peer-to-Peer Technology will enable the Data Pirate to Make Content Available. However this question could provoke further related questions. Is the availability of the Peer-to-Peer network alone sufficient to Make Content Available Peer-to-Peer? Doesn't the Data Pirate also need to acquire the content? In order to Obtain Pirated PC Products from the Data Pirate, doesn't the PC User also need access to Peer-to-Peer Technology? Is it necessary to represent this explicitly, or does the model imply it?

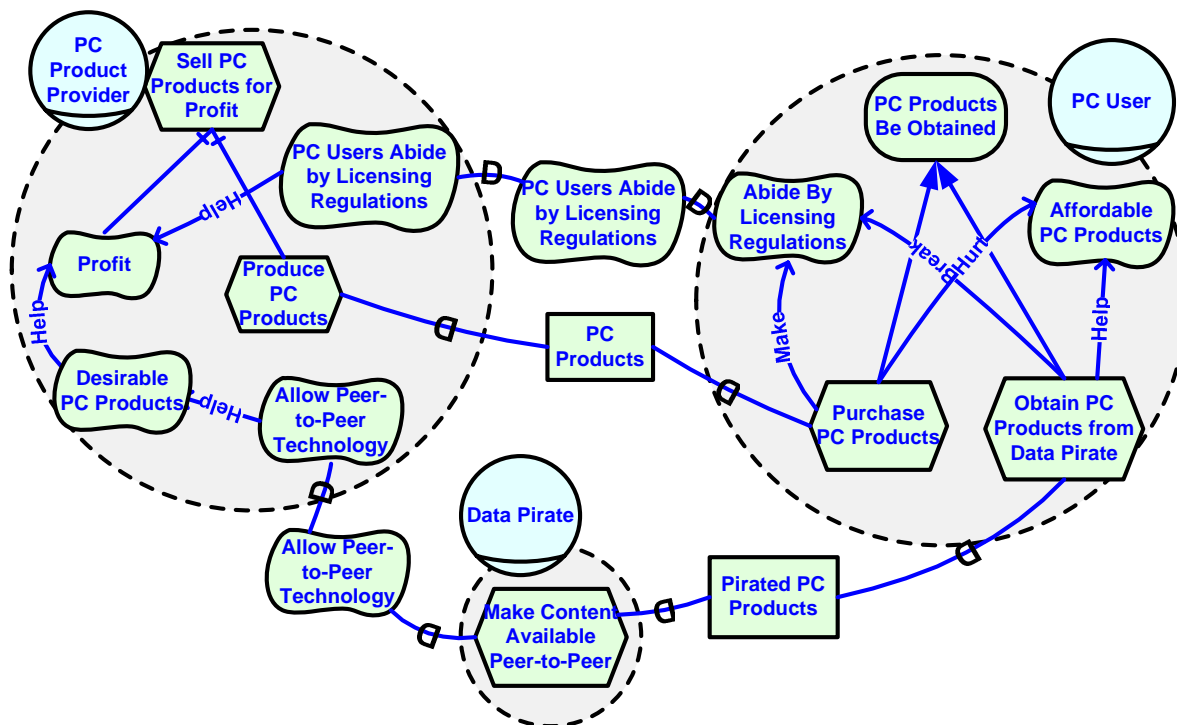


Figure 2.16: SR Model Example, a Simplified Version of a Model from the Trusted Computed Case Study (Repeated from Figure 2.9)

Questions such as this could promote iteration on the model in Figure 2.16, with example results shown in Figure 2.17. This expansion of the model is based on the idea that the Data Pirate and the PC User are often played by the same agent. The agent, Jill in this case, would obtain some content legally, and then play the role of the Data Pirate by distributing it. So the role of a Data Pirate depends on the role of the PC User to obtain PC Products. The model could be expanded further, as shown in Figure 2.18, if it is believed that the PC User's dependency on the PC Product Provider to Allow Peer-to-Peer Technology is important enough to be modeled explicitly.

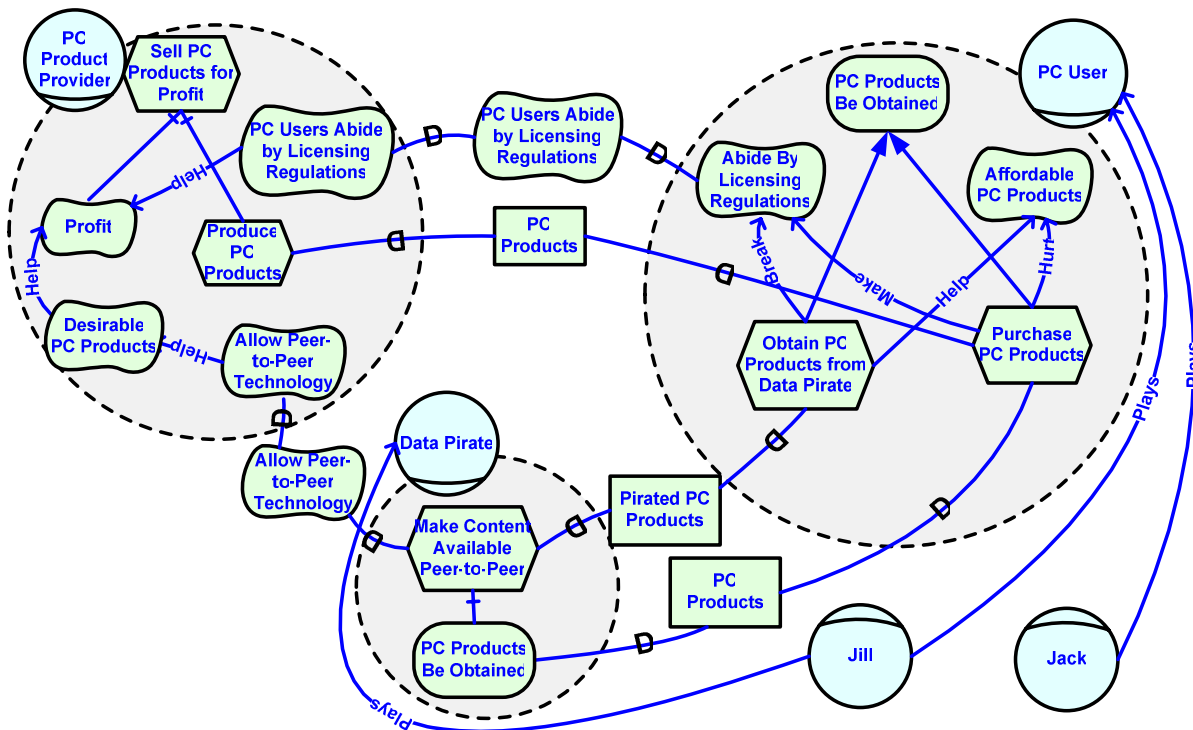


Figure 2.17: Possible Iteration on Figure 2.16 inspired by Strategic Questions

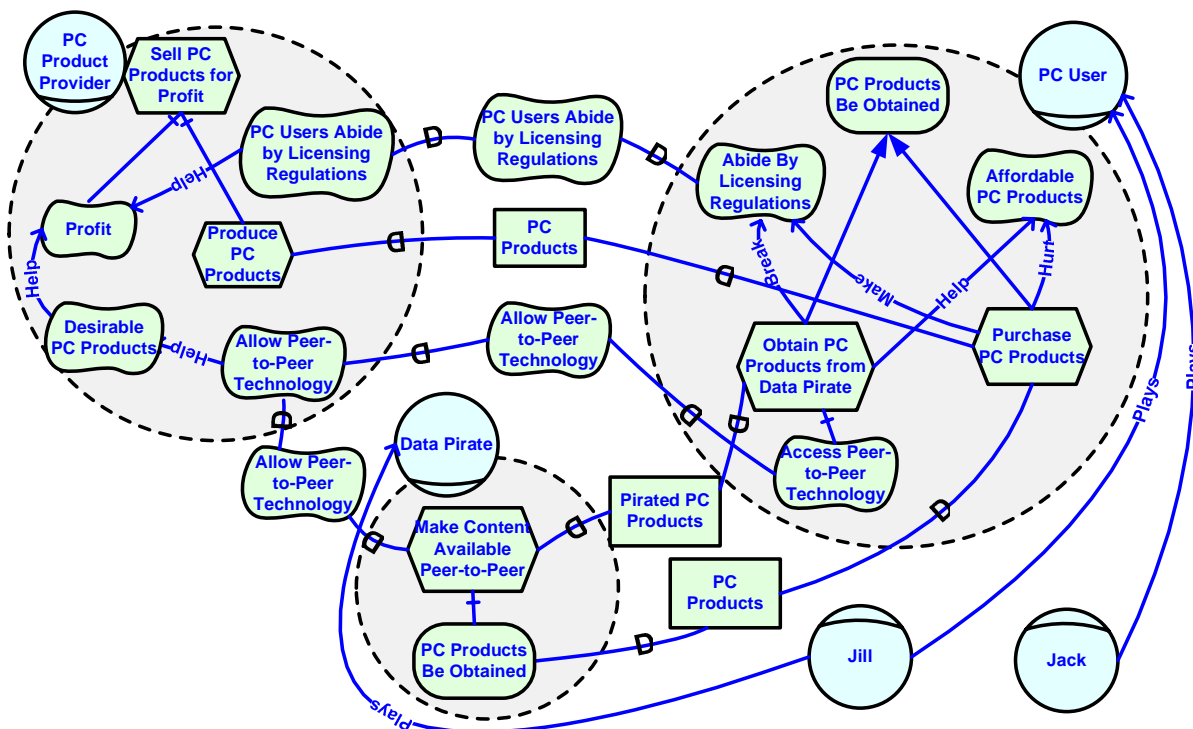


Figure 2.18: Another Possible Iteration on Figure 2.16 inspired by Strategic Questions

Unfortunately, some questions prompted by i^* models are not as simple to answer as the examples we have considered so far. These questions have involved elements whose connections involve only a few links where the structure of the model generally makes the relationships between these elements clear. If we again consider Figure 2.16, we could ask questions whose answers require tracing through multiple links, often through multiple paths. In these cases the relationships between elements is not obvious by studying the model, as the effort required to comprehend and resolve the multiple connections involves a high cognitive load. For example, how does Obtaining PC Products from the Data Pirate effect the PC Product Provider's high-level task of Selling PC Products for Profit? Tracing through the links, we can see that Obtaining PC Products from a Data Pirate breaks the Abide by Licensing Regulations softgoal. This softgoal is depended upon by PC Users Abide by Licensing Regulations, which helps Profit. Profit, which now has a partially negative value, is a decomposition element of Sell PC Products for Profit. Therefore there is a negative effect on this task. However, the presence of partial contribution links leads us to believe that this negative effect is only partial. Therefore, according to our model, Obtaining PC Products from a Data Pirate has a partial negative effect on Sell PC Products for Profit, and this result seems to reflect reality.

Now, what if the PC Product Provider decides to not Allow Peer-to-Peer Technology? What effect will this have on Sell PC Products for Profit? Allow Peer-to-Peer Technology is now denied, and this softgoal helps the Desirable PC Products softgoal, and therefore this softgoal is also partially denied. The existence of the help link to Profit also makes this node partially denied. However, we can also see that the Data Pirate depends on Allow-Peer-to-Peer Technology in order to Make Content Available Peer-to-Peer, and the PC User depends on this task for Pirated PC Products in order to obtain PC Products from a Data Pirate. If the PC User is not able to Obtain PC Products from the Data Pirate, it will likely Purchase PC Products, and therefore Abide by Licensing Regulations is satisfied, and the PC Product Provider's softgoal of PC Users Abide by Licensing Regulation is satisfied, which helps Profit. So Profit is partially satisfied, but if we recall the previous path through Desirable PC Products, it is also partially denied; therefore there exists a conflict for the Profit softgoal.

than peer-to-peer technology, then we would need to adjust the model to reflect this belief, and re-evaluate the results until they conform to our ever changing perception of reality.

The previous examples help to illustrate what types of questions can be answered easily, and what types of questions are more difficult to answer, when examining a model. Questions which involve simply tracing through one or two links, or questions which involve only one simple path of analysis could be answered effectively by simply studying the model, although this depends on the user's familiarity with i^* . However, questions involving longer paths of links, questions that require values to be inversed, questions involving multiple paths or conflicts are all difficult to answer without a systematic way to perform analysis.

Generally, we can divide the types of questions we may ask about the contents of a model into two categories, namely *bottom-up* and *top-down* questions. The questions we have asked so far have looked at the effects of elements "lower" in the model, such as Allow Peer-to-Peer Technology on high-level elements, such as the Sell PC Products for Profit task. We can categorize these questions as bottom-up, generally asking how a contributing element affects an element which receives a direct or indirect contribution from that element. Such questions are often framed as "What if element X were satisfied/denied?" or "How does element X effect element Y?"

In addition, we could ask questions which instead focus on the set of possibilities for an element's satisfaction. Instead of asking how an element affects another element, we ask, "Is it possible for an element to take on this value?" or "What is necessary for one or more elements to be satisfied?" These questions are directed in a top-down manner, looking at the necessary combinations of values for contributing elements needed for the satisfaction of higher-level elements receiving these contributions. For example, what values would lower level elements need to take on in order for Sell PC Products for Profit to be fully satisfied? Upon examining the model we can see that Profit must be fully satisfied in order for this to occur, and in order for Profit to be fully satisfied PC Products must be Desirable, and PC Users must Abide by Licensing Regulations. Further study will show that in order for this to happen, the PC Product Provider must allow Peer-to-Peer Technology, but the PC User must choose to Purchase PC Products, and not Obtain

PC Products through a Data Pirate. It becomes apparent that this sort of top-down analysis would often be non-trivial, and can involve multiple, potentially conflicting paths.

2.3.1 Desired Qualities for an i^* Evaluation Procedure

From work with these and other examples it becomes possible to derive a set of desirable qualities for an i^* evaluation procedure. Research in requirements engineering over the last two decades has produced a proliferation of modeling techniques see (van Lamsweerde, 2000) for examples. Despite this, few of these modeling frameworks have seen widespread industry adoption. One can suggest many reasons for this lack of acceptance, but it seems likely that the perceived cost of using such methods, in terms learning and application, has not surpassed the perceived benefit. By an up-front definition of the qualities for i^* evaluation which would provide significant benefits while maintaining a manageable cost, we hope to develop an evaluation procedure for i^* which is practical for widespread adoption.

In the examination of these desired qualities, it is clear that the emphasis for an i^* evaluation procedure must be put on its ability to facilitate analysis. The model must be able to provide useful answers to strategic domain questions, such as the examples provided earlier for Figure 2.16. In the pursuit of this higher level goal, we can categorize our desired qualities into those qualities that are essential for i^* analysis, those that are beneficial for i^* analysis, but which are not essential, those that facilitate analysis indirectly by addressing the usability of the procedure, and those that facilitate analysis indirectly by improving the quality of the model in terms of how well it captures reality.

Qualities Essential for i^ Analysis*

(i) Element Evaluation. As introduced in Section 2.2, one of the fundamental concepts of i^* involves the notion of satisfaction or denial of an element. This satisfaction indicates whether it will be possible to accomplish an element given the state of achievement and links between other elements in the model. Consequently, an evaluation procedure for i^* should have a way to indicate the level of satisfaction of particular elements. It is clear that there must be a means to keep track of intermediate

satisfaction and denial values, as values from multiple paths are propagated. We can decompose this criterion further, into two specific qualities of element evaluation.

(i.a) Evaluation Value Expressiveness. The means to represent the satisfaction level of an element should be able to cover a range of possible satisfaction and denial values, including unknown and conflicting values. The greater the range of values, the greater the power of expressiveness with these values, up to a certain point of saturation. This is especially true in the high-level analysis for which i^* is intended, as the lack of concrete evidence may make the differences between numerous grades of qualitative evaluation measures meaningless.

(i.b) Overall Evaluation Value. Included in this quality is the ability to arrive at an overall satisfaction or denial value for an element. Specifically, this addresses the frequent presence of multiple contributions to an element. The evaluation of an element should reflect the presence of multiple sources of evidence, including the potential combination of partial evidence into a full evaluation value, avoiding a proliferation of evaluation values representing only partial evidence. Such partial evaluation values offer less potential for conclusive analysis results. An overall assessment of the achievement of model elements is essential for analysis. From this assessment, one can explore the effectiveness of design alternatives in terms of the satisfaction of elements representing the intentional desires of domain actors.

(ii) Clear Procedural Guidelines. In order to facilitate the determination of the satisfaction or denial of an element, the meaning of each combination of link and element must take on a precise definition in terms of this satisfaction and denial. Such meanings are often already embedded in i^* syntax, for example the "and" nature of decomposition links and the "or" nature of means-ends links. The definitions must be translated into a set of *propagation* rules indicating how evaluation values will be propagated through the links in the model to all recipient elements. These rules indicate the satisfaction and denial results received by an element, given the combination of the contributing label, source elements and contributing link type.

(iii) Allowance for Human Intervention. It is desirable, considering the qualitative nature of softgoals, and the high-level analysis facilitated by i^* models, to allow human intervention into the evaluation process. In the early analysis facilitated by

i^* modeling, detailed information such as quantitative measures are often not available, leading to the use of a qualitative framework. It is likely that when creating such models for early analysis, confidence in the correctness of models may not be high, as the relationships between higher-level, abstract elements and actors are often more difficult to capture and formulate. In addition, when modeling the complex interactions of an organizational network, it is infeasible to create a model which captures the domain completely. Human intervention in the evaluation process can be used to compensate for the potential lack of precise information, the possible deficiency in confidence concerning model correctness, and the aspects of the domain missing from the model. Specifically, intervention may be needed when determining a final evaluation value for elements, as described in quality (i), when contributing evidence is partial or contradictory.

(iv) Accuracy. The results of i^* evaluation must be sufficiently accurate, assuming the relative accuracy of the model in depicting the domain. It is necessary here to explore the meaning of accuracy. Accuracy could be considered as a match between the results of model analysis and the analysis results expected by the modeler. However, it is more useful to consider accuracy as a match between real-life results and the qualitative results provided by the model. As intentional and organizational models are an abstraction of a real and likely complex domain, the match between these results may not be exact. However, we would like the predicted positive and negative effects of design alternatives as reported by the model to sufficiently match these effects in the real world. In this way the modeler is able to make interesting discoveries via evaluation, when the model reveals analysis results which are accurate as compared to the domain but were not expected by the modeler.

(v) Usefulness in Multiple Contexts. The functionality provided by the evaluation procedure should be applicable in the various contexts in which i^* can be applied. For example, while analyzing software design processes, the evaluation procedure may be especially useful for ensuring satisfaction of both system-wide and individual goals. In Requirements Engineering a critical functionality provided by an i^* evaluation procedure may be the ability to analyze alternatives and find conflicts. In the analysis of trust, privacy, and security, the ability of the evaluation procedure to provide a

judgment on the level of satisfaction or denial of these elements, represented as softgoals, may be especially beneficial. In general, an evaluation procedure should provide a useful tool to promote high-level design and analysis for all types of applications.

Qualities which are Beneficial for i* Analysis

(vi) Modes of Analysis. Ideally an i* evaluation procedure will facilitate both bottom-up and top-down analysis, allowing questions such as "What happens if...?" and "Is this possible...?". By combining these methods an analyst can ask "Are there alternatives which satisfy my target goals?" and then "What affect does this solution have on all goals in the graph?"

(vii) Traceability. When performing analysis, in addition to knowing the evaluation result for each element, it may be useful to easily see the initial and intermediate sources which contributed to this result. In this way, one can determine which evaluation the elements which affected the outcome of a particular element.

(viii) Conflict Detection. When analyzing a model, it may be useful to have a specific indication of which elements receive conflicting contributions. In this way, one can better identify areas in the model where the determination of overall element evaluation values may be difficult, or depend heavily on domain knowledge.

(ix) Constraints on Values. Providing the ability to constrain the results of the evaluation procedure could be useful for analysis. In this way a user could indicate that certain elements should or should not take on certain evaluation values, with the procedure indicating whether or not these constraints can be met by the resulting evaluation values.

(x) Facilitating Cost Analysis. It may be useful to analyze the relative costs of design solutions. Likely, in the high-level analysis for which i* is intended, specific cost information may not be available, or the measurement for such cost will not be universally applicable to all model elements. However, it would be useful to be able to represent and analyze the relative costs of each solution, perhaps using more approximate measurements.

Qualities Addressing the Usability of i* Analysis

(xi) Simplicity. Considering the overall desire to minimize the effort required to apply i* evaluation, it is important that an evaluation procedure for i* be devised to be as simple as possible. This simplicity should include the ability to apply the procedure manually on models of small to medium size, (up to 50 elements), without great difficulty. If the procedure is kept simple, it is far more likely that it will be adopted, as its benefits will be seen to outweigh its costs.

(xii) Automation. In order to address the cost of evaluation in terms of time, it is essential that the procedure be automatable. Manual propagation of evaluation values via an application of the propagation rules could be potentially tedious in models of any substantial size. Therefore, automation is essential for the evaluation of medium to large models, and is likewise useful for the application of many different potential evaluations for small to medium models.

Qualities Addressing Model Quality

(xiii) Syntax Checking. Applying an evaluation procedure has the potential to prompt a systematic and detailed examination of model constructs, bringing to light syntax errors. For example, incorrect link types between nodes, wrong link directions, and inappropriate element types are all likely to become apparent during an application of an evaluation procedure. In addition, application of an evaluation procedure may push modelers to avoid using i* constructs which are precise and unambiguous. We would like to develop an evaluation procedure which makes it easier to pick out syntactic errors in the model.

(xiv) Semantic Improvement. In addition to the detection of syntax errors, we would like to develop an evaluation procedure which facilitates the detection of semantic faults in i* models. Upon the consideration of evaluation results it could become apparent that these results do not sufficiently reflect reality, potentially illuminating semantic faults in the underlying model constructs. In this case the model can be iterated until the perceived gap between the model and reality is filled. Of course the model can

never completely and accurately reflect reality, especially the complex social driven systems modeled by i^* ; therefore, the modeler must judge whether the level of completeness and accuracy is sufficient for analysis. The modifications made to the model in Figure 2.16, shown in Figure 2.17 and 2.18 are examples of this sort of iteration, prompted by semantic deficiencies. In general, we would like to produce an evaluation procedure which provokes this type of beneficial iteration.

2.3.2 Desired Qualities: Synergies and Conflicts

When describing the desired qualities of an i^* evaluation procedure, complementary and detrimental interactions between these qualities become apparent. By defining Clear Procedural Guidelines (ii) we facilitate the receipt of contributions allowing Element Evaluation (i). The Allowance for Human Intervention (iii) may allow the evaluation results to better reflect reality, improving Accuracy (iv). In general, any quality providing analysis capabilities will help to facilitate the Usefulness of an Evaluation Procedure in Multiple Contexts (v). It could be asserted that the Allowance for Human Intervention (iii) helps to facilitate Syntax Checks (xiii) and Semantic Improvement (xiv) by prompting the modeller to carefully consider model constructs when intervention is required. In addition, the improvement of model quality through Syntax (xiii) and Semantic (xiv) changes could help to improve the Accuracy (iv) of the evaluation results.

However, conflicts exist between many of the qualities desired for i^* evaluation. For instance, the desire for defined Clear Procedural Guidelines (ii) contradicts our desire to Allow Human Intervention (iii). If all propagation and label resolution were defined through guidelines, human intervention would not be needed. If human intervention is allowed, there is the potential for this intervention to supersede the defined procedural guidelines.

In addition, we can see a conflict between the desire to Allow for Human Intervention (iii) and the desire for Automation (xii) in i^* evaluation. If one fully automates the procedure, rules are needed to eliminate the need to allow human intervention of any kind. If, on the other hand, one intervenes in the procedure, the

procedure may become only semi-automated, reducing efficiency and increasing the effort required in order to perform evaluation. Similarly, we can see a potential conflict between Automation (xii) and the desire for Syntax (xiii) and Semantic (xiv) changes. When applying the evaluation procedure manually, the modeler will likely have to consider and review model constructions, in order to perform the manual propagation. This careful consideration may lead to the discovery of model faults. However, if the procedure is automated, the user may not study the model in detail unless evaluation results are unexpected.

One can see multiple qualities which have the potential to conflict with the desire for Simplicity (xi). It is important to keep this desired quality in mind when defining the clear procedural guidelines, when outlining the role of human intervention, and when considering the addition of all of the qualities which may provide additional capabilities for analysis (vi – x). Especially problematic may be the desire to facilitate both directions of analysis, as the top-down determination of potential evaluation combinations producing a desired result is a computationally difficult problem. If it were very difficult for an evaluation procedure to easily facilitate both types of analysis, the bottom-up analysis is likely the most important for the high level analysis required by i^* , as this is the type of analysis which best facilitates the selection of alternatives.

In general, just as the evaluation of a design alternative within an intentional model should attempt to find the most appropriate alternative by examining the achievement of various, often conflicting, non-functional requirements, we must produce an evaluation procedure which effectively balances the desired qualities of i^* evaluation.

2.4 Conclusion

In this chapter we have introduced the application of system modeling, motivating the need for modeling frameworks which provide intentional ontologies, such as goal modeling frameworks, and organizational ontologies, such as i^* . We have briefly introduced contexts of i^* application in order to provide a foundation domain knowledge required in evaluation.

The constructs of the i* Framework have been described in detail. We have used a simple i* example from the Trusted Computing domain to demonstrate the need for evaluation by highlighting the types of questions which are difficult to answer without a systematic analysis procedure. Finally, we have identified a series of desired qualities for an i* evaluation procedure, and explored the synergies and conflicts between these qualities. These qualities shall be used a basis for the evaluation of goal model evaluation procedures in the next chapter, and will be used to assess the effectiveness of the procedure developed in this work.

Chapter 3: Evaluation in Goal Modeling Frameworks

3.1 *Evaluation for Goal Models vs. Evaluation for i**

The need for a procedure for model evaluation is not unique to the *i** Framework. Research in goal modeling has included methods of model evaluation since its conception (Mylopoulos, Chung, & Nixon, 1992). Various goal model evaluation methods have been proposed for goal models, including the NFR and KAOS Frameworks (Chung, Nixon, Yu, & Mylopoulos, 2000), (Letier & van Lamsweerde, 2004). Although it may be possible to transfer one or more of these methods directly for use with the *i** Framework, we must first consider whether such methods meet the desired qualities for evaluation in *i**, and whether the intentions of these frameworks and their evaluation methods are similar enough to the intentions of *i** and *i** evaluation.

Goal models are mainly intended for prescriptive design, capturing the structure of a potential new system, typically starting at high-level system goals and moving towards operationalizations. As described, goal model evaluation procedures are used to ensure that top-level goals are sufficiently met by design choices. The process of choosing between design alternatives is facilitated by the assessment of goal achievement for each alternative. Choosing between alternatives using goal models becomes a process of trade-off and negotiation. The objective is to find the design solution which best addresses the goals captured in the model.

On the other hand, the *i** Framework is intended to first capture the current domain of the potential system, or current system for which changes must be made. It focuses on social interactions and an in-depth understanding of the goals and interactions that motivate the problem that the system addresses. It is used to discover goals which may not have been apparent and to discover, as well as explore, design alternatives. As a result of the differing intentions of goal modeling frameworks and the *i**, the desired qualities we have outlined for an evaluation procedure in Chapter 2 may not match the desired qualities for goal model evaluation procedures. For instance, as the nature of goal modeling is more prescriptive, evaluation procedures may be less favorable to the idea of

Allowance for Human Intervention (iii). In addition, evaluation for goal modeling has not explicitly addressed a desire to provoke iteration, improving the quality of models through evaluation (xiii, xiv).

Despite the differences between the intentions of i^* and other modeling frameworks, there exists significant overlap in the desired qualities for both types of procedures. Both evaluation in the i^* Framework and evaluation for goal models should include capabilities for analysis, including an indication of the satisfaction of elements (i), set Clear Procedural Guidelines (ii), Accuracy (iv), and other helpful features such as Conflict Detection (viii) and Traceability (vii). As a result, even though goal model evaluation methods may not be appropriate for direct adaptation for i^* use, the formulation of an i^* evaluation procedure may be aided by choosing the most appropriate goal model evaluation procedure to serve as a base. This may facilitate familiarity with the evaluation procedure for those familiar with goal model evaluation. The expansion of the base evaluation method to i^* syntax should be done in such a way that it facilitates the high-level, social analysis for which i^* is intended. In this chapter, we shall explore existing goal model evaluation in order to select a base for i^* evaluation in Chapter 4. Beneficial elements of other procedures will be considered for future inclusion in the i^* evaluation procedure in Chapter 9.

3.2 Goal Model Evaluation Procedures

This section explores evaluation procedures for goal models, including models in the NFR and KAOS frameworks, in order to gain a broader picture of the possibilities for goal model evaluation. As the different approaches offered by each evaluation procedure are likely to satisfy a different set of desired evaluation qualities, it is useful to explore, contrast, and compare these methods. The evaluation of these methods against the desired qualities for i^* evaluation will aid in the choice of a method to act as a foundation for the i^* evaluation procedure.

3.2.1 The CNYM Method

The method proposed for goal models using the NFR Framework is described in depth in (Chung et al., 2000). We shall refer to this method as the CNYM method, after its authors. This method uses the concept of *satisficed* to represent sufficient evidence of goal satisfaction. The converse of *satisficed* is represented by the term *denied*. In this method, all goals are softgoals, as defined in Chapter 2. The procedure uses six qualitative labels to represent fully *satisficed*, partially *satisficed*, *unknown*, *conflict*, partially *denied* and *denied*. Partial *satisficing/denial* refers to the situation where there exists positive/negative evidence towards the satisfaction/denial of a goal, but this evidence is not sufficient to judge the goal as fully *satisficed/denied*. Here, as in Chapter 2, we use the term *fully* to distinguish evaluation values which are judged as sufficiently *satisficed* or *denied* from values which are partially *satisficed* or *denied*. *Unknown* represents the case where no positive or negative evidence exists, and is the default label for the evaluation procedure. *Conflict* indicates the presence of both positive and negative evidence for a goal.

The evaluation proceeds by placing labels on a subset of goals. This set of initial labels indicates the high-level *design alternative* currently being evaluated. This is distinguished from the use of the term *alternative* to represent an alternative task or means, which satisfies a goal through a means-ends link. These initially labeled goals are often leaf nodes in the graph. For example, in the NFR model shown in Figure 3.1, if one wanted to evaluate the effect of choosing *CertificationBy [HighSummary.Manager]* over *CertificationBy [LowSummary.Manager]* without *Confirmation [Summary]*, the evaluation procedure would start with the *HighSummary* goal given a label of *satisficed*, and the *LowSummary* and *Confirmation* goals given a label of *denied*. Not all leaf nodes need to be assigned explicit labels, as they are given a default label of *unknown*, shown by the *Certification [Summary]* goal in Figure 3.1. The labels are propagated from offspring to parent goals in a procedure involving two steps. Both propagation rules and human judgment are used. The rules indicate what labels are propagated, given the label of the offspring and the type of link. See Table 3.1 for a description of these rules and the graphical representation of the node labels. In step one of the evaluation procedure, all

current values are propagated from offspring to parent using the propagation rules. As a softgoal may receive more than one label via more than one contribution link, these labels must be combined into a single label, possibly requiring human judgment. Step two involves assigning such a label to the parent nodes. The procedure suggests collecting the labels for one parent node in a bag, allowing for duplicates. All partial values are combined together into one or more full, unknown or conflict labels and the final result is the minimum of these combined labels, with an order of:

Conflict \leq Unknown \leq Denied \approx Satisfied.

If both a satisfied and denied label remain, the result is called a *conflict*. These steps are repeated until all values have been propagated.

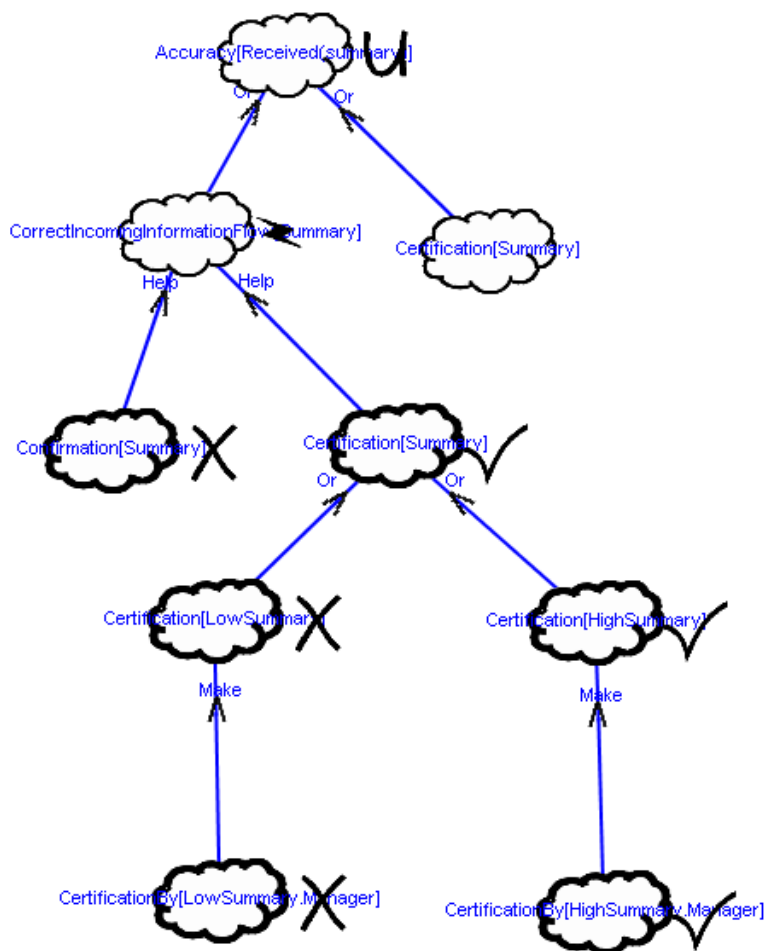


Figure 3.1: An example NFR model reproduced from (Chung et al., 2000)

The rules for the procedure propose the promotion or demotion of partial values to either satisfied, denied, unknown, or conflict, resulting in the avoidance of partial values as final labels for a goal. Here, *promotion* would occur when a partially satisfied value is taken as sufficient evidence for satisfied, or when a partially denied value is taken as sufficient evidence for denied. Likewise, *demotion* would occur when these values are treated as no evidence, resulting in unknown. However, in the rest of the book, the procedure is expanded to allow for partial values as a final label of a node, as it allows the propagation of extra information. The rules shown in Table 3.1 allow for such an expansion by using partial values as input labels.

The human judgment required in this procedure is a point of interest. It is up to the analyst to merge multiple, often conflicting values into one. This process should make use of domain knowledge, including knowledge of the relative importance of each contributing goal. Such human judgment is necessary as it is not possible to automate this step of evidence merging, without ignoring the specific source of the contributions, causing the evaluation decision to disregard the domain.

Table 3.1: CNYM Method Evaluation Labels and Propagation Rules, reproduced from (Chung et al., 2000)

Child Node Label	Contribution Type and Parent Result							
	Label Name	Make	Break	Help	Hurt	Some+	Some-	?
√	Satisfied	√	X	W+	W-	W+	W-	U
W+	Partially Satisfied	W+	W-	W+	W-	W+	W-	U
C	Conflict	C	C	C	C	C	C	U
U	Unknown	U	U	U	U	U	U	U
W-	Partially Denied	W-	W+	W-	W+	W-	W+	U
X	Denied	X	W+	W-	W+	W-	W+	U

We can assess this evaluation procedure in terms of the desired qualities for i^* evaluation as described in Chapter 2. **(i) Element Evaluation.** The procedure provides the means to evaluate the achievement of elements via use of the six qualitative labels. **(i.a) Evaluation Value Expressiveness.** Although these labels could be seen to be

sufficiently expressive for a high-level qualitative analysis, their level of expressiveness could potentially be increased by including other values indicating the relative strength of positive or negative evidence. **(i.b) Overall Evaluation Value.** The allowance for human intervention allows for the combination of multiple sources of evidence into an overall evaluation value for each goal. The possibility for manual promotion of evaluation values helps to avoid a proliferation of partial values.

(ii) Clear Procedural Guidelines. As described in Table 3.1, this method defines propagation guidelines in terms of the relationship between the evaluation value of contribution goals, links between goals, and the evaluation label received by the recipient goal. In addition, methods are described to combine multiple evaluation labels into a single overall label, often involving human intervention.

(iii) Allowance for Human Intervention. This method allows the intervention of users in order to make decisions concerning the combination of multiple sources of evidence. In this way, domain knowledge which may not be explicitly included within the model can be used to derive accurate evaluation results. **(iv) Accuracy.** The description of the NFR evaluation procedure provided in (Chung et al., 2000) argues for the accuracy of the results through the use of analytical arguments and successful applications to multiple examples. A further application of this procedure to multiple detailed examples, similar to what is done in Chapter 7 of this work, would be beneficial.

(v) Usefulness in Multiple Contexts. The application of this procedure to multiple examples in (Chung et al., 2000) could be seen as a demonstration of its usefulness in multiple contexts. However, this framework is typically proposed for use in the analysis and design of technical systems. Applications to other contexts, such as Knowledge Management or Business Process Analysis, although possible, have not been extensively explored.

(vi) Modes of Analysis. This particular procedure provides only a bottom-up or cause and effect mode of analysis. Adaptation to a top-down mode of analysis would be difficult to due to the allowance for human intervention in the procedure. The additional desired qualities which may add analysis capabilities to an evaluation procedure such as **(vii) Traceability, (viii) Conflict Detection, (ix) Constraints on Values, and (x) Facilitating Cost Analysis** have not been explicitly included in this procedure.

(xi) Simplicity. The use of a small set of qualitative evaluation values, in conjunction with the use of the intuitive meanings of terminology such as help, hurt, make and break, produce a procedure which is relatively simplistic. Although the decisions required in the intervention of human judgment may be difficult, the role of human judgment within the procedure is straightforward, required when evidence must be combined. **(xii) Automation.** Due to the need to allow human intervention, the procedure can be only partially automated. The viability of this partial automation has been proven through an implementation in a conceptual modeling tool (<http://www.cs.toronto.edu/km/ome/index.html>).

(xiii) Syntax Checking and **(xiv) Semantic Improvement.** The procedure does not explicitly mention the possibility for checking the syntax and improving the semantics of models through application of evaluation. However, as explored in Chapter 2, the allowance for Human Interventions (iii) in the procedure may facilitate the awareness of such faults through the required careful examination of modeling constructs.

3.2.2 The GMNS Method

The goal model evaluation procedure introduced in (Giorgini, Mylopoulos, Nicchiarelli, & Sebastiani, 2002) and expanded on in (Giorgini, Mylopoulos, Nicchiarelli, & Sebastiani, 2004), which shall be referred to as GMNS after its authors, contains many similarities to the CNYM procedure. In fact GMNS could be seen as an adaptation or formalization of this earlier procedure. Both procedures are qualitative, and can be seen as “bottom-up”, propagating labels from model leaf goals to high-level goals. However, some significant differences between the methods exist. In the GMNS method the degree of satisfaction or denial is represented as a predicate over a goal. These predicates range from full evidence of satisfaction, FS, to partial evidence of satisfaction, PS, to partial evidence of denial, PD, to full evidence of denial, FD. In this case the term satisfaction, meaning that there is least full evidence that a goal is satisfied, is used to represent the achievement of a goal, as opposed to satisficing as used in the NFR procedure. Each goal

is assigned two variables, Sat and Den, over the range of $\{F, P, N\}$, representing the level of evidence for the satisfaction and denial of a goal, with F, P and N representing full, partial, or none. The predicates FS(G), PS(G), PD(G), and FD(G) are defined as $Sat(G) \geq F$, $Sat(G) \geq P$, $Den(G) \geq P$, and $Den(G) \geq F$, respectively. In this procedure the distinction between “hard” goals vs. softgoals is not made explicit.

Formalizing the satisfaction of goals in this manner creates a separation of negative and positive evidence. This allows the procedure to be fully formalized and automated by a set of rules, shown in Table 3.2, as there is no need to deal with the presence conflicts before propagating evaluation values. Note that the goal models used in the GMNS method allow for *non-symmetric* contribution labels. Labels of S or D on the link indicate that the Sat or Den values are propagated, respectively. An absence of any letter on the link indicates that the values are propagated *symmetrically*, meaning both Sat and Den values are propagated. We can see an example goal model for this procedure in Figure 3.2.

Table 3.2: Propagation Rules for the GMNS Method, taken from (Giorgini et al., 2004)

	$(G_2, G_3) \xrightarrow{and} G_1$	$G_2 \xrightarrow{+S} G_1$	$G_2 \xrightarrow{-S} G_1$	$G_2 \xrightarrow{++S} G_1$	$G_2 \xrightarrow{--S} G_1$
Sat(G_1)	$\min \left\{ \begin{array}{l} Sat(G_2), \\ Sat(G_3) \end{array} \right\}$	$\min \left\{ \begin{array}{l} Sat(G_2), \\ P \end{array} \right\}$	N	Sat(G_2)	N
Den(G_1)	$\max \left\{ \begin{array}{l} Den(G_2), \\ Den(G_3) \end{array} \right\}$	N	$\min \left\{ \begin{array}{l} Sat(G_2), \\ P \end{array} \right\}$	N	Sat(G_2)
	$(G_2, G_3) \xrightarrow{or} G_1$	$G_2 \xrightarrow{+D} G_1$	$G_2 \xrightarrow{-D} G_1$	$G_2 \xrightarrow{++D} G_1$	$G_2 \xrightarrow{--D} G_1$
Sat(G_1)	$\max \left\{ \begin{array}{l} Sat(G_2), \\ Sat(G_3) \end{array} \right\}$	N	$\min \left\{ \begin{array}{l} Den(G_2), \\ P \end{array} \right\}$	N	Den(G_2)
Den(G_1)	$\min \left\{ \begin{array}{l} Den(G_2), \\ Den(G_3) \end{array} \right\}$	$\min \left\{ \begin{array}{l} Den(G_2), \\ P \end{array} \right\}$	N	Den(G_2)	N

As has been done with the previous procedure, we shall assess this method using the desired qualities for i*evaluation. **(i) Element Evaluation.** The procedure provides the capability to assign values indicating the achievement and denial of goals, including indicators of full and partial evidence. **(i.a) Evaluation Value Expressiveness.** The expressiveness of the values used in this procedure is equivalent to the expressiveness of the CNYM values. **(i.b) Overall Evaluation Value.** In this scheme, conflicts occur when a goal is fully or partially satisfied, (FS(G) or PS(G)), and fully or partially denied, (FD(G) or PD(G)). The separation of negative and positive evidence in this method

causes such conflicts to be propagated throughout the graph, without resolution. In addition, when faced with multiple contributions to a single goal, the GMNS algorithm takes the maximum contribution, meaning that multiple partial values are not promoted to a full value. These effects, resulting from the continuous separation of positive and negative evidence, result in the frequent proliferation of partial or conflicting overall evaluation values, making it more difficult to derive conclusive analysis results.

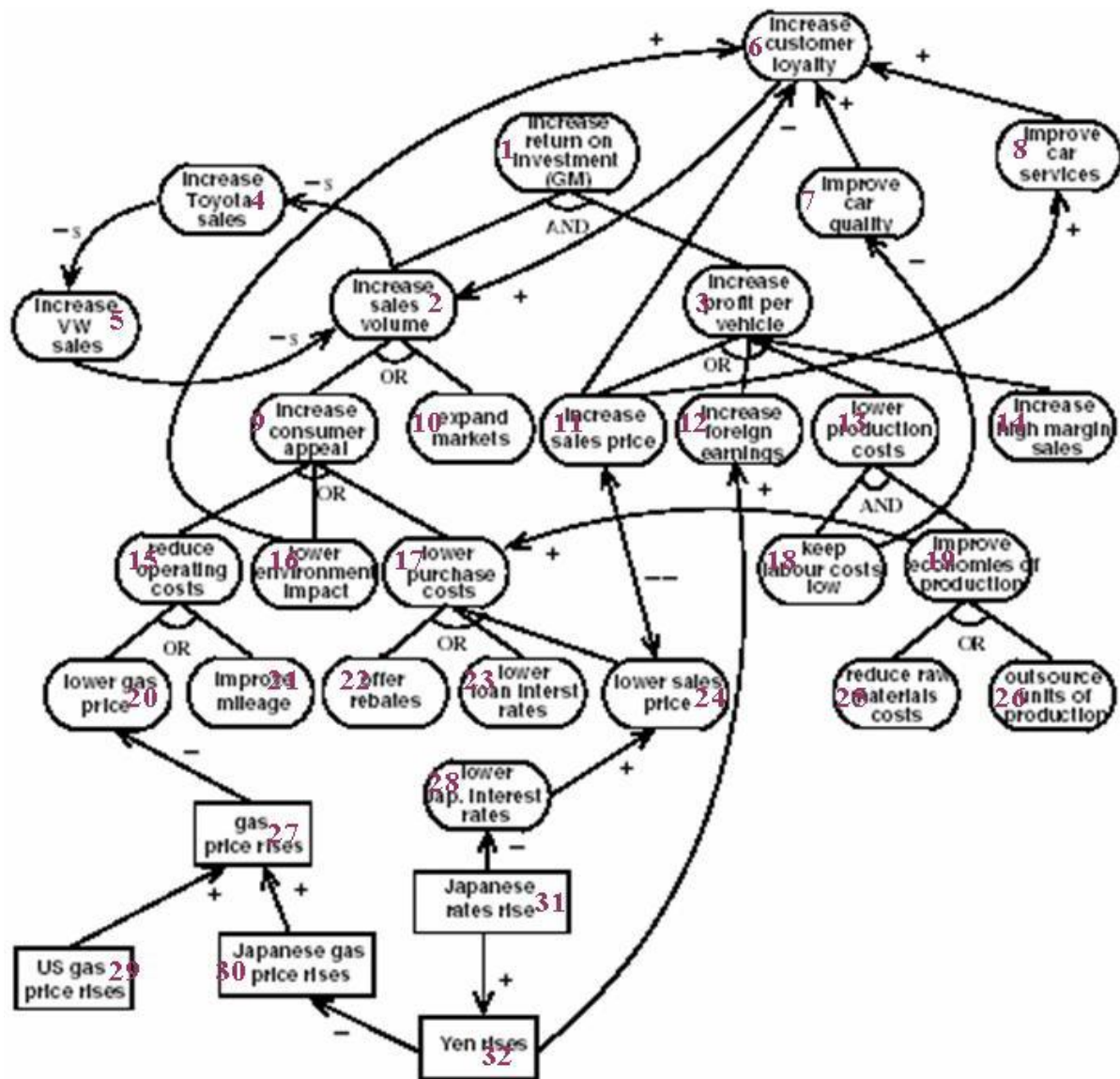


Figure 3.2: A partial goal model for GM, appearing in (Giorgini, Mylopoulos, Nicchiarelli & Sebastiani, 2002) and (Giorgini, Mylopoulos, Nicchiarelli & Sebastiani, 2004)

(ii) Clear Procedural Guidelines. Clearly defined propagation rules for this procedure are provided, as described in Table 3.2. **(iii) Allowance for Human Intervention.** This procedure does not include a role for the intervention of users. This is mainly due to the separation of positive and negative evidence, meaning that human judgment is not needed to combine conflicting evidence. Additionally, there is no role for human intervention in combining, promoting or demoting partial values. **(iv) Accuracy.** The argument for the accuracy of the evaluation results in this procedure is made via the successful application of the procedure to experimental examples described in (Giorgini, Mylopoulos, Nicchiarelli, & Sebastiani, 2004) and (Giorgini, Mylopoulos, Nicchiarelli, & Sebastiani, 2002). However, it is not clear whether this method has been applied to a real-life application. **(v) Usefulness in Multiple Contexts.** An assessment of the usefulness of this method in multiple contexts via applications to multiple examples or case studies has not been performed. Therefore it is difficult to determine whether or not this quality is achieved.

(vi) Modes of Analysis. The description of this particular method is directed to bottom-up analysis; however, this method has been adapted for top-down evaluation, as is described in the Section 3.2.4 concerning the SGM-TD Method. This method does not explicitly describe any methods to provide **(vii) Traceability**, to **(x) Facilitate Cost Analysis**, or to put **(ix) Constraints on Values**. **(viii) Conflict Detection.** Although an overt method for detecting conflicts is not described, the separation of evidence in this procedure makes it easy to detect conflicts, when both positive and negative evidence are present in the same goal.

(xi) Simplicity. The basic operations involved in this procedure are relatively simplistic. In comparison to the CNYM Method considered previously, the avoidance of human intervention could be seen to make this procedure simpler, as the user does not need to intervene with potentially complex decisions. However, the propagation of two types of evidence for every node may be seen as more complex than the single-value-per-node CNYM Method. Additionally, the full automation of this procedure may contribute to its simplicity. **(xii) Automation.** As human intervention is not required, this procedure can be fully automated. Tests performed in (Giorgini et al., 2002) and (Giorgini et al., 2004) show that the procedure is guaranteed to terminate, having gone

through a reasonable amount of iterations, making the running time of the procedure manageable, indicating that the implementation is practical for real use.

(xiii) Syntax Checking and (xiv) Semantic Improvement. As the procedure does not enforce a careful consideration of some aspects of the model due to the need for human intervention, it could be argued that the procedure does not explicitly facilitate the discovery of syntax and semantic improvement. However, if the evaluation results are surprising, an examination of the model is likely to occur, and this examination may result in such discoveries.

3.2.3 The GMNS-# Method

The GMNS procedure is adapted to produce a quantitative version, described in (Giorgini et al., 2002) and (Giorgini et al., 2004), and referred to here as GMNS-#. In order to propagate quantitative values, the goal model contribution links must be adjusted to contain numerical weights. Goals are again given *Sat* and *Den* variables, but this time their values range over a numerical interval, $[\text{inf}, \text{sup}]$, where *inf* represents no evidence and *sup* represents full evidence. In the examples, a range of $[0, 1]$ is used, both for the satisfaction and denial of goals and the weights of contribution links. The rules are adjusted to deal with these numerical values via the introduction of the \oplus operator, used as disjunction or "max", and the \otimes operator, used as conjunction or "min". The \otimes operator is defined as typical multiplication. The \oplus operator is defined as follows:

$$x \oplus y = x + y - x \cdot y.$$

In this scheme, the results of contributions indicate the conditional probability of the parent goal being satisfied, given the child goal. Therefore, the application of this numerical model to a goal graph creates a Bayesian network. The propagation rules for this method are shown in Table 3.3.

Table 3.3: Propagation Rules for the GMNS-# Method, taken from (Giorgini et al., 2004)

	$(G_2, G_3) \xrightarrow{\text{and}} G_1$	$G_2 \xrightarrow{w+s} G_1$	$G_2 \xrightarrow{w-s} G_1$	$G_2 \xrightarrow{++s} G_1$	$G_2 \xrightarrow{--s} G_1$
$Sat(G_1)$	$Sat(G_2) \otimes Sat(G_3)$	$Sat(G_2) \otimes w$	$Sat(G_2) \otimes w$	$Sat(G_2)$	$Sat(G_2)$
$Den(G_1)$	$Den(G_2) \oplus Den(G_3)$				
	$(G_2, G_3) \xrightarrow{\text{or}} G_1$	$G_2 \xrightarrow{w+p} G_1$	$G_2 \xrightarrow{w-p} G_1$	$G_2 \xrightarrow{++p} G_1$	$G_2 \xrightarrow{--p} G_1$
$Sat(G_1)$	$Sat(G_2) \oplus Sat(G_3)$		$Den(G_2) \otimes w$		$Den(G_2)$
$Den(G_1)$	$Den(G_2) \otimes Den(G_3)$	$Den(G_2) \otimes w$		$Den(G_2)$	

(i) Element Evaluation. In this procedure, a measure of the goal achievement is provided by separate quantitative values for positive and negative evidence. **(i.a) Evaluation Value Expressiveness.** The quantitative values provide a higher level of expressiveness than comparable qualitative measures, such as in the GMNS or CNYM procedures. It is more expressive to describe a node as 0.2 or 20% satisfied than it is to describe a node as partially satisfied. However, unless such numbers are grounded in domain evidence, the increase in the level of expressiveness may be misleading, indicating only a more fine-grained qualitative measure. **(i.b) Overall Evaluation Value.** Like the GMNS method, multiple partial values are not promoted to full values by the algorithm. For example, if a goal were to receive the partial positive values of 0.5, 0.2, and 0.3, the final value would be 0.5, not 1.0. Therefore, the overall evaluation values are more likely to be partial, and may be less helpful in facilitating analysis.

(ii) Clear Procedural Guidelines. This procedure provides clear propagation guidelines as described in Table 3.3. **(iii) Allowance for Human Intervention.** As in the GMNS procedure, the separation of positive and negative evidence means that conflicts are not resolved. As a result, human intervention is not required. However, at least some of the criteria used in human judgment in the qualitative methods, such as relative importance of goal offspring, can be explicitly encoded in the model.

(iv) Accuracy. As in the GMNS method, arguments for the accuracy of this procedure are made by the successful application to examples. In addition, one could argue that the results of this procedure are more accurate, in comparison to the qualitative procedures, as they take into account the distance of propagation. It is likely that the propagation of evidence through multiple links decreases its relevance on recipient goals. In a qualitative framework, partial evidence propagated through ten positive contribution links arrives as P, the same strength as partial evidence propagated through only one link. In the quantitative methods, the “diluting” of values propagated over long distances is

accomplished by numeric partial contribution labels, which can consecutively reduce the strength of the evidence.

The use of quantitative evidence may lead one to believe that the evaluation results have a higher accuracy in general. However, one must consider the source of the numbers used in this procedure. Ideally the numbers used in both contribution weights and initial propagation values will be rooted in reality, perhaps as indications of variables such as cost or time. However, each goal may involve a different variable; one goal may be Decrease Time to Learn while another is Decrease Cost. As a result, finding a grounded universal measurement for contributions is difficult. The quantitative values are likely to be determined by domain experts in a relative comparison of contributions. By this line of thought, the quantitative values are only fine-grained qualitative values, in that the numbers on the graph are likely to have no correspondence to real life measurements. Unfortunately, this lack of correspondence may not be intuitively clear to all users of the procedure, and there are dangers in putting too much trust in accuracy of the numerical results.

As this procedure is based on the GMNS Procedure, the assessment of qualities including **(v) Usefulness in Multiple Contexts**, **(vi) Modes of Analysis**, **(vii) Traceability**, **(viii) Conflict Detection**, and **(ix) Constraints on Values** are similar to the assessment for the GMNS Procedure. **(x) Facilitating Cost Analysis**. The use of numerical propagation may facilitate the inclusion of cost information into the propagation. However, as was explored in the assessment of accuracy, even if cost information can be obtained it may not be sensible to propagate this information to other goals in the model which do not directly relate to cost.

(xi) Simplicity. Although the propagation rules used in this method are not especially complex, they require calculations above and beyond what is required in the qualitative methods. Nevertheless, it could be argued that the avoidance of human intervention and the automation of this procedure produce a method which is simple. **(xii) Automation**. This procedure inherits the automation benefit of the GMNS qualitative method.

(xiii) Syntax Checking and (xiv) Semantic Improvement. Like the qualitative version of this method, the discovery of model faults is not encouraged by the automation of the procedure.

3.2.4 The SGM-TD Method

The GMNS procedure is again adapted to produce a top-down procedure that searches for combinations of lower-level goals which would produce desired high-level values. This procedure is described in (Giorgini, Mylopoulos & Sebastiani, 2004), and shall be referred to as SGM-TD. The scheme of predicates over goals with Sat and Den variables, as well as the propagation rules, remain the same as in the GMNS procedure. However, the propagation in this method is intrinsically more complex, as the search for lower-level values producing acceptable higher-level values can produce multiple possibilities.

The procedure uses propositional satisfiability (SAT), the problem of finding a satisfying assignment of variables in a boolean formula. It uses a state-of-the-art SAT solver which makes calculation practically accomplishable, despite the NP-complete nature of the problem. The structure of the goal graph, the desired set of goal values, as well as the axioms for backward propagation are converted into a formula, which can be used as input for the SAT solver. In addition, this method allows for the addition of constraints and a restriction of conflict levels. **Constraints on Values (ix)** can be added by specifying whether or not certain goals should take on certain values, for example: $PS(G1) \text{ or } \neg FD(G2)$. Conflict avoidance can be specified by avoiding all strong conflicts (FS and FD), all strong and "medium" conflicts (FS and PD, or FD and PS), or all conflicts. The SGM-TD procedure also allows for consideration of costs. Leaf goals in the goal graph can be assigned relative costs. The evaluation procedure is adjusted to find the minimum cost solution which will satisfy constraints and result in the desired top-level values.

It is relevant to note that although a description of an SGM-TD-# procedure is not described in (Giorgini, Mylopoulos & Sebastiani, 2004), this method is reportedly under development by the authors.

As this procedure is a further adaptation of the GMNS procedure, its assessment in terms of **(i) Element Evaluation**, **(i.a) Evaluation Value Expressiveness**, **(i.b) Overall Evaluation Value**, **(ii) Clear Procedural Guidelines**, **(iii) Allowance for Human Intervention**, **(iv) Accuracy**, **(v) Usefulness in Multiple Contexts**, **(vii) Traceability**, **(viii) Conflict Detection**, **(xiii) Syntax Checking**, and **(xiv) Semantic Improvement** is analogous to the assessment for the GMNS procedure, and will not be repeated here.

(vi) Modes of Analysis. Unlike the previous methods we have investigated, this method facilitates a top-down mode of analysis, answering questions such as “Given the structure of a graph, can a goal be achieved?” As mentioned in the description of the procedure, this procedure both **(ix) Constraints on Values** and **(x) Facilitates Cost Analysis**.

(xi) Simplicity. In terms of the simplicity of the procedure, the conversion of the graph into a boolean formula, in conjunction with the use of a SAT solver, make this procedure quite complex. In fact, the level of complexity is so great, that it would be difficult to perform this procedure manually on any graph which is not very small. **(xii) Automation.** This procedure has the benefit of full automation; with experimental running times for large graphs of less than five seconds.

3.2.5 The Jarvis Method

The final goal evaluation procedure is a modification of the GMNS procedure described in (Jarvis, 1992), and given here the name of Jarvis, after its author. This procedure emphasizes traceability in the source of positive and negative evidence. Each goal is given a reference number. Propagation rules similar to GMNS are used, but the procedure propagates multiple positive and negative labels, each with the source goal number. Therefore, at the end of propagation, the analyst knows not only what qualitative labels a goal has, but also where they have originated. This facilitates a different sort of analysis, and makes it easier to apply judgment if the aim is to come up with a final merged label.

In a large model high-level goals may have numerous labels. In AND and OR links, more than one number should be propagated. This can result in a "flat" representation of much of the graph's tree structure in the resulting labels. As this procedure is again an adaptation of the GMNS procedure, we will only consider the qualities whose assessments differ from the GMNS procedure assessment.

(ii) Clear Procedural Guidelines. The propagation guidelines differ from the GMNS procedure as values for all sources of evidence are propagated, as opposed to one value for all sources having the same evidence label. However, the propagation guidelines concerning how to represent the source of evidence for And or Or links are not clearly defined; however, rules concerning the other aspects of propagation can be derived from the GMNS Procedure.

(vii) Traceability. The major distinguishing feature of this procedure is its provision of traceability. However, as mentioned, deciphering this traceability could be complicated when labels contain multiple sources due to And or Or links. **(xi) Simplicity.** Although the basics of this method are simplistic, like the GMNS method, this method can be considered complicated due to the proliferation of labels and the difficulty in processing sources through And or Or links.

3.2.6 Evaluation for the KAOS Framework

Evaluation for the KAOS Framework has traditionally been relatively simple, due to the presence of formally defined goals and only binary AND and OR links. Recently, however, partial satisfaction for goals in KAOS has been introduced in (Letier & van Lamsweerde, 2004), providing tools to evaluate alternatives in terms of these partial values.

The procedure for evaluating the partial satisfaction of goals as a means to choose between design alternatives in KAOS involves the creation of a probabilistic model, similar to a Bayesian Network. Each goal is given a set of cumulative distribution functions, called objective functions, over random variables, called quality variables. The objective functions have a mode indicating whether they should be minimized or maximized, a target probability, and a current probability. Objective functions do not

need to be defined for every goal, as the propagation is performed over quality variables. Quality variables for leaf nodes need to be probabilistically independent, and if they're not they need to be more fully refined.

The procedure uses domain specific equations called refinement equations to relate the quality variables of a parent goal to the quality variables of its sub-goals. In alternative goal refinements, different refinement equations are needed for each alternative. Sometimes these equations can be complex, using probability distributions and additional variables. The actual propagation is performed by acquiring distribution estimations for leaf nodes and reformulating refinement equations into probability density functions in order to compute objective functions. The measure of partial satisfaction for a goal is equivalent to the probabilities for the objective functions. This method can be applied in both a bottom-up and top-down manner.

Making comparisons between the methods of evaluation for NFR models and KAOS models is problematic due to the different fundamental concepts underlying the Frameworks and their evaluation methods. In the KAOS Framework all goals, even those that represent NFRs, are "hard" goals, in that their satisfaction can be judged via clear-cut criteria. Where the NFR Framework may have a softgoal Increase Profit, the equivalent KAOS goal may be something like Profit increased by 12% by the First Quarter of 2006. In fact, it may seem counterintuitive to assign a value of partially satisfied to such a hard goal, as its nature indicates a binary level of satisfaction; either it is true or it is not. This is one of the motivations for introducing soft, qualitative goals, so that a value of partial satisfaction could be applied when precise measurements were unavailable. This is especially useful during system development, when precise measurements are difficult to acquire (Mylopoulos et al., 1992).

The introduction of partial goal satisfaction in KAOS, may seem at first to be attributing a qualitative judgment of partial satisfaction to hard goals. However, upon closer examination the NFR and KAOS Frameworks assign different meanings to the notion of satisfaction. In the NFR Framework satisfaction indicates that there exists either sufficient (full satisfaction) or insufficient (partial satisfaction) evidence towards the satisfaction of a goal. In KAOS the notion of satisfaction refers to the probability of a variable associated with a goal being in a precise range. If this probability is 1.0, the goal

is fully satisfied, if it is less than 1.0; the goal is partially satisfied, specified quantitatively by the probability. One could actually see the two definitions as complimentary, satisfaction using qualitative judgments for softgoals, and satisfaction using probabilities for hard goals.

We can assess this procedure using our desired qualities for i^* evaluation. **(i) Element Evaluation.** The KAOS procedure provides a representation of element achievement by providing a probability for the satisfaction of an element. In this case the representation of negative evidence for a goal could be seen as $1 -$ the probability of the objective function for a goal. **(i.a) Evaluation Value Expressiveness.** As the measurement of satisfaction involves a probability, the value is expressive. **(i.b) Overall Evaluation Value.** As the representation of satisfaction involves only a single value, not separating positive and negative evidence, and as the refinement equations can account for all sources of evidence, the evaluation values can be considered to represent the overall evaluation of a goal.

(ii) Clear Procedural Guidelines. The method describes propagation guidelines in terms of the probabilistic model, including random variables, cumulative distribution functions, and refinement equations. **(iii) Allowance for Human Intervention.** As the relationships between all goals are clearly defined before propagation using refinement equations, and as the definition of such equations eliminates the notion of a conflict, human intervention is not required in this method.

(iv) Accuracy. The KAOS evaluation method includes the ability to provide a more accurate estimation of the results of a design decision on non-functional requirements. Certainly saying that there is an 80% chance that profit will increase by 12% by the First Quarter of 2006 is more precise and descriptive than saying that Increase Profit is partially satisfied, or even that Increase Profit is satisfied by 0.7. In fact the authors of the KOAS method state that the non-measurable nature of goals and quantitative satisfaction levels for other frameworks "violates a fundamental principle of requirements engineering calling for precise and measurable requirements and specifications" (Letier & van Lamsweerde, 2004, p. 3).

However, it is explained that leaf quality variable distributions are estimates and refinement equations use simplifying assumptions. Therefore the final numbers are not

intended to be accurate, and are intended to be used only to compare relative strengths of alternatives. But this is exactly the intention behind the qualitative and quantitative evaluation procedures for the NFR Framework. Although there is no precise meaning to saying a softgoal such as Increase Profit is partially satisfied, or is satisfied by 0.7, the usefulness of these values are in their comparative power for alternatives. Therefore, although this procedure may have more capability for accuracy than the qualitative procedures, its accuracy is still limited.

(v) Usefulness in Multiple Contexts. The application of this procedure to a detailed example concerning the London Ambulance Service has been described in (Letier & van Lamsweerde, 2004). Beyond this example, it is difficult to determine whether this method will be useful in many contexts, especially as it requires the ability to obtain specific measurements of domain aspects.

(vi) Modes of Analysis. This procedure facilitates both bottom-up and top-down analysis. **(vii) Traceability.** The refinement equations used in this procedure specifically define the relationships between directly related goals. However, a method for tracing these relationships throughout the structure of the model is not described. **(viii) Conflict Detection.** As this procedure does not include the notion of a conflict, this feature is not relevant. **(ix) Constraints on Values.** Constraints in this procedure could be factored into the various equations used. In fact, the use of a target value for goal objective functions could be seen as the incorporation of a constraint. **(x) Facilitating Cost Analysis.** The procedure does not explicitly address an analysis of costs.

(xi) Simplicity. This method requires knowledge of probability theory in order to be applied correctly. Although this may not be an unreasonable demand on researchers, a systems analysis may not be familiar with these theories, depending on their area of expertise. In addition, it can be asserted that the KAOS evaluation method requires a lot of precise information such as objective functions, quality variables, refinement equations, and leaf distributions, all of which may be difficult and time consuming to obtain, depending on the system.

(xii) Automation. Like several of the other methods, this method is fully automatable. **(xiii) Syntax Checking and (xiv) Semantic Improvement.** This method requires a detailed and careful definition of the relationships between goals, in order to

express them in probabilistic equations. If we assume that the definition of these relationships is not a part of the evaluation procedure itself, but instead part of the processes of creating the model, then the automatic procedure is not likely to prompt the discovery of model faults unless the evaluation results are surprising. If, instead, we consider the definition of this information as part of the procedure itself, then the syntax and semantics of the model could be greatly improved and refined.

3.2.7 Experimental Result Comparison for the NFR Framework Evaluation Methods

In order to better understand and compare these methods, we have evaluated an example model using equivalent initial evaluation values. Evaluation results for Figure 3.2, a model taken from (Giorgini et al., 2002) and (Giorgini et al., 2004), are provided in Table 3.5. This table compares the CNYM, GMNS, and GMNS-# methods. The results for the Jarvis method are shown in Table 3.6, due to lack of space. The KAOS method is not included in this comparison, as it is difficult to make a correlation between the results of this procedure and the results of the other procedures. In addition, we are not able to acquire the specific probabilistic information for the example model.

In order to perform a method comparison, equivalences in notation and qualitative and quantitative labels must be made. These equivalences are shown in Table 3.7. The graphical labels for the CNYM procedure are abbreviated textually as FS, PS, PD, and FD. To compare qualitative and quantitative values, we have assumed that partial contribution links are equivalent to the midpoint between [inf, sup], in this case 0.5. Likewise, partial positive and negative evidence is given a default value of 0.5.

The initial and final values for each procedure are given in the Init and Fin columns in Table 3.5. The results for the GMNS procedures, which use separate positive and negative values, are given a final merged value. This merging is done with the assumption that FS and PD = PS, PS and PD = N, PS and FD = PD, and FS and FD = N. Applying a formulaic merging of the values goes beyond the specifications of the GMNS procedure, but is necessary in order to make a comparison of the overall evaluation

results. The final column in the table indicates, given the equivalencies, whether there is a difference in results between the three methods.

Table 3.4: Results Comparison of Three Goal Model Evaluation Procedures

Goal Name and Number		GMNS Qualitative ¹					CNYM Qualitative		GMNS Quantitative					Dif Y/ N
		Init		Fin		Fin Mgd	Init	Fin	Init		Fin		Fin Mgd	
Internal Goals		S	D	S	D				S	D	S	D		
Increase return on investment (GM)	1	N	N	F	P	PS	N	PS	0	0	1.0	0.5	S 0.5	N
Increase sales volume	2	N	N	F	P	PS	N	PS	0	0	1.0	0.5	S 0.5	N
Increase profit per vehicle	3	N	N	F	N	FS	N	FS	0	0	1.0	0	S 1.0	N
Increase customer loyalty	6	N	N	P	P	Con	N	PD	0	0	0.5	0.25	S 0.25	Y
Improve car quality	7	N	N	N	P	PD	N	PD	0	0	0	0.5	D 0.5	N
Improve car services	8	N	N	N	P	PD	N	PD	0	0	0.25	0.5	D 0.25	?
Increase consumer appeal	9	N	N	F	N	FS	N	FS	0	0	1.0	0	S 1.0	N
Lower production costs	13	N	N	F	N	FS	N	FS	0	0	1.0	0	S 1.0	N
Reduce operating costs	15	N	N	F	N	FS	N	FS	0	0	1.0	0	S 1.0	N
Lower purchase costs	17	N	N	F	N	FS	N	FS	0	0	1.0	0	S 1.0	N
Improves economies of production	19	N	N	F	N	FS	N	FS	0	0	1.0	0	S 1.0	N
Lower gas price	20	N	N	P	N	PS	N	PS	0	0	0.125	0	S 0.125	?
Gas price rises	27	N	N	N	P	PD	N	PD	0	0	0	0.25	D 0.25	?
lower Japanese interest rates	28	N	N	N	N	N	N	N	0	0	0	0	0	N
Japanese gas price rises	30	N	N	N	P	PD	N	PD	0	0	0	0.5	D 0.5	N
Increase Toyota Sales	4	F	N	F	P	PS	FS	PS	1.0	0	1.0	0.5	S 0.5	N
Increase VW Sales	5	F	N	F	P	PS	FS	PS	1.0	0	1.0	0.5	S 0.5	N
Increase Sales price	11	N	P	N	F	FD	PD	FD	0	0.5	0.0	1.0	D 1.0	N
Increase foreign earnings	12	N	F	P	F	PD	FD	PD	0	1.0	0.5	1.0	D 0.5	N
Lower Sales price	24	F	N	F	N	FS	FS	FS	1.0	0	1.0	0.0	S 1.0	N
		Initial and Final		Merged		Initial and Final		Initial and Final		Merged				
Leaf Goals		S	D					S	D					
Expand markets	10	P	N			PS	PS	0.5	0			S 0.5		
Increase high margin sales	14	N	P			PD	PD	0	0.5			D 0.5		
Lower environment impact	16	F	N			FS	FS	1.0	0			S 1.0		
Keep labor costs low	18	F	N			FS	FS	1.0	0			S 1.0		
Improve mileage	21	F	N			FS	FS	1.0	0			S 1.0		
Offer rebates	22	P	N			PS	PS	0.5	0			S 0.5		
Reduce raw materials cost	25	F	N			FS	FS	1.0	0			S 1.0		
Yen Rises	32	F	N			FS	FS	1.0	0			S 1.0		

It is interesting to analyze where the differences among methods occur, and why. We can see that in Table 3.5 the three methods produce results which are similar with a

few exceptions. For example, in the CNYM method a final value of partially denied is given to Increase Customer Loyalty, taking into account the partial positive contribution of Lower Environment Impact and Lower Sales Price and the partial negative contribution due to the denial of Improve Car Quality and Improve Car Services. In this case human intervention (iii) was used to judge that that the Improve Car Quality and Improve Car Services goals are more important for Customer Loyalty than the other two goals, and therefore Increase Customer Loyalty is partially denied. In the GMNS method the combination of opposite partial values for Increase Customer Loyalty produces a conflict. The quantitative method produces yet another result for this goal, taking into account the longer propagation distance for the negative evidence, giving the positive evidence a slightly higher strength.

However, the relative importance of Car Quality and Services could have been expressed in this method by increasing the partial contribution weights, making them greater than 0.5. There are also cases, indicated with a ‘?’ in the Dif column, where the quantitative method produced results that were of the same polarity as the qualitative methods, but possessed a finer precision of values. In general, this example shows that although the similar principles behind the methods often produce similar results, but the inclusion of human judgment in the CNYM procedure and contribution weights in GMNS-# have the potential to produce results which are significantly different from the GMNS method.

Table 3.6 shows the results of the Jarvis procedure applied to the same model in Figure 3.2. The initial values are equivalent to those in Table 3.5, making the results comparable to the other procedures in Table 3.5. We can see that the presence of multiple labels, although slightly cumbersome, gives a better understanding of the different sources of evidence that contribute towards the satisfaction or denial of a goal, facilitating traceability (vii). In the GMNS method, we know only that goal 6 was partially denied, but here we can see that goals 18, 11, and 24 all have a partial contribution to this denial, providing specific domain information which may facilitate a more in-depth analysis.

Table 3.5: Results for Jarvis Method on Model in Figure 3.2

Goal Name and Number		Jarvis			
		Init		Fin	
Internal Goals		S	D	S	D
Increase return on investment (GM)	1	N	N	F(16/24/21):(18:25)	P18, P5, P24, P11
Increase sales volume	2	N	N	P24, P11, P16, F16/24/21	P18, P24, P5, P11
Increase profit per vehicle	3	N	N	F18:25	N
Increase customer loyalty	6	N	N	P24, P11, P16	P18, P11, P24
Improve car quality	7	N	N	N	P18
Improve car services	8	N	N	N	P11, P24
Increase consumer appeal	9	N	N	F16/24/21	N
Lower production costs	13	N	N	F18:25	N
Reduce operating costs	15	N	N	F21	N
Lower purchase costs	17	N	N	F24, P25	N
Improves economies of production	19	N	N	F25	N
Lower gas price	20	N	N	P32	N
Gas price rises	27	N	N	N	P32
lower Japanese interest rates	28	N	N	N	N
Japanese gas price rises	30	N	N	N	P32
Increase Toyota Sales	4	F	N	F4	P11, P16/24/21, P16, P24
Increase VW Sales	5	F	N	F5	P4
Increase Sales price	11	N	P	N	F24, P11
Increase foreign earnings	12	N	F	P32	F12
Lower Sales price	24	F	N	F24, P11	N

Table 3.6: Required Equivalences for Evaluation Method Comparison

GMNS Notation	CNYM Notation	GMNS-# Notation
Satisfaction and Denial Values		
Sat(G) \geq F	FS	Sat(G) = 1.0
Sat(G) \geq P	PS	Sat(G) = 0.5
Sat(G) or Den(G) = N	N	Sat(G) or Den(G) = 0.0
Den(G) \geq F	FD	Den(G) = 1.0
Den(G) \geq P	PD	Den(G) = 0.5
Contribution Link Labels		
++ (s, d, or no label)	++	+ 1.0
+ (s, d, or no label)	+	+ 0.5
- (s, d, or no label)	-	- 0.5
-- (s, d, or no label)	--	- 1.0

3.3 Choosing a Goal Model Evaluation Procedure as a Basis for *i** Evaluation

We will use the assessment of the goal model evaluation procedures in terms of the desired qualities for *i** evaluation to help ensure that the most appropriate goal model evaluation procedure is chosen as a base of adaptation for *i** evaluation. In order to summarize our assessments and facilitate a final decision, we express the effectiveness of each procedure in achieving the desired qualities in tabular form. As our desired qualities can be considered softgoals, for each quality and each procedure we will assign one of the qualitative values used in the CNYM procedure, using the textual representations from the GMNS procedures. Namely, S for satisfied, PS for partially satisfied, C for conflict, PD for partially denied, D for denied, and U for unknown. These assessment values, based on the arguments in the above sections, are contained in Table 3.7.

Table 3.7: Assessment of Evaluation Methods using the Desired Qualities for *i Evaluation**

	Essential Qualities							Beneficial Qualities					Usability		Model Quality	
	i	i.a	i.b	ii	iii	iv	v	vi	vi	vii	viii	ix	x	xi	xii	xiv
	Evaluation Value	Expressiveness	Overall Value	Procedural Guidelines	Human Intervention	Accuracy	Multiple Contexts	Modes of Analysis	Traceability	Conflict Detection	Constraints	Cost Analysis	Simplicity	Automation	Syntax Checking	Semantic Improvement
CMNY	S	P	S	S	S	P	U	C	D	D	D	D	S	PS	PS	PS
GMNS	S	P	P	S	D	P	U	C	D	PS	D	D	S	S	C	C
GMNS-#	S	S	P	S	D	P	U	C	D	PS	D	D	PS	S	C	C
SGM-TD	S	P	P	S	D	P	U	C	D	PS	S	S	PD	S	C	C
Jarvis	S	P	P	P	D	P	U	C	P	PS	D	D	PD	S	C	C
KAOS	S	S	S	S	D	P	U	S	D	U	PS	D	D	S	C	C

Overall, we can see that the KAOS procedure generally achieves the qualities necessary for evaluation, with the exception of an Allowance for Human Intervention

(iii), and provides the capabilities for Modes of Analysis (vi), Constraints on Values (ix) and Automation (xii). However, as it is not effective in addressing our desire to Allow for Human Intervention (iii), the qualities identified as beneficial other than Constraints on Values (ix), and our desire for an improvement in model quality. In assessing the general effectiveness of the Jarvis procedure, it seems that in comparison to the GMNS procedure, the gains the Jarvis procedure makes in Traceability are not worth the losses in terms of Simplicity (x) and clarity in Procedural Guidelines (ii).

Concerning the GMNS-# procedure, although this procedure satisfies most of our desires for the essential evaluation qualities, it is not as simplistic as the CMNY and GMNS procedures, does not Allow for Human Intervention (iii) and does not clearly provide an improvement in model quality. In addition, although this procedure contains evaluation values which are more expressive, this extra expressiveness may not be useful in the context of high-level analysis where specific domain measurements are difficult to obtain, as we have mentioned previously. Overall, this procedure is likely not the most appropriate choice for adaptation for i^* evolution.

The GMNS and GMNS-TD Procedures use the same representation of evaluation and propagation rules, they share the ability to provide Evaluation Values (i), Clear Procedural Guidelines (ii), implicit Conflict Detection (vii), and full Automation (xi). However, they lack the ability to Allow for Human Intervention (iii), a sufficient means to derive an overall evaluation value, and an improvement in model quality. The GMNS Procedure offers Simplicity (xi), while the GMNS-TD procedure offers beneficial qualities such as Constraints on Values (ix) and Cost Analysis (x). As we have indicated a preference for bottom-up as opposed to top-down analysis in i^* evaluation, and as simplicity is a critical quality for i^* evaluation, the GMNS-TD procedure shall not be chosen for adaptation to i^* .

Out of all the procedures, the CMNY procedure addresses the qualities essential for evaluation most effectively, with the exception of Expressiveness (i.a). However, as we have mentioned, the qualitative values it provides may be sufficient for early analysis. This procedure satisfies our criteria for Simplicity (xi) and is the most effective at addressing an improvement of model quality. However, this procedure does not

explicitly address many of the beneficial qualities for i^* evaluation and provides only partial automation.

Although qualities vi to ix may be beneficial to evaluation in i^* , they are not essential. In addition, using ideas derived from the other goal model evaluation procedures, an evaluation procedure for the i^* Framework based on the CNYM procedure could potentially be expanded to include these desired qualities.

If we choose the CNYM procedure over the GMNS procedure we make a trade-off of partial Automation (xii) for Allowing Human Intervention (iii) and the effective derivation of an overall evaluation value (i.b). Based on our knowledge concerning the early, high-level analysis of i^* models, involving a unavoidably incomplete depiction of the domain, and our desire to facilitate decisive evaluation, the tradeoff between these elements seems reasonable. Therefore we shall adapt the CNYM procedure for i^* evaluation, returning to the possibility of incorporating other beneficial qualities such as traceability and conflict detection in Chapter 9.

3.4 Conclusion

By exploring existing goal model evaluation procedures in terms of the desired qualities for i^* evaluation, we have chosen the CNYM as a base for the i^* evaluation procedure. In the next Chapter, we will expand this procedure to deal with i^* syntax and to ensure that it continues to effectively address the desired qualities.

Chapter 4: The i^* Evaluation Procedure

4.1 Introduction

In this Chapter we address several of the qualities for i^* evaluation which are essential to facilitate evaluation, such as Element Evaluation (i), Propagation Guidelines (ii), and Human Intervention (iii). As described in Chapter 3, we shall adapt the CNYM procedure for the NFR Framework for the evaluation of i^* models. This requires us to review the procedure and elaborate on aspects of the procedure which were not previously specified, such as the frequency of required human judgment, the meaning of and the choices for initial labels, and the placement of initial labels on non-leaf nodes.

The newly elaborated procedure is then expanded for use with i^* . First we consider the set of evaluation labels and their visual representations, specifically addressing the Element Evaluation (i) quality. Next, we explore the differences between goals in the NFR Framework and the various types of elements in i^* , clarifying the differences between “soft” and “hard” elements and making choices concerning the use of partial evaluation values for “hard” elements in i^* . Our focus then turns to the various links in i^* , defining Propagation Guidelines (ii) for these links beyond the rules provided by CNYM. We consider potentially complex link syntax situations such as using links to other links and having a single element as a recipient to a mixture of link types. We determine the role of actor boundaries in the i^* evaluation procedure, opting to use them as a consideration factor for human intervention. Finally, we address Human Intervention (iii) specifically in an i^* context, describing the areas where human judgment is specifically required and where it can be avoided through the application of automatic cases.

Pseudocode describing the implementation of the evaluation procedure is provided. Issues with this implementation, including insuring the convergence and termination of the algorithm, are described. In order to help the reader understand the mechanism of the evaluation procedure described in this Chapter, we provide a full example of a model evaluation, using a model presented in an earlier Chapter. Finally we

briefly suggest ways in which this procedure can be used and adapted in the various context in which i^* is used.

4.2 The CNYM Method Revisited and Elaborated

We have previously outlined the goal model evaluation procedure included in the NFR Framework, and referred to as CNYM, in Section 3.2.2. We will review the main points of the procedure in this section, and provide additional details necessary for expansion to i^* . In the next section we will go beyond the CNYM procedure and elaborate on aspects that are not well-defined, in order to form a basis for i^* evaluation.

4.2.1 A Review of the CNYM Procedure

The CNYM procedure uses qualitative labels to represent the satisfaction or denial of goals. It distinguishes between full and partial satisfaction and denial, and as well as conflict and unknown. The six labels and their label names are shown in the first two columns of Table 4.1, repeated from Chapter 3.

Table 4.1: CNYM Method Evaluation Labels and Propagation Rules, adapted from (Chung et al., 2000)

	Contribution Type and Parent Result							
Contributing Label	Label Name	Make	Break	Help	Hurt	Some+	Some-	?
√	Satisfied	√	X	W+	W-	W+	W-	U
W+	Partially Satisfied	W+	W-	W+	W-	W+	W-	U
C	Conflict	C	C	C	C	C	C	U
U	Unknown	U	U	U	U	U	U	U
W-	Partially Denied	W-	W+	W-	W+	W-	W+	U
X	Denied	X	W+	W-	W+	W-	W+	U

The evaluation procedure contains propagation rules that dictate the transmission of labels from one node to another via contribution links, these rules are shown in Table

4.1. Here we shall elaborate on the motivations for the rules in this table. We have previously described the meanings of the contribution links on the top of this table, propagating positive or negative, full or partial evidence. When these links are combined with a contributing evaluation label, the results take into account both the value of this label and the meaning of the contribution link. For positive contributions such as *Make*, *Help*, and *Some+*, the polarity of the evidence is retained, with the *Make* link preserving the strength of the evidence and the *Help* and *Some+* links weakening the evidence. For the negative contributions such as *Break*, *Help* and *Some-*, the polarity of the evidence is reversed, with the *Hurt* and *Some-* links weakening this inversed evidence. The *Break* link contributes the non-weakened inverse of the evidence, except in the case of a denied value, with the idea that the denial of *Break* is not quite equivalent to a *Make* link, but instead has a positive contribution which is only partial. Here the *Unknown* link always propagates an unknown value, and a conflict value is always propagated as a conflict value, unless by an *Unknown* link.

In addition, the procedure supports And and Or links, taking the minimum and maximum child label, respectively, using the following ordering:

$$X < U \approx C < \checkmark$$

Here the “ \approx ” symbol indicates that there is not specific ordering relationship between the unknown and conflict values.

Initial labels are manually placed on selected goals to start the evaluation procedure. These goals are typically leaf goals, although initial labels may be placed on intermediate goals in the graph. Leaf goals without initial labels receive a label of unknown. In the first step of the procedure, all labels are propagated from child to parent as intermediate values, using the Table 4.1 rules. In the second step, the labels for each parent goal are collected in a bag, allowing for repeats, and a final label for each parent goal is determined. In the cases described in Table 4.2, this final label can be determined automatically.

Table 4.2: Cases where Final Labels for Parent Goals can be Automatically Determined in Step 2

Case	Resulting Label
1. If the bag contains an unknown label	Unknown
2. If the bag contains a conflict label, without the presence of an unknown label	Conflict
3. If the bag contains full labels which are of both polarities, such as $\{\surd, X\}$	Conflict
4. If the bag has only one label which is not a partial label such as W^+ or W^-	the single label
5. If the parent goal has multiple full labels of the same polarity, such as $\{\surd, \surd, \surd\}$ or $\{X, X\}$	the full label
6. If all labels in the bag are of the same polarity, and a full label exists in the set of labels, such as $\{W^+, \surd, W^+\}$ or $\{X, W^-\}$	the full label

In other cases, the final label needs to be determined with the aid of human judgment based on domain knowledge and the interpretation of the parent and contributing goals. Namely, human judgment is needed when the bag is absent of conflict or unknown labels, and has at most one of \surd or X , and has a potential variety of partial labels.

The primary version of this procedure, described in (Chung et al., pp. 70), recommends against giving a parent goal a final label that is partial (W^+ , W^-). To this end, all partial labels in the label bag are promoted or demoted to \surd , X , C or U labels, requiring human input. At this point, with only full, conflict or unknown labels remaining in the bag, the above rules can be applied to automatically determine a final label.

Steps 1 and 2 are repeated until all labels are propagated and a final label has been decided for all goals. It is important to note that it is the final label resulting from step 2 which is propagated in the next round of steps 1, and not the potentially multiple labels in the goal bag.

In the description of the evaluation procedure the CNYM authors note that it would be possible, and more descriptive, to allow partial final parent labels (Chung et al., pp. 86). If this were the case, in Table 4.2, Case 1 would need to be adjusted, allowing single labels of W^+ , W^- to be the final labels for parent goals. If partial labels are allowed, the human judgment aspect of the procedure changes as well, as the evaluator no

longer needs to promote or demote all partial labels in the bag. If partial labels remain after the step of promotion or demotion, then the evaluator must use his/her judgment to determine a final label by “merging” or blending the labels together using domain specific information. For example, if the bag of labels contained $\{X, W^+, W^+, W^-\}$, the evaluator may decide that the goal which produced the third and fourth labels are of roughly equal importance, and that the first contribution is more important than the second, judging the final value to be W^- or X , depending on how much more important the first contribution is in comparison to the second.

The description of the CNYM procedure does not include a semi-formal or pseudocode description of its algorithm. As we intend to formulate such a description for the i^* evaluation procedure, and as we are using the CNYM procedure as a base for this procedure, it is useful to define such an algorithm here. The pseudocode representing the version of the CNYM algorithm that does not allow for partial final values is shown in Appendix A, Table A.1. The change required to this pseudocode which allows partial final values is described in Appendix A, Table A.2. In the first version we prompt for individual promotion of partially satisfied values, in the second version when the automatic cases do not apply, we show all contributing goal and their labels and ask for a merged value.

4.2.2 An Elaboration of the CNYM Method

Formulating this procedure into a semi-formal algorithm brings to light issues in the algorithm not addressed in the NFR Framework. These issues are discussed in the following subsections. Resolutions are proposed based on experience gained through case studies described in Chapter 7.

4.2.2.1 Requiring Human Judgment Multiple Times for one Goal

As initial evaluation values are propagated throughout the graph, all of the contributing labels for a parent goal will not arrive at the same time. For example, in Figure 4.1, the goal Lower Sales Price receives a contribution from Increase Sales Price, a

leaf goal, and Lower Japanese Interest Rates, which in turn receives a contribution from Japanese Rates Rise. If Increase Sales Price and Japanese Rates Rise are given initial values, during the first iteration of the CNYM algorithm Lower Sales Price will receive a contribution from Increase Sales Price and Lower Japanese Interest Rates will receive a contribution from Japanese Rates Rise, potentially prompting human judgment. It is only during the second iteration of the algorithm that the value from Lower Japanese Interest Rates will arrive at Lower Sales Price. Therefore contributions can, and often do, arrive at different “times”, or different iterations of the algorithm. This phenomenon creates issues concerning human judgment. If the both values arriving at Lower Sales Price require human intervention to resolve, then the evaluator is prompted for judgment twice. Potentially, depending on the structure of the graph, an evaluator may be prompted to judge the value of a single goal many times, potentially annoying the user.

How should the algorithm deal with this? Should it try to wait until all of the children have made a contribution before prompting the user? Should it prompt the user each time a new contribution has arrived and human judgment is needed? How does the algorithm use the previous human judgment when new evidence arrives? In the Table A.2 algorithm we have taken the simplest implementation, re-evaluating the parent value each time a new contribution arrives, with or without human judgment, depending on the contents of the bag. The previous human judgment is discarded, and the cumulative bucket contents are used for a new decision. Although this is the simplest implementation, it is not the most convenient for the user. When implementing this algorithm, expanded for i^* , in Chapter 6, we shall consider ways to reduce the frequency of human input required.

evaluation of goal models, the word alternative can be interpreted in two different ways, the specific alternatives in an *Or* relationship which would satisfied a particular goal, or a higher-level design alternative representing an overall choice of many alternatives.

We return to Figure 4.2 as an example, repeated from Chapter 3. In Figure 4.2 we can see some clear specific alternatives within the model. For example, in order for Accuracy [Received (Summary)] to be satisfied, we could Certify the Summary or Correct the Incoming Information Flow of the Summary by satisfying the appropriate sub-goals. In addition, the alternatives may not be exclusive. Presumably, we could both Certify the Summary and Correct the Incoming Flow of the Summary, and although in this particular model, it would not make a difference to the values of other goals in the model, in other models it might. In many cases, the specific alternative is not represented clearly by an *Or* link, but represented implicitly by the presence of elements which could be marked as satisfied or denied. The element Confirmation [Summary], is a leaf node, and therefore must receive human input if it is to have a value other than unknown. This element is not involved in an alternative in terms of a variety of ways to accomplish something; however, the choice of whether or not to mark this node as satisfied can be seen as a type of specific alternative. We either want to evaluate the effects of confirming the summary or not.

In the same model, we could consider an example of an overall design alternative as the satisficing of Certification By [HighSummary.Manager], the denial of Certification By [LowSummary.Manager], the denial of Confirmation [Summary], and a lack of an evaluation value for Certification [Summary], as is shown in Figure 4.2.

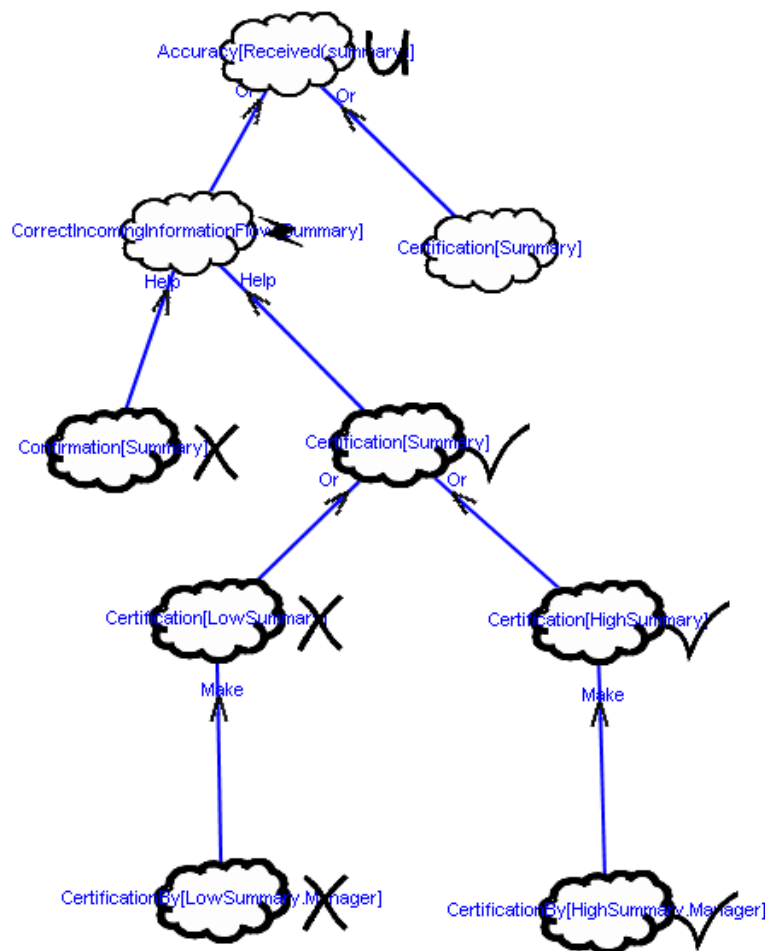


Figure 4.2: An example NFR model reproduced from (Chung et al., 2000)

4.2.2.3 Initial Labels for Non-Leaf Nodes

As well as placing initial values on leaf nodes, a modeler could place such values on intermediate nodes that have contributing children. This sort of input is similar to the case with Certification [Summary] where the modeler wants to indicate in the evaluation that something is or is not satisfied. In this case we may ask whether the contributions that are made to this node after it has been given its initial value are included in the determination of a final label.

By placing a value on a non-leaf element, the modeler may be claiming that an additional contribution, not included in the model, exists, and the effects of this contribution are being evaluated. In this case we recommend that the modeler revise the

model to explicitly include such contributions, improving the quality and comprehension of the model and its evaluation to outside parties. Alternatively, a modeler could be asking a question such as "Even though we do not currently have the means to cause this element to have this value, what if we did have such means?" This type of question may be a valid reason to give initial labels to non-leaf nodes.

In the algorithm in Table A.1, the human input to a non-leaf node is not used directly as the final label for that node. Instead we include the initially placed label as a label in the bag of contributing labels, possibly prompting the modeler for human input. Therefore the child contributions could affect the manually placed label. If a human judgment situation is created, the user can decide whether to apply the results of the initially placed label or the results of the contributing elements.

4.2.2.4 Choosing Alternatives to Evaluate

When the model is simple and contains only one alternative, the meaning of initial labels applied to leaf nodes is relatively clear as it represents the choice of an alternative. The situation becomes more complex when the model becomes larger and contains multiple alternative or alternatives for more than one decision. In these cases, the possible combinations of alternatives grow, with a maximum of x^i combinations, where i is the number of decisions and x is the number of alternatives for each decision. Of course the number of alternatives for each decision is likely to vary, not remaining at a common x for all choices, but this value gives an idea of the large number of possible evaluations that can be performed on a simple model. Furthermore, for each alternative, we could place one of six initial labels, depending on whether we want to evaluate that label as satisfied, denied, partially satisfied or denied, conflicted or unknown.

With the possibly exponential number of evaluation possibilities, how does an evaluator know where to begin? It is feasible that all possible combinations of leaf node values could be automatically used in successive evaluations, in sort of a brute-force bottom-up evaluation. But this kind of technique contrasts with the intention of the evaluation procedure as a tool to help promote domain analysis and discovery. The starting points of an evaluation rely on human judgment, more so than the application of

the procedure itself. The combination of initial evaluation values should correspond to a real life domain analysis question. And the choices of which possible evaluation to perform should be directed by the presence of interesting domain questions. For example, in Figure 4.1, there are 11 leaf nodes and therefore 2^{11} or 2048 possible evaluations, assuming we use only full satisfaction and denial, but of course it is not realistic to perform them all, even for such a small model. Instead, maybe the evaluator is particularly interested in how to increase the profit per vehicle, and therefore will perform an evaluation for each option, with all other leaf nodes marked as satisfied. Or perhaps the evaluator wants to know the effects of Keep labour costs low on Increase customer loyalty, and will therefore perform two evaluations, with this element satisfied and denied and all other elements denied. In a way, the choice of initial values can be seen as an experiment, where initial values that are changed between evaluations are the independent variables, the other initial values are the controlled variables, and the results of the evaluation are the dependent variables.

Although we are discussing initial values in terms of the NFR Framework and CNYM evaluation, these points will likely apply to the version of this procedure adapted for i^* , as we are retaining the structure of initial and resulting evaluation values.

4.3 Expanding and Adapting Rules for i^* Constructs

In this section we expand and adapt the CNYM method to i^* . We must make basic additions corresponding to the elements in i^* which are additional to the NFR Framework. Such elements can be classified under the headings of elements and links. In addition, we will consider the adaptation of NFR labels for i^* evaluation.

We are choosing to adapt the version of the CNYM algorithm that allows partial values as final element labels, as this version provides greater descriptive capability in terms of the amount of evidence available.

The conventions for the i^* evaluation procedure proposed in this section should be considered as guidelines or recommended practices in the evaluation of i^* models. Such guidelines should aid in the promotion of consistency between evaluations performed by different individuals; thereby facilitating model transfer and general

comprehension. However, cases may exist where the modeler or evaluator feels that these guidelines must be broken in order to produce the evaluation value best corresponding to reality. The occurrence of such situations may point to the need for model iteration and expansion in order to make the concepts within the model clearer. There may be times however, when this expansion is inconvenient due to space constraints. We will explore the issues concerning model changes prompted by evaluation more thoroughly in Chapter 5. At this point it is sufficient for the reader to keep in mind that the specifics of the evaluation procedure are presented as recommended guidelines as opposed to rules meant to be strictly followed in all cases.

4.3.1 Labels

In the previous applications of an i^* evaluation procedure based on the CNYM procedure, the graphical notation, (visual appearance), of the labels were modified. The primary modification involved the replacement of the W^+ and W^- labels with a checkmark and cross, both with a dot underneath. In the determining the appearance of evaluation labels, we can see several reasons to adopt these adapted labels. First, these labels are "bolder", consisting of thicker lines, making them easier to pick out in models. Secondly, since the time that the above work has been published, users of an i^* evaluation procedure (even though one has not been formally defined) have become accustomed to these labels. Thirdly, the software in which we intend to implement the i^* evaluation procedure, OpenOME, already makes use of these new labels, so for implementation simplicity, we will use them as well. Lastly, the new symbols used for partially satisfied and denied look similar to the values of satisfied and denied, making it easier to identify that the labels are of the same polarity, and making the meaning of such labels more obvious. Table 4.3 compares CNYM evaluation labels to the evaluation labels we will use in the i^* evaluation procedure.

Table 4.3: Labels for CNYM and i* Evaluation

Label Name/Description	Satisfied	Partially Satisfied	Conflict	Unknown	Partially Denied	Denied
CNYM Label	✓	W ⁺	⚡	U	W ⁻	X
i* Label	✓	✓.	⚡	?	✗	X

4.3.1.1 The Propagation of Unknown vs. Unlabelled

When considering the propagation of evaluation values through the various types of i* links, we are led to question whether a node without a label shall be treated as having an unknown label. One can discern a semantic difference between the two, with the explicit placing of an unknown label indicating that evidence is lacking, or that a decision has been deferred. On the other hand an unlabelled element may not be explicitly unknown having not been explicitly addressed by the evaluator. This is especially likely if the model is very large. A lack of evaluation value can be used to indicate that this node does not have an effect in the current evaluation, and therefore should be ignored. This will effectively allow the user to evaluate a subset of the model, choosing to exclude the effects of some elements.

Taking these considerations into account, we resolve this issue by avoiding the explicit propagation of unknown values unless they are involved in an And or Or relationships, where their value maybe be the result of this relationship. In this way, elements may be marked as having no effect unless their status is explicitly required.

4.3.2 Elements

The NFR Framework, for which the CNYM procedure was intended, contains only goals that can be considered "soft", meaning that their satisfaction is not precisely determinable, motivating the notion of satisfied and the inclusion of partial satisfaction and denial values. In other types of goal models, such as in the KAOS Framework, goals are "hard", in that their satisfaction can always be precisely determined. The i* framework includes both types of goals, as well as tasks, resources, and beliefs. For our

purposes, we will refer (nonsoft) goals, tasks and resources as hard elements. This designation has commonly been considered to mean that these elements they are either performed/furnished or not. However, the mixing of hard and soft elements in the i^* Framework forces us to reconsider the precise desired meanings of such elements. Specifically, can the satisficing of a softgoal result in the satisfaction of a hard element. And similarly, if we propagate partial values, what does this mean when these values are passed on to "hard" elements? Do we allow such propagation?

As it turns out, if the rules of i^* syntax are followed precisely, the only way this issue arises is when a softgoal is made to be a decomposition element of a task, or when a hard element depends on a softgoal. These situations are depicted in Figure 4.3 and Figure 4.4. However, recommending the avoidance of such situations is not an optimal solution. There are cases where a modeler may wish to depict a situation where a hard element is dependent upon, or requires the satisfaction of, a soft element. For example, in an excerpt from a Trusted Computing case study shown in Figure 4.5 and to be explored in Chapter 7, we can see that it is necessary for technology to be Desirable and Trustworthy in order to Purchase Technology. Desirability and trust are best represented as softgoals, and that the purchase of technology can be accurately represented by task, yet the softgoals are necessary in order to satisfy the hard task. In this case satisficing of softgoals is sufficient for the satisfaction of a hard element, because if Trust and Desirability are fully satisficed, technology will be purchased. But what if Trust or Desirability are only partially satisficed or denied, how will this affect Purchase Technology?

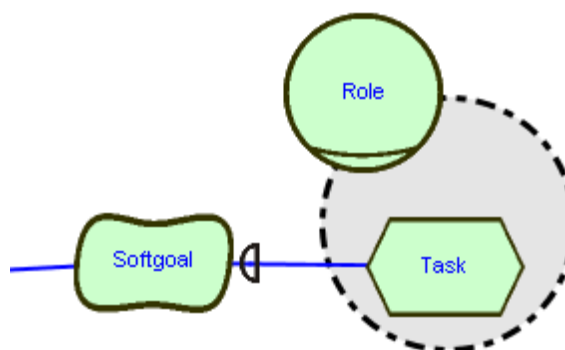


Figure 4.3: Mix of Hard and Soft Elements

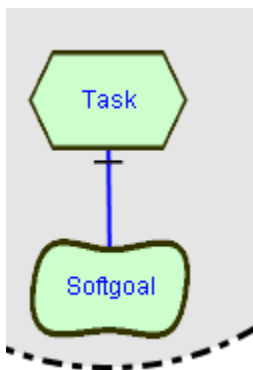


Figure 4.4: Mix of Hard and Soft Elements

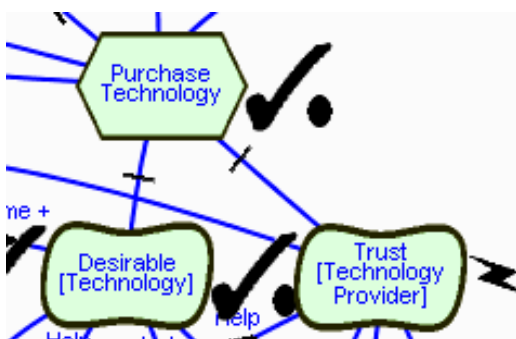


Figure 4.5: Excerpt from Trusted Computing Case Study

The resolution of this issue can take two forms, one "strict", and the other more "loose". In the strict resolution, hard elements are treated strictly as hard, only allowing full values of satisfaction or denial. In this case satisficing is sufficient to cause satisfaction, but partial values may not be sufficient. In the "strict" resolution, similar to the second step in the first version of the CNYM procedure, before a value of partially satisfied or denied is propagated it must be rounded either up, or down, based on human judgment, and resulting in a value of satisfied, denied, conflict or unknown. In this way only hard elements ever receive full labels.

In the "looser" resolution, hard elements such as hard goals, tasks and resources are allowed to take on partial values. At first, this convention may seem contradictory to the definition of such elements. If a task represents a physical action, what does it mean if this action is partially satisfied or denied? Likewise with a goal or resource, how do partially accomplish something that is meant to be binary? However, if you consider the hard element as encompassing the qualities represented by the soft elements, then partial labels are sensible. We can see an example of this in Figure 4.5. Having a partially

satisfied value for Purchase Technology may be sensible to indicate that only some technology was purchased, due to partial values in Trust or Desirability, but not enough technology was purchased to judge the task fully satisfied. Alternatively, the partially satisfied value may represent the fact that some of the instances of this actor purchased technology, enough that this task is partially satisfied, but some instance did not, keeping the task from being fully satisfied. Another example appears in the excerpt from the Montreux Jazz Festival Case Study shown in Figure 4.6. Although the task is Perform, as in perform a concert; the intention is not just to put on a performance, but to also have a Good Performance, represented by the softgoal in the decomposition. As Although it is possible to Perform without it being a Good Performance, the inclusion of Good Performance as a sub-element of Perform indicates that Good Performance is part of what is meant by Perform, and therefore if the performance is not good, Perform should not be satisfied. In this case it is sensible for this “hard” task to take on a partially satisfied value, meaning the performances are not sufficiently positive to be fully satisfied. Likewise, partially negative means that there was partial evidence to suggest the performance was not good, or was deficient in some other way, represented by another one of Perform’s sub-elements. If the Perform task did not decompose to the Good Performance softgoal, or any other softgoal decomposition, then it would likely be referring to the binary act of the performance occurring or not.

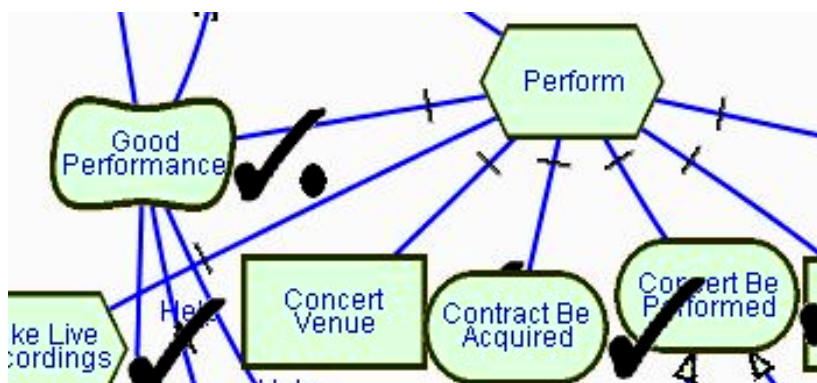


Figure 4.6: Excerpt from Montreux Jazz Festival Case Study

In general, allowing partial values for hard elements provides a greater degree of expressiveness in the evaluation procedure, as we would like to know about and propagate contributions, even if they are weak. In choosing an interpretation for the i^*

evaluation procedure, we shall choose the "looser" interpretation in order to provide the greatest flexibility in evaluation results. If a user wants to use only the "strict" interpretation of hard elements, they could either promote or demote all partial values before they arrive at hard elements, or they could avoid the *i** syntax shown in Figure 4.3 and Figure 4.4 that makes this issue relevant. For example, if the user intends for Perform to only represent the binary physical act, then they should not make a softgoal such as Good Performance a decomposition element of this task. If this sort of convention is adopted, the resulting model may contain two decomposition structures or *trees* of elements. One tree would contain purely functional elements such as Perform and its functional sub-components; whereas the other tree would contain non-functional aspects of the performance such as Good Performance, as well as other, more specific non-functional decomposition elements of Good Performance. An example of this type of model structure for a Office Support System is shown in Figure 4.7, originally presented in (Mylopoulos, Chung, & Yu, 1999). Here the functional goals are the dark ovals, and the non-functional softgoals are the cloud-like ovals.

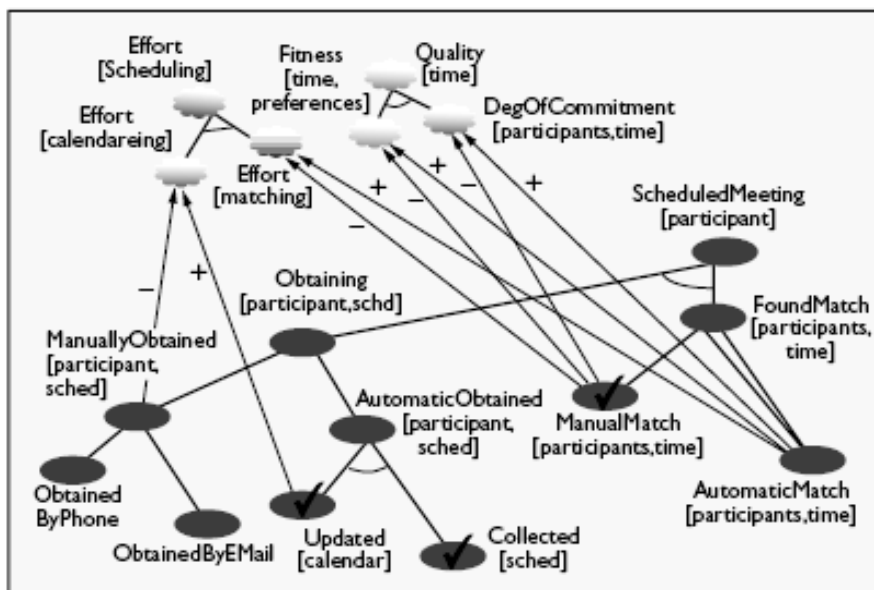


Figure 4.7 NFR Model of an Office Support System Showing Functional and Non-functional Hierarchy from (Mylopoulos, Chung, & Yu, 1999)

Furthermore, implementing the "strict" interpretation would increase the amount of human intervention required for automatic evaluation, as partial labels would have to be manually promoted or demoted. If there is enough demand for an implementation of

the "stricter" interpretation, it may be possible to add this in as a global option to the i* evaluation procedure.

4.3.2.1 Beliefs

As well as softgoals, goals, tasks, and resources, the i* Framework contains the notion of a belief, represented as an element which may contribute to softgoals. These elements are similar to claims in the NFR Framework. The nature of this element leads one to treat it as a “soft” element, naturally possessing partial evaluation values such as partially satisfied. However, if partial final values are allowed, making the distinction between hard and soft is not necessary. As with claims in the NFR Framework, beliefs in i* can receive contributions from other beliefs; however, it is not specified whether these elements can receive contributions from other elements. However, experience with i* case studies have shown that occasionally beliefs, like softgoals, do receive contributions from other elements. If this were to occur in a model, for evaluation purposes, this element should be treated similarly to a softgoal, with its value decided via human judgment if necessary.

4.3.3 Links

In order to adapt the CNYM evaluation procedure for the i* Framework, we must explore the treatment of various link types between elements in the evaluation procedure, including contribution links, correlation links, dependency links, decomposition links, and means-ends links. In addition, we provide conventions to deal with links whose target is another link, and the evaluation of elements who are the recipients of more than one link type.

4.3.3.1 Contribution Links

In examining the contribution links in the NFR and i* Frameworks, we see that both frameworks consist of Make, Help, Some+, Unknown, Some-, Hurt, Break, And, and Or links. These links have equivalent meanings between frameworks. As a result,

in formulating the propagation rules for links in the i^* evaluation procedure, we can adopt the CNYM rules in Table 4.1 without changes.

4.3.3.2 Correlation Links

Although correlation links in i^* are meant to represent side effects or unintentional effects, these effects are not necessarily weaker, stronger, or otherwise different than regular contribution links. For the purpose of the i^* evaluation procedure, keeping in mind simplicity, the propagation effects of correlation links will be the same as contribution links, described in the above section.

4.3.3.3 Dependency Links

The i^* Framework introduces the notion of dependencies between actors, consisting of a depending element, the depender; the element depended upon, the dependum, and the element which fulfills the dependency, the dependee. When expanding the CNYM evaluation procedure to i^* , we must consider how evaluation values should be propagated through dependencies. The nature of a dependency seems to indicate that if the thing in which you are depending on is satisfied then your element will also be satisfied, with the same logic applying for other evaluation values. This reasoning points towards a direct transfer of the evaluation value from dependee to dependum to depender. See Figure 4.8 for a graphical example.

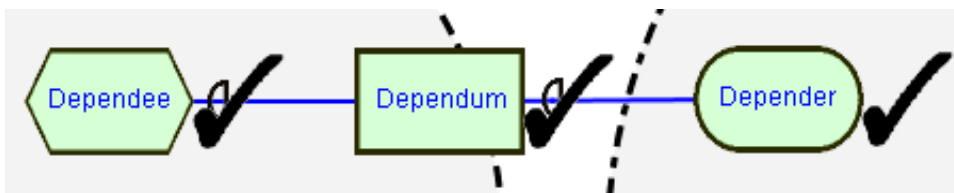


Figure 4.8: Example of Propagation through Dependency Links

4.3.3.4 Decomposition Links

In i^* , decomposition links are introduced to represent the elements necessary to accomplish a task. The syntax is meant to represent the need for all elements in order for the task to be achievable. This leads us towards the treatment of decomposition links as an And relationship in propagation of evaluation values. As in the CNYM method, And is used to indicate the selection of the "minimum" value amongst the values of all of the contribution elements in the And relation. Therefore we must consider the min/max ordering of evaluation values in the i^* evaluation procedure. The CNYM procedure uses the following ordering:

$$X < U \approx C < \checkmark$$

We can expand this order to allow final partial values:

$$X < W^- < U \approx C < W^+ < \checkmark$$

We need to consider whether this ordering is appropriate for the i^* evaluation procedure, based on our experience in i^* application. The order involving negative and positive values is sensible given the meaning of these labels in terms of full and partial positive and negative evidence ($X < W^- < W^+ < \checkmark$). In considering the placement of the unknown and conflict values, a conflict involving both negative and positive values would be better or "greater" than both full and partial negative evidence. In addition, such a conflict would be less desirable or "less" than both full and partial positive values, giving us $X < W^- < C < W^+ < \checkmark$. Where should the unknown value fit into this ordering? It is more beneficial in an And relationship to know that the parent element is denied than to not know the satisfaction level of the element, therefore when choosing a minimum:

$$X < W^- < U.$$

The ordering defined in Section 4.2.1 does not provide guidance as to which value is chosen over the other when resolving decomposition links. In this case, as it is more informative to know that a node has conflicting evidence than to know that part of the evidence is incomplete, we shall chose to propagate conflict over unknown. Our analysis results in an overall ordering for i^* evaluation as follows:

$$X < \text{✗} < ? < \text{✗} < \checkmark \bullet < \checkmark$$

The above ordering shall also be used for And and Or contribution links.

An example of the propagation for decomposition links is shown in Figure 4.9.

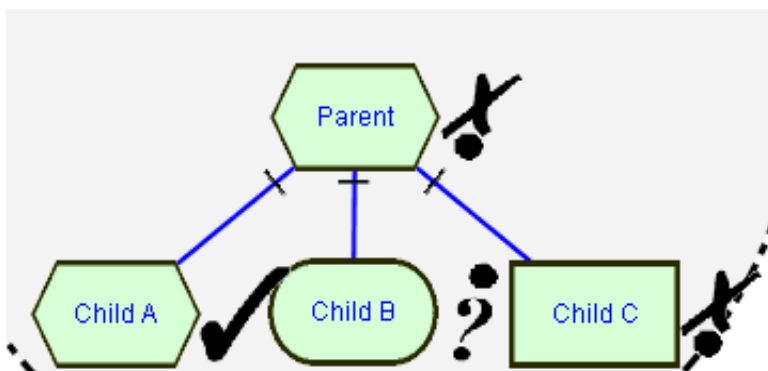


Figure 4.9: Example of Propagation through Decomposition Links

4.3.3.5 Means-ends links

Like the i^* decomposition links, the means-ends links have an underlying meaning which leads us naturally towards a method of evaluation value propagation. The means-ends relationship is meant to depict the alternative tasks which are able to satisfy a goal, and this depiction of alternatives indicates an Or relationship in evaluation. Specifically, this relationship will take the "maximum" value of its children, using the same ordering as decomposition links, described in the above section. An example of Means-ends propagation is shown in Figure 4.10.

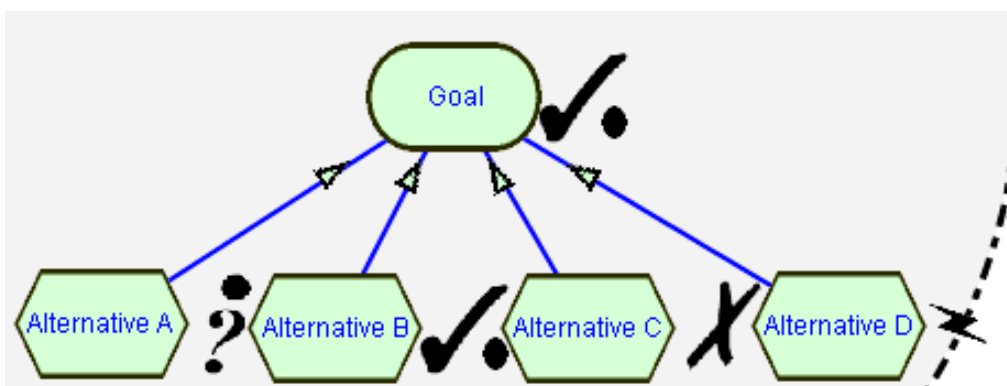


Figure 4.10: Example of Propagation through Means-Ends Links

4.3.3.6 Links to Other Links

As mentioned in Chapter 2, it sometimes seems necessary in i^* to draw links to other links, representing the effects of an element on the relationship between other elements. In the description of the CNYM procedure, evaluation values are given to links as well as goals, although this convention is not explored in great detail. Should the i^* evaluation procedure adopt this convention, perhaps as a way to mitigate the need for links to other links?

4.3.3.6.1 *The Need for Links to Other Links*

When considering this question, we can see that both methods could be used to produce the same evaluation results. For example, in the left hand side of Figure 4.11, the hurt link to the help link mitigates the effect of the help link, making the softgoal value unknown. In the right hand side the same effect is accomplished by declaring that the link is partially denied, mitigating its positive effects. However, we can see that the right hand side holds a semantic deficiency when compared to the left. We know that the help link is partially denied, but we don't know why, or at least the intention is not explicitly included in the model. In the left hand side we know the link is hurt because of the satisfaction of the goal. On the right we are unaware of any relationship between the goal and the other elements. The additional intention information included in links to other links demonstrates the continued need for such links. In addition, allowing links to other links may be simpler than placing evaluation values directly onto links. The meaning of values placed on links can always be expressed explicitly by redrawing the link. For example, in Figure 4.11, partially denying the help link is approximately equivalent to not having the link at all. Partially denying a make link may be equivalent to having a help or some+ link. The only thing we lose by removing the capability to assign values to links is the ability to evaluate varied link strengths without redrawing the links.

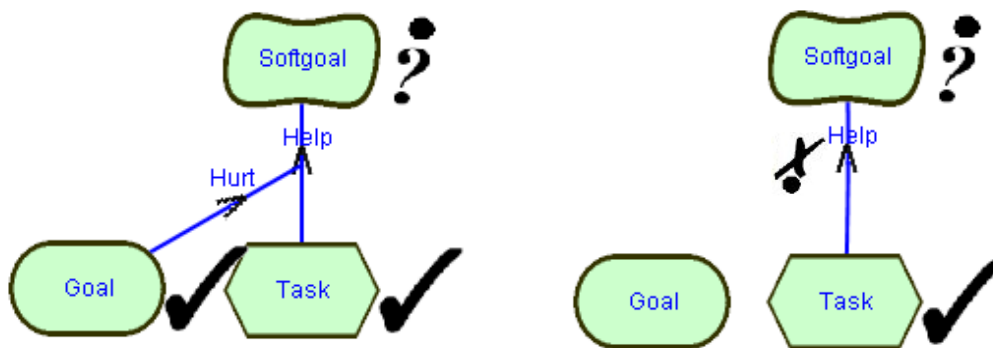


Figure 4.11: Links to Links vs. Evaluation Values for Links

4.3.3.6.2 Evaluation for Links to Other Links

Now that we have shown the need for links to other links, it is necessary for us to precisely define the affects of these links on the propagation of evaluation values. What does it mean to *hurt*, *break*, *help* or *make* a link? The meaning of hurt and break seems somewhat intuitive; the strength of the recipient link is reduced, either fully, or partially. So a make link may be reduced to help or to no effect, and a help link may be reduced to no effect. However, it does not seem intuitive for negative affects on links to be strong enough to inverse the effects of such links, turning *help* or *make* links into *hurt* or *break* links. What if a link is helped or made, should the link become stronger? Is it ever necessary to represent a case where the satisfaction of an element positively increases the strength of a relationship between two other elements, promoting a hurt or help link to break or make? Or can this type of relationship be better modeled by other means? If an element makes the effects of an element on another element stronger, than perhaps this is just a direct effect on the other element. In Figure 4.12 we show an example of an element making a link on the left, where the intention is to promote the help link to make, and an alternative representation on the left, where the make link is demoted to a help and contributes directly to the parent. Although these two cases are not exactly semantically equal, they are semantically similar enough that we will assume that a link to another link will not be used to increase the strength of another link, as an alternative and simpler notation exists.

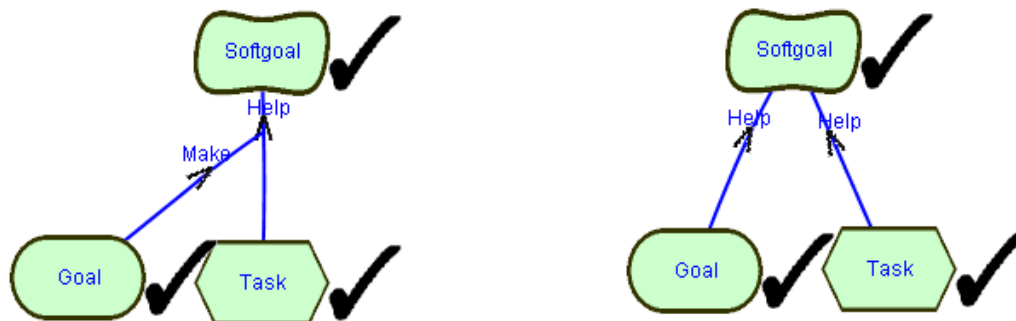


Figure 4.12: Example of Potential Link Promotion

In Figure 4.13 we can see that the demotion of the strength of a link cannot be represented via the same conversion as in Figure 4.12. Therefore the convention of using links to links to make links weaker should be retained. Of course the situation in Figure 4.10 could become a case of link demotion, and likewise Figure 4.13 could become a case of potential link promotion, if the Goals were denied instead of satisfied. However, it can be seen in these cases that the clusters on the right will always show similar evaluation results when promotion occurs, in other words, if the goal in Figure 4.13 was denied, the Softgoal would be either satisfied or partially satisfied on both sides of the figure. Again this demonstrates that the structure on the right of the figures is closely equivalent to link promotion, whether it comes from the satisfaction of a positive contribution or the denial of a negative contribution.



Figure 4.13: Example of Link Demotion

Although we have shown that links to other links may be necessary for link demotion, because of the potential confusion in such links it is recommended that they be used only if necessary. If an alternative method of modeling with the same effect can be found, it should be used instead.

Now that we have determined the nature of links to other links, we can formally describe the recommended effects of links on other links. In Table 4.4, the leftmost column describes the contributing labels when the element label and contribution link type of the element contributing to a label is derived. This derivation is done by using the typical Table 4.1 rules. The origin of the contributing labels is shown in Figure 4.14 by the area enclosed in the red circle. The last column of the table describes the effects of this value on the first contribution link, shown in Figure 4.14, given its original label type in the middle column. In demoting the strength of links, we have chosen to partially demote links to the next strongest link for partial negative contributions, and for full negative contributions, the effects of the link are eliminated. As is apparent in the table, links to other links can be considered to have an asymmetric effect, as the effects of positive contributions and negative contributions are not opposites of each other, as is the norm for i^* links.

Table 4.4: Propagation Rules for Links to Links

Contributing Label	First Link Contribution	Actual Contribution of First Link
✓	Make	No Change
	Some+/Help	
	Some-/Hurt	
	Break	
✓.	Make	No Change
	Some+	
	Help	
	Hurt	
	Some-	
	Break	
?	Anything	Unknown
↗	Anything	Conflict
⚡	Make	Some+
	Some+	Help
	Help/Hurt	None
	Some-	Hurt
	Break	Some-
✗	Anything	None
Anything	Unknown	Unknown

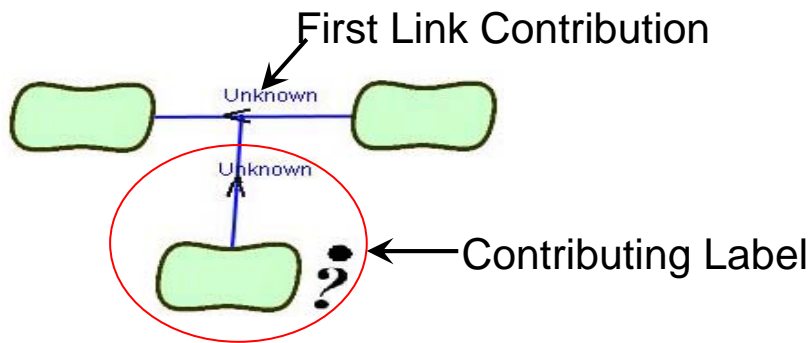


Figure 4.14: Description of Columns in Table 4.4

4.3.3.7 Contributions from a Mixture of Link Types

So far, by exploring the variety of links in i^* , we have examined the cases where a node receives input from a single type of link, either a dependency, decomposition, means-ends, or potentially many contribution links. However, it is common in i^* to see a single element involved in more than one type of link relationship. Contribution links can have the same target as any other type of link, and dependencies, means-ends or decomposition links often have the same target. Figure 4.15 depicts common cases of mixing links. Figure 4.16 shows an example in the context of the Montreux Jazz Festival.

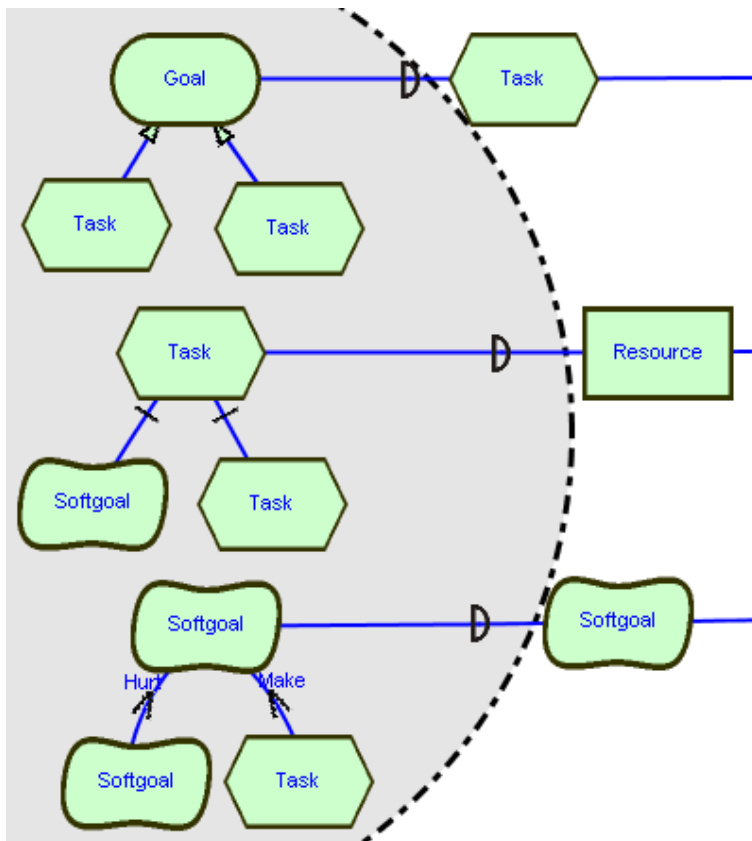


Figure 4.15: Common Cases of Mixing Links

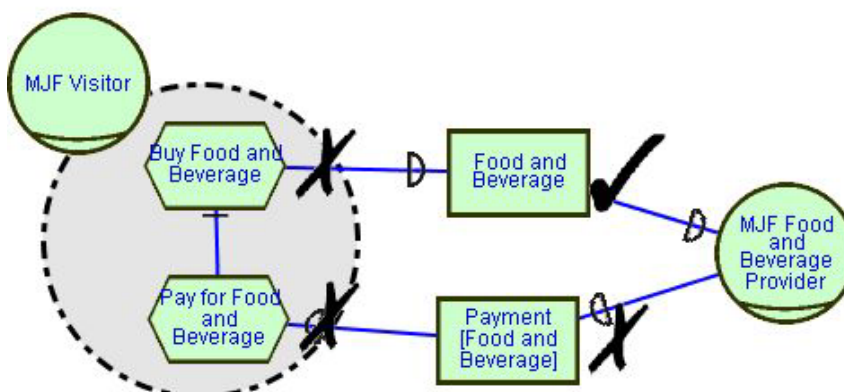


Figure 4.16: Example from the Montreux Jazz Festival of a Mixture of Links

The use of more than one type of link for a node requires us to define the effects of these combinations on the evaluation procedure. When dependency links are mixed with means-ends or decomposition links, the nature of such links seems to indicate that the results of each individual link type should be combined with an And relationship.

This is interpreting the syntax as indicating that as well as the max/min of the Or/And relationship in the means-ends/decomposition link; the parent element also requires another element to be satisfied. These situations are depicted in the top two clusters of Figure 4.15. In the case of mixing contribution links and dependency links, shown in the bottom cluster of Figure 4.15, the nature of softgoals makes them more appropriate for qualitative contributions as opposed to binary relationships. Therefore, it is recommended that the dependency is treated as an additional contribution, such as would be made by a *make* link. However, as this convention may not be intuitive to all *i** users, it is recommended that this notation be avoided if possible. Namely, if it is the modelers intention that the dependency should be treated as an "And" relationship with the other contributions, this can be modeled explicitly by adding additional elements, as shown in Figure 4.17.

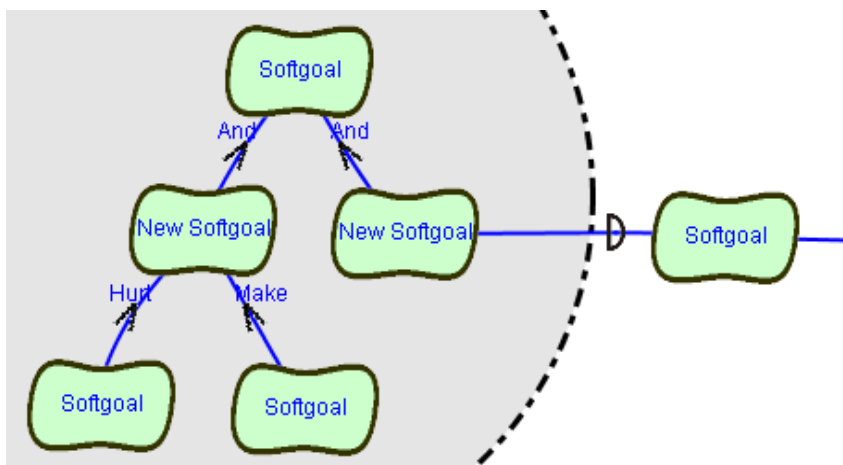


Figure 4.17: A Rearrangement of the Bottom Cluster of Figure 4.13 to Avoid Mixing Links

Fortunately, if one follows *i** syntax, complex situations such as a mixture of means-ends and decomposition links or a mixture of contributions, dependencies, means-ends or decompositions would not appear.

4.3.4 Actor boundaries

When adapting a goal model evaluation procedure to *i**, we are led to consider the effect of actor boundaries on propagation. If the standard *i** syntax is followed,

propagation through actor boundaries should only occur through dependency links. In our consideration of such links in Section 4.3.3.3 we have indicated that evaluation values are propagated from dependee to depender as is. However, it is possible that passing through the boundary representing the perception of one actor through to another boundary may have an effect on this value. This is especially interesting to consider in the deviation of i^* syntax where "internal" links are used outside of actor boundaries. A common example of this is the use of contribution links across boundaries to indicate attack and defense scenarios, such as we will see in the Trusted Computing Case Study in Chapter 7. According to the rules we have defined so far, such links are treated the same as they would be if they were inside of a single actor.

Keeping in mind our desirable quality of simplicity in an evaluation procedure, we shall retain the convention of effectively ignoring actor boundaries in label propagation in this version of the i^* evaluation procedure. That is not to say that such boundaries are irrelevant to the evaluation procedure. In the analysis of evaluation results and in the application of human judgment, the presence of such boundaries and general location of elements should have a major influence on human decisions. However, to the automatic component of the procedure, these boundaries currently have no meaning. We leave the further exploration of this topic to future work.

4.3.5 Human Intervention

The application of human judgment to evaluation has been discussed extensively in our review of the CNYM procedure and our discussion of initial values. In the i^* evaluation procedure human judgment is used in much the same way, in deciding on a final label for parent nodes when propagation rules don't clearly apply, and in deciding the set of initial values based on strategic domain questions. When analyzing the use of such judgment in the CNYM procedure to determine a parent label in Step 2 of the procedure, we believe it is preferable to relax some of the automatic cases shown in Table 4.2. Many of the rules in this table express the CNYM procedure's method of taking the minimal element from the bag of contributing labels, with the following ordering:

Conflict \leq Unknown \leq Denied \approx Satisfied

In other words, if a conflict is present, it will always be the final label, and if an unknown value is present without a conflict, it will also always be the final label. In the i^* evaluation procedure, we suggest leaving this choice to human judgment. Our motivation for this is twofold; first, favouring the propagation of unknown over satisfied or denied values is less informative for the evaluator. Although the final value could change if the unknown value becomes known, it may also be the result of an unlabelled leaf node, and therefore will remain unknown. If many leaf nodes are left unlabelled, and they are involved in And/Or relationships (as described in Section 4.3.1.1), the propagation may see a proliferation of unknown values. The evaluation procedure is more informative if the evidence that is present is propagated. Secondly, in situations where a conflict is propagated, there may be enough positive or negative evidence in the bag to reasonably choose a positive or negative value. This is especially true when there are many contributions to a node. For example, if the bag contained $\{\checkmark, \checkmark, \checkmark, \times, \checkmark, \checkmark\}$, a single value of conflict should not automatically indicate a final value of conflict. The evaluator should be able to choose full or partial satisfaction.

In addition, the presence of a fully satisfied and fully denied value should not automatically indicate a conflict, as the presence of multiple positive or negative labels may make a non-conflict value more appropriate. For example, in a label bag of $\{\checkmark, \checkmark, \checkmark, \checkmark, \checkmark, \times\}$, an evaluator may decide that full or partial satisfaction is more appropriate than a conflict. As a result of these changes, we are left with fewer cases where final labels can be determined automatically, the cases left over from the CNYM procedure are described in Cases 1, 2, and 3 in Table 4.5.

In order for the evaluation procedure to be convenient for the modeler, the procedure should only ask for human judgment when necessary. In a few cases, when human judgment has already been applied, upon the arrival of new evidence we can use the previous result to determine the next result automatically. These cases occur when values in the element bag have been promoted, and future evidence is of the same polarity as the original evidence. For example, if the user decides that a bag of $\{\checkmark, \checkmark, \checkmark\}$ will result in \checkmark , future contributions of \checkmark or \checkmark will not change this judgment. The same logic holds for the opposite polarity. As a result of this, we can add a new case to

Table 4.5. We will consider the issue of reducing the amount of human judgment needed again in Chapter 6, when discussing the possibility of delaying human judgment until all the evidence for a parent node has been collected.

Table 4.5: Cases where Final Labels for Parent Goals can be Automatically Determined in Step 2

Case	Resulting Label
1. If the bag has only one label	the single label
2. If the parent goal has multiple full labels of the same polarity, and no other labels, such as {✓, ✓, ✓} or {X, X}	the full label
3. If all labels in the bag are of the same polarity, and a full label exists in the set of labels, such as {✓, ✓, ✓} or {X, X}	the full label
4. If the previous human judgment produced ✓ or X, and a new contribution is of the same polarity	the full label

Our discussion of human judgment to this point has focused on the cases where human judgment is explicitly required. In other cases, it may be used to justifiably adjust the guidelines described in the evaluation procedure thus far in order to produce a value that the evaluator feels better reflects reality. As mentioned, this likely indicates the need for model expansion, as to be described further in Chapter 5.

Although we have stressed the importance of allowing human judgment in the i^* evaluation algorithm, the use of such judgment in model evaluation may imply that the evaluator possesses knowledge which is not yet captured in the model. This potentially indicates that the model may be less transferable or understandable by parties other than the modeler, as specific domain knowledge is needed to understand choices in the evaluation. In this sense the evaluation procedure may help to indicate the areas in the model that could be expanded or clarified. This subject will be returned to in Chapter 5 when exploring the benefits and effects of the evaluation procedure.

4.4 The i^* Evaluation Algorithm

The i^* evaluation algorithm will account for the expanded flexibility in human judgment as described in Section 4.3.5. For the sake of simplicity, it will assume that the

input graph follows correct i^* syntax as described in Chapter 2. The algorithm will be based on the CNYM algorithm, described in Appendix A, Table A.2 and Table A.2. Our changes to the CNYM algorithm account for the expanded i^* syntax described in this chapter. Such expansion raises minor issues in the implementation. For instance, the addition of "hard" elements leads us to consider how such elements will be treated in the algorithm. Although "hard" elements are permitted to have partial values, we have decided that these elements will not make use of an element bag, and therefore human judgment should never be required to determine the value for a "hard" element. In addition, the inclusion of hard elements raises issues when initial values are placed on non-leaf hard elements. Should the final value for the element be the initial value, or the result of the means-ends or decomposition relationship? As previously explored in Section 4.2.1, a modeler may want to ask a question such as "Even though we do not currently have the means to cause this element to have this value, what if we did have such means?" In order to allow the evaluation of such questions, we will retain them in the evaluation of "hard" elements, even if the graph structure results in a different value. The treatment of initial values for non-leaf soft elements remains the same as in the CNYM algorithm. They are treated as a contribution and placed in the element's label bag. The pseudocode for the i^* evaluation algorithm appears in Table A.3 in Appendix A.

The implementation of the evaluation algorithm in the software tool OpenOME, will be described in Chapter 6. In Chapter 8 we will explore the expansion of this algorithm to adopt various potentially useful conventions inspired by the goal evaluation procedures in Chapter 3, such as traceability and quantitative propagation.

4.4.1 Convergence

i^* models differ from traditional tree-like graphs in that they often consist of loops or circular contributions. When dealing with loops the issue of convergence arises. How do we know that the evaluation algorithm does not continually fluctuate between values, causing an infinite loop? For example, in Figure 4.18, one could imagine that a negative value could be propagated from Softgoal A to B, then a negative value would be

propagated back to A, causing a positive value to propagate back to B, causing a negative value to be propagated to A, and so on, in an infinite loop. Fortunately, in the way we have set up the evaluation procedure a loop such as this always starts with an initial or contribution value, and this value is remembered by the algorithm and applied to the determination of the final label each time this label is determined. In other words, the contents of the contributing label bag for a softgoal are not removed after a human decision has been made; they are available in each subsequent judgment. Values in the label bag are only removed when they are replaced by a value from the same element. As a result of this, situations where a loop is infinite always involve repetitive calls for human judgment. This allows the user to detect the presence of an infinite loop and choose a value which results in convergence. Likely a situation such as this will prompt modification to the model.

For example, the loop in Figure 4.18 could be started with a fully satisfied value for Softgoal B, and this would propagate partially satisfied to A, which would propagate partially denied to B. Then, in softgoal B the label bag contains the initial value of fully satisfied and the partially denied value from A, so human judgment would be prompted, likely resulting in partially satisfied or a conflict. If a conflict is the result, both softgoals would converge to a conflict. If partially satisfied is the result, the value would come around to B again with a partially denied value. If the user makes the same judgment as previously, a value of partially satisfied would again be propagated to A. In this case, assuming the human decision does not change, the values have converged to partially satisfied.

If, on the other hand, when prompted for human judgment for softgoal B with a label bag of fully satisfied and partially denied from A, the user chooses partially denied, the loop will not converge. In this case partially denied will be propagated to A, which will propagate partially satisfied back to B. At that point the label bag for B will be satisfied (the initial value), and partially satisfied, which shall be automatically resolved to satisfied by the automatic cases in Table 4.4. Partially satisfied will be propagated to A and the partially denied will be propagated back to B. At this point human judgment is requested again with the same label bag as in the previous prompt for B: {✓, ✗}. If the human judgment is the same (✗) this loop will occur again, prompting

with the same label bag. Therefore we rely on the human to detect the presence of the loop via the presence of repeated prompts for the same human judgment and either choose an evaluation value which causes the loop to converge (such as conflict or partially satisfied), or stop the evaluation procedure altogether. We explore this situation further in Chapter 6, Section 6.3.

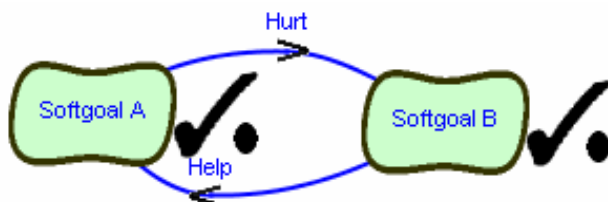


Figure 4.18: A Simple Example of a Loop with Inversing Links

The instances where the propagation value is not inverted, such as in Figure 4.19 are less interesting, as their convergence is more obvious.

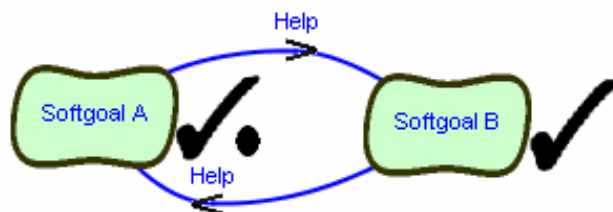


Figure 4.19: A Simple Example of a Loop without Inversing Links

4.4.2 Termination

In order for the i^* evaluation algorithm to be viable, it must be guaranteed to terminate. Part of this guarantee involves convergence, addressed in the previous section, as the algorithm will not terminate if it does not converge to a set of values. However, even if the algorithm does converge, it may not terminate. It is possible that the procedure will continue to place labels in the queue of labels to propagate indefinitely. In order to ensure that propagation terminates, we will prevent an element from propagating the same label as it propagated previously. For example, if Softgoal B in Figure 4.19 has propagated partially satisfied the last time it propagated a label, then if the next propagation calls for it to do so again, this value will not be placed in the queue for propagation. This, along with convergence, ensures that the queue of labels to

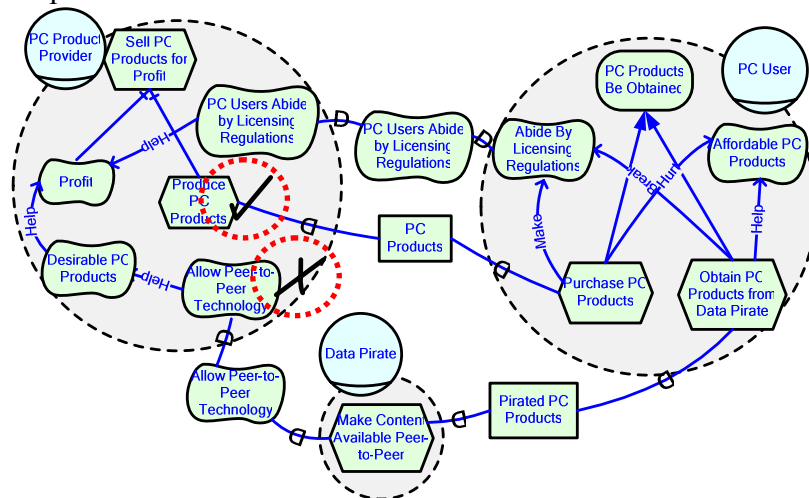
propagate will eventually become empty. The i^* evaluation algorithm in Appendix A, Table A.3 has been adjusted to include this check.

4.5 Evaluation Example

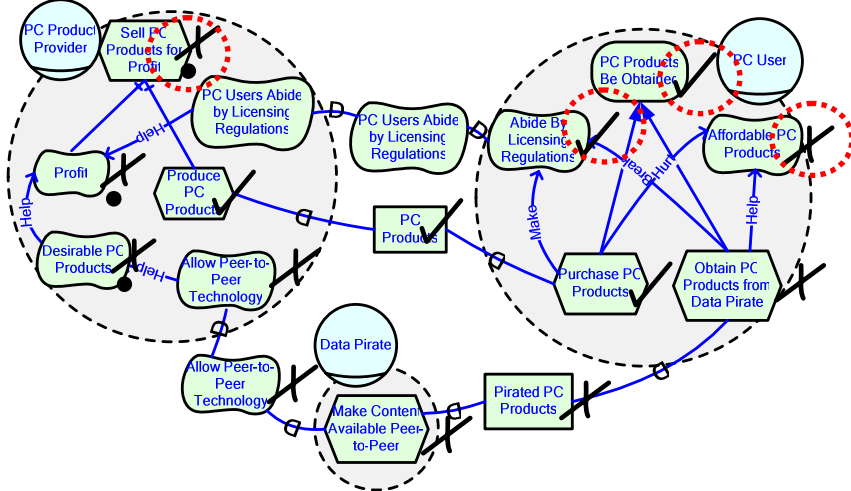
In order to better illustrate the evaluation procedure, we will provide a detailed example of its application on the example model from Chapter 2 concerning Trusted Computing. In Section 2.3 we asked: “If the PC Product Provider decides to not Allow Peer-to-Peer Technology, what effect will this have on Sell PC Products for Profit?” Here we shall attempt to answer this question via the application of the i^* evaluation procedure.

In the first stage, we identify the initial values which correspond to our question. In this case the softgoal Allow Peer-to-Peer Technology is denied. We can see that Purchase PC Products is a leaf goal, and will likely need an initial value, but we hold out on placing this value until we see whether or not it is possible to Obtain PC Products from the Data Pirate. We mark the leaf task Produce PC Products as satisfied. The propagation of the initial values through the model is shown in Figure 4.20, with the new labels in each figure circled in red. The analysis capabilities provided by these results will be explored in the next section.

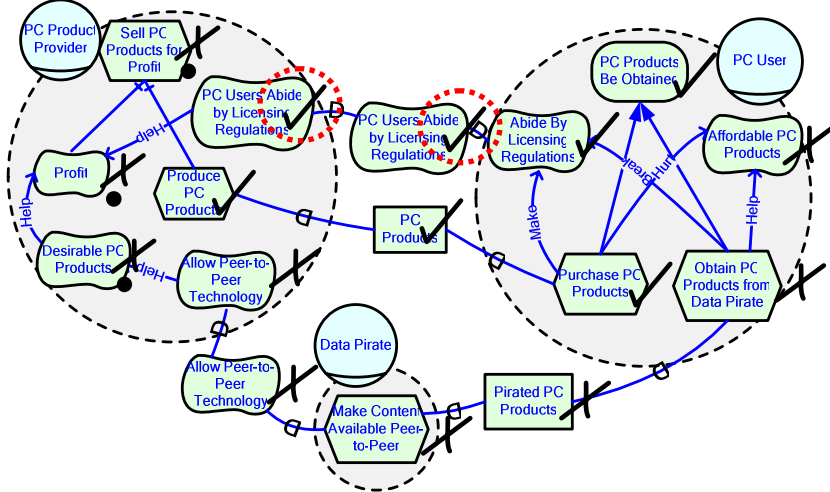
Step 1



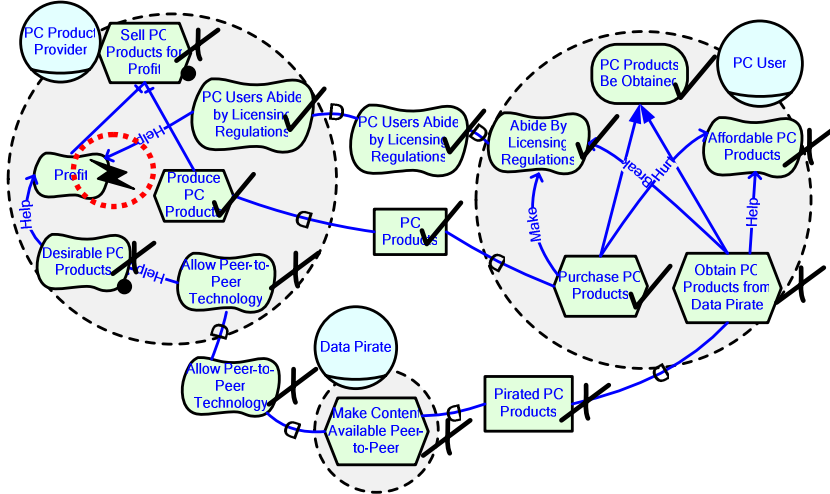
Step 5



Step 6



Step 7



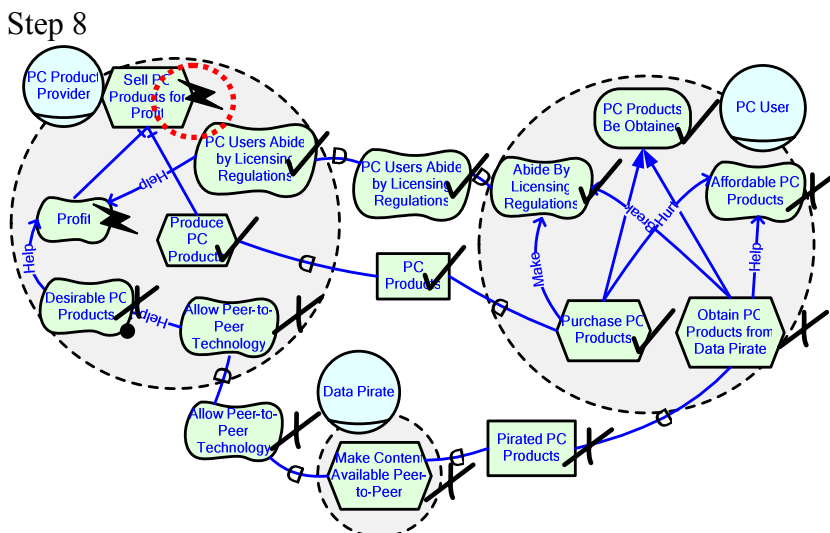


Figure 4.20: Propagation Steps in Example Trusted Computing Model

4.6 Domain Analysis

A significant benefit of i^* evaluation is its ability to facilitate analysis by providing answers to strategic questions. The provision of this ability is the motivation for the desired quality criteria defined in Chapter 2. This capability for analysis can help the modeler formulate a high-level system design that makes the most effective tradeoff between the wishes and needs of actors. The modeler is now able to find answers to complicated questions that are difficult to discern from simply studying the model. Often such analysis facilitates further model iteration as improved design configurations are sought. We shall demonstrate the capabilities of i^* evaluation for analysis using examples, as in the previous sections. Additional examples of domain analysis using analysis may be found in Chapter 7, where case studies will be examined in greater detail.

In this example, we return to the evaluation results for the model in the previous section, repeated here in Figure 4.22. When examining even this relatively simple model we can think of several analysis questions relating actor options. For example the question evaluated in the previous section: If the PC Product Provider decides to not Allow Peer-to-Peer Technology, what effect will this have on Sell PC Products for Profit? By manually tracing through the links, we were able to guess that Profit would have a

conflicted value. Now, with evaluation, we can see these results directly. Namely, as the PC User is not able to Obtain PC Products from the Data Pirate, in order for PC Products to be Obtained, the PC User must Purchase PC Products. This has a positive effect on Profit; however the final results show a conflict for Profit as a result of the Desirable PC Products Softgoal. The overall result is a conflict value for Sell PC Products for Profit. If we abstract these results away from the constructs of the model, we can see that preventing the use of peer-to-peer technology will reduce piracy, but will also make products less desirable to users; therefore the overall effect on business profit for PC product providers is both positive and negative.

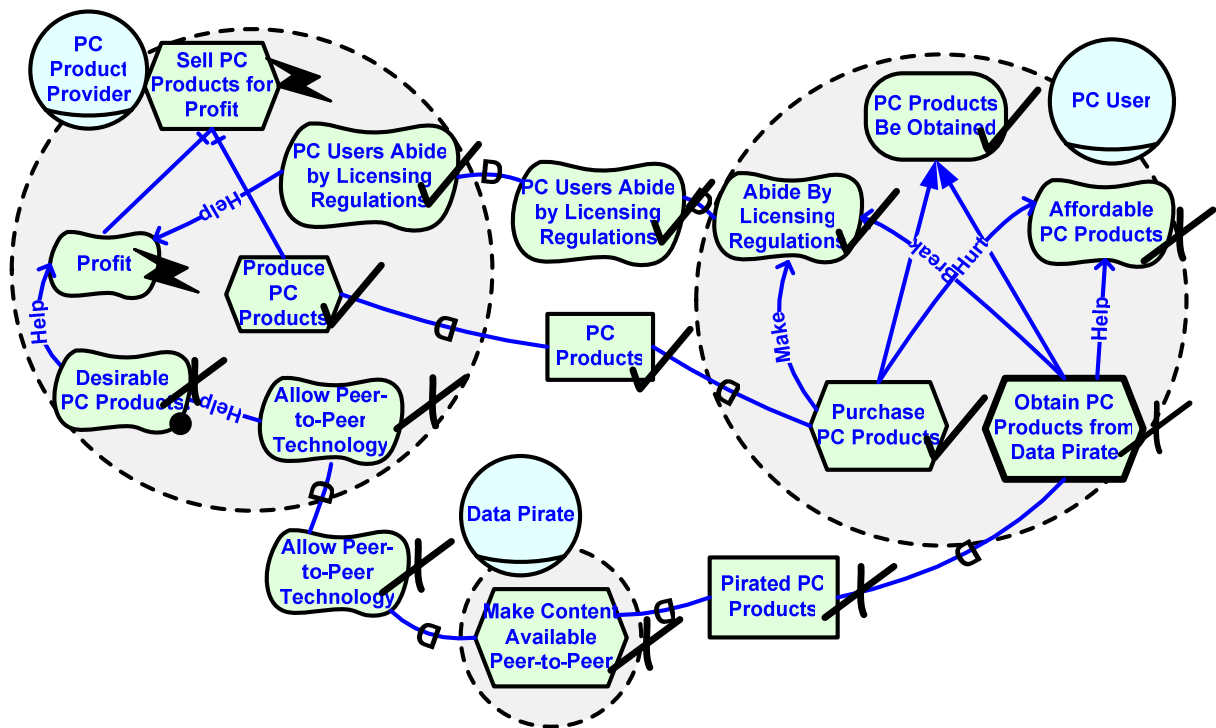


Figure 4.21: SR Model Example, a Simplified Version of a Model from the Trusted Computed Case Study

In a more complex model, evaluation can produce analysis results that are difficult to acquire by manually examining the model. For example, in Figure 4.23, we show an excerpt of the full version of the Trusted Computing Model from Figure 4.22. Here we are evaluating the results of the implementation of TC Technology on the Technology User. We can see that the Technology User no longer has a Rich Selection of Content from the Data Pirate, meaning they are not likely to Obtain Technology from the Data Pirate. The satisficing of Lock-in means that they will Purchase Technology even though

they do not Desire it or Trust the Provider. This explains why Trusted Computing opponents believe that users will purchase products with Trusted Computing technology, even though these products harm many of the user's goals.

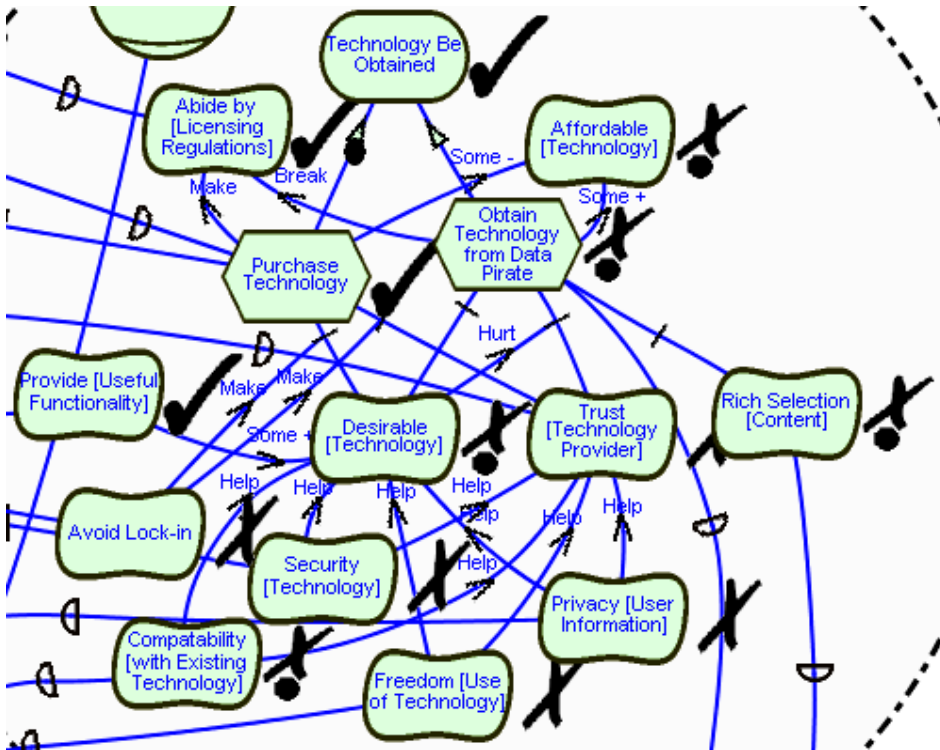


Figure 4.22: Excerpt from Full TC Opponents Model

The example in Figure 4.24 concerns Privacy in E-Commerce and shows the affects of an Online Corporation Using Customer Information Internally and Externally on the Privacy of the Online Consumer and their Trust in the Online Corporation. Based on this domain description, we may be prompted to ask several questions. For instance, what if Customer Information was used only Internally and not Externally? To evaluate this question, the tasks Using Customer Information Externally (circled near the top of the model) are marked as denied, and the tasks Using Customer Information Internally (circled near the bottom of the model) are marked as satisfied. In the results of this evaluation, shown in the same Figure, we can see that because of the And relationship used to determine the overall value of Privacy (highlighted by a square), as long as Customer Information is still Used Internally, not using Customer Information Externally makes no difference to the overall values of Privacy and Trust. In other words, only one type of customer information usage

can see that in both cases, the employee's goal of Profit is partially satisfied. In addition, the employee is seen as Competent resulting in Retaining Employment. According to our analysis, there is no motivation for an employee to choose the best product when that product is from a little known company. This result is interesting, and would likely provoke further investigation to determine its accuracy.

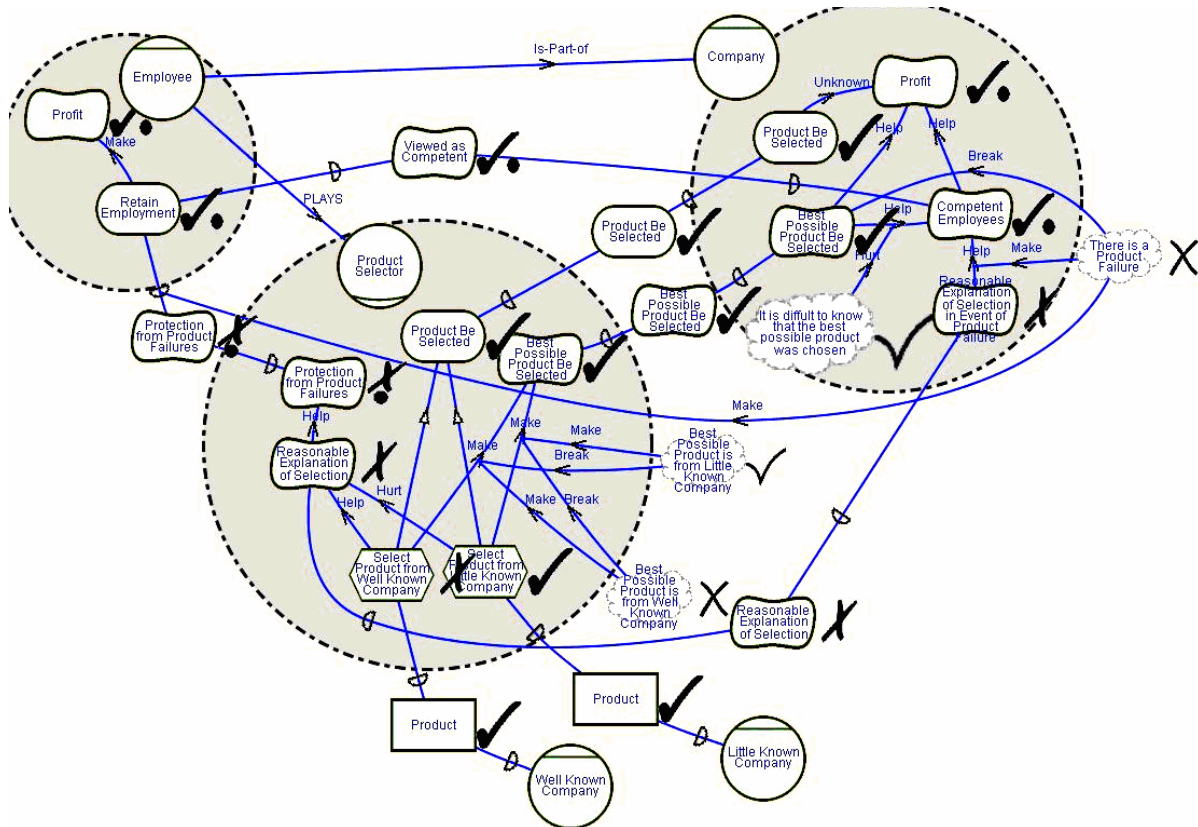


Figure 4.24: Excerpt from an Economic Information Security Model showing the Evaluation Results

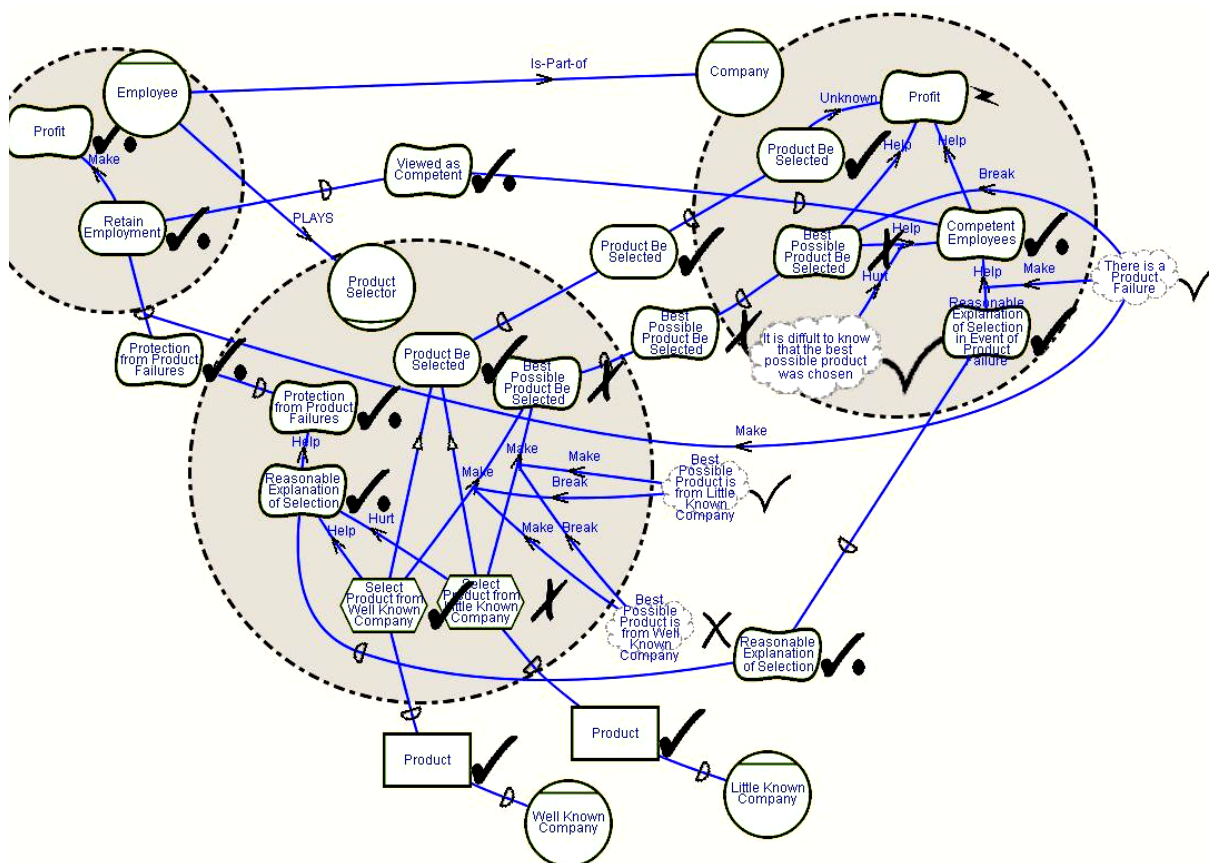


Figure 4.25: Excerpt from an Economic Information Security Model showing the Evaluation Results

4.7 Evaluation Adaptation

As we have described in Chapter 2, Section 2.1.3, since its introduction, i^* has been applied to numerous fields including Knowledge Management, Requirements Engineering, and security analysis. As the intention of i^* in each of these fields may differ slightly, the way in which i^* is used may also differ. For example, in Requirements Engineering i^* is used as a way to capture and explore the social domain for a new system in order to explore high-level system alternatives before generating specific requirements, emphasizing the use of system goals as a source for decomposition into requirements. Whereas in security analysis i^* is used to examine vulnerabilities, security threats and the effectiveness of countermeasures, emphasizing the use of links in attack and defense.

As the capabilities of the i* Framework are employed differently in different areas, the use of evaluation of i* models may also differ. For example, evaluation in Requirements Engineering may be focused on the evaluation of non-functional requirements when finding an optimal system design, while evaluation in security analysis will focus on the degree of satisfaction of the various aspects of security, given the attacks of malicious parties. It is our intention that the evaluation procedure as described in this chapter should act as a general guideline for application of evaluation, with the potential for useful adaptation in the differing areas in which i* is applied.

4.8 Conclusion

By expanding and adapting the CNYM procedure to evaluate i* models, we have created an i* evaluation procedure which achieves several of the essential qualities of an evaluation procedure such as Element Evaluation (i), Propagation Guidelines (ii), and Human Intervention (iii). Other essential qualities of the procedures, such as Accuracy (iv) and Usefulness in Multiple Contexts (v) will be addressed in our exploration of evaluation case studies in Chapter 7. Further qualities will be addressed in the remainder of this work.

Chapter 5: Improving Model Quality as a Result of Evaluation

5.1 Introduction

The process of evaluation provokes deeper consideration of aspects in the potential system environment, promoting a greater understanding of the domain. This greater understanding is likely to prompt the modeler to modify the model. We use the term “model iteration” to refer to the iterative refinement of a model assisted by model evaluation. We argue that such iteration creates higher quality models. In addition, use of the i^* evaluation procedure can prompt an expansion of the syntax of a particular model, in order to better facilitate the propagation of evaluation values. In this chapter we shall investigate the syntax and semantic changes prompted by evaluation, illustrating their effects on the quality of models and on modeler domain knowledge using multiple examples from our i^* evaluation case studies, the case studies are covered more extensively in Chapter 7.

5.1.1 Model Quality

In order to assess model quality, and specifically the change in model quality prompted by evaluation, we enumerate aspects which contribute to the quality of models.

(1) Accuracy. We can consider two types of accuracy in terms of models, accuracy in terms of correctly reflecting the modeler’s perception of the domain, and accuracy in terms of correctly reflecting the domain itself. A high quality model will satisfy both types of accuracy, as ideally the modeler’s perception of the domain will be near to the realities of the domain. In fact, we claim that the process of iteration through evaluation will help to bring the modeler’s perception closer to reality, as issues brought to light by evaluation force the modeler to ask interesting questions and further investigate reality.

(2) Comprehensibility. How easy it is to understand the elements and relationships portrayed in a model is a reflection of its quality. As with accuracy, there

are two facets of comprehensibility, how comprehensible the model is to the modeler(s), and *transferability*, or how comprehensible the model is to others. Included in this aspect are the notion of *detail* and *simplicity*, both of which can be necessary to promote comprehension, but which conflict with each other, making careful tradeoffs necessary.

There also exists a potential conflict between accuracy and comprehensibility, as complete accuracy calls for more and more model detail, the addition of which, past a certain threshold, will make the model hard to comprehend. Therefore, we look for a balance between sufficient accuracy and viable comprehension, making appropriate tradeoffs depending on the context of model application, and the identity of the model audience, i.e. system analysts vs. users.

Unfortunately, it is difficult to definitively measure levels of accuracy and comprehensibility in models. One possibility would be to perform various experiments testing the accuracy of model and the ease of comprehension. However, this sort of rigorous validation is out of the scope of this work, and left for future investigation.

5.2 Expanded *i Syntax Prompted by Evaluation**

Use of *i** evaluation occasionally produces situations where the modeler is inclined to add additional information to the model, making propagation of evaluation values more intuitive. During the first drafts of a model, when the modeler is trying to apply *i** concepts to form a picture of some aspect or section of the domain, the modeler is likely to create a model which uses the notations which are the simplest and easiest to apply. This may be done without a deep consideration of the meanings conveyed by the *i** syntactic constructs. When the evaluation procedure is applied to such models, certain changes such as the removal of circularity, the removal of multiple dependency links to one element and the repetition of softgoals across dependencies may occur. We shall explore these potential changes in the sections below.

5.2.1 Circularity

A model may contain a cycle of links having no obvious beginning or end. This situation may seem acceptable in the early stages of a model, but confusion may arise when a modeler attempts to place initial labels. In this case the modeler may pick some arbitrary starting point, or they may be inclined to adjust the model to add a starting point for evaluation. For example, in Figure 5.1, in order to Purchase PC Products, PC Products must be obtained from the PC Product Provider, depending on its task of Sell PC Products for Profit, which must receive payment from the PC User in order to be satisfied. This circular structure is typical for a model depicting a direct exchange of resources. Although this may seem sensible, it is not clear where to place an initial value to start the evaluation. The evaluator may choose an arbitrary starting point, such as the PC Products resource, or they may redraw the model, adding in a node that acts as a starting point. Before PC Products can be provided, they must be produced, and this production can be seen as a part of Sell PC Products for Profit. Figure 5.2 reflects these changes. The Produce PC Products task now provides an obvious place to start an evaluation of this model.

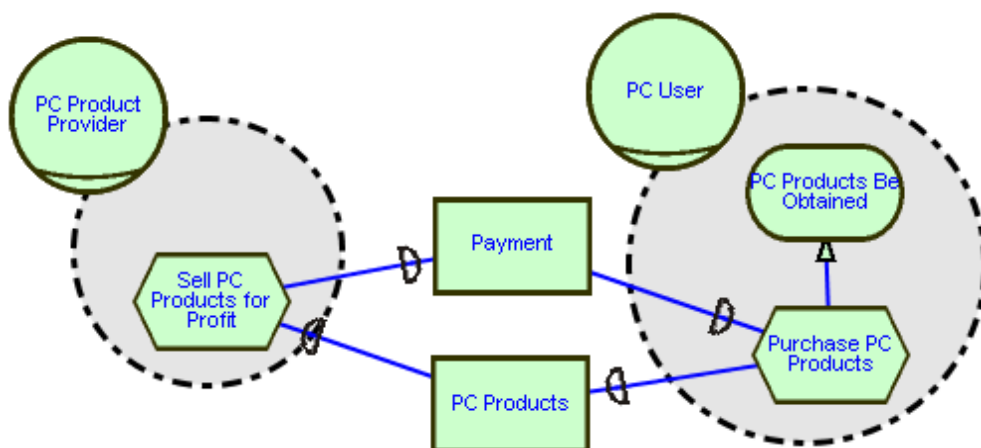


Figure 5.1: Excerpt from the Trusted Computing Domain showing Model Circularity

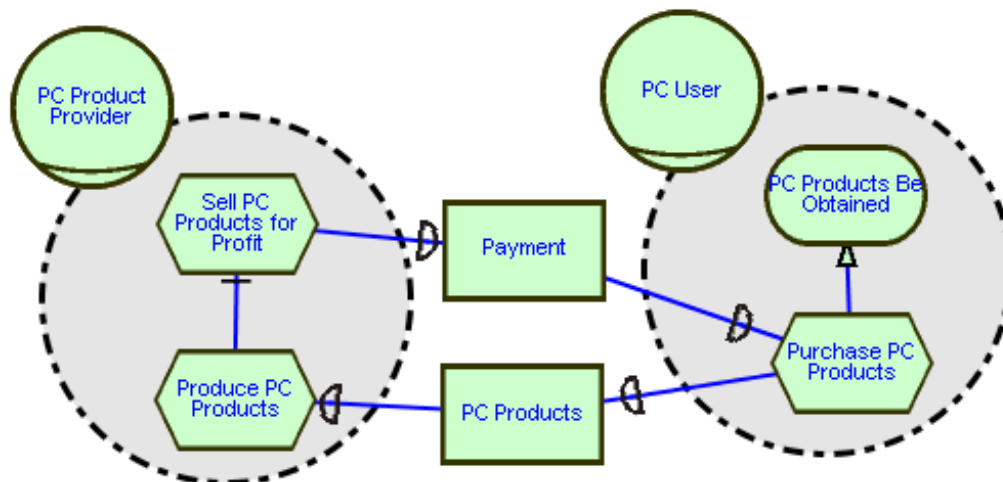


Figure 5.2: Excerpt from the Trusted Computing Domain redrawn to avoid Circularity

5.2.2 Single Dependent with Multiple Dependents or Dependees

A further example of *i** syntax which may be expanded in order to facilitate evaluation involves the presence of multiple dependency links to a single dependent. If the same dependent depends on two different dependees, often they are drawn with multiple dependency links. This has the potential to cause confusion when evaluation is applied, as the values from multiple dependees must be somehow combined together. For example, in Figure 5.3, the Online Consumer depends on Privacy from both the User of Customer Information Externally and User of Customer Information Internally roles. Drawing the model this way before the application of evaluation may seem sensible, as the Consumer's Privacy depends on both of these actors. However, as evaluation is applied, the modeler must determine how to combine the partial negative and partial positive evaluation values as shown in the Figure 5.3 example. In order to avoid this confusion, the model can be redrawn with more detail, as in Figure 5.4. In this case the Consumer's Privacy is decomposed into different sources of Privacy, each of which depends separately on Privacy from the respective roles. In this case we have decomposed Privacy with an And relationship, but other relationships such as Make, Help or Some+ could be used, depending on the judgment of the modeler.

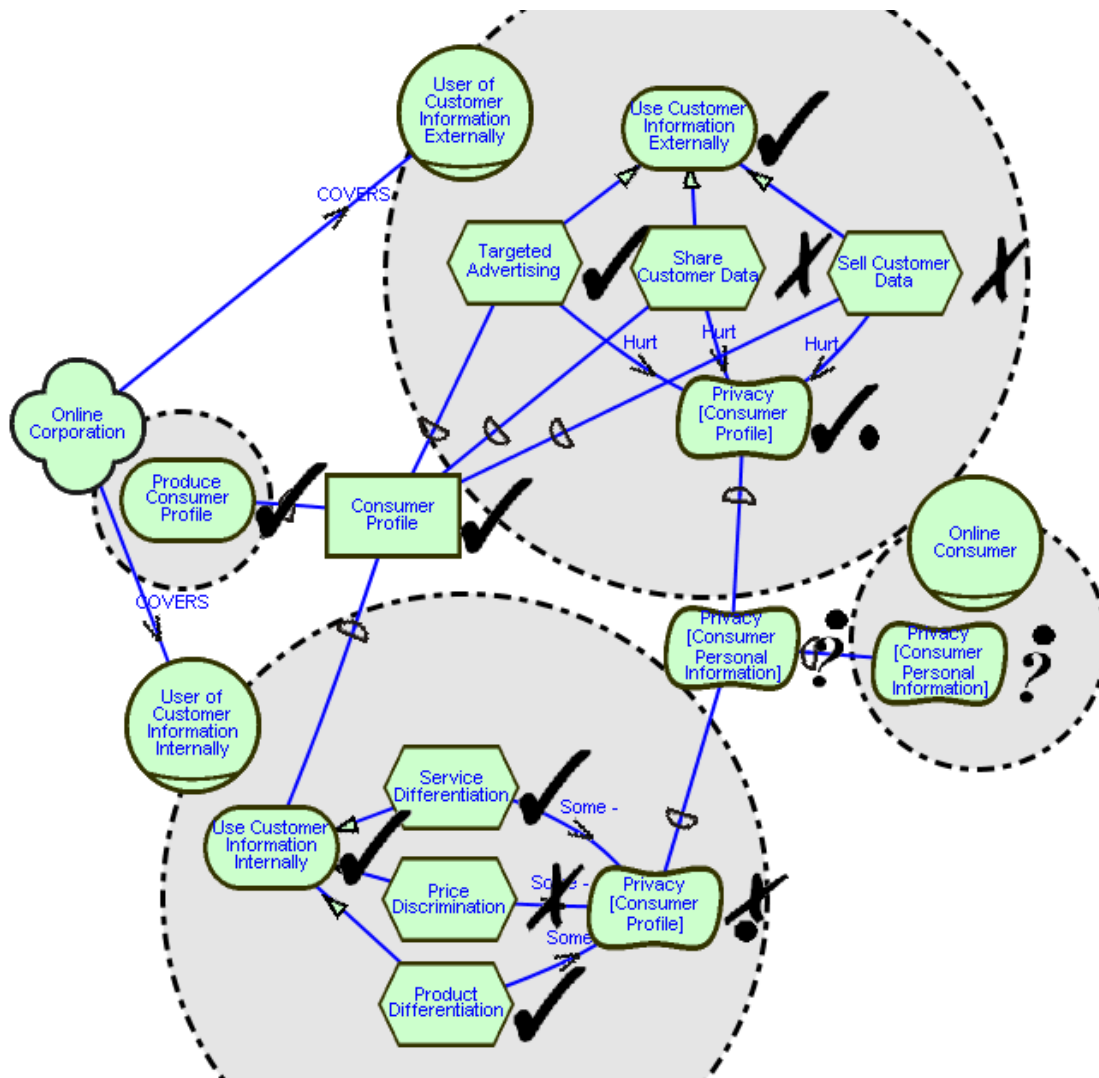


Figure 5.3: Excerpt from Privacy in E-Commerce Example showing Multiple Dependency links to Single Elements

These examples contain another instance where an element is involved in multiple dependency relationships. The Consumer Profile resource is depended upon by four different elements in the model. However, in this case it would be impossible for each dependum to have different valuation values as they each have the same source, Produce Consumer Profile. Therefore it is less likely that this syntax will be expanded by adding four separate dependums.

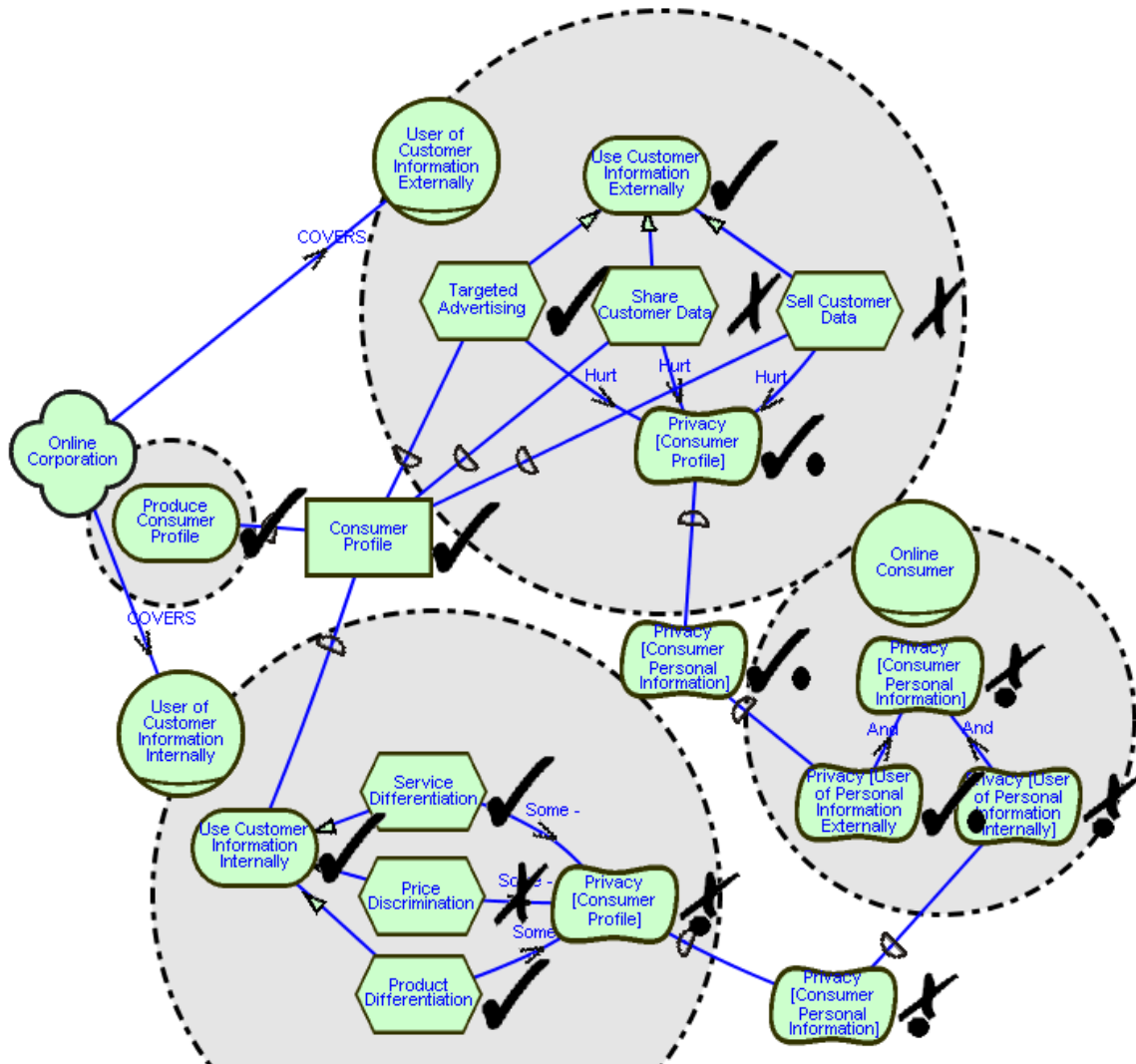


Figure 5.4: Excerpt from Privacy in E-Commerce Example where Multiple Dependency links to Single Elements are partially removed

5.2.3 Repeating Elements across Dependencies

There are many instances where elements should be repeated across dependencies. The Privacy softgoals in Figure 5.4 serve as examples. Even though elements repeated in dependencies may have the same name, these elements have a different meaning in each actor, as they depict the fact that an actor explicitly wants an element accomplished. Inside each actor, different elements may affect these softgoals, and as a result these elements may be given different evaluation results. Despite this, these values are still semantically related, even though they are not identical, and in

evaluation the syntax of repeating these elements as a dependum is necessary to propagate evaluation values for these related softgoals across actors. For example, in Figure 5.4, even though the Privacy softgoals are semantically distinct in each actor, dependencies link the evaluation values for these elements, as the satisfaction of Privacy in one actor depends at least partially on the satisfaction of Privacy in another actor.

5.2.4 Non-Typical i* Syntax not Hindered by Evaluation

It is interesting to point out that there exists model constructions which do not follow typical i* syntax, but which may not necessarily be detrimental to i* evaluation. This may include incorrect link and element combinations, such as a means-ends link to a task or a decomposition link from a goal, as well as the use of non-dependency links across actor boundaries.

5.2.4.1 Non-Dependency Links between Actors as Shorthand

In Figure 5.5 we see an example of the use of non-dependency links between actors, where contribution links are drawn from the internal elements of one actor to another, showing that the different means of Using Customer Information Internally all have a negative effect of unknown strength on Privacy of Consumer Personal Information. The presence of links across actors in Figure 5.5 would not deter the application of the evaluation procedure, as the procedure currently ignores actor boundaries. However, models such as this can often be expanded to avoid such syntax. In a second version of this model, shown in Figure 5.6, the contribution links are moved to a Privacy dependee within User of Customer Information Internally. A closer examination will show that although the redrawn version of the model in Figure 5.6 follows typical i* syntax by avoiding links across actors, the meaning of this model is not identical to the meaning of Figure 5.5. In Figure 5.5 the User of Customer Information Internally specifically wants Privacy to be accomplished, perhaps to contribute to a goal not shown in the excerpt, where as in Figure 5.6, this role is not concerned with Privacy.

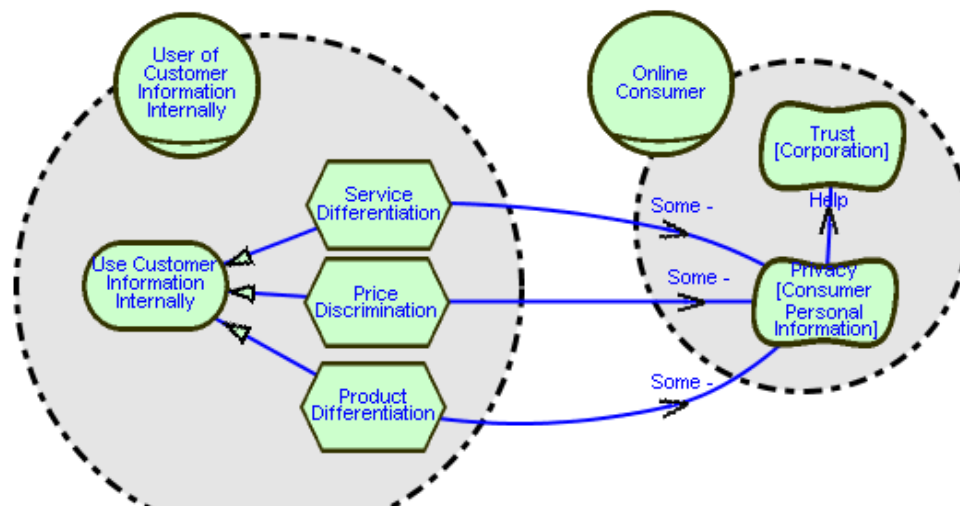


Figure 5.5: Excerpt from the E-Commerce Privacy Case Study Showing Links across Actors

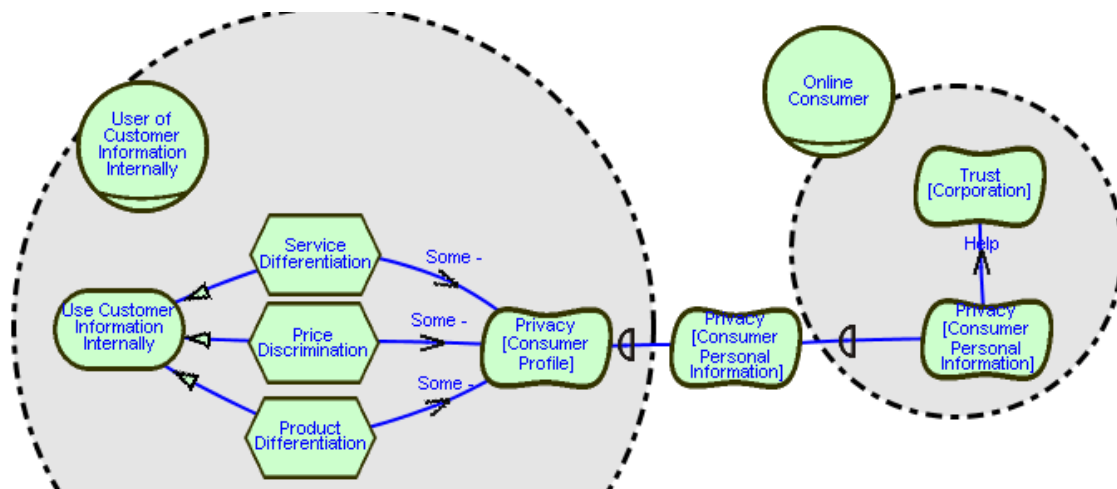


Figure 5.6: Excerpt from the E-Commerce Privacy Case Study with Links across Actors Removed

5.2.4.2 Non-Dependency Links between Actors in Attack/Defense Models

There are cases where contribution links between actors may not be seen as a shorthand for existing i* syntax. For instance, when they are used in an attack and/or defense situation, where an actor does not explicitly depend on an element, but where the actions of a malicious party have a negative effect on an actor's internal elements. In turn, a defender may have a negative effect on these effects. This is demonstrated in Figure 5.7. As with the previous example, use of the i* evaluation procedure with links of this type does not produce issues that may prompt changes in the model.

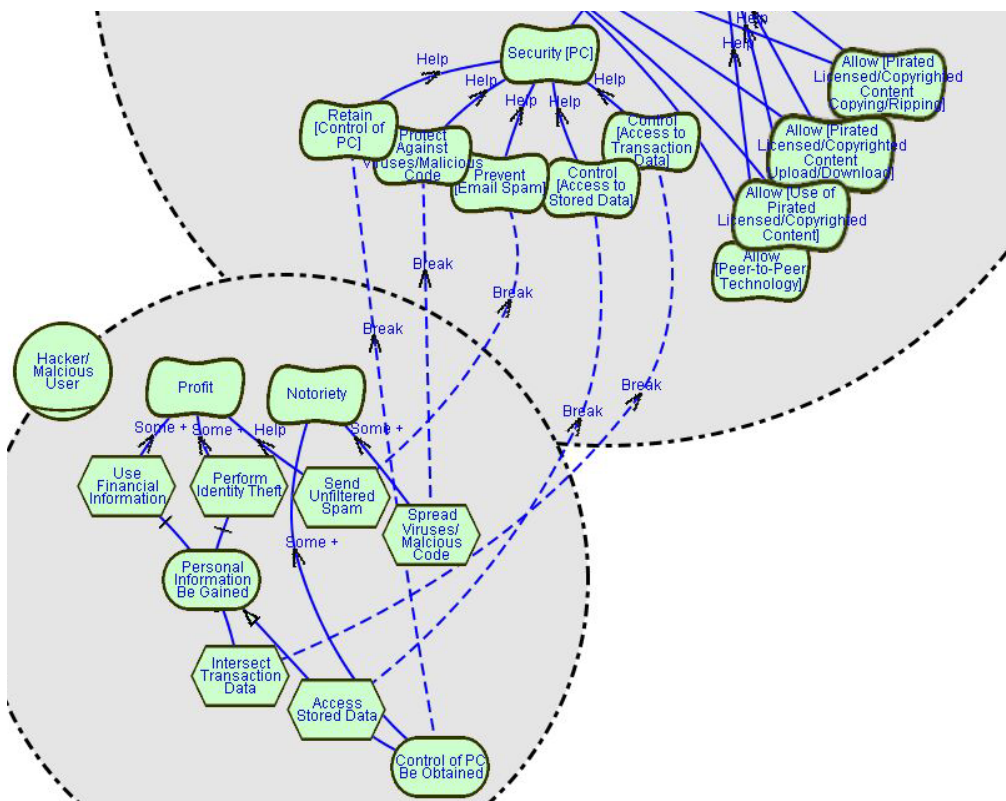


Figure 5.7: Trusted Computing Example Showing Links Across Actors as Attacks

5.2.5 Discussion

It is useful to assess whether the syntax expansion in each model prompted by evaluation produced a model which is of better quality. In the examples of syntax changes in the first three sections, it is clear that version of the model after syntax changes prompted by evaluation contains more detail than the first. One could argue that this extra detail makes the model more **Accurate (1)**, encoding more of the modeler's tacit knowledge concerning the domain. For example, in Figure 5.2, we now know that the PC Product Provider must Produce PC Products (as opposed to purchasing them from a third party). The extra detail may aid in the transferable aspect of **Comprehension (2)**, there may be less ambiguity in meaning due to the avoidance of shorthand syntax. Differences between the modeling styles of different i* modelers may be reduced by expanding model syntax to use standard constructs. The resulting models may more closely resemble models that would be included in a catalogue, as they contain typical syntactical constructs.

On the other hand, it can be argued that the extra detail added for evaluation is unnecessary for understanding the important aspects of the model, and that it may in fact add clutter to the model, hindering **Comprehension (2)**. This last point may not be obvious in most of the examples presented due to their reduced size. However, in larger models, if evaluation prompts the addition of many extra elements, this may contribute to problems with model scalability. Such scalability problems have been recognized in previous work (Maiden, Jones, Manning, Greenwood, & Renou, 2004), (Easterbrook et al., 2005). By adding more elements, simplicity is violated, and the models may be more difficult to understand, as it may take more time to understand the interactions between all elements. Additionally, there is a danger when adding more tacit modeler knowledge to the model, if answers to questions are not taken back to stakeholders for confirmation; more of the modeler's opinions are inserted in order to compensate for missing information. Such opinions may not accurately correspond to the domain if they are not confirmed by consulting with domain stakeholders.

It is difficult to determine definitively whether this extra detail improves the overall quality of the model, due to the tradeoff between the detail needed for **Accuracy (1)** and the balance between simplicity and detail needed for **Comprehension (2)**. In general there are times when each type of modeling style may be more appropriate. In the very early, exploratory stages of modeling, when the modeler is first trying to formulate his/her perceptions using *i** syntax, the shorthand style may be the easiest to apply and understand. However, when the model or models have moved to a stage of partial completion where the basic structure has been established, the modeler may wish to apply evaluation in order to check the semantic correctness of the model and begin to answer questions concerning the domain. At this point, likely prompted by initial attempts at evaluation, the model may need to be expanded, adding in extra elements which allow evaluation and conform more closely to typical *i** syntax.

In the iterative process of *i** modeling, as alternative functional elements and actor responsibilities are discovered, the simpler syntax may be used to add to or adjust existing models. When these alternatives have been established, the models can be expanded to facilitate easier evaluation. In this way the modeler may switch back and forth between styles in cycles of discovery and analysis.

5.3 Semantic Improvements Prompted by Evaluation

The application of evaluation to a model can help ensure model quality by addressing **Accuracy (1)**. Applying the evaluation procedure to an i* model often reveals that the model does not accurately capture the modeler's notion of the domain, prompting changes to the model. Additionally, applying evaluation may illuminate areas where the modelers knowledge is incomplete, driving a return to the stakeholders to confirm domain knowledge. This can help to achieve a partial validation of the model, helping to insure that it accurately reflects reality. Although such inaccuracies could be discovered by a deep examination of the model, these discoveries are made easier by applying the i* evaluation procedure described in Chapter 4.

In addition, applying the evaluation procedure makes it clear what sections of the model require human judgment in order produce an evaluation value that corresponds to real-life phenomena. These areas of the model should be considered for expansion, potentially being adjusted to reduce the dependency on modeler knowledge, making the meaning of model constructs less ambiguous and the model more transferable. By these changes the model will better capture domain phenomena, reducing the dependency on modeler knowledge. The interactive process of modeling and evaluation provoke questions that deepen domain understanding. These ideas concerning semantic model improvement aided by evaluation are explored in the following sections.

5.3.1 Semantic Validation

As final and intermediate evaluation results are analyzed, it may become apparent that these results do not reflect the modeler's perception of reality. Although it is possible that this may result in a revelation concerning the domain, often it is an indication that the model is omitting or misrepresenting some aspect. The types of semantic issues that may be noticed via the application of evaluation include:

- (a) **Appropriate Leaf Elements**
- (b) **Linking Softgoals with Similar Meanings**

- (c) **Soft Element Depending on Single Hard Element**
- (d) **Dead-End Elements**
- (e) **Missing Links**
- (f) **Inappropriate Element Types**
- (g) **Soft Element Depending on Single Hard Element**
- (h) **Differentiating between Semantically Similar Elements**
- (i) **Analyzing Direct and Indirect Contributions**
- (j) **Adding Missing Elements**
- (k) **Removing Redundant Elements**
- (l) **Using Unknown for Missing Information**
- (m) **Multiple Paths from Same Element (Redundant Links)**
- (n) **Link Completeness**
- (o) **Need for Decomposition**

Unfortunately, as our procedure ignores actor boundaries, semantic errors involving the construction of actors are harder to find via application of evaluation.

When issues are discovered, the modeler should iterate on the model, changing the model to produce evaluation results that better reflect reality. We claim that these iterations increase the quality of the model, in terms of its **Accuracy (1)**, and that these changes could have a positive effect in terms of model **Comprehensibility (2)**.

Additionally, the interactive process of evaluating and adjusting the model prompts the modeler to continually question his/her knowledge of the domain, defining terms and concepts more precisely. Ideally, these questions would fuel further elicitation, returning to stakeholders to clarify domain concepts. We shall attempt to demonstrate the accuracy of our claims concerning semantic validation by presenting an example.

We turn to the model in Figure 5.8, an expanded version of the model in Figure 5.4, showing a model after syntax expansion provoked by evaluation. This Figure represents a version of a model which was considered sufficiently complete and accurate, but to which the evaluation procedure had not yet been applied. We shall see that by applying the evaluation procedure to this model, semantic issues are discovered, leading to significant changes.

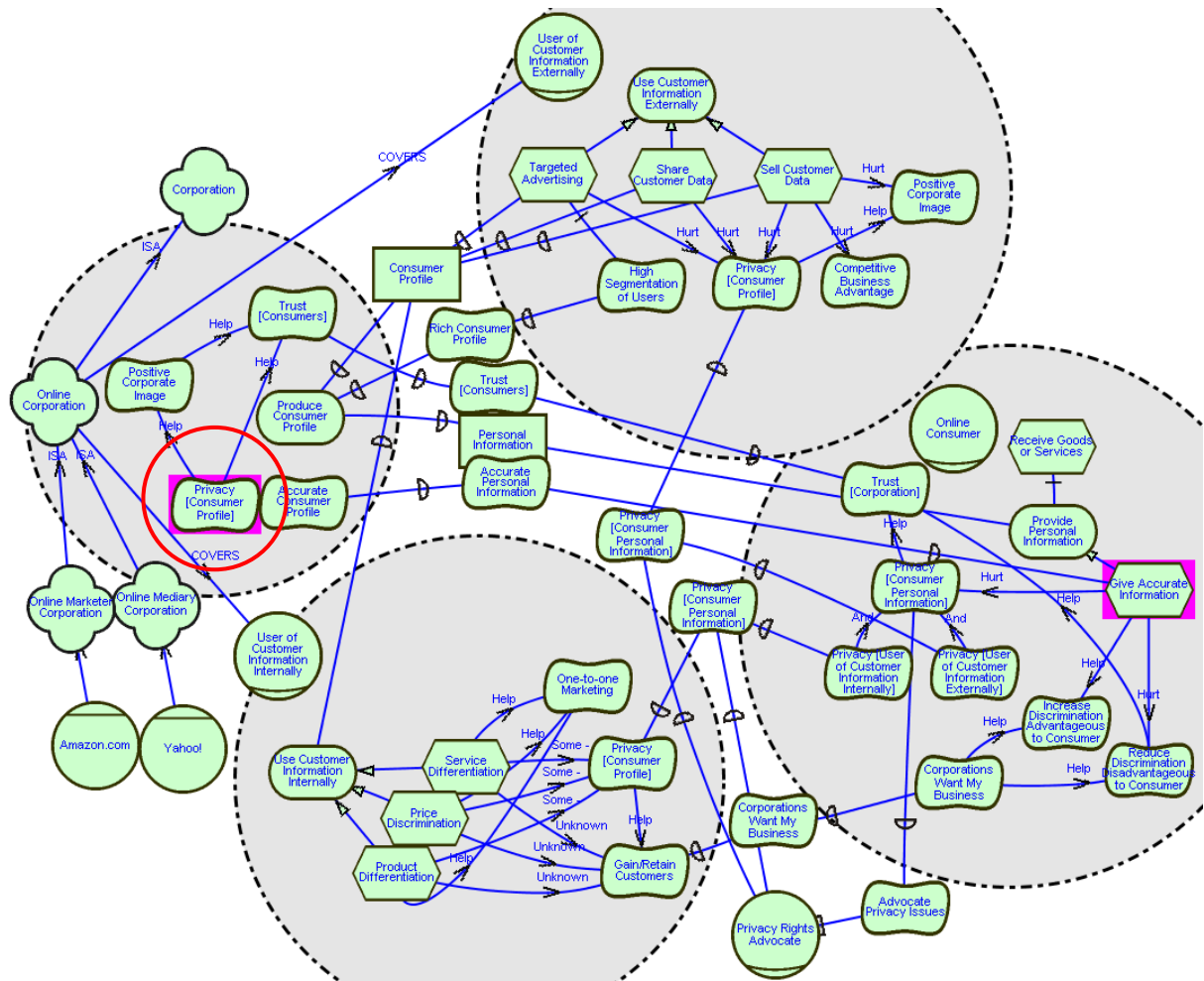


Figure 5.8: Privacy in E-Commerce Example after Syntax Changes Prompted by Evaluation

(a) Appropriate Leaf Elements and (b) Linking Softgoals with Similar Meanings. First, the evaluator identifies model leaf elements, highlighted here in pink, and applies values that correspond to a question in the domain. In this case, the Online Consumer can decide whether or not to Give Accurate Information. We see that the Privacy softgoal within the Online Corporation is also a leaf element. However, it seems that this element should be connected to Privacy in the roles covered by the Online Corporation: User of Customer Information Internally and User of Customer Information Externally. We change the model to reflect this by decomposing Privacy within the Online Corporation and having it depend on Privacy from each of the two roles, similar to the situation inside the Online Consumer. The evaluation then commences with the assumption that the Online Consumer Gives Accurate Information. These changes are reflected in Figure 5.9.

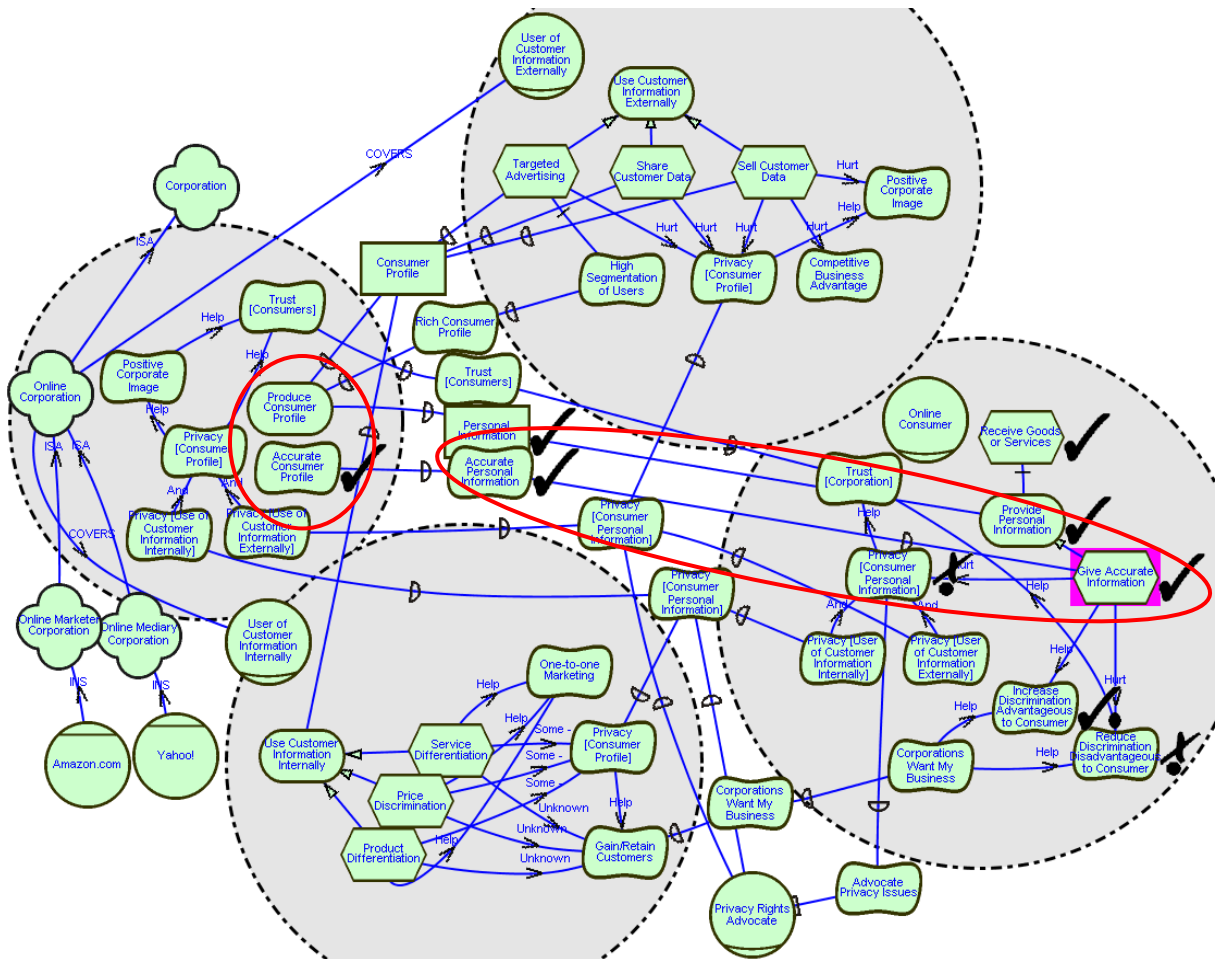


Figure 5.9: Privacy in E-Commerce Example after Semantic Changes Prompted by Evaluation

(c) **Soft Element Depending on Single Hard Element.** While propagating values from Give Accurate Information, the modeler may notice that the Accurate Personal Information softgoal depends solely on the Give Accurate Information task. As we are allowing hard elements such as tasks to take on partial values, this could be valid, but for clarity the Accuracy component of Give Accurate Information could be modeled separately within the Online Consumer. (d) **Dead-End Elements.** In relation to this, the Accurate Consumer Profile softgoal inside of the Online Corporation is a “dead end”. Although the Corporation certainly would want Accurate Information, likely they would want it in order to accomplish some higher-level goal. (e) **Missing Links** and (f) **Inappropriate Element Type.** As this Accuracy goal is concerning consumer profiles, it could be part of Produce Consumer Profile. This would indicate that this element, currently a goal, is not simply a

binary accomplishment, but dependent on the softer quality of Accuracy. Examining the effects of this element throughout the model, it seems sensible that the ability of the other actors to perform tasks such as Service Differentiation and Share Customer Data would be qualitatively affected by the Accuracy of the Consumer Profile.

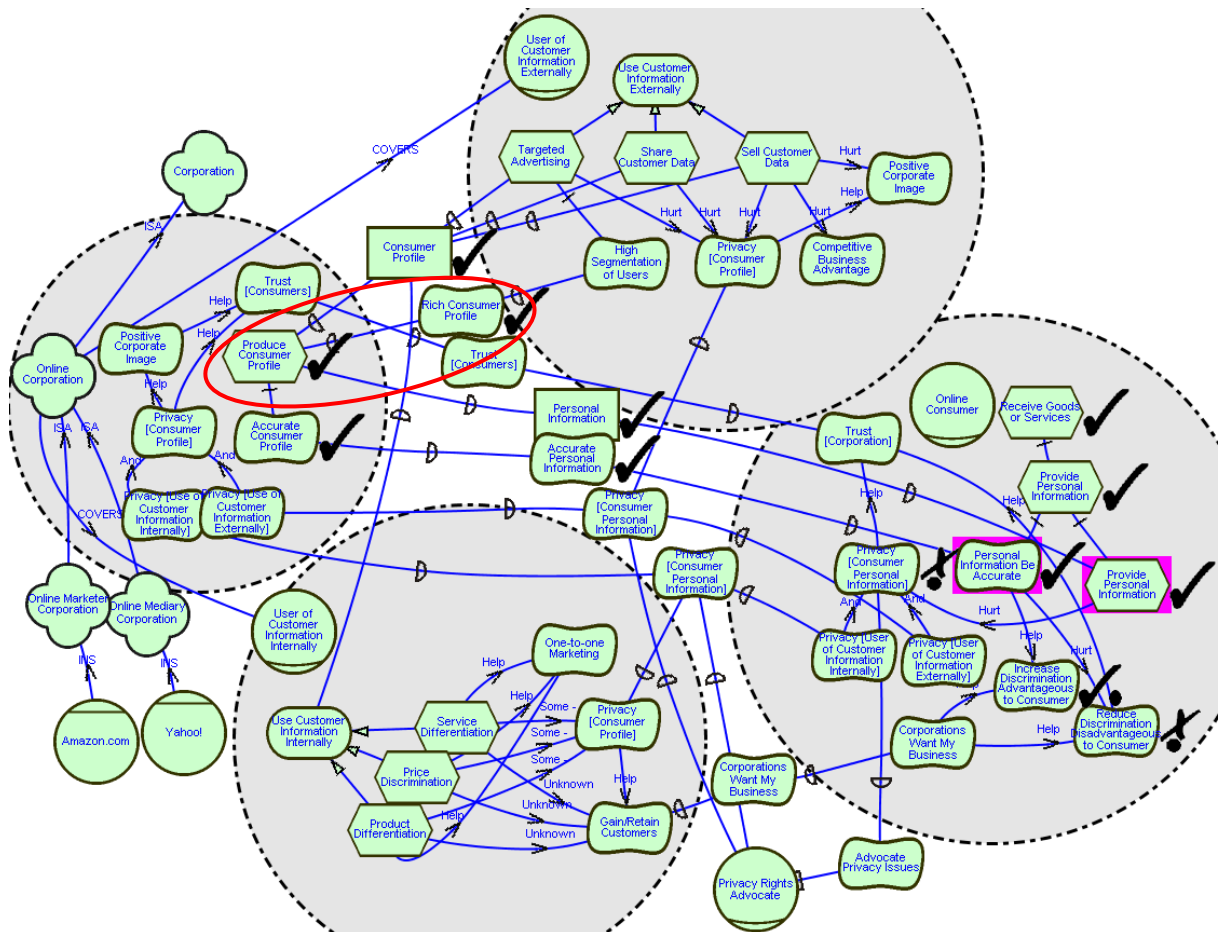


Figure 5.10: Privacy in E-Commerce Example after Semantic Changes Prompted by Evaluation

(g) Soft Element Depending on Single Hard Element. Now the modeler may notice that Rich Consumer Profile depends solely on Produce Consumer Profile, meaning that if the profile is provided and it is Accurate then the profile will be Rich as well. **(h) Differentiating between Semantically Similar Elements.** It seems that reality may be better reflected if we consider additional aspects that may result in a Rich consumer profile, forcing us to more clearly define the differences between Richness and Accuracy in Consumer Information. Perhaps if the Online Customer provides complete information, then

the profile will be Rich, indicating that another qualitative aspect of Consumer Profile must be considered in addition to Accuracy. **(i) Analyzing Direct and Indirect Contributions.** It appears that Richness or Completeness would also affect the effectiveness of tasks such as Service Differentiation and Sell Customer Data, indicating that this softgoal can be considered a part of the Consumer Profile entity. However, we can see from the model that Richness is necessary for High Segmentation of Users that is specifically required by Targeted Advertising. By making Completeness or Richness a part of Consumer Profile we are effectively making the claim that this aspect is as important for all of the tasks within the two roles as it is for Targeted Advertising. Is High Segmentation of Users important for the quality of Product Differentiation, Price Discrimination, Service Differentiation, Sharing Customer Data and Selling Customer Data? Having greater customer segmentation will enable differentiation and discrimination to be done more effectively, and will allow customer data to be more valuable for sharing and selling.

(j) Adding Missing Elements and (k) Removing Redundant Elements.

Therefore we shall include this aspect as part of Customer Profile, and eliminate the specific requirement of High Segmentation of Users for Targeted Advertising. These changes and the continuation of propagation are shown in Figure 5.10.

were not the same, it should be possible to combine these differences through model links. Upon further consideration of the domain, it appears that the Online Corporation in fact depends on the User of Customer Information Externally for a Positive Corporate Image through their use of Customer information.

(m) Multiple Paths from Same Element (Redundant Links). After this dependency is added, we are lead to consider the links to Positive Corporate Image. Within both the Online Corporation and the User of Customer Information Externally, Privacy has a positive effect on this softgoal. Selling of Customer Data, however, has a negative effect. At present, due to the presence of multiple paths, External Usage Privacy has a “double” effect on this goal, and Internal Usage Privacy a “single” effect on this goal. We must consider whether each of the external and internal tasks have a negative or positive effect on Corporate Image. It is difficult to determine the effect of the internal usage of customer data on Corporate Image, as we have not differentiated between effects that are positive or negative for the consumer. However, external usage such as sharing or selling customer data would have a negative effect on image, likely more so than Targeted Advertising. As a result of this, we eliminate the links to Positive Corporate Image from within Online Corporation and modify these links within User of Customer Information Externally.

(m) Multiple Paths from Same Element (Redundant Links). During the propagation of evaluation values a modeler may notice an area of confusion surrounding the Privacy [Consumer Personal Information] softgoal within Online Customer. This softgoal is satisfied both via an And relationship from the two users of its information and by the act of giving Accurate and Complete information. We can see by tracing through the links that as Accuracy and Completeness are already reflected in the Consumer Profile, these values would affect the effectiveness of the tasks within both roles, and consequently would affect both sources of Privacy. Therefore the Accuracy and Completeness of information contribute to the Privacy goal within Online Consumer via two different paths. In some ways this may seem valid, as the links within Online Consumer indicate the opinion of this role, that providing Accurate and Complete information will hurt Privacy. The links external to this role indicate the dependency on the roles of the Online Corporation to ensure Privacy. However, the Online Consumer cannot be sure that providing such information will actually harm Privacy, as it depends on precisely what the

corporation does with this information. One can imagine a scenario in which the corporation decides not to use personal information, in order to gain business by ensuring the Trust of the consumer. In this case Privacy may be satisfied even though the information provided is Accurate and Complete. This situation would be aided by a more precise definition of Privacy. Is the Online Consumer concerned with their predictions concerning Privacy, or the actual effects of the Online Corporation's actions on Privacy? In this case a modeler may favor the second interpretation, and therefore the internal links to Privacy are removed.

(m) Multiple Paths from Same Element (Redundant Links). Similar questions arise with the Trust softgoal within the Online Corporation. Privacy and Positive Corporate Image affect Trust from within this actor, but this softgoal also depends on the Online Consumer's perception of Trust, which in turn depends on Privacy and Reducing Discrimination Disadvantageous to Consumers. We can see here that Privacy has a "double" effect on this softgoal. In order for Trust within the Online Consumer to accurately reflect this role's criteria for Trust, the links to Trust within this role should be retained. Does the Online Corporation actually depend on the Online Consumer for Trust, or does it in fact "depend on itself" in order to accomplish the goals necessary to acquire Trust from the Consumer? Is it possible for the Online Corporation to determine if it has the Trust of the Consumer? These questions may lead the modeler to remove this dependency altogether, making Consumer Trust internally accomplishable for the Online Corporation. The model resulting from these changes, as well as the completion of the evaluation procedure, is shown in Figure 5.12.

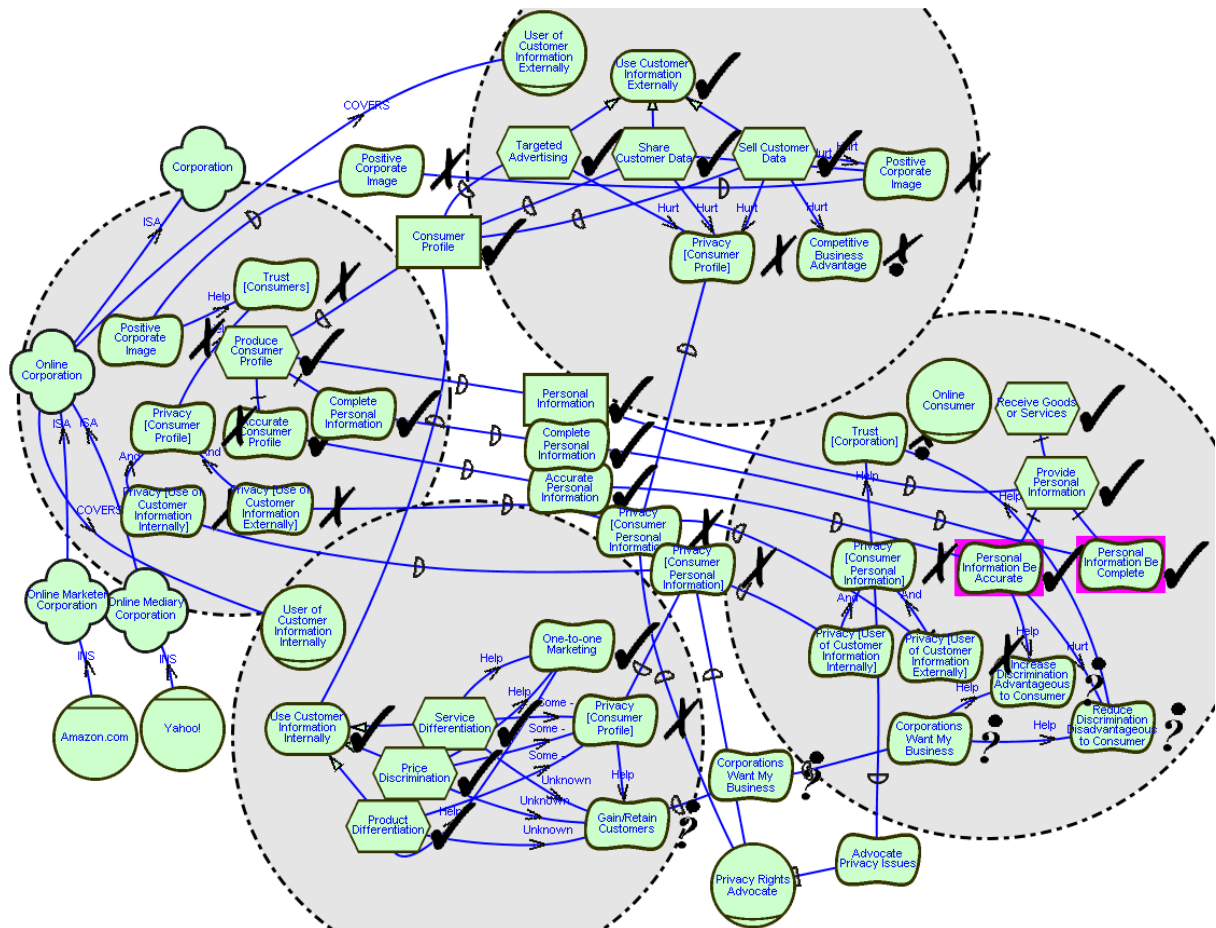


Figure 5.12: Privacy in E-Commerce Example after Semantic Changes Prompted by Evaluation

Model iteration for this example could be continued indefinitely. The results of our evaluation show that collecting an Accurate and Complete Consumer Profile results in a denial of Consumer Trust. In this case, why would the Corporation want such a Profile in the first place, and why are they concerned with Trust? Why do the roles of the Corporation want a Competitive Business Advantage or One-to-one Marketing? Aspects representing financial gain are obviously missing from the model, and may be the subject of future iterations. In addition, future evaluations of different domain questions may produce additional changes. For example, if the Personal Information provided is not Accurate or Complete, perhaps elements such as Positive Corporate Image are not affected in the ways indicated by the existing links? Questions such as these may prompt further semantic changes.

(n) Link Completeness. In addition to a check of detailed semantics, evaluation can provide a higher-level semantic check by indicating link completeness. For example, in an evaluation of a large model in the Kids Help Phone study, shown in Figure 5.13, only 62% of the model elements received an evaluation value, meaning 38% of the model was not connected to elements receiving initial values. Although this is an extreme example, evaluation can identify clusters of elements that are isolated from the rest of the graph. This may indicate semantic deficiencies, as the presence of these clusters within the same physical model likely indicates they are related, and should therefore have an effect on each other. Most likely this measure indicates that a search should be done for missing links. This is especially significant as missing links may affect the accuracy of evaluation values. However, if the model is very large, such as in Figure 5.13, there may be a certain point where the addition of more links has little effect on the overall evaluation results. The contributions from these links are effectively “drowned” among the large numbers of other links. For example, we can see in Figure 5.14 an excerpt from the Figure 5.13 model showing the large number of links arriving at the Efficiency softgoal. In this example, the addition of more links to this softgoal may make little difference to the final evaluation value. More investigation is needed into how strongly link incompleteness actually affects evaluation results.

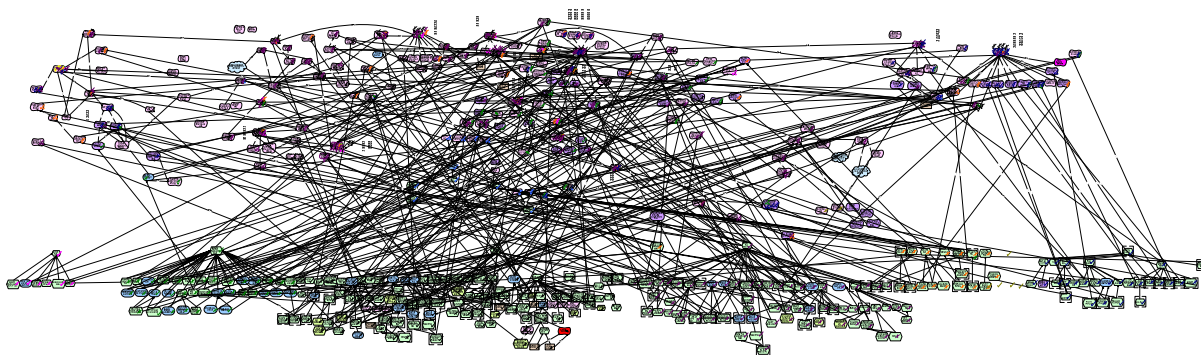


Figure 5.13: Kids Help Phone Example showing Link Incompleteness

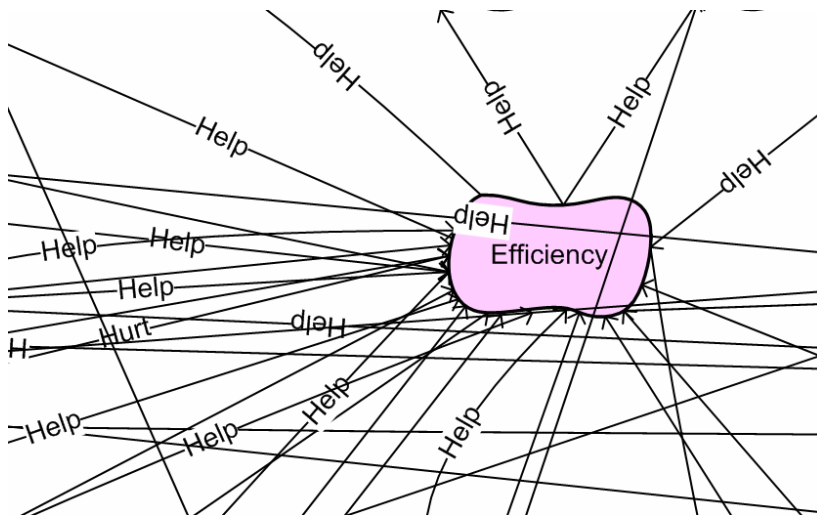


Figure 5.14: Kids Help Phone Example Excerpt Showing Large Number of Contribution Links

(o) **Need for Decomposition.** Figure 5.14 also demonstrates the difficulty a modeler may have in applying human judgment to determine final element labels in large models, as it is difficult to take into account the semantic meaning of each label source when the number of sources is so numerous. This difficulty, made clearer when evaluation is applied, indicates that the model should ideally be expanded and decomposed. In this case, for example, one could decompose Efficiency by exploring the many different types of Efficiency, such as Efficiency for a Counseling Phone Call, Efficiency for Writing a Web Post, Efficiency for Editing and Reviewing a Web Post, etc. Moving the contributions to Efficiency to these elements would significantly reduce the number of contributions to a single node, making human judgment using domain knowledge easier to apply. However, as the size of this model is already very large, adding more detail may be difficult or infeasible. Again, we see a tradeoff between facilitating easier evaluation and model complexity. In this particular case, the model may have to be broken up into related conceptual chunks in order to allow for expansion.

5.3.1.1 Discussion

We have claimed that the changes prompted by evaluation produce a model that is of a higher quality. In order to validate this claim, we have defined the notion of quality in models as including **Accuracy (1)** and **Comprehension (2)**. It is difficult to

definitively show an increase in such aspects of the model. Nevertheless, we can argue, based on the reasons for semantic changes presented in the above sections, that the model produced after iteration is more accurate, both in terms of the modeler's notion of the domain and the domain itself. It is more difficult to argue that such models are easier to comprehend, as, similar to the discussion on syntax changes; the details added to the model could potentially have had a positive or negative contribution to comprehension.

However, as the semantic iteration prompted by evaluation has likely provoked the user into thinking deeply about the model and its constructs, the resulting better-thought-out model is likely to be more comprehensible. As mentioned in Section 5.1.1, experimentation is needed to confirm our claim concerning an improvement in model accuracy, and to determine whether changes to models resulting from evaluation produce more comprehensible models.

It is easier to confirm our other claim, that the process of evaluation and model iteration provokes deeper thought and understanding concerning the domain. In our example we were forced to consider the qualities that make using customer information possible, the differences between accuracy and richness or completeness in information, the precise meaning of privacy, and the role of trust in the domain.

Although we have not explicitly mentioned stakeholder intervention in our example, one can see many instances where, instead of using modeler domain knowledge to make changes to modeler, one could use the considerations made during evaluation to formulate detailed questions to pose to system stakeholders. For example, what is the relationship between Richness and Accuracy in Consumer Information? Or, do all ways of Using Customer Data Externally have an equally negative affect on Corporate Image? By such questions the process of model evaluation can help direct further acquisition of domain knowledge. Figure 5.15 compares our example before and after the changes prompted by evaluation.

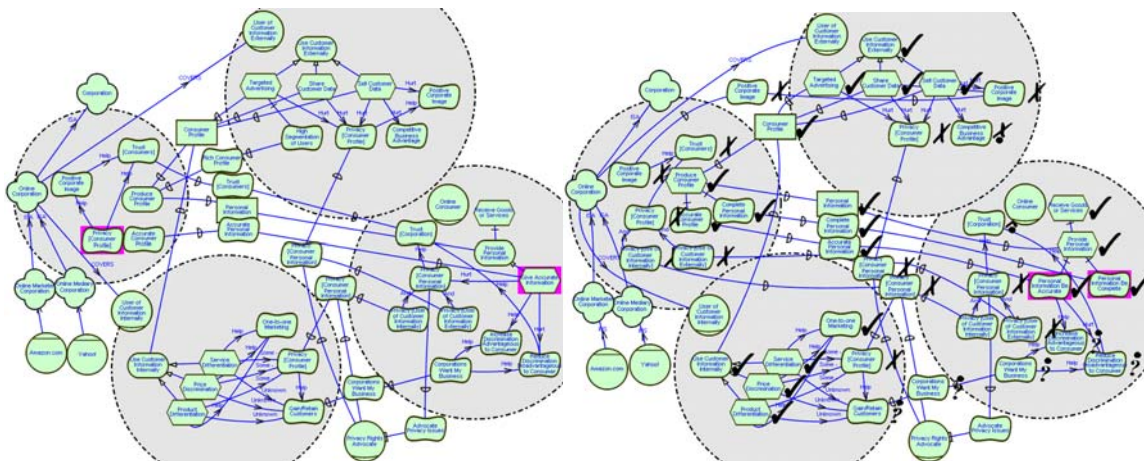


Figure 5.15: Privacy in E-Commerce Example before and after Semantic Changes Prompted by Evaluation

5.3.2 Reduction of Ambiguity

Although we have stressed the importance of Allowing Human Intervention (iii) in the i^* evaluation algorithm, as identified in Chapter 2, Section 2.3.1, the use of such judgment implies that the evaluator possesses specific domain knowledge which is not yet captured in the model. Therefore, although human judgment is useful for working iterations of the model and the evaluation procedure, the objective should be to produce a model that is **Accurate (1)** enough so that the need for human judgment is reduced. In this sense the evaluation procedure helps to indicate the areas in the model that could potentially be expanded or clarified. In some cases, the meanings of contributions are apparent even though human judgment is required, especially when multiple partial contributions are used. For example, in Figure 5.12 human judgment is required to determine the value for Trust in both the Online Consumer and the Online Corporation. In this case the meanings of the help links are relatively clear, and the resulting judgment of denied and partially denied closely reflects the structure of the model. In other cases, the modeler may be using tacit knowledge that can be included in the model. However, given the high-level nature of analysis in i^* and the scalability issues inherent in SR models, it may not be possible, necessary or desirable to include all relevant tacit knowledge in the model. The strategic nature of i^* models indicates that modelers have to make a difficult choice between detail and complexity, the aspects of

Comprehensibility (2), strategically choosing which aspects of the domain to include and exclude from the model. Specifically the modeler must attempt to determine which potential elements will have a significant affect on other elements in the model.

We provide some examples of model expansion prompted by the need for human judgment by examining the models produced in our various case studies. An example of this situation has been given in the previous section by Figure 5.14 from the Kids Help Phone domain. The need for human judgment in evaluation indicates that this element may be a candidate for refining. In this particular case we can explore the different types of Efficiency.

In another example from the Trusted Computing domain, shown in Figure 5.16, Profit is given a conflicted value based on the values of four contributing nodes. The complexity of this judgment may indicate that a refinement of Profit would be beneficial in understanding the evaluation decision. In this case we refine Profit to Profit from Sales and Profit from Protection of Content, shown in Figure 5.17. It is our claim that this refinement makes the model and the judgment it contained clearer and more transferable.

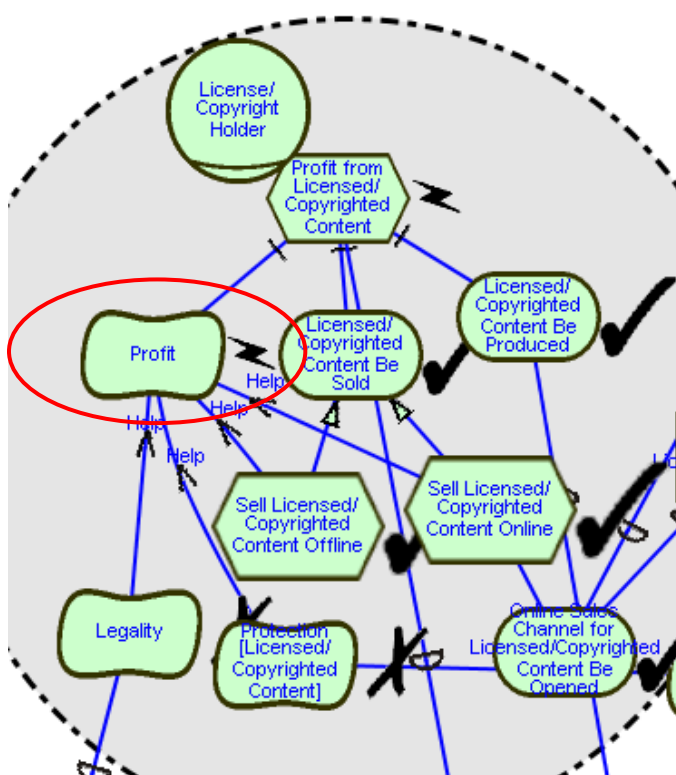


Figure 5.16: Excerpt from a Trusted Computing Model showing Human Judgment before Refinement

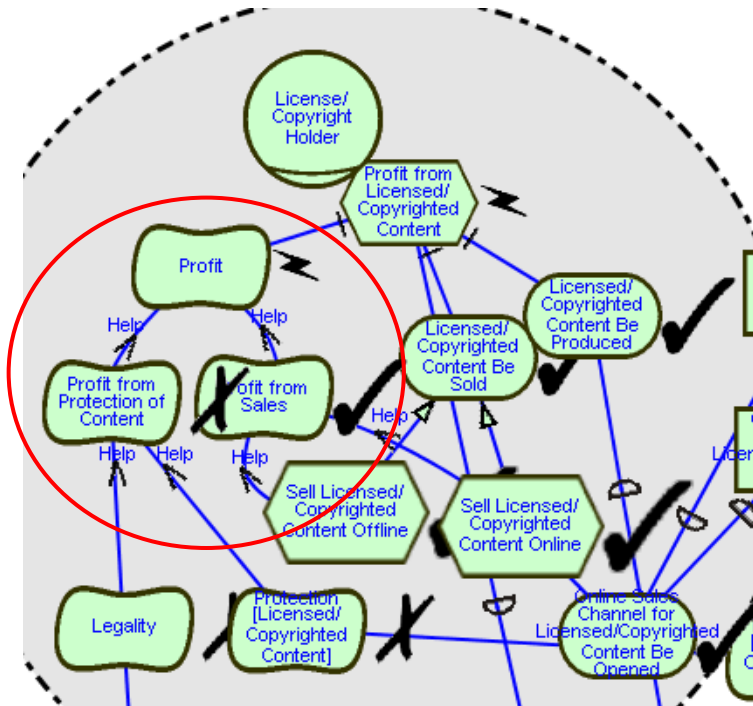


Figure 5.17: Excerpt from a Trusted Computing Model showing Human Judgment after Refinement

In a similar example, the excerpt from a model in the Montreux Jazz Festival study in Figure 5.18 shows a conflict value for the Reduce Expenses softgoal given the input of many other elements. This softgoal can be refined to explore the different kinds of expenses that can be reduced, or to explore the specific goals that can help to Reduce Expenses. For example, the Jazz Festival can Acquire Free Services, Reduce Expenses for Physical Items and Reduce Labor Costs. These elements are added to a refined version of the model in Figure 5.19. We claim that the refinement makes the human judgment in this case easier to understand, as now only a few contributions are combined together to produce a resulting value.

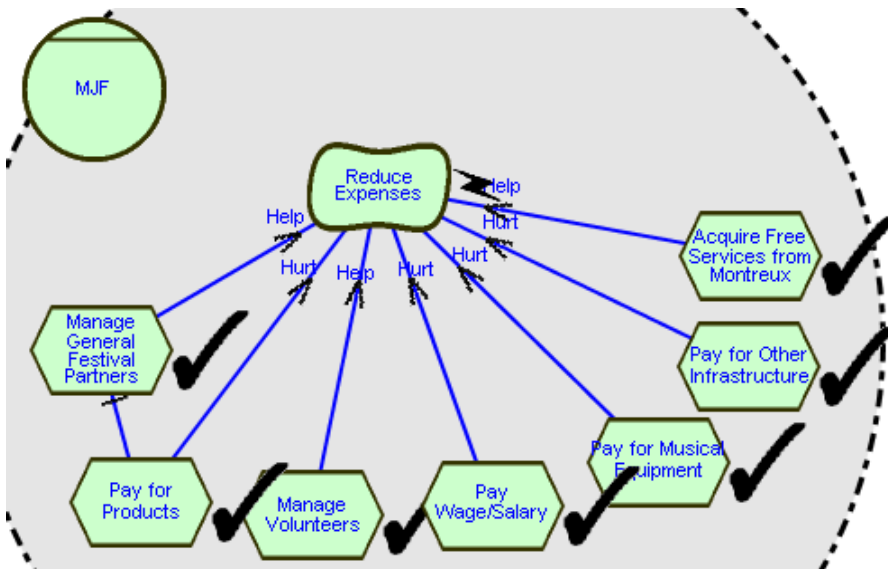


Figure 5.18: Excerpt from a Montreux Jazz Festival Model showing Human Judgment before Refinement

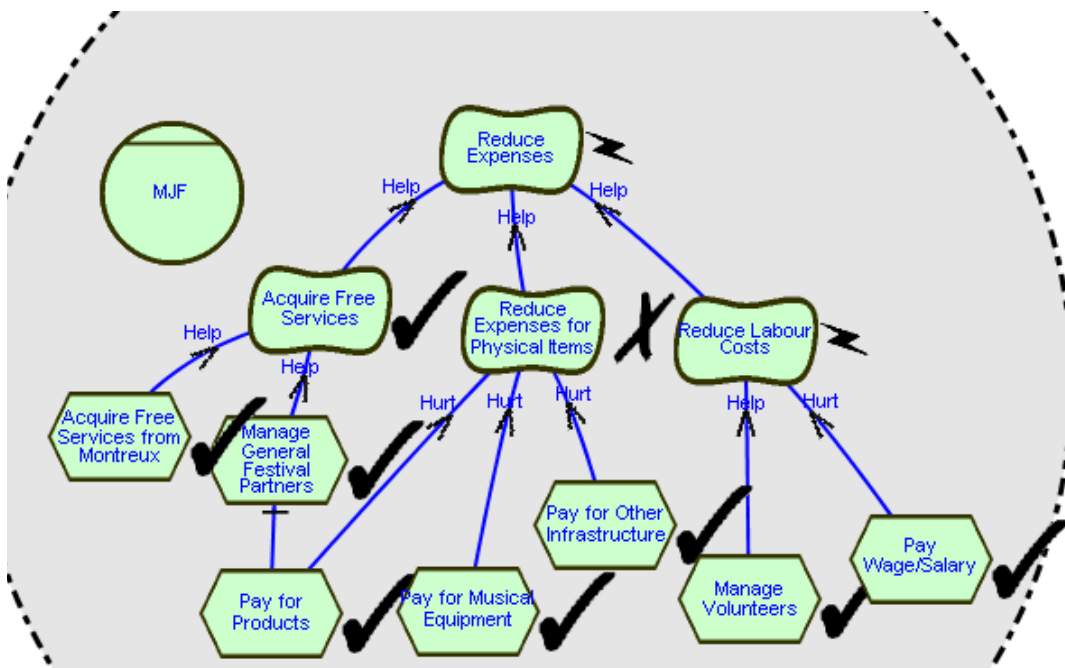


Figure 5.19: Excerpt from a Montreux Jazz Festival Model showing Human Judgment after Refinement

5.4 Conclusion

In this Chapter we have presented extensive examples to demonstrate the changes in syntax and semantics prompted by evaluation, and the resulting improvement in model quality, specifically in terms of **Accuracy (1)** and **Comprehensibility (2)**. Although we

have discussed the syntactic and semantic effects of i^* evaluation separately it is clear that these effects are highly interrelated and often not distinguished from each other in practice. The interactive process of evaluation often prompts syntactic and semantic changes simultaneously, and it is common for the results of an analysis question to provoke further changes in the model. The argument could be made that if the evaluator is continually adjusting the model to correspond with his/her notion of the domain, subsequent evaluations will not produce results that bring revelations or new information, only what the modeler already knows. We argue that as the modeler adjusts the model to conform to his/her perception of the domain, the modeler's perception of the domain is also altered, and for the better, as the entire interactive process forces the modeler to ask and find answers for specific questions about the relationships between entities. Ideally these questions will prompt the modeler to return to the system stakeholders, filling in the gaps from previous rounds of elicitation.

In addition to the improvements in model quality prompted by evaluation, we hope that the examples within this chapter have demonstrated that the value of evaluation is as much in the ideas and discoveries which arise during the process of evaluation, as it is in the results of the evaluation, or the resulting model. The process of modeling along with evaluation allows domain analysts to explore and expand their knowledge of the domain, likely resulting in systems that better meet stakeholder needs.

Chapter 6: Implementation

6.1 Introduction

In this Chapter, we describe the implementation of the evaluation procedure presented in Chapter 4. The implementation serves to demonstrate that the automated part of the procedure is implementable in software, and that the overall procedure is practical. We also aimed to discover hidden issues. The software implementation is described in Section 6.2. The issues discovered during implementation are explored in Section 6.3. In Section 6.4 we describe tests of the implementation and in Section 6.5 we explore various ways to reduce the need for human judgment.

The procedure was implemented as an extension to the existing OpenOME software tool for i^* modeling. An evaluation procedure, based on CNYM, has been previously implemented in OME, another version of the i^* modeling tool. We first considered whether the new algorithm could be implemented as a modification of the existing procedure in OME.

OME (Organization Modelling Environment) was initiated at the University of Toronto Knowledge Management Lab and produced two major versions, version 2 in 1998 and version 3 in 2000 (<http://www.cs.toronto.edu/km/ome/index.html>). This application, implemented in the Java programming language, allowed users to draw and manipulate both SD and SR i^* models. An assessment of the evaluation algorithm contained in this application was undertaken. This procedure prompts for human judgment in order to resolve means-ends, decomposition, and dependency links, including a mixture of link types. The implementation does consider the effects of links to other links, as described in our evaluation procedure. The algorithm does not make use of the cases where softgoals can be resolved automatically, as shown in Chapter 4, Table 4.5, resulting in a need for human judgment in all softgoal resolutions. When dealing with cycles, initial values are not retained in the element bag during user prompts. As a result, a cycle where the value is inversed is not likely to converge, unless the user decides to manually converge through human judgment. In addition, the procedure only

works on models created in the GRL (Goal-Oriented Requirement Language) Framework (<http://www.cs.toronto.edu/km/GRL/>), a version of i^* syntax. The differences between the algorithm implemented in OME and the algorithm described in this work indicates that the OME implementation cannot be used as a sufficient instantiation of the algorithm described in Chapter 4.

In order to ensure continued development and increase the pool of users for OME, an open source version of OME, OpenOME, was initiated in 2004 (<http://www.cs.toronto.edu/km/openome/>). OpenOME retains the majority of the functionality of OME and includes additional features such as integration with Protégé, a conceptual modeling tool; and Eclipse, a software development environment. As the OpenOME tool remains under development, it is not as stable as a typical commercially available software tool, or as the previous OME application. In addition, the presence of legacy code, supporting features which may no longer be present, often makes it difficult to understand its composition. However, as the core features of the application stabilize OpenOME has the potential to become a widely shared and widely used tool for i^* modeling and reasoning. In addition, the code for the implementation of the GMNS, GMNS-# and GMNS-TD goal evaluation methods, described in Chapter 3, have been partially integrated into OpenOME. These factors, together with the presence of an active development community, make it an appropriate vehicle for integration of the i^* evaluation algorithm described in this work.

6.2 Implementation in OpenOME

As OpenOME is a modification and expansion of OME, it was necessary to consider whether the OME evaluation algorithm code was suitable as a basis for modification for the implementation of algorithm in this work. This code had been deprecated in OpenOME, and was no longer called by the application. Upon locating and examining this code, it was determined that the complexity of the implementation of their algorithm, along with the use of potential depreciated data structures, made it a difficult candidate for modification. It was determined that implementing the pseudocode in Figure A.3 of Appendix A would be simpler than modifying the pre-

existing code. However, the implementation of the graphic pop-up window used in this algorithm was borrowed and heavily modified to avoid re-implementation of such GUI elements.

Figure 6.1 shows a class diagram of the i* evaluation algorithm implementation derived automatically from the Java code. The actual set of classes for OpenOME is large and complex; therefore we present a simplified view by showing only the essential classes that are involved in the evaluation algorithm. Classes shown with a blue background are new to OpenOME, with elements in white existing before the implementation of this algorithm. The Figure shows relationships between classes such as *inheritance*; *import*, here indicating usage; and *instantiation*, here meaning containment as an attribute. Figure 6.2 shows the relationships between classes that are part of the newly added HumanInterventionReasoning Java package, a subset of the classes in Figure 6.1. The classes in Figure 6.1 that are not contained in Figure 6.2 belong to various packages in the pre-existing edu.toronto.cs.ome.editor project.

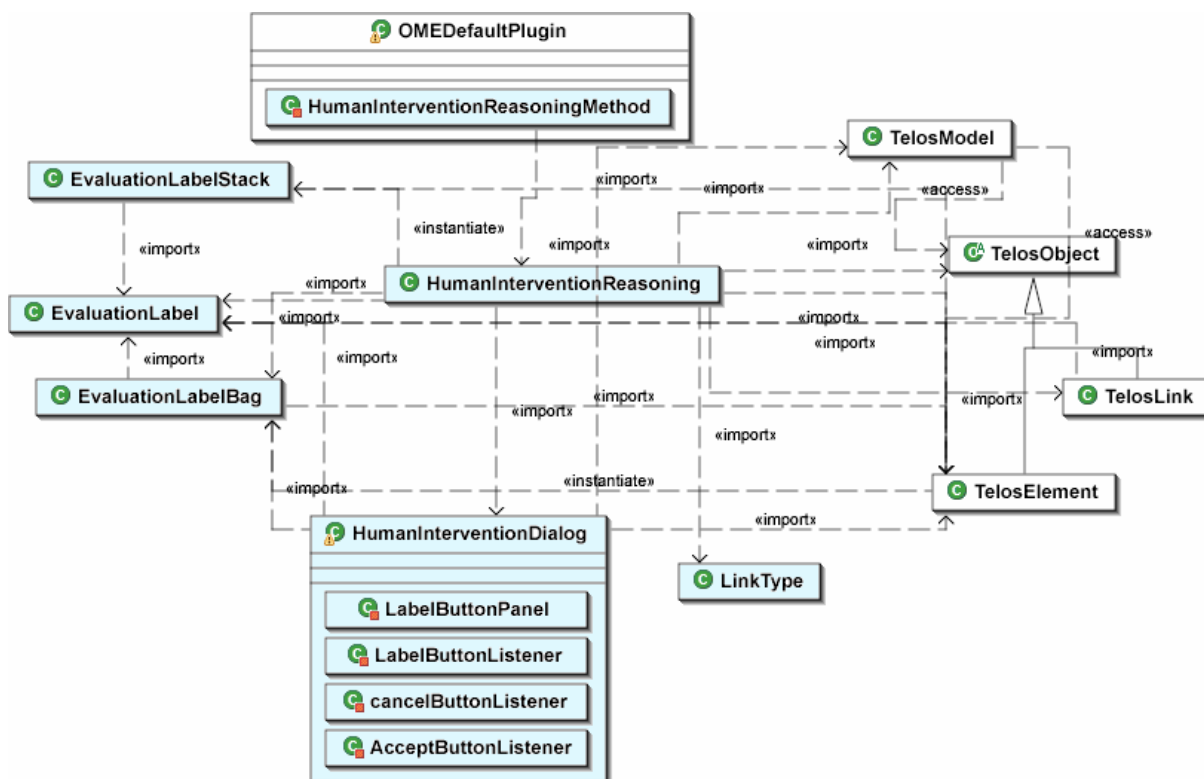


Figure 6.1: Classes Created or Used in the Evaluation Algorithm in OpenOME

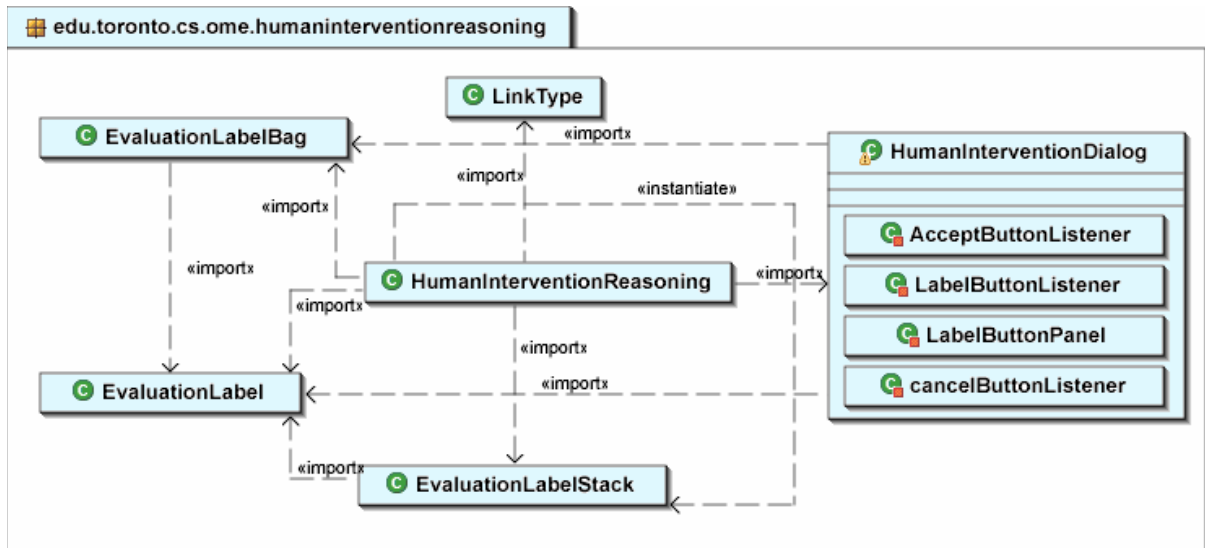


Figure 6.2: New Classes in the New HumanInterventionReasoning Package

In Figures 6.3 through 6.11 we see the expanded version of each class, showing the private and public attributes and methods. In the classes existing before the implementation of the evaluation algorithm the attributes and methods added specifically for the algorithm are shown in blue. We shall provide a brief description of the functionality of each class in the i* evaluation algorithm.

The `HumanInterventionReasoning` Class, shown in Figure 6.3, contains the major functionality of the algorithm. This class corresponds closely to the pseudocode in Figure A.3 of Appendix A, but with the greater detail required for implementation. This class is instantiated in the `OMEDefaultPlugin` Class, where the `propagate` method is called, starting the evaluation. The queue of labels to propagate is contained in a class of type `EvaluationLabelStack`. A list of elements and a reference to the `TelosModel` is maintained.

In Figure 6.4 we see the `EvaluationLabelStack` mentioned above. This class is responsible for storing the queue of labels to propagate. It differs from a typical Java Stack in that the elements are arrays of two objects, an `EvaluationLabel` representing the label to propagate and a `TelosElement` representing the element source of the label.

Figure 6.5 shows the construction of the `EvaluationLabel` class, representing the set of six possible qualitative evaluation labels. As well as the expected methods

referring to the label type, this class contains `isGreaterThan` and `isLessThan` methods to allow comparisons of labels given the ordering described in Chapter 4, Section 4.3.3.4.

Figure 6.6 provides the description of the `EvaluationLabelBag` Class, which is contained within each `TelosElement`. This class is similar to the `EvaluationLabelStack` class in that its objects are `(EvaluationLabel, TelosElement)` tuples. However, it differs in the behavior of object addition and removal, behaving as an unordered bag, as opposed to a first-in first-out queue. In addition, this class provides the functionality to save previous bag states and check for a match with a previous bag state, in order to avoid duplicate human judgment on identical bag contents.

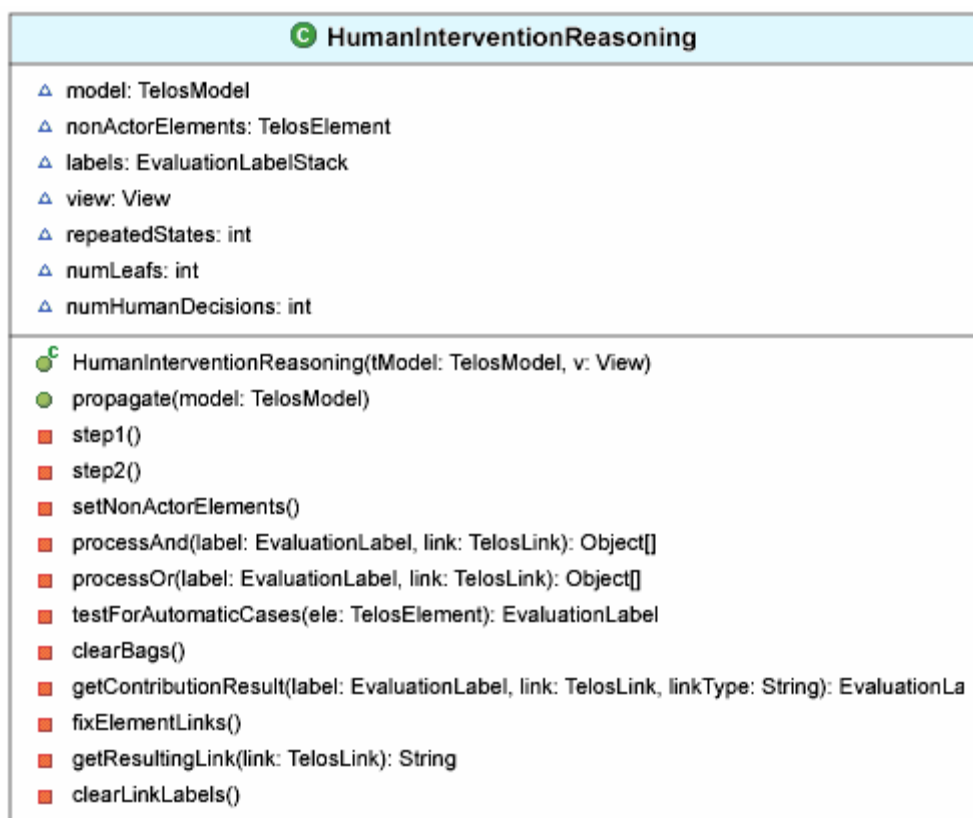


Figure 6.3: The `HumanInterventionReasoning` Class from the Newly Implemented `HumanInterventionReasoningPackage`

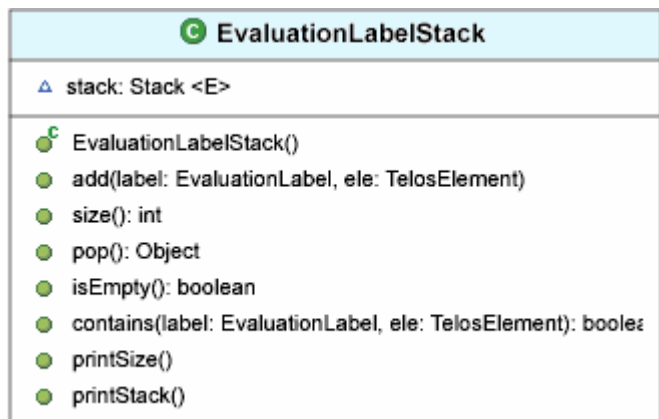


Figure 6.4: The EvaluationLabelStack Class from the Newly Implemented HumanInterventionReasoningPackage

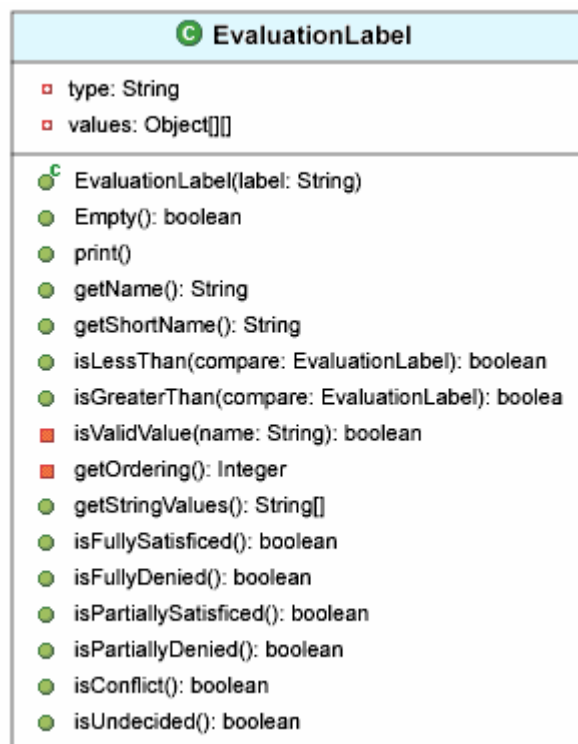


Figure 6.5: The EvaluationLabel Class from the Newly Implemented HumanInterventionReasoningPackage

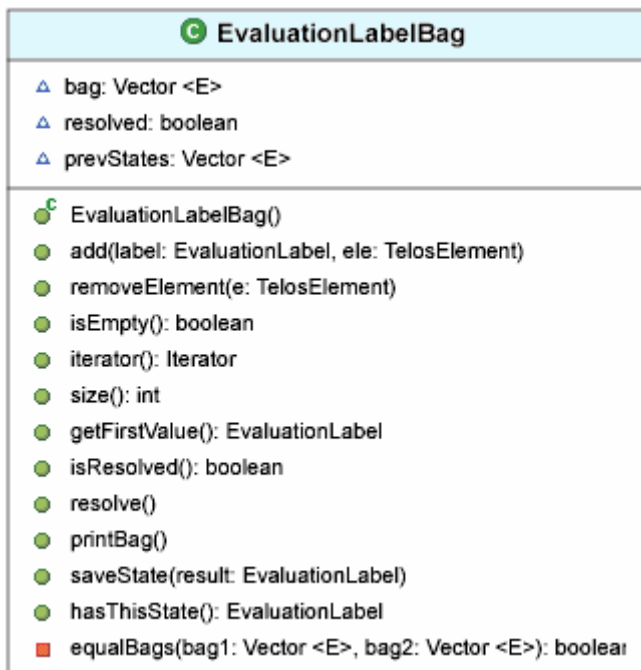


Figure 6.6: The EvaluationLabelBag Class from the Newly Implemented HumanInterventionReasoningPackage

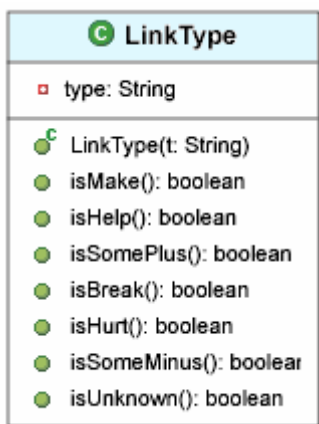


Figure 6.7: The LinkType Class from the Newly Implemented HumanInterventionReasoningPackage

Figure 6.7 describes the `LinkType` class, which mimics the behavior of the `TelosLink` class in confirming or denying its type. The motivation for this class may not be immediately clear. It was added in order to assist in the processing of links to other links. When the strength of a link is modified by another link, a `LinkType` string represents the new strength, as the creation of a whole new `TelosLink` object would modify the underlying model representation.

The `HumanInterventionDialog` Class, shown in Figure 6.8, provides the functionality for creating the GUI window that is used to prompt for human decisions. This class was modified from the evaluation implementation in the original OME application. The Class file contains multiple subclasses in order to deal with dialog events such as the selection of a label, or pressing the cancel or accept buttons. In addition, the `LabelButtonPanel` separately represents the section of the dialog displaying buttons corresponding to evaluation labels.

The `TelosModel` class represented in Figure 6.9 existed as part of the previous OpenOME implementation and was not modified by this implementation. However, as this class holds the general structure of an i^* model, including links and elements, it is used in the `HumanInterventionEvaluation` Class as a means to access the model. The `TelosElement` and `TelosLink` classes described in Figure 6.10 and 6.11 also existed in OpenOME before the current implementation. However, significant additional functionality was added to these classes in order to hold the information required for evaluation, such as the current and previous `EvaluationLabel`, the `EvaluationLabelBag`, and the type of element or link. One may also notice many additional methods added to deal with the insertion and retrieval of links. Although some of these methods were explicitly required for the evaluation algorithm, such as `getLinksTo`, other methods were inserted to correct errors in the default OpenOME implementation concerning the linkage between elements and their links. In future, more stable versions of OpenOME some of this functionality can be phased out.

Finally, Figure 6.12 shows the `OMEDefaultDialog` Class that contains the `HumanInterventionReasoningMethod` Class. As mentioned in conjunction with the `HumanInterventionReasoning` Class, this class is used to instantiate and evoke the evaluation algorithm. This invocation occurs when the OpenOME user presses the Human Intervention Reasoning button in the application GUI.

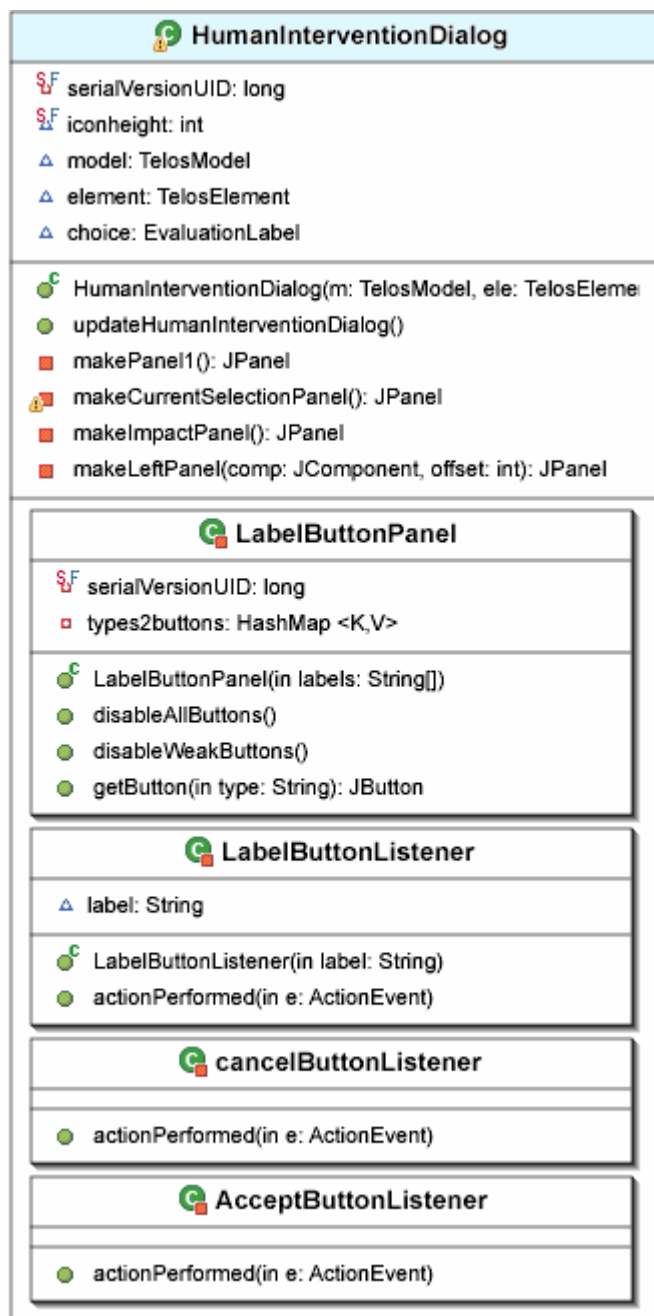


Figure 6.8: The HumanInterventionDialog Class from the Newly Implemented HumanInterventionReasoningPackage

C TelosModel	
<ul style="list-style-type: none"> ▣ kb: KB ▣ elements: Collection <E> ▣ links: Collection <E> ▣ dirty: boolean ▣ elementcount: int ▣ linkcount: int ▣ props2objects: Map <K,V> 	
<ul style="list-style-type: none"> ● C TelosModel(kb: KB, framework: TelosFramework, mm: ModelManager) ● getViewNumbers(kb: KB): int ● getPlugins(): Collection ● createElement(type: Object): ModelElement ● createLink(type: Object): ModelLink ● deleteElement(element: ModelElement) ● deleteLink(link: ModelLink) ● deleteObject(mo: ModelObject) ● getFramework(): OMEFramework ● getElements(): Collection ● getLinks(): Collection ● load() ● save(f: File) ● saveasxml(f: File) ● layout(f: File) ■ unmangleID(s: String): int ● isDirty(): boolean ● getElement(p: Proposition): OMEElement ● getLink(p: Proposition): OMELink ● getObject(p: Proposition): OMEObject ● getModelManager(): ModelManager ● getViewManager(): ViewManager ● saveasvdx(f: File) ● saveasdot(f: File) ● saveasxml(f: File) ● saveasq7(f: File) ● saveasdat(f: File) ● saveasxmi(f: File) 	

Figure 6.9: The TelosModel Class from the Preexisting edu.toronto.cs.ome.model Package

C TelosElement	
<ul style="list-style-type: none"> ▣ children: Collection <E> ▣ prevEvalLabel: EvaluationLabel ▣ initEvalLabel: EvaluationLabel ▣ evalLabelBag: EvaluationLabelBag ▣ EvalResultFromWhichLinkType: String ▣ toLinks: HashSet <E> ▣ fromLinks: HashSet <E> 	
<ul style="list-style-type: none"> ● getModel(): OMEModel ● getType(): Object ● setType(newtype: Object) ● getID(): int ● setID(id: int) ● getLinks(): Collection ● addLink(l: OMELink) ● C TelosElement(kb: KB, framework: TelosFramework, model: TelosModel, id: int, type: Object) ● C TelosElement(kb: KB, framework: TelosFramework, model: TelosModel, id: int, p: Propositio 	
<ul style="list-style-type: none"> ● getEvalLabel(): EvaluationLabel ● setEvalLabel(eLabel: EvaluationLabel) ● getElementType(): String ● getLinksFrom(): HashSet ● getLinksTo(): HashSet ● isLeaf(): boolean ● setInitialEvalLabel(eLabel: EvaluationLabel) ● getInitialEvalLabel(): EvaluationLabel ● setPreviousEvalLabel(eLabel: EvaluationLabel) ● getPreviousEvalLabel(): EvaluationLabel ● addToBag(label: EvaluationLabel, element: TelosElement) ● removeFromBag(e: TelosElement) ● isLabelBagEmpty(): boolean ● getLabelBag(): EvaluationLabelBag ● emptyLabelBag() ● printAllLinks() ● hasLink(link: TelosLink): boolean ● addLinkTo(link: TelosLink) ● addLinkFrom(link: TelosLink) ● setEvalResultLinkType(st: String) ● getEvalResultLinkType(): String ● isSoftgoal(): boolean ● isActor(): boolean ● labelBagStateHasOccured(): EvaluationLabel ● labelBagStateResolved() 	

Figure 6.10: The TelosElement Class from the Preexisting edu.toronto.cs.ome.model Package

C TelosLink	
▣	evalLabel: EvaluationLabel
●	getModel(): OMEModel
●	getType(): Object
■	toProposition(type: Object): Proposition
●	setType(newtype: Object)
●	getLinks(): Collection
●	addLink(l: OMELink)
●	getID(): int
●	setID(id: int)
●	getTo(): OMEObject
●	setTo(e: OMEObject)
●	getFrom(): OMEObject
●	setFrom(o: OMEObject)
●	TelosLink(kb: KB, framework: TelosFramework, model: TelosModel, id: int, type: Object)
●	TelosLink(kb: KB, framework: TelosFramework, model: TelosModel, id: int, p: Proposition)
●	getLabel(): TelosParserIndividual
●	setLabel(newname: TelosParserIndividual)
●	isDependency(): boolean
●	getLinkType(): String
●	printLink()
●	isContribution(): boolean
●	isAndContribution(): boolean
●	isOrContribution(): boolean
●	isMake(): boolean
●	isHelp(): boolean
●	isSomePlus(): boolean
●	isBreak(): boolean
●	isHurt(): boolean
●	isSomeMinus(): boolean
●	isUnknown(): boolean
●	isMeansEnds(): boolean
●	isDecomposition(): boolean
●	setEvalLabel(label: EvaluationLabel)
●	getEvalLabel(): EvaluationLabel

Figure 6.11: The TelosLink Class from the Preexisting edu.toronto.cs.ome.model Package



Figure 6.12: The OMEDefaultPlugin Class from the Preexisting edu.toronto.cs.ome.controller Package

Figure 6.13 shows a screen shot of the OpenOME application with a model evaluation in progress. The HumanInterventionDialog is shown prompting the user for a decision concerning the element Increase [Profit from Visitors]. The contributing labels and their sources are shown within the dialog. The user then selects the button corresponding to their decision and clicks Accept. Ideally, the graphical version of the evaluation labels should be displayed instead of the text name, but this implementation detail has been left to future versions. At the top of the application, one can see button

icons containing arrows, (circled in red). The third arrow, pointing in the up direction and drawn with a dotted line, invokes the evaluation procedure described in this chapter.

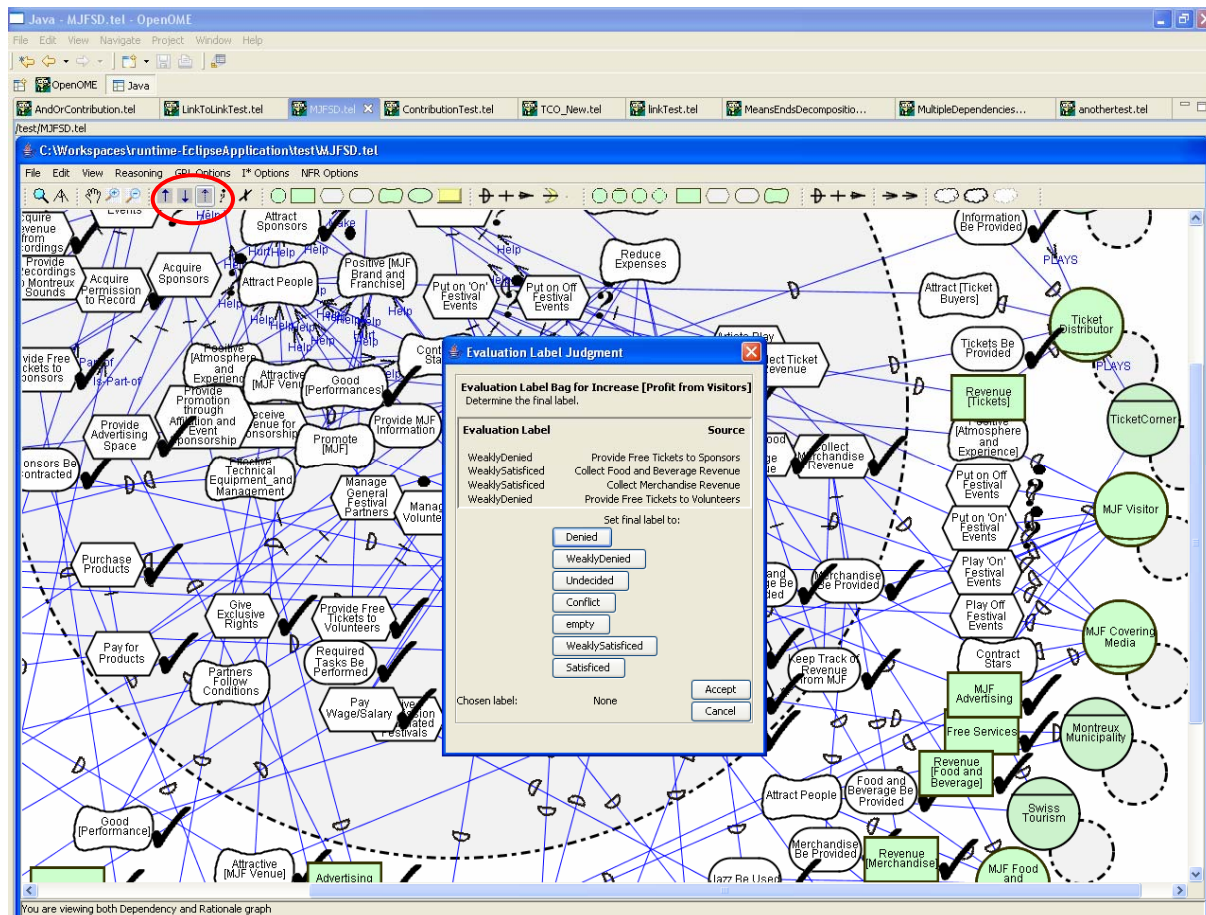


Figure 6.13: A Screenshot of the OpenOME Implementation of the i*Evaluation

6.3 Issues Discovered During Implementation

During the implementation of the evaluation algorithm a number of unforeseen issues arose, causing us to make minor adjustments to the algorithm outlines in Chapter 4. For instance, the rules laid out in Chapter 4 for the algorithm assume that the rules of i* syntax are followed; however, OME and OpenOME allow some flexibility with these rules. Therefore, the treatment of unconventional i* syntax in the evaluation algorithm must be considered. For example, what if contribution links have a non-softgoal as their destination? In this particular case, although we have stipulated that “hard” elements will not have label bags, we bend this rule in order to allow the collection of multiple labels.

In effect, “hard” elements in this situation are treated as softgoals. An example of this situation is shown in Figure 6.14 in an excerpt from a large model in the Trusted Computing domain. Here Control of Technology Be Obtained, a hard goal, is the destination of correlations links. Likely this situation arose as an error; however, the algorithm should deal with such errors gracefully. Similarly, if a means-ends link is made to a task, or a decomposition link to a goal, the resolution of these links will be added to the target element. Effectively, the algorithm uses the element types implied by the link type before the actual type of the target element.

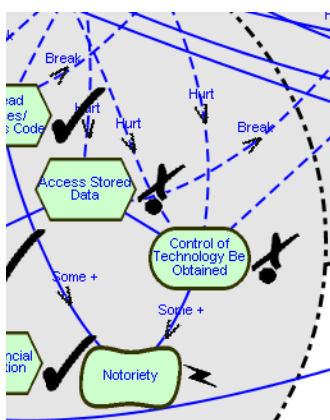


Figure 6.14: Example of Contribution Links to “Hard” Element

An additional issue became apparent when implementing the storage of previous human judgment situations. This functionality was added in order to attempt to reduce the amount of times the application prompts for human judgment, as mentioned in Chapter 4, Section 4.3.5. When a decision is made for an element, given a certain set of labels from a certain set of sources, this decision and the contents of the label bag are stored. Upon future decisions for that element, the stored label bags are checked to see if they match the contents of the current bag, in which case the previous decision is used without prompting the user. However, it has become apparent that in certain situations this implementation results in an infinite loop. Take, for example, a cycle where evidence is inverted, such as in Figure 6.15, repeated from Chapter 4. If an initial label of satisfied is given to softgoal B, human judgment is prompted for B with a label bag containing satisfied (the initial label) and partially denied. If the user chooses partially denied as a resulting label, the next iteration will prompt for human judgment of softgoal B with a label bag of satisfied and partially satisfied. If at this point the user chooses

satisfied, the next human judgment for softgoal B will have already occurred, resulting in partially denied; and the following human judgment will have also have already occurred, producing a result of satisfied. As a result, the application will enter an infinite loop, continually flipping the evaluation values based on stored human judgment situations.

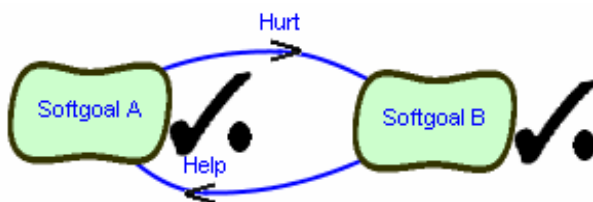


Figure 6.15: An Example Cycle which may Result in an Infinite Loop if Previous Human Judgment is Stored

However, if previous human judgment situations are not used to resolve successive judgments, there is a hope of avoiding an infinite loop, as the user will be continually prompted for a decision concerning softgoal B. After a few iterations it will become obvious that the decisions made do not converge, at which point the user will likely revise one of the decisions or exit the evaluation altogether, (likely prompting an adjustment in the model). As a result of the potentially dangerous situation caused by applying previous judgment situations automatically, this feature has been removed from the current implementation. In Section 6.4, we will consider the topic of reducing human judgment more closely, assessing whether storing previous judgment situations provide a significant reduction in the prompting of users.

During implementation, a mechanism to reduce the need for human judgment, which was not previously included in the Chapter 4 algorithm, became apparent. Namely, an evaluation bag is marked as “decided” once a decision has been prompted for. When the nodes containing non-empty bags are iterated on, only the “undecided” bags prompt for human judgment, with the assumption that the decided bags have already placed their decision into the label queue. When the contents of the bag are modified, the bag is again marked as “undecided”. The pseudocode in Appendix A, Table A.3 now contains this feature.

When describing the specifics of the i^* evaluation procedure in Chapter 4, Section 4.3, we presented the propagation rules as recommended guidelines, allowing adjustment in cases where the evaluator judges it to be necessary. As a result, our evaluation implementation should also allow for such adjustments. However, as it is difficult to foresee the nature of these adjustments, in the current implementation disagreements with the conventional rules of propagation have to be propagated manually. This can be done by explicitly changing the evaluation values individually for specific nodes. Unfortunately, the nature of the current implementation does not allow these changes to be propagated in conjunction with previous evaluation results. In other words, once an evaluation is completed, successive evaluations are unaware of the results or process of the previous evaluation, meaning that all results of a previous evaluation are considered as initial labels in a new evaluation. The algorithm is currently not able to propagate the impact of a single change. A future beneficial feature may allow propagation starting from a single specified node, facilitating these types of isolated changes.

6.4 Testing

In order to ensure that the OpenOME implementation of the evaluation algorithm is sound, sufficient testing is required. During the development individual pieces of functionality were tested as they were created. For example, propagation through each type of link, propagation when a mixture of links is present, correct label bags contents in human judgment situations, and propagation through links to other links. Figure 6.16 shows an example model used to test for the correct propagation through contribution links, the prompting of human judgment, and the convergence and termination through circular models.

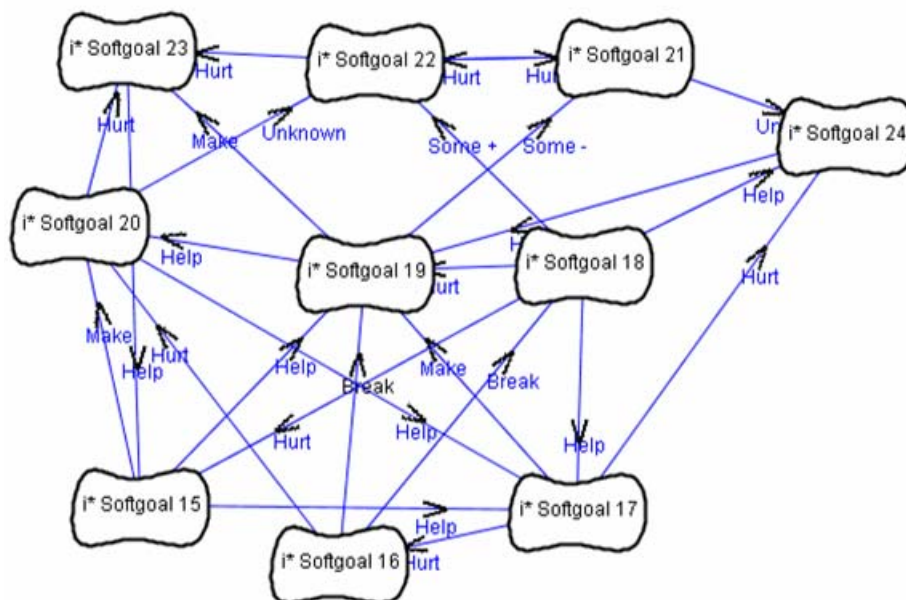


Figure 6.16: Example Model used to Test for the Correct Functionality of the Evaluation Algorithm

Once the performance on these individual “unit” tests was correct and the algorithm was complete, more extensive tests were performed by applying the implementation to example models from the i* evaluation case studies, which shall be described in Chapter 7. The code was instrumented to record the elapsed time for each evaluation, as well as the number of times the algorithm prompted for human judgment. It should be noted that the duration of an evaluation depends heavily on how long it takes for a user to resolve label bags. In this case, since the user evaluating the model was the same person who constructed the model, such decisions came relatively quickly. If the user were unfamiliar with the model, it would likely take more time to evaluate than the results reported here.

As the feature which stored human judgment situations that had previously occurred had been implemented and disabled, the algorithm was still able to report how many times an identical human judgment situation was repeated. Finally, the results of the evaluation were analyzed manually to determine if evaluation procedure rules were followed in the propagation. In the first and second examples, the results were compared to results from a manual application of the evaluation procedure, given the same human judgment decisions. In the third example, manual evaluation of the model was

considered too difficult a task to perform, given the difficulty in visually tracking the links.

The first application to a real life model was performed on the small model example from Chapter 4, repeated in Figure 6.17. This particular model has 16 elements, 2 of which are leaf nodes. Evaluation, as predicted in Chapter 4, Section 4.5, required two requests for human judgment. The entire evaluation took 11.6 seconds. As the time between human judgment promptings seemed minimal, one may assume that this indicates an average of approximately five seconds per judgment for this particular model. In addition, the evaluation results corresponded to the results from manual algorithm application.

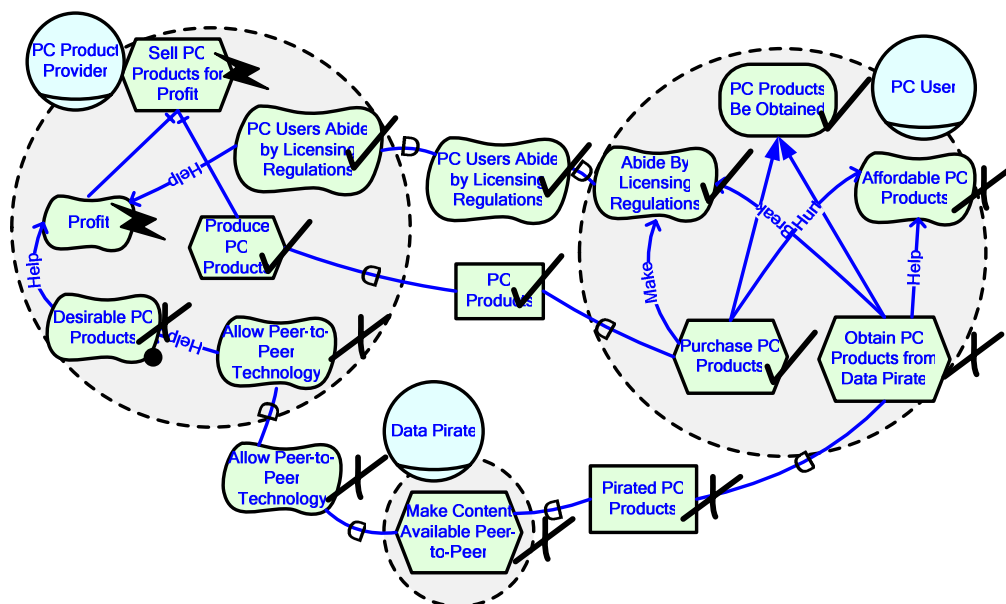


Figure 6.17: Example Trusted Computing Model Used to Test the Evaluation Procedure Implementation

The first large example used to test the implementation also comes from the Trusted Computing domain. The OME version of this model is shown in Figure 6.18, (OpenOME does not allow exporting to images). This particular model contains 109 elements, 22 of which are leaf nodes. This model serves as a particularly interesting test of evaluation due not only to its large size, but also to the presence of multiple actors and the cross-actor attack and defense situations resulting in complex contribution links. As explored in the initial value section of Chapter 4, there are an exponential number of

possible evaluations that could be performed on such a model. In this case, we chose to mark all leaf goals as satisfied, except for the unreciprocated dependency of Avoid Lock-In, which was marked as denied. This evaluation was performed several times, taking a range of time from 4 minutes 15 seconds to 6 minutes 46 seconds. The algorithm prompted for human intervention 45 times, resulting in an approximate average of 7seconds per human judgment. One situation with identical human judgment arose. However, multiple judgments with differing bag contents arose for elements higher in the graph such as Profit in the License/Copyright Holder, prompted for three times; Desirable [Technology], prompted for twice; and Desirable to Technology Users [Technology], prompted for twice. Upon examining the manual application of evaluation employing the same human judgment decisions, it was found that the results would have matched exactly if not for a dependency link which was pointing in the wrong direction. It is interesting to note that this was not noticed in the manual evaluation, but became apparent upon application of the semi-automatic evaluation procedure, when the results were unexpected. This is in contradiction to the idea put forth in Chapter 3 and Chapter 4 that the manual application of the evaluation procedure is more likely to result in the correction of model semantic errors. Perhaps this is an indication that there exist situations where such errors are not visually obvious, and therefore better detected via the results of an automated algorithm.

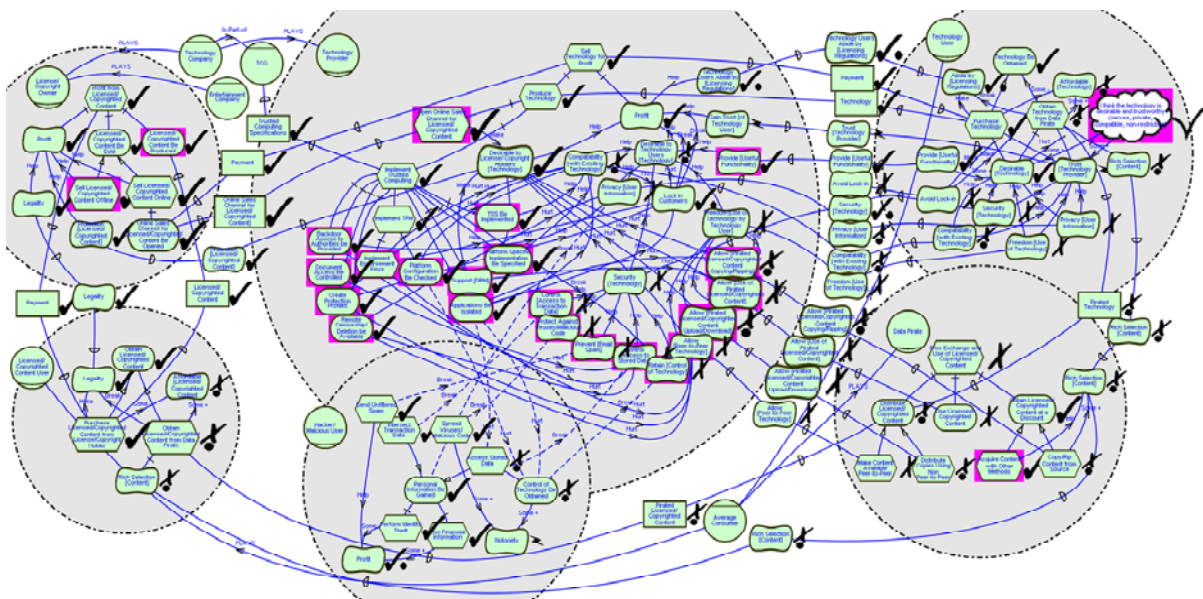


Figure 6.18: Trusted Computing Example used to Test the Evaluation Implementation

The next test of the evaluation implementation on a large model was performed on a model from the Montreux Jazz Festival (MJF) case study containing 111 nodes, 15 of which were leaf nodes. The OME version of this model is shown in Figure 6.19. This model is particularly useful in testing an evaluation situation where multiple input and output values come from actors whose internal rationale is not displayed. In this case we chose the initial values to be the 15 leaf nodes, as well as the incoming dependums. Outgoing dependums were left without labels. A particular evaluation of this model took 1 minute 6 seconds to complete. The algorithm prompted for human judgment 14 times, with one repeated bag state. As in the previous example, resolution was prompted for higher-level elements multiple times. Profit, for example, was the subject of human judgment four times, comprising 29% of the human judgment required for the model. This model serves as an example of the usefulness of evaluation automation. As one can imagine from examining the model in Figure 6.19, manual evaluation would be especially problematic given the difficulty in tracing links. However, with partial automation, the procedure was completed in little more than a minute.

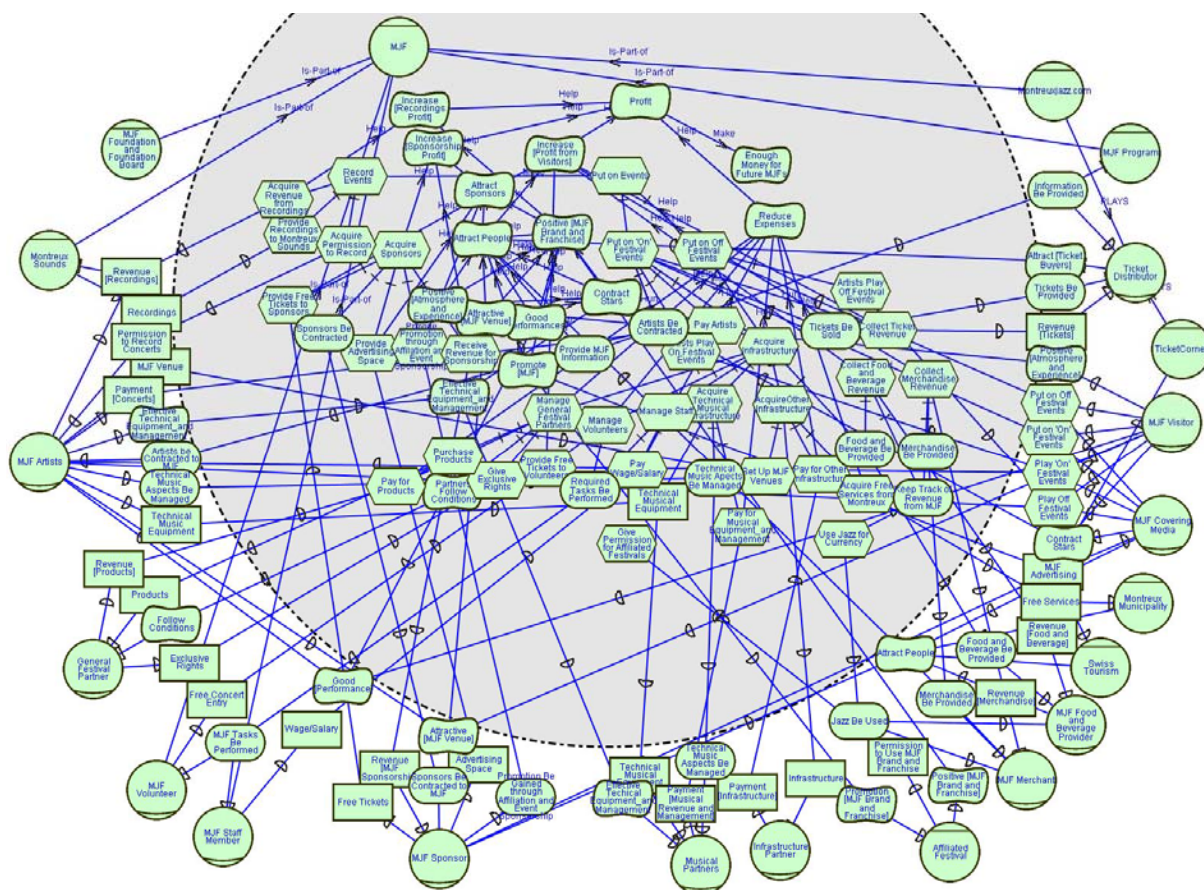


Figure 6.19: Example MJF Model used to test the Evaluation Procedure Implementation

Ideally, we would test the implementation of the evaluation algorithm on the models from the Kids Help Phone domain, as they provide the largest examples of i^* models available to us. However, these models were created using Microsoft Visio, and not OME, due to the difficulty OME has in scaling to very large models. Although work has been done to facilitate the importing of Visio models saved in XML format to OpenOME, this feature is not yet stable enough to use reliably.

6.5 Reducing the Need for Human Judgment

As mentioned, the primary factor contributing to the time taken for semi-automated model evaluation is the need for human judgment. Although it is difficult to reduce the time an individual may take in making a domain-knowledge-based decision,

we can attempt to reduce the number of times human judgment is required in the implementation of the evaluation procedure.

We have previously explored one potential way to accomplish this, by saving states of human judgment for each element and using the saved results to avoid prompting for the same answer. However, as mentioned in Section 6.2.1, this implementation brings the risk of an infinite evaluation loop. In addition, the examples in the previous statement seem to demonstrate that identical situations do not arise frequently. In each model, with 45 and 14 human judgment situations, only one was identical to a previous situation in each case. Repeated bag states are likely to occur when evaluation values fluctuate from one value to another. A decision is made for a particular element in one iteration, and then in the next iteration evidence changes resulting in a new decision, then, if the evidence reverts back to the previous state, the same decision must be made again. However, this is precisely the situation where the application is in danger of going into an infinite loop. Until, in future work, a method is devised to avoid this danger; the benefits provided by the implementation of this feature do not appear to be worth the risks.

As mentioned in Section 1.2.1, the current version of the algorithm includes another method in order to reduce the amount of human judgment required. An evaluation bag is marked as “decided” once a decision has been prompted for, with only the “undecided” bags prompt for human judgment. The bag is marked as “undecided” when the contents of the bag are modified. The usefulness of this feature depends on the specifics of the evaluation algorithm implementation. Concerning the application of algorithm step 1, where labels are propagated; and step 2, where bags are resolved; two ordering options exist. After all labels are propagated in step 1, step 2 could resolve only a single bag before it returns to step 2, where the results of this bag are propagated. This would result in a cycle where, after the propagation of initial labels, step 1 and 2 propagate and resolve only one label in each iteration. Alternatively, after all labels are propagated in step 1, all pending label bags could be resolved in step 2. These options correspond roughly to a depth first and breadth first propagation, respectively.

In the first option, the implementation of the “decided” feature is essential, although the termination clause of not re-propagating the same label as previous will

prevent the propagation of the same decision more than once; the same label bag may be resolved each time in step 2. In the second option this feature is still helpful, as a decision may be prompted for multiple times, even if step 1 did not result in changes to its label bag. As a result, the “decided” feature will remain as an important aspect in the implementation.

However, we have yet to determine which of the step 1 and 2 ordering alternatives performs optimally. The test results presented in the previous section correspond to an implementation of the breadth first method, where all possible label bags are resolved during each iteration of step 2. If the alternative, breadth-first option is used in the evaluation of Figures 6.18, the number of user decisions remains at 45. However, the number of identical bag states the user is prompted for rises from 1 to 5. In Figure 6.19, the number of user decisions with the second option rises from 14 to 15, with the number of identical user decisions remaining at one, and the Profit softgoal still being the object of judgment four times. These limited tests seem to show little difference between the two implementation options, with the exception of a significant rise in identical judgment situations for Figure 6.18. This seems to indicate that if the possibility of an infinite loop is somehow mitigated, and the avoidance of identical judgments is re-implemented, the number of human judgments may fall. For now, as the majority of i^* models do not grow beyond 100 elements, the efficiency differences caused by these options do not seem overly significant. As a result, a more in-depth comparison of these implementation options is left to future work.

Multiple decisions for the same higher-level elements, as shown in the results of the previous section, may indicate an even greater potential for optimization. In an ideal implementation human judgment for a particular element would only be prompted for once. In order to accomplish this, the algorithm would have to “wait” until all children contributing to a decision contain a value. For example, the user would not be prompted to resolve the Profit softgoal in Figure 6.19 until values for all four of its children are present. However, in models with circularity this may never occur, as the child results for one element may depend on resolving another element whose children have not yet received complete evidence. In addition, a child element may not ever receive a value other than unknown, due to an incomplete coverage in initial values. Furthermore, how

does the algorithm know if a value for a child element is the final value? Perhaps the current value may be updated upon further propagation. Overall, an implementation that accomplishes this ideal behavior is far from trivial.

However, one can think of implementation options that may partially accomplish this behavior. For example, human judgment could not be prompted for while child evidence is missing, but only while there exists other candidates for judgment and subsequent propagation. In other words, if all label bags are waiting for further evidence, one such bag shall be chosen as a candidate for resolution based on incomplete evidence, in the hope that the results propagated from this resolution will fill in the missing evidence for some other label bag. Although this procedure is feasibly implementable, it is difficult to determine how effective it will actually be in reducing the amount of human judgment. Presumably the results depend highly on which bag is chosen as a candidate for a human decision with incomplete evidence. Yet it seems that it would be difficult to determine heuristics for this selection without a detailed examination of the model structure. The implementation and testing of this and other possible algorithm adjustments for the purpose of optimization is left for future work.

6.6 Conclusion

The implementation of the i^* evaluation algorithm described in this work as part of the OpenOME software demonstrates its practical potential as a tool for model analysis. As the OpenOME project has potential for eventual widespread use, we can hope that this implementation will make the procedure widely available. This may increase the use of i^* model evaluation in situations where i^* modeling is currently being used, and may influence potential future users to employ the framework due to the presence of a viable reasoning support tool.

Chapter 7: i* Evaluation Case Studies

7.1 Introduction

As mentioned throughout this work, the desired qualities for and details of the evaluation procedure were derived from the application of i* and evaluation to multiple examples. In order to demonstrate the viability of i* evaluation algorithm described in this work for projects involving real-life elements, we shall return to five case studies involving examples of i* modeling and evaluation, and test the procedure developed in this work against these studies. Extensive case study application will help to demonstrate that the i* evaluation procedure described in Chapter 4 captures the desired qualities of Accuracy (iv), by showing that the domain specific evaluation results were sufficiently accurate to provide a useful analysis, and Usefulness in Multiple Contexts (v), by showing the successful application of evaluation in a variety of domains. The description of such studies will effectively illustrate the analytical capabilities of the i* evaluation procedure.

7.1.1 Discovering Issues in i* Modeling and Evaluation

As the specifics of the i* evaluation procedure were still under various degrees of development during the execution of each case study, various issues and challenges were uncovered and addressed using a range of solutions. The application and degree of success of such solutions worked towards the formation of the specifics of the algorithm described in Chapter 4 and implemented in Chapter 6. However, some of the problems observed are not yet sufficiently addressed by the evaluation work in this study, and are therefore identified as potential areas of future research. In addition, issues were discovered with the expressive capabilities of the Framework itself, often having implications on evaluation. As we identify various **i*** and **evaluation issues** and their potential solutions in the context of case studies, we will highlight them using **bolded titles**. The issues, challenges, and potential solutions that were identified in the application of evaluation to the various case studies include:

- **Blind Tests for Evaluation Accuracy**
- **Actor Multiplicities**
- **Circular Evaluation**
- **Exclusive vs. Inclusive Or**
- **Links to Other Links**
- **Unknown vs. No Label**
- **Iteration of Evaluation Values**
- **Qualitative Expressiveness**
- **Representing Events**
- **Capturing the Process of Changing Evaluation Values**
- **Model Views Hindering Evaluation**
- **Contributions from a Mixture of Link Types**
- **Repetition of Softgoals across Actors**
- **Marking Initial Values**
- **Propagation Rules as Guidelines**
- **Semantic Improvement**
- **Model Size Hindering Evaluation**
- **Model Slices**
- **Long Contribution Paths**
- **Evaluation of Scenarios**
- **Evaluation Metric**
- **Model Layering**
- **Tables to assist Human Judgment**
- **The Affects of Link Completeness**

Unfortunately, the implementation of the evaluation algorithm was not available at the time when each Case Study was completed; therefore the lessons learned centre mainly on the manual application of the i* evaluation procedure. In the future, we hope to complete a further Case Study in the Kids Help Phone domain using the OpenOME evaluation implementation, thereby testing its practical feasibility.

7.1.2 Exploratory Case Studies

Regarding our use of the phrase “Case Study”, recent work has urged a more formal, empirical meaning for studies of this type, having a predefined objective and experimental design (Kitchenham, Pickar, & Pfleeger, 1995). In this light, the first four of our studies could be considered as “exploratory” case studies, where we explore the viability of the application of i^* and the evaluation procedure to real-life phenomena. However, these studies lacked the explicit experimental design required by the empirical notion of a Case Study. The Kids Help Phone study can also be considered exploratory, although one section of this study intended to test a more specific pre-defined theory concerning viewpoint modeling (Easterbrook et al., 2005). This study can be more accurately called a Case Study in the empirical sense, as it involved a careful experimental design.

7.1.3 Overview of the Case Studies

We first explore the “Privacy in E-Commerce” study, demonstrating the ability of i^* to portray the current privacy concerns in E-Commerce, including alternative solutions to these problems. Next, we look at the “Economic Information Security” study, which attempts to express and evaluate economic patterns. The “Montreux Jazz Festival” study depicts the complex interactions between multiple actors involved in this large festival, evaluating the effects of various stakeholder choices on overall business success. The “Trusted Computing” study explores the actors involved in the Trusted Computing debate, depicting technical attack and defense situations. This case study explores the ability of i^* and the i^* evaluation procedure to depict the differences between conflicting viewpoints. These first four studies utilized various existing documentation as sources, while the last study, the Kids Help Phone Study, is based on evidence collected from a requirements elicitation project with a real-life non-profit organization. In this study we push i^* modeling and i^* evaluation to its full capacity in terms of model size, analyzing the effects of viewpoint modeling, and exploring the conversion of i^* models into artifacts used in a software requirements specification.

7.2 Privacy in E-Commerce

This particular study is based primarily on a single source, *User Agents in E-Commerce Environments: Industry vs. Consumer Perspective on Data-Exchange* by Spiekermann, Dickinson, Günther, & Reynolds (2003). Additional ideas concerning the representation of security, privacy and trust in i* models were adopted from the work of Yu et al in sources such as (Yu et al., 2003) and (Yu et al., 2001). The work of Spiekermann provides an interesting example of phenomena involving multiple actors, including multiple possible solutions to problems involving the introduction of new actors and the shifting of responsibilities. As a result, this paper offered an excellent opportunity to test i*'s ability to accurately model these situations and solutions, with the aim of determining whether the conclusions reached by the models matched those in the paper, where a mismatch would indicate either deficiencies in the modeling framework or flaws in the paper. Although this study is on a small scale, the conclusions are not obvious, providing a potentially interesting example for evaluation.

When this study was originally executed, analysis using an evaluation procedure was not performed on the models. This initial lack of evaluation makes this study a prime candidate to demonstrate the changes to models prompted by evaluation. In Chapter 5 we have used a model from this study to serve as an example of the syntactic and semantic changes prompted by evaluation. Here, we provide a brief background on the study in order to place this model and its contents in the context of the domain, and we use the results of this evaluation to demonstrate the contextual analysis capabilities of i* evaluation.

7.2.1 Privacy in E-Commerce: The Current Situation

The paper explored the situation involving online corporations and their need for consumer personal information in order to perform directed marketing, both internally, through service differentiation, product differentiation and price discrimination; and externally through targeted advertising, selling and sharing customer profiles. Often, in order to receive products online, it is explicitly necessary for consumers to provide

personal information. However, online consumers, with the help of privacy advocates, want to protect the privacy of their personal information, and specifically avoid detrimental marketing, such as price discrimination against a consumer, while maximizing beneficial marketing, such as price discrimination in favor of consumers. The paper first describes the current situation, and then presents two potential solutions. In the current situation, the online consumer is totally dependent on the online corporation for protection of its privacy, a situation which is problematic for consumers. We have developed an i* model, (Figure 7.1, repeated from Chapter 5), which attempts to capture this situation.

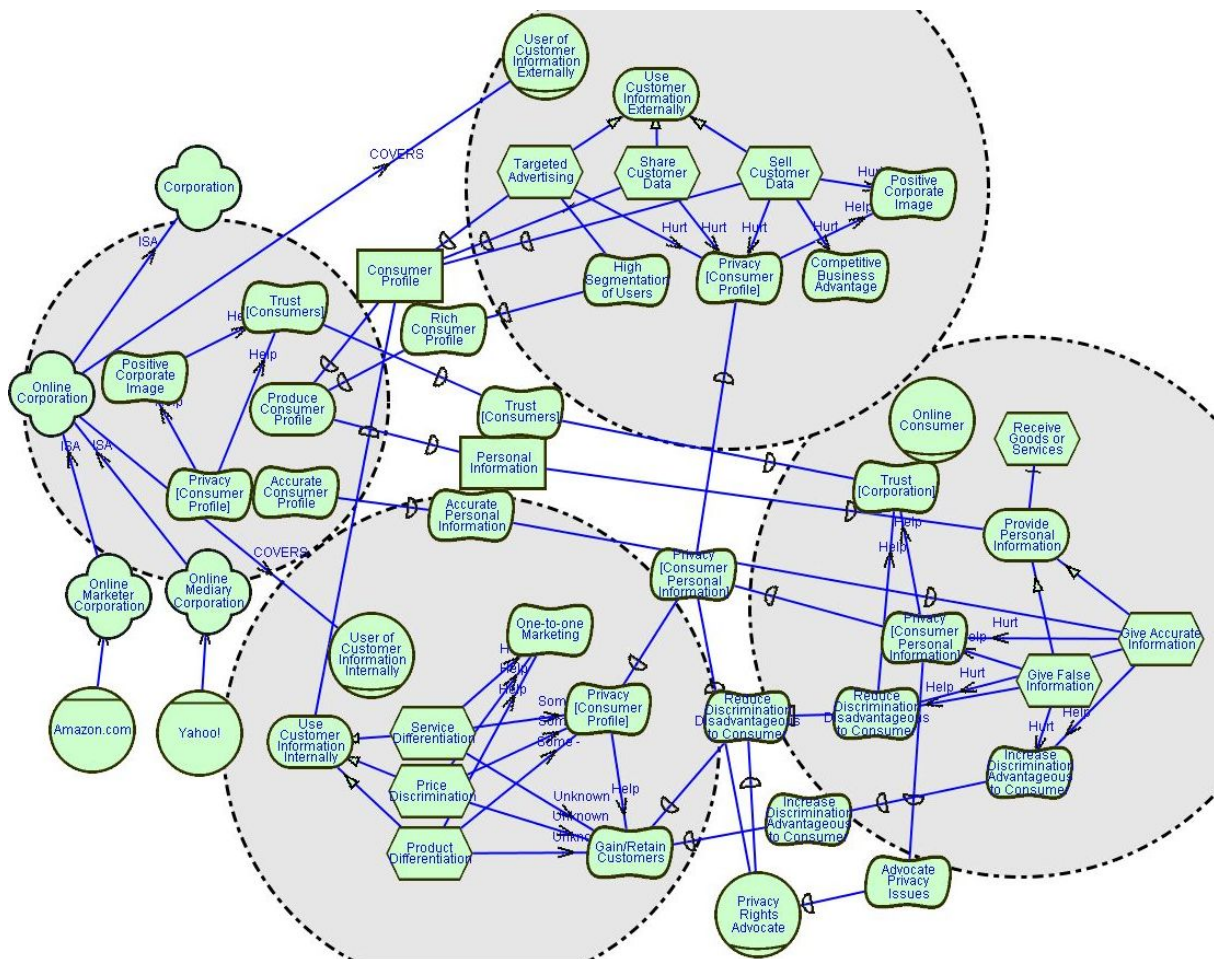


Figure 7.1: From the Privacy in E-Commerce Case Study Representing the Situation before Potential Solutions to Privacy Problems

7.2.2 Potential Solutions to Protect Privacy

The paper then goes on to explore two potential solutions to the problems concerning the privacy of personal information for online consumers. First, it explores the possibility of using a pseudonym, in either a long or short-term situation, as is often done when consumers provide false information. This solution creates problems when it comes to physical product delivery, and in the acquisition of beneficial marketing for short-term pseudonyms. This situation is modeled in Figure B.1, included in Appendix B, as the evaluation procedure had not been applied. The second, novel solution proposed by the paper involves the use of privacy-protecting agents who provide personal information to the corporations on behalf of the consumer, depending on an identity chosen by the consumer, as well as an analysis of the privacy policy of the particular corporation. This solution is modeled in Figure B.2 of Appendix B. In addition to the concerns suggested by the paper, a further model was created to explore the effects of the proposed solution on security. However, this model, included in Appendix B as Figure B.3, was only created in SD form.

7.2.3 Discussion: Demonstration of Changes Prompted by Evaluation

In general the conclusions reached by the models concerning the viability of these solutions matched the conclusions reached by the paper, with the agent idea providing a potentially viable solution to the consumer privacy problems. However, as we have mentioned, an evaluation of model satisfaction was not performed, due to unfamiliarity with evaluation procedures at the time the case study was completed. We can return to these models and evaluate them, paying particular attention to the changes made, such as with Figure 7.1 in Chapter 5. In Figure 7.2 we repeat Figure 5.17 from Chapter 5, showing the results of evaluation for the model in Figure 7.1 after the extensive changes prompted by evaluation.

This adjusted model supports the claims of the Spiekermann paper concerning the privacy concerns in the current e-commerce situation, as the elements representing privacy and trust are denied. One could argue that while performing iteration on the

model, we changed the model, (either consciously or subconsciously), to deliberately produce results which coincided with the conclusions of the paper. This phenomenon is actually quite likely; however, we argue that as the modeler iterates on the model in order to make the model conform to his/her perception of the domain, the modeler will run into interesting questions which force him/her to question domain knowledge. In this case, we may be prompted to ask interesting questions not covered by the Spiekermann paper, such as: “What does it mean for a Consumer Profile to be Rich?” and “Do all ways of using Customer Information Externally depend equally on these qualities?” In a case with real stakeholders, we would use these questions to frame further elicitation. In general, we argue that if the analysis results that the modeler is shaping the model to coincide with are flawed, the modeler will have a very difficult time producing these results in the model, unless inaccuracy is built into the model. As we have described in Chapter 5, the application evaluation can help to identify such semantic flaws when localized results do not match knowledge of the domain; however if the model is not accurate and evaluation results do not prompt iteration, as they are not surprising, there are possibilities for erroneous conclusions. Therefore the accuracy of the evaluation is dependent on the ability of the modeler to detect model inaccuracies through model analysis or stakeholder validation.

Blind Tests for Evaluation Accuracy. In this particular case it would have been more effective if we were able to model the concepts in the domain without being aware of the conclusions the authors make concerning each proposed solution. Then, after the models had been evaluated, we could attempt to confirm both the accuracy of our evaluation procedure and the accuracy of the conclusions by comparing our evaluation results to the conclusions made in the paper. If these results did not match, and we are confident our analysis results are correct, we could potentially use our models to point out flaws in the arguments of the paper. Conversely, if we were confident that the conclusions of the paper were correct, we could try and determine whether the mismatch was a modeling error or a limitation of i^* or the evaluation procedure. This type of useful test is left for future investigation.

If the results of this Case Study were to be readdressed extensively, this process of evaluating and model iteration, would be repeated on the additional Case Study models

in Appendix B which explore the potential solutions, in order to further confirm that the results of evaluation match the results in the paper.

Although the models in the studies to follow underwent significant changes during the application of evaluation, we were unfortunately not meticulous enough to store the “before” and “after” versions of such models, in order to examine and quantify the changes made. In fact, we made an effort to maintain consistency across different models, propagating changes in one model to other models containing the same information. As a result, those studies do not provide good examples of the iteration prompted by evaluation, such as is provided by this study.

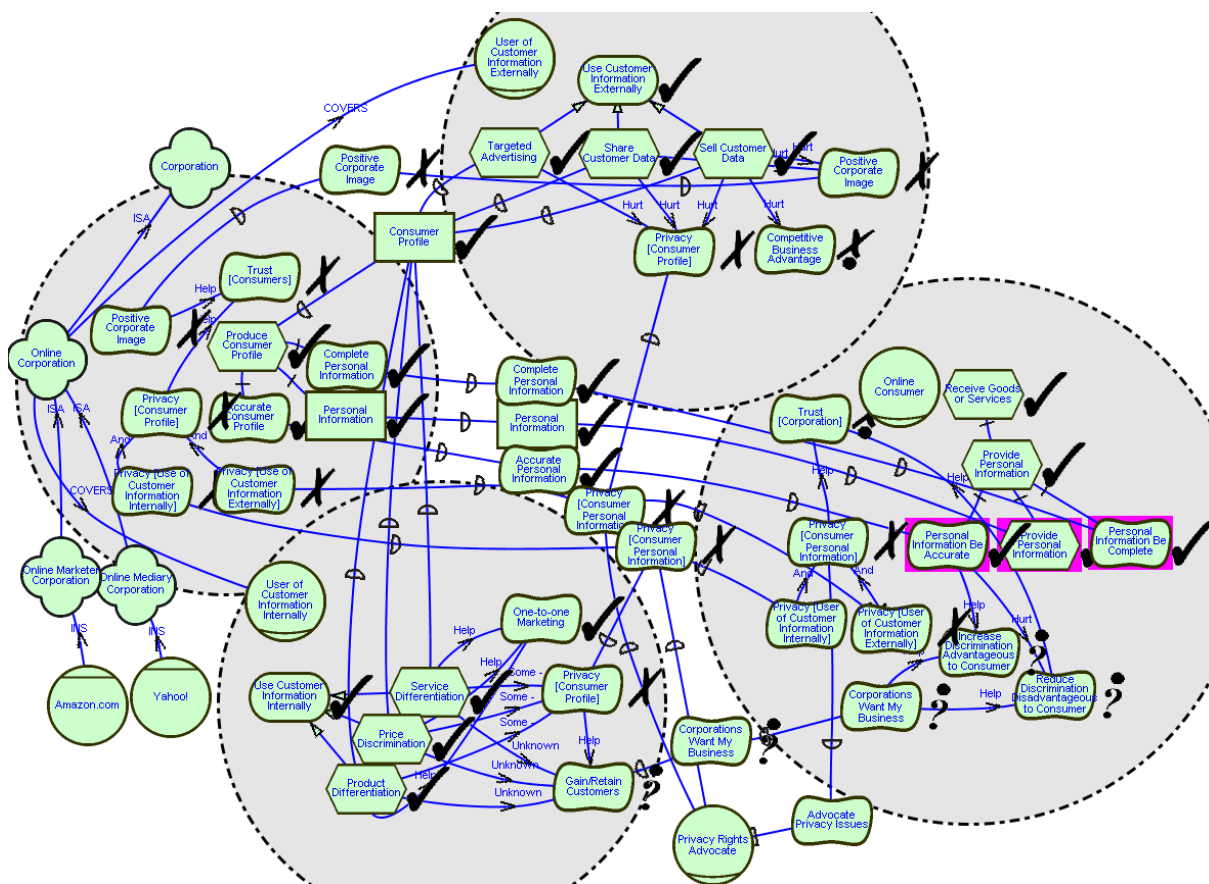


Figure 7.2: Privacy in E-Commerce Example after Semantic Changes Prompted by Evaluation

7.3 Economic Information Security

This case study is similar to the previous in that it is based primarily on the information contained in one paper, in this case Anderson's *Why Information Security is Hard – An Economic Perspective* (2001). This paper makes the point that an understanding of information security is better facilitated by examining economic patterns, as opposed to the current emphasis on technical solutions such as cryptography and access control models. The economic patterns described in this paper, including network externalities, asymmetric information, adverse selection, liability dumping and the tragedy of the commons, provided an opportunity to test the expressive limits of i^* modeling and evaluation, as the framework had not been extensively applied to economic examples. For each pattern an attempt was made to create a general model; this model was then expanded to represent some of the specific examples provided in the paper. The majority of the models were evaluated, producing interesting issues concerning iteration, actor multiplicities, and evaluation values. In the presentation of this study, we have chosen to include the most revealing and interesting models in terms of evaluation for each individual economic pattern in the main body of the Chapter. Further models for each pattern are provided in Appendix C.

7.3.1 Network Externality

The first economic pattern modeled was Network Externality, involving the idea that a network becomes more valuable as more individuals use it. Usage of a network increases, which makes the network more valuable, which convinces more individuals use it, and so on. The general model for this pattern is shown in Figure 7.3, with the specific example models shown in Figure C.1 and C.2 in the Appendix C.

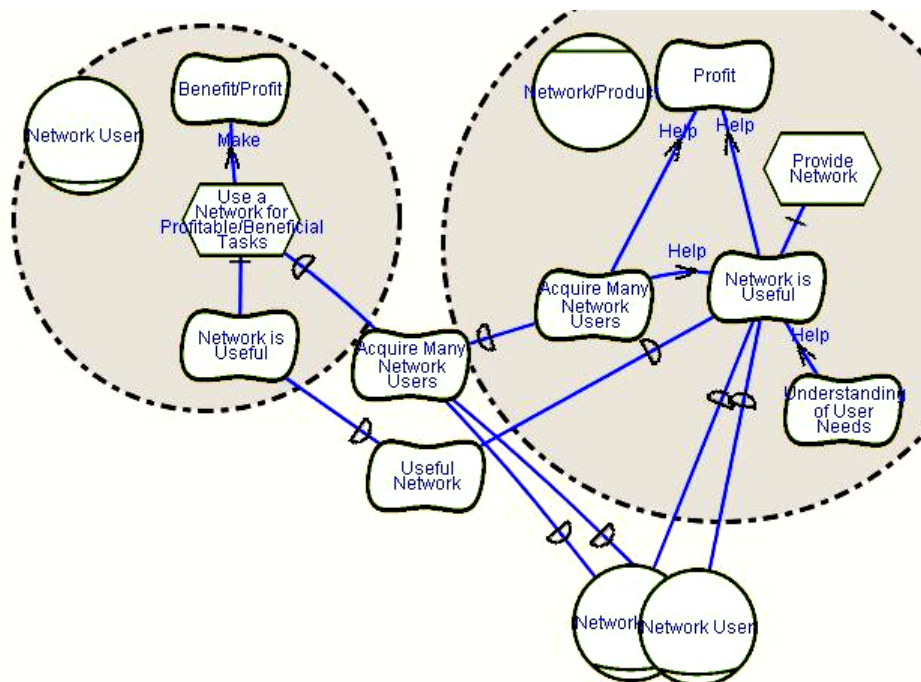


Figure 7.3: Information Security Network Externalities General Model

Actor Multiplicities. One of the fundamental difficulties with modeling this pattern arises in the representation and subsequent evaluation of actor multiplicities. As the full effect of the economic pattern occurs when the number of Network Users increases, it is difficult to model this pattern using only one Network User. This problem relates to the instance vs. class nature of i* models. Most actors represent a class of actor, with agents potentially representing their instantiation. However, in some cases we would like to model the instances of actors, in order to indicate that multiple instances occur. In this particular case, we have attempted to solve this difficulty by modeling multiple copies of the same actor, attempting to represent multiple instances. This issue carries over to evaluation, as the evaluation results conceptually depend on whether the model represents the actions of a class of actors, a specific actor instantiations, or a group of specific actor instantiations, potentially making different choices. We shall return to this issue when it arises again in the Trusted Computing Case Study.

Circular Evaluation. In addition, this pattern pointed out issues involving initial values for circular evaluations, as previously explored in Chapter 5, Section 5.1. We can see that Figure 7.3 contains a cycle from Network is Useful to Use a Network for Profitable/Beneficial Tasks and back. In order to deal with the evaluation of this cycle, an

explicit starting point was added in the form of the Understanding of User Needs softgoal, representing the idea that a network must provide for at least some user needs even before it is widely adopted, otherwise it would have no initial adopters.

7.3.2 Asymmetric Information

The next economic pattern, Asymmetric Information, describes a situation where one party knows more than another party, and the “knowing” party must make a decision whether or not to reveal this information to the other party, based on the predicted consequences of revealing or not revealing this information. This pattern may be best explained through the concrete example of a car dealership, shown in Figure 7.4. In this model, the salesperson knows whether or not the car for sale is a “lemon”, (of poor quality), or a “plum”, (of high quality), but must choose whether or not to tell the Car Buyer. The additional models depicting general Asymmetric Information and a Government Asymmetric Information example are included in Figures C.3 and C.4 of Appendix C.

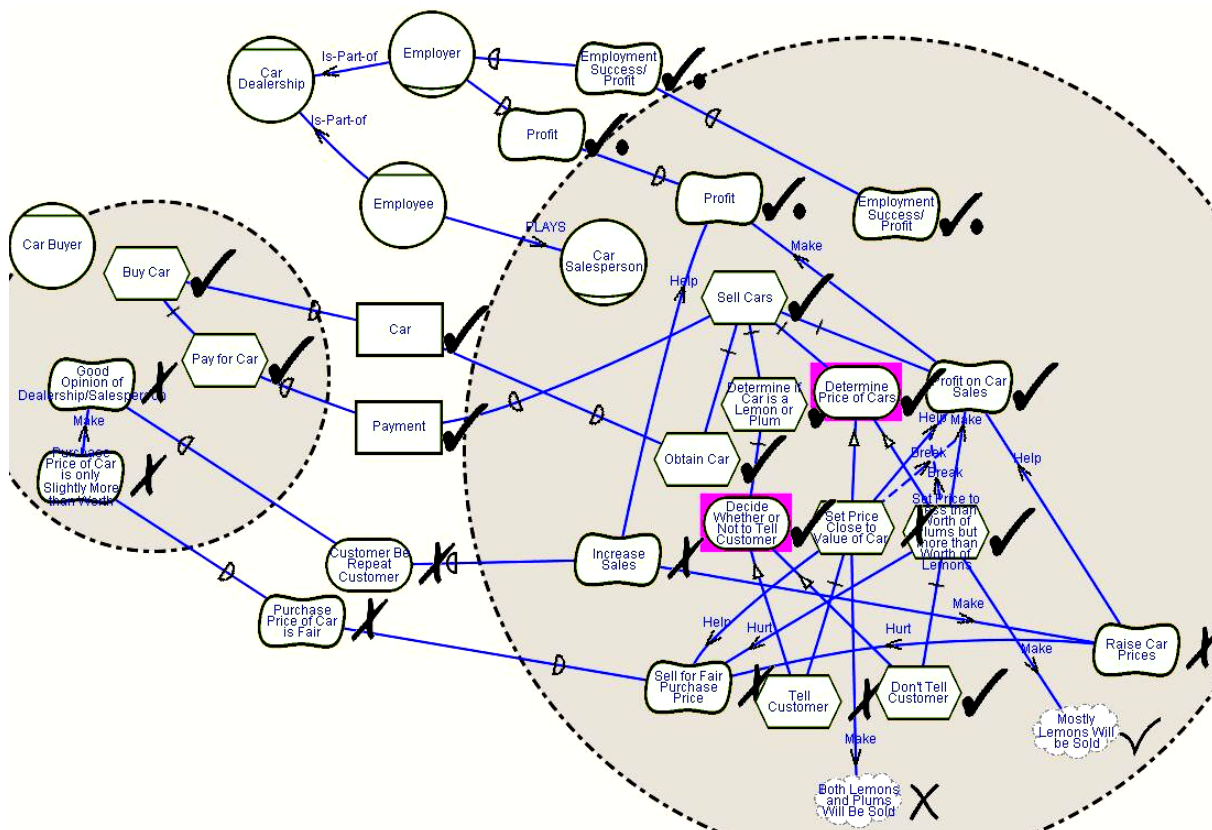


Figure 7.4: Information Security Asymmetric Information for a Car Salesperson

Exclusive vs. Inclusive Or. The modeling and evaluation of this pattern raised multiple issues. The issue of an exclusive vs. inclusive OR relationship for means-ends links, for instance. In this particular case, the choice of whether or not to reveal the status of the car to the customer is exclusive, yet there is no way to explicitly insure that these tasks receive opposite evaluation values. This issue has been left open in the evaluation procedure, leaving it up to the user to ensure that values given to means are semantically sensible.

Links to Other Links. These models were one of the first instances where links to other links seemed to be semantically necessary. In this case, concerning the Determine Price of Cars decision, the selection of one option did not seem to imply the effects of the negation of the other option. In other words, when Set Price Close to Value of the Car was chosen, this did not imply that the link from Set Price to Less than Worth of Plums but More than Worth of Lemons to Profit on Car Sales propagated a denied value. In this case, the proposed solution was to insert break links from each option to the link from the other option, meaning that selection of one option broke the effects of not selecting the other

option. Another possible solution may be to avoid marking the alternative not selected as denied, and instead leave this alternative without an evaluation value. However, if the strict form of propagating unknown values were followed, this alternative would result in an unknown value for Profit on Car Sales.

Unknown vs. No Label. This situation may indicate that a semantic difference is needed between an unknown evaluation value and an unmarked node, with no value being treated as having no effect, as explored in Chapter 4, Section 4.3.1.1.

Iteration of Evaluation Values. The cycles present in the Information Asymmetry models raise issues concerning the iteration of evaluation values. Specifically, there are situations, especially in economics, where prices increase or decrease in a cyclical manner, in order to continually optimize profit. The car sales example provides an illustration of this, as prices are made as high as possible for as long as possible, until the consumer becomes sufficiently annoyed and business starts to suffer. At which point the prices are lowered until business increases, and at which point the prices begin to rise again. It is difficult to represent these kinds of economic cycles in i^* models, and it is difficult to demonstrate the effects of such cycles with i^* evaluation. In Figure 7.4 we attempt to show the first stage of such a cycle, where prices are high and profit is temporarily high, but the Car Buyer is becoming displeased with the situation. As the values in a situation such as this never converge, the best we can do with i^* evaluation is to show “stages” or snapshots of the values, such as we have done in Figure 7.4. In the case study, we include a second evaluation of this model showing the next iterative stage in the economic network, included in Figure C.5 of Appendix C.

Qualitative Expressiveness. In addition to the complexity of non-converging cycles, this example illustrates the potential inadequacies of the qualitative evaluation values we have chosen. In each iteration of such a cycle we would like to show that Profit on Car Sales and Good Opinion of Dealership/Salesperson rises or falls incrementally. However, we are only able to jump from fully satisfied to partially satisfied to conflict. It is unlikely that profit in such a cycle is ever fully satisfied. Likely such a value ranges in different degrees of partial satisfaction, and occasionally partial denial. This situation serves as a good example for when a quantitative evaluation may be more appropriate, especially if we can ground such evaluation in data concerning real car prices, creating

links which approximately mimic the effects in the rise and fall of the cycles. We shall return to the subject of incorporating quantitative values into i^* evaluation in Chapter 9.

7.3.3 Adverse Selection

The next economic pattern explored through models was Adverse Selection, which involved a certain party making a decision, where the goals and benefits of the decision maker are not the same for the party depending on the decision. Therefore there is a lack of adverse consequences for making the wrong decision. In fact, often the consequences of making a decision that is wrong for the dependor are more beneficial to the decision maker than making a decision that is right for the dependor. This general situation, as well as a specific example concerning software selection, is included in Figure C.6 and C.7 of Appendix C. A specific example of Adverse Selection concerning product selection by an employee is shown below in Figure 7.5. In this case, Anderson makes the claim that it is more likely for an employee to pick an ill-suited product from a well-known company than an effective product from a little-known company, in order to appear competent in the event of a product failure.

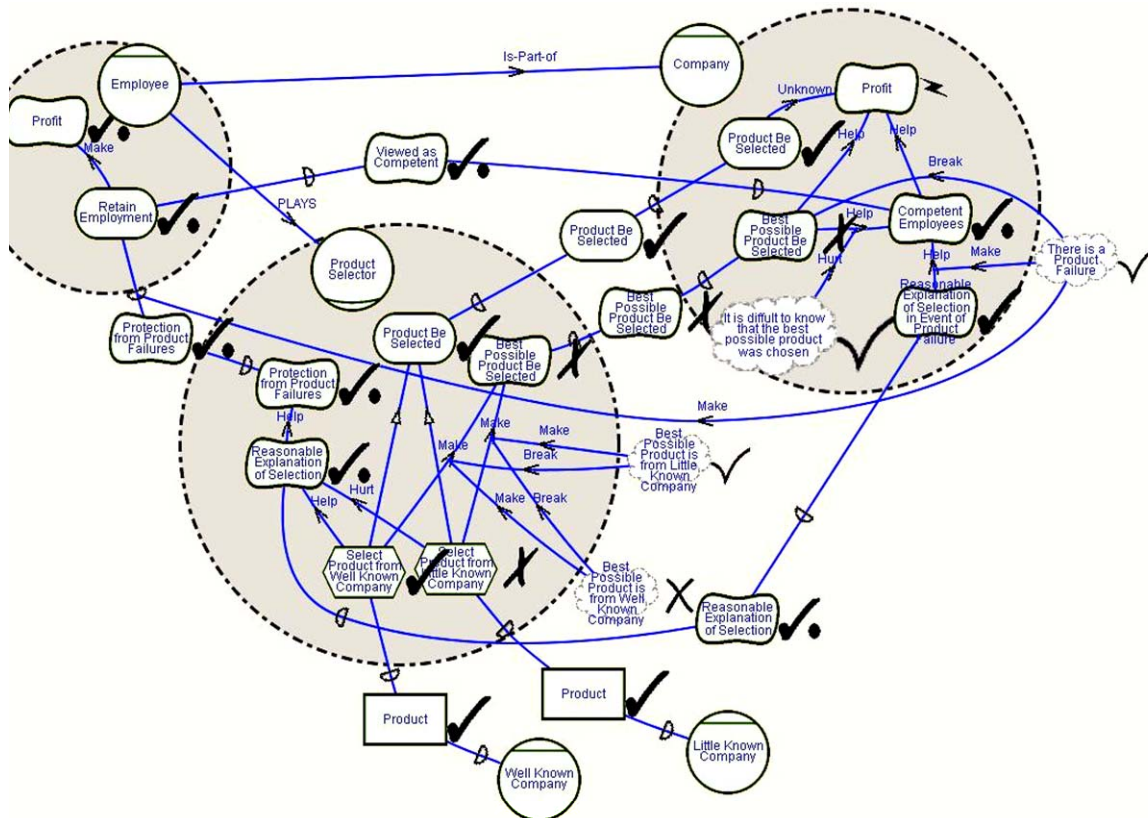


Figure 7.5: Information Security Adverse Selection in Product Selection

Representing Events. In modeling this situation, difficulty was encountered in representing the occurrence of a specific event, and the effects of this event. In this case, difficulties arose in representing the event of a product succeeding or failing. In this particular case we chose to model this event as the belief, There is a Product Failure, which is then marked as satisfied or denied depending on whether or not the event occurs. The effects of this event, both negative and positive, are represented by links from the belief to other elements in the model, including other links. During evaluation, the effects of this event on the model would be determined and analyzed. An alternative way to address this issue may have been to create two separate models, one for each event. This may have been necessary if the effects of an event occurring and not occurring were not symmetric, i.e. could not have been represented by the same set of links, although some of this non-symmetry could have been dealt with by links to other links.

7.3.4 Tragedy of the Commons

The next pattern, the Tragedy of the Commons, involves the situation where many users use a common resource. If users increase their use of that resource slightly, it might not be worth the effort for other users to prevent this increase, as it results in only a minor effect on the shared resource. However, as it is generally inconvenient to regulate other users, the summation of this effect results in the depletion of the shared resource. The general model for this pattern is shown in Figure 7.6, with a specific example for grazing sheep shown in Figure C.8 of Appendix C.

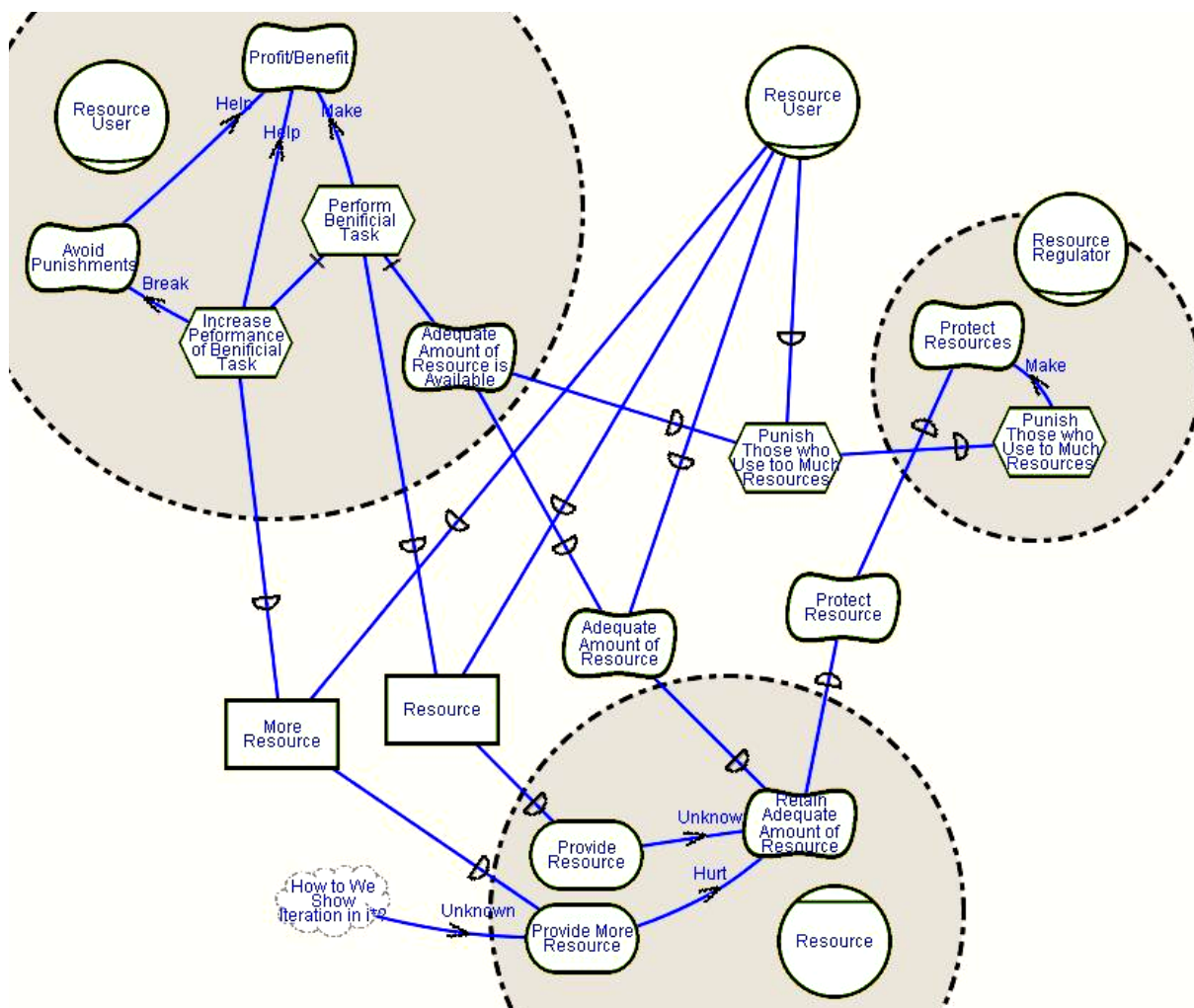


Figure 7.6: Tragedy of the Commons General Pattern

Qualitative Expressiveness. In this example, we see an issue similar to one which arose in the Asymmetric Information example, where the gradual depletion of a resource is difficult to depict with qualitative evaluation values. In this particular case, we attempted to model a gradual increase in resource usage with the addition of an additional resource element More Resource. Although this may serve to get the point across, the specific quantity of resource may, again, be better represented through numerical propagation.

Capturing the Process of Changing Evaluation Values. In addition, although this particular evaluation would converge to a depletion of the resource, the interesting phenomenon is the manner of depletion more so than the end result. These intermediate states are difficult to capture accurately with i^* evaluation.

7.3.5 Liability Dumping

The fifth type of economic pattern explored through i^* modeling, Liability Dumping, involves the situation where the party who suffers from a lack of security is not the same party who is responsible for the security. The models for this pattern are included in Appendix C in Figures C.9 and C.10.

7.3.6 Discussion

This case study provided us with the opportunity to test the applicability of the evaluation procedure to economic patterns, revealing multiple issues fueling various aspects of the procedure, and multiple areas of future investigation. Overall, although work may be needed in order to adapt the procedure for use in this context, including the use of numeric measures, the descriptive abilities of this evaluation application show promise in representing the effects of economic patterns on the intentional motivations for such patterns.

7.4 *Montreux Jazz Festival*

The Montreux Jazz Festival (MJF) study was performed with two aims, first, attempting to represent a domain which had been described using E-Business models, with the intention of comparing the two representations; second, to provide a shared tutorial on i* modeling and evaluation for those involved in the Kids Help Phone research project. The information concerning the Jazz Festival, including the actors involved and the aspects involved in the business was obtained from the PhD thesis of Alexander Osterwalder (2004), exploring E-Business models. In the presentation of this study, we include a subset of the models which highlight interesting aspects discovered during the application of evaluation.

7.4.1 MJF Background

The Montreux Jazz Festival, as the name implies, is an annual music festival held in Montreux, Switzerland (http://www.montreuxjazz.com/index_en.aspx). The business interactions of this large festival are complex, including sponsors, municipal agents, volunteers, artists, vendors, and of course visitors; making this an interesting domain to express via organizational modeling. The models were created as part of one large conceptual model; however, due to the large size of the domain, the models were presented in many separate files, each file displaying a different partial view. Evaluation was used to assess the effects of certain events, such as an artist making a poor performance.

7.4.2 i* and Evaluation Issues Encountered

During the modeling exercise multiple issues involving model evaluation arose.

Actor Multiplicities. First, issues involving the multiplicity of actors, or instances vs. class, similar to the Privacy in E-Commerce models were detected. For instance, when a Visitor is modeled with the option of attending On-Festival events, Off-Festival events, or both, how can one evaluate the effects of these choices on the profit of

the MJF? The evaluation of one Visitor actor choosing one type of event and not the other would certainly affect the value of Profit, but this seems to imply that all Visitors make this choice. What if some visitors choose to attend both, some one or the other, and some none, how can we evaluate Profit? When attempting to portray the affects of such a choice, the Visitor role could be divided into two types of visitors, the On-Festival and Off-Festival Visitor, with each type attending only one type of event. However, even with this division, how do we know what proportion of actors made what choices? Although using numerical evaluation instead of qualitative evaluation may provide descriptive power in this situation, the question of multiplicity still arises: How many actors made each choice? As a result of these issues, the evaluation performed on such models can only demonstrate the effect of the choices of a single instance of an actor, or of all actors (the class) making the same choice. This situation is shown in Figure 7.7, where a festival visitor makes choices concerning what event to attend and what to purchase.

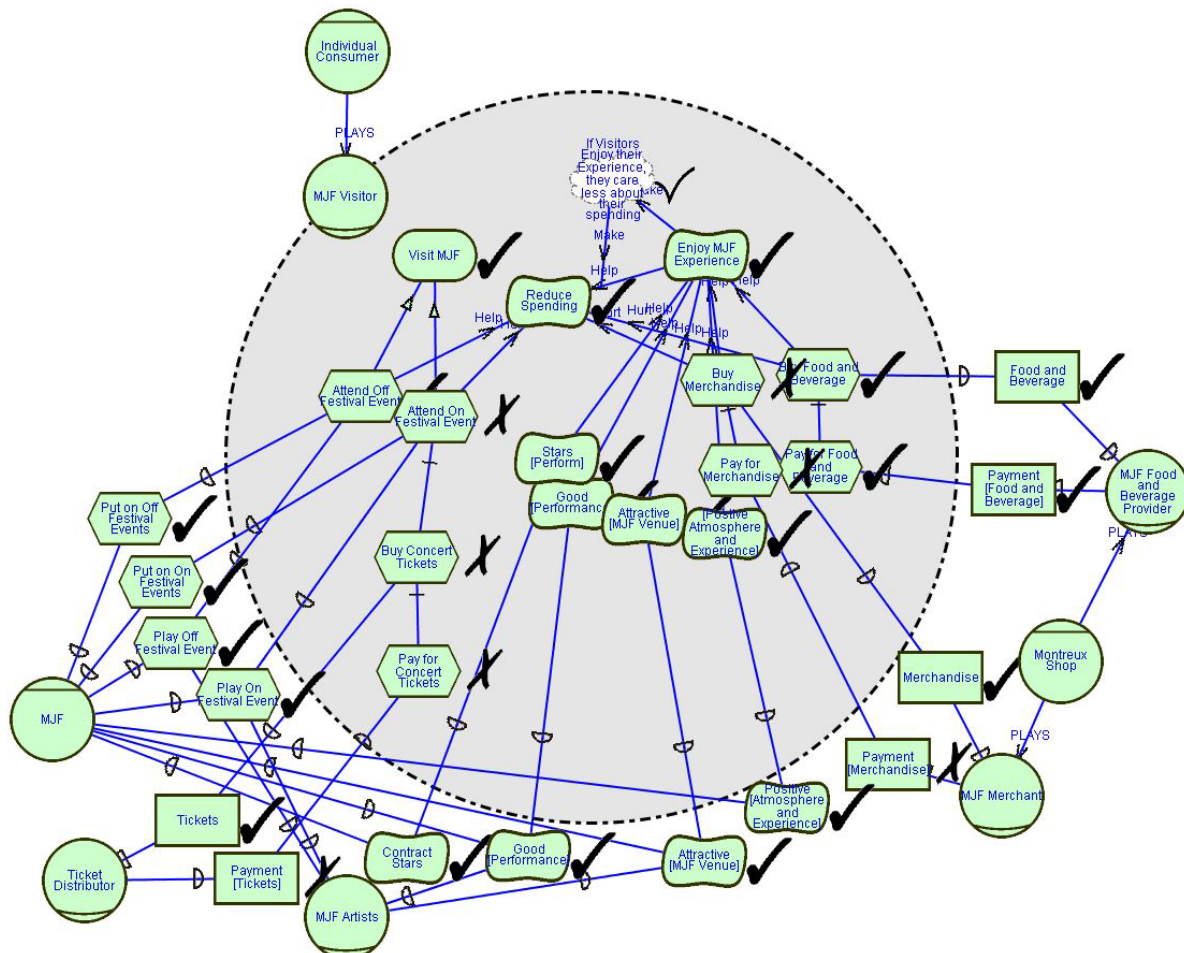


Figure 7.7: Montreux Jazz Festival Visitor Evaluation

Model Views Hindering Evaluation. The use of multiple physical files to break up the large models in this study made a thorough evaluation of all elements in the conceptual model difficult. For example, the evaluation in Figure 7.7 shows the internal results in the MJF Visitor, and also shows how these results affect and are affected by their interaction with other actors. However, we are not able to see in the same model how these evaluation values affect the internal elements in other actors, or vice versa, as we would be able to in a “full” SR model. In order to evaluate the effects of an option on the entire conceptual model, we need to “import” and “export” evaluation values to other physical files. For example, given the results of the evaluation of the MJF Artist shown in Figure 7.8, we would export the values shown into the MJF agent as shown in Figure 7.9 (note that this view of the MJF agent involves only elements relating to the MJF Artist). The relationship between these two actors is circular in terms of contributing and receiving evidence through evaluation values, so we would have had to first assume that certain values such as MJF Venue were available from the MJF agent. In general, we have not yet arrived at a solution to this problem of propagation across models more sophisticated than the manual export and import shown here. However, the ultimate goal for the OpenOME application is to support multiple views for one large conceptual model, in which case propagation across such views may be done automatically.

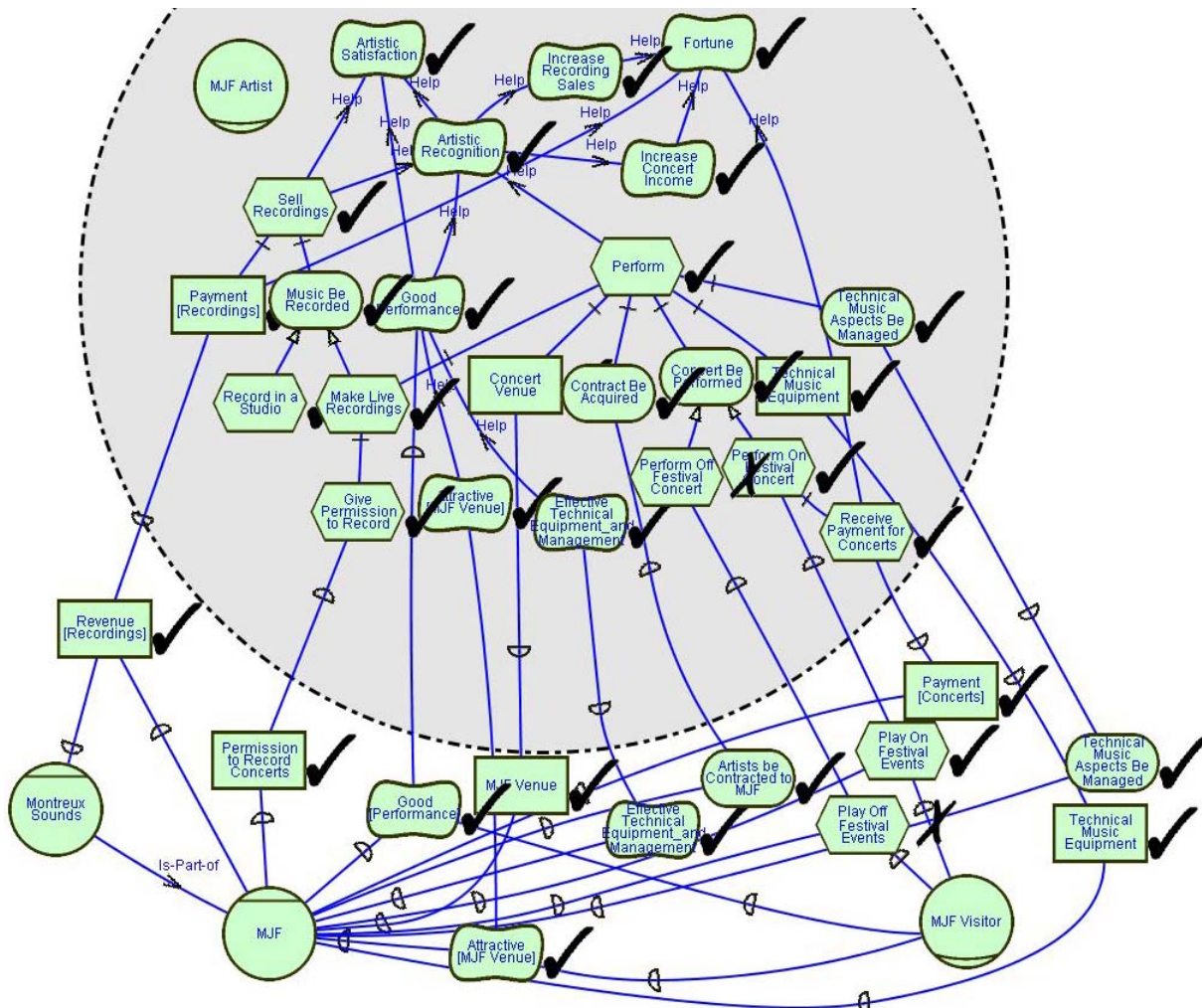


Figure 7.8: Montreux Jazz Festival Artist Evaluation

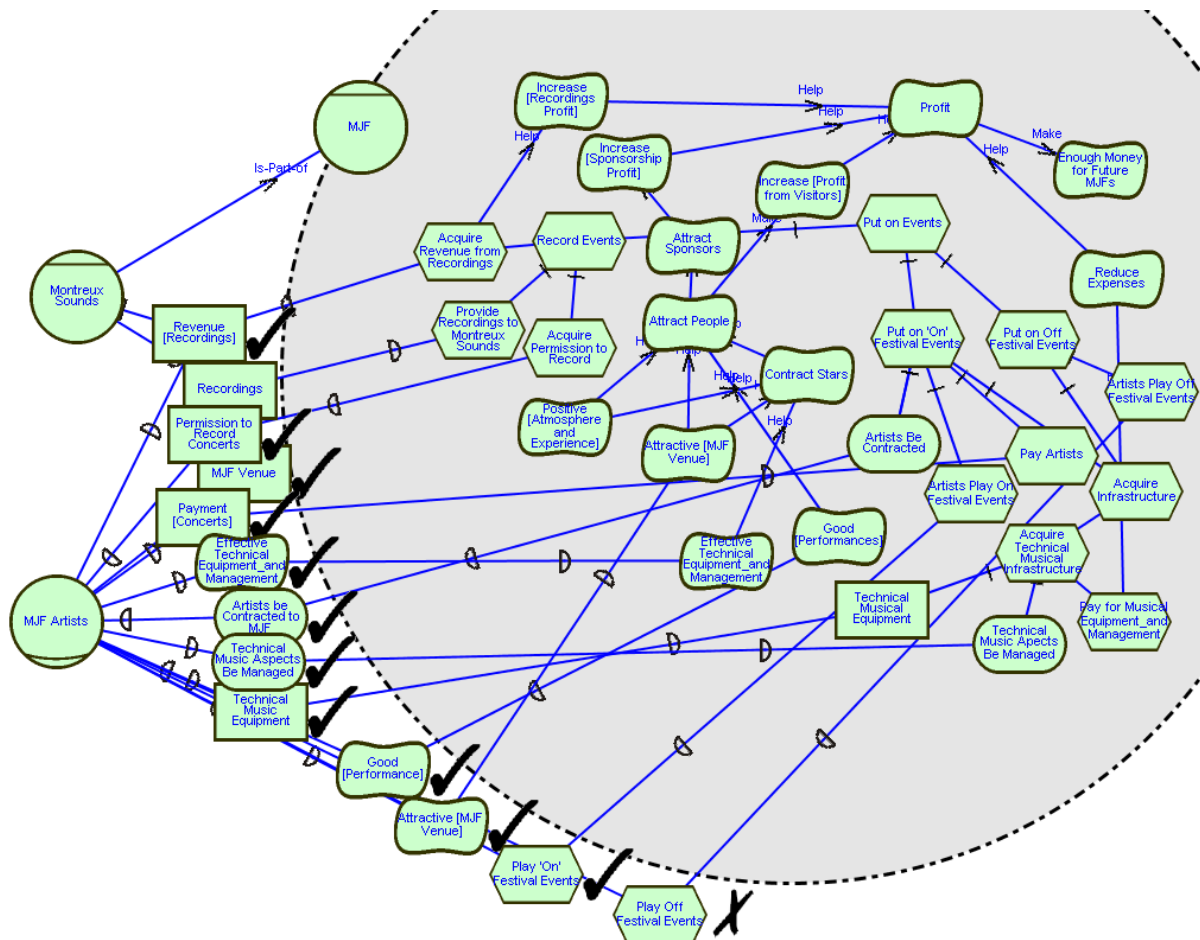


Figure 7.9: Montreux Jazz Festival MJF in Relation to Visitor Evaluation

7.5 Trusted Computing

The Trusted Computing modeling project began as an exercise in order to push the limits of i^* modeling by attempting to model the debate concerning the Trusted Computing technology proposed by corporations such as Microsoft, IBM and Intel as part of the Trusted Computing Group (TCG). This debate provides an interesting domain for modeling due to its complexity, the presence of at least two conflicting viewpoints, and the existence of attack and defence situations. We have abstracted this complex and multifaceted domain into two viewpoints, opponents and proponents of Trusted Computing. Although it is likely that parties which fit within these viewpoints do not themselves have a uniform viewpoint, we have used available documentation from both sides of the debate in an attempt to model the prevailing opinions on each side.

7.5.1 Sources

The document sources for the proponent point of view include technical reports or FAQ's of the TCG or TCG members such as (<http://www.microsoft.com/technet/Security/news/ngscb.msp>) and (<https://www.trustedcomputinggroup.org/>). The proponent point of view was based mainly on Anderson's Trusted Computing FAQ (<http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html>).

7.5.2 Background

In general, proponents of Trusted Computing (TC) claim it will promote Security for the average user by protecting Personal Information and PC Control from Malicious Users. This group claims that TC will protect against Identity Theft and promote Secure Online Transactions as well as potentially protect against Spam and Viruses/Malicious Code. However, in contrast, opponents of Trusted Computing claim it will take away the Security of the Technology User by giving control of their technology to the Technology Providers (such as Microsoft or Intel) as well as to authorities such as the Government or the Police. In addition, proponents claim that TC technology will increase market shares for TCG Member Companies by Locking In Users. One of the fundamental motivations for TC technology, according to its opponents, is to combat Software Piracy and further implement Digital Rights Management (DRM), protecting the technical products and increasing profits for TCG member companies.

7.5.3 Case Study Presentation Style

In order to try and accurately capture this complex situation using models, we presented the models incrementally, focusing on the relationship between two actors at a time, and then gradually expanding the models to include interactions with other actors. However, due to problems arising from making such views consistent, the majority of the modeling was done using one very large model, broken into smaller views for

presentation. In this presentation, we have opted to include, for each major section of the study, the larger models which encapsulate the most social interaction, and which provide demonstrate the applicability of i^* evaluation to complex models. The smaller model views, which are a subset of these larger models, are included in Appendix D.

7.5.4 Shared Viewpoints

The first stages of presentation involved elements which were “neutral” or non-controversial, shared between both viewpoints. This included the basic relationship between the Technology Producer and the Technology User, between the Technology Producer and the License/Copyright Holder, and between the License/Copyright Holder and the Content User. These views are included in Figures D.1, D.2, and D.3 of Appendix D. The combined view of these four actors is shown below in Figure 7.10. The major conclusions brought about by the evaluation of this model was that the relationships among these actors was generally non-problematic, with the exception of the PC User and the Content User's desire to save money by acquiring more Affordable products.

Contributions from a Mixture of Link Types. We can see that the model in Figure 7.10 contains multiple instances of a mixture of different types of links for one element, most commonly the mix between decomposition and dependency links for a higher-level task. During the evaluation of this case study it became clear that standard recommendations to deal with label propagation in these types of situations were needed, resulting in the guidelines presented in Chapter 4, Section 4.3.3.8.

Repetition of Softgoals across Actors. Models in this case study, such as Figure 7.10, serve as examples of the repetition of softgoals across actors, as discussed in Chapter 5, Section 5.1. Between the Technology Provider and the Technology User we can see the repetition of four softgoals, Security [PC], Freedom [Use of PC Products], Abide by Licensing Regulations [PC Users], and Compatibility [with Existing PC Products]. The differing location of each of these softgoals implies differing meanings, as indicated by slight naming changes. For instance Compatibility within the Technology Provider refers to whether this softgoal is judged to be Compatible by the standards of that particular role, while Compatibility within the Technology User may be judged by different criteria. These

interpretation differences can manifest in detailed models as the presence of differing decomposition elements within each actor, although this situation is not present in Figure 7.10. Although these elements may mean different things within each actor, they are still conceptually linked. The presence of a dependum of the same name linking elements of this type is motivated by the need for a linkage between the evaluation values of both elements. Therefore the Technology User depends on the Technology Provider for Compatibility. When the Technology Provider does not satisfy this element, this value will be reflected within the Technology User, although it is possible that Compatibility could obtain a different evaluation value if it received contributions internal to the Technology User. The repetition of softgoals across actors is common in TC models, as well as in models for the Kids Help Phone study, as described in next section.

Marking Initial Values. During the application of evaluation to the large models in the TC case study, it became apparent that explicitly marking the starting points of evaluation would be beneficial for future examination of the model. This would make it easier for model users, especially those unfamiliar with the models, to follow the propagation paths, tracing through the cause and effect relationships contained within the model. In the TC models created after this discovery, the starting points of evaluation are indicated with pink rectangles, such as is seen in Figure 7.10 below.

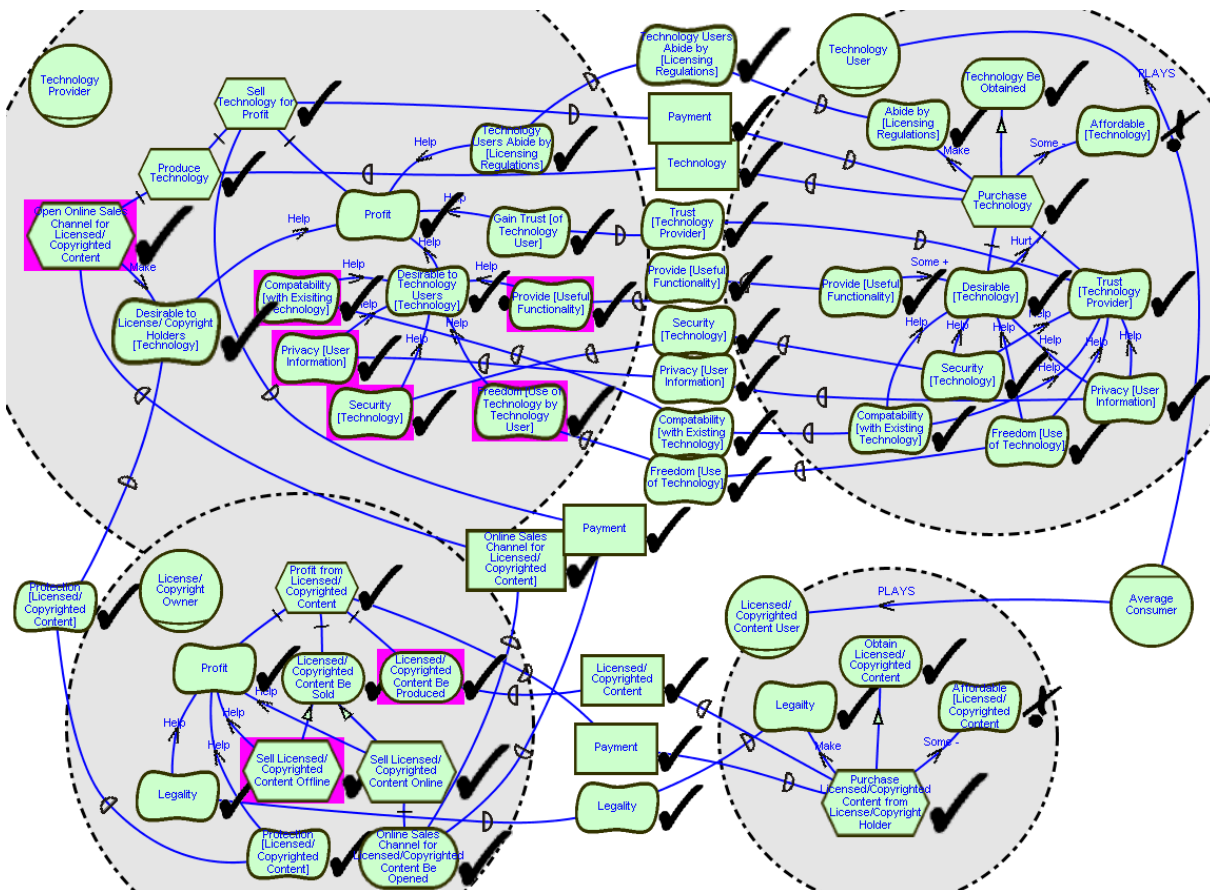


Figure 7.10: Trusted Computing Case Study Model Showing the Shared View of the Technology Producer, Technology User, License/Copyright Holder and Content User

7.5.4.1 How do the Consumers satisfy their Desire for Affordable Products?

The denial of Affordable motivates the involvement of the actor we consider next, the Data Pirate. Continuing with the view shared by the TC proponents and opponents, we include the actions and motivations of this actor into our previous representations. First we examined the relationship between the Technology User, Content User and the Data Pirate. The interactions between the Data Pirate and the Technology Provider were modeled, showing the technological freedom required by the Data Pirate in order to accomplish its piracy tasks. The models showing these smaller views are included in Figure D.4, and D.5 of Appendix D. Finally, we added the License/Copyright Holder into this picture, describing the interaction between all five actors. This combined model is shown in Figure 7.11. Here, evaluation reveals that although the introduction of the Data Pirate has

provided a means to satisfy the Affordable softgoals of the Technology User and Content User, the Technology Provider and License/Copyright Holders main tasks of selling their products or content for profit is now denied, fueling further modification to the situation via the introduction of TC technology.

In a note concerning the naming of model elements and actors, as the Trusted Computing Study has undergone various revisions, names of elements and actors have been modified slightly across the diagrams presented here. Namely, the word PC used in earlier models has been replaced with the more general term Technology, both in element and actor names.

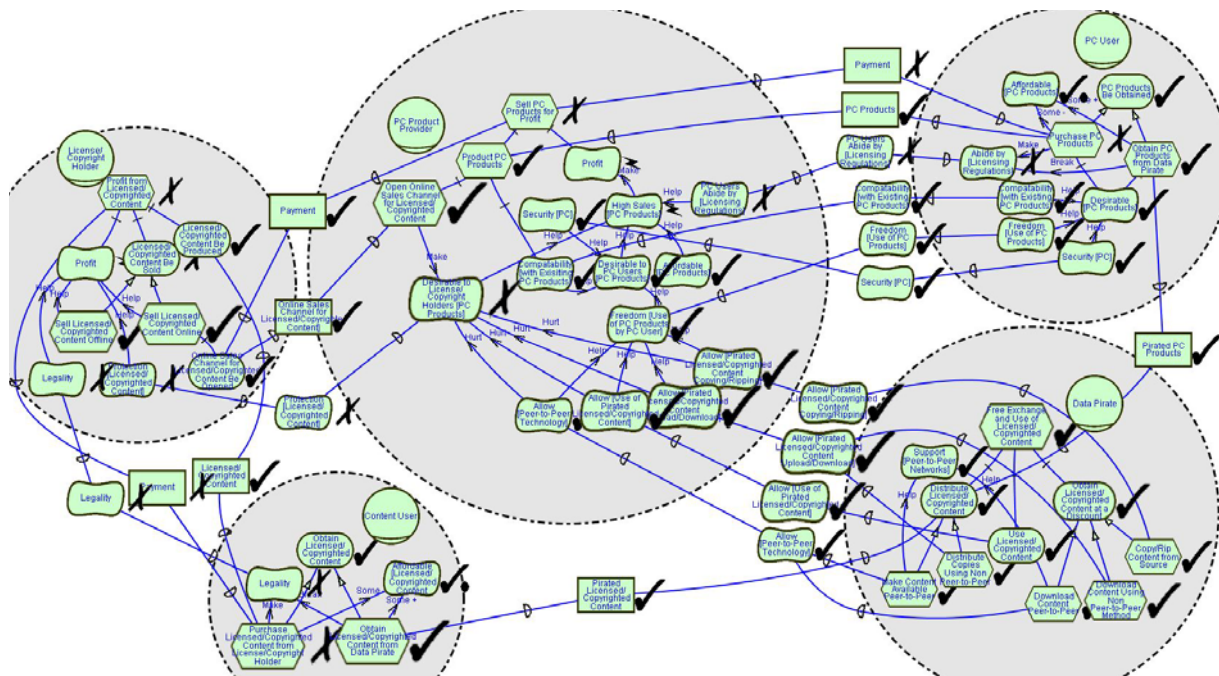


Figure 7.11: Trusted Computing Case Study Model Showing the Interaction of the Data Pirate with the Actors Considered Previously
Note: The names of actors and some nodes are modified slightly across figures, as explained in the text.

Actor Multiplicities. While modeling the effects of the Content User and Technology User's decision to obtain products from the Data Pirate instead of from the Technology Provider or License/Copyright Holder, the issues of multiplicity among actors again became apparent. When a Content User decides to avoid legal purchases by acquiring technology or content from a Data Pirate, this obviously has a detrimental effect on the Profit of other actors. However, do we really want to evaluate the situation where

all Content Users make this choice? Is this realistic? What if some Content Users obtain products illegally and some do not? What if the majority of Content Users purchase some products legally and, at the same time, obtain other products illegally from a Data Pirate? In the second version of Figure 7.11, shown in Figure 7.12, we have attempted to model some of these effects by drawing two instances of the Technology User and Content User, one obtaining products illegally, and the other legally. This is meant to demonstrate the effect of a split in the decisions made by these roles, with the human decision for the PC Users Abide by Licensing Regulations softgoal reflecting this knowledge. In this way, our use of human intervention in evaluation compensates for a potential lack of expressive power in i^* models. Alternatively, we could have created two different types of roles for each existing role; for instance: Legal Content User and Illegal Content User; and Legal Technology User and Illegal Technology User. This would be especially effective if the internal goals and motivations of each role sub-type were different. However, as the internal motivations of these types of roles did not differ significantly, we chose to only create a single version of these roles (with multiple instances).

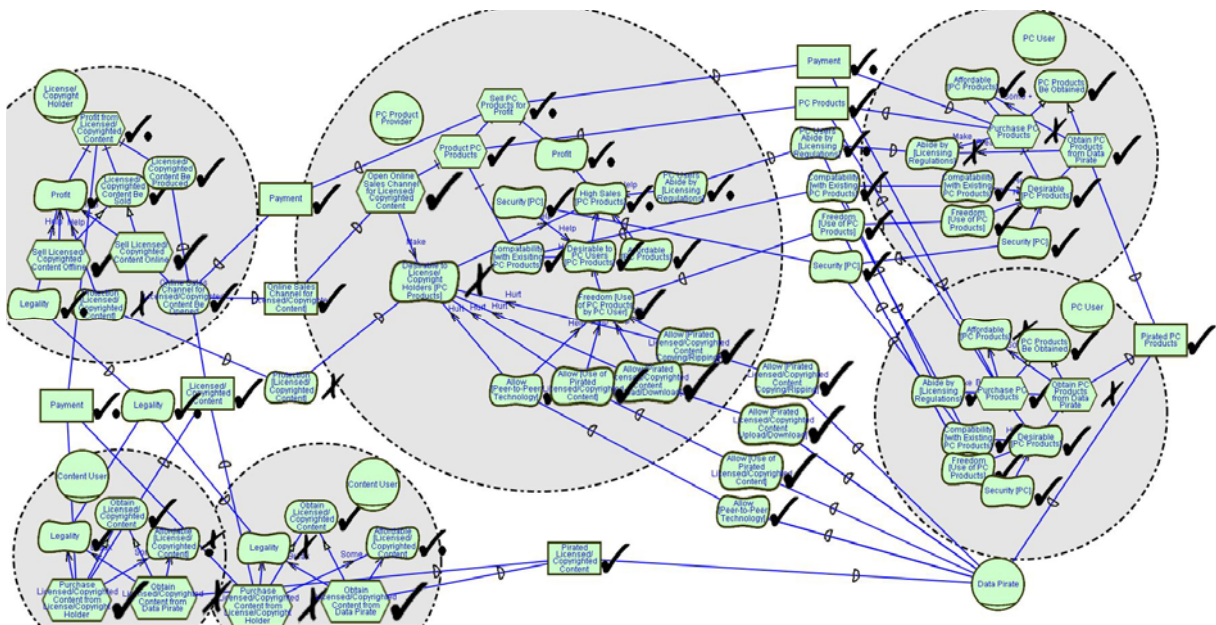


Figure 7.12: Trusted Computing Case Study Model Showing the Interaction of the Data Pirate while Considering the Multiplicity of Actors

Propagation Rules as Guidelines. When presenting propagation rules in Chapter 4, Section 4.3, we stated that such rules should be treated as guidelines, and

should be broken in rare situations when the modeler considers it necessary. Figure 7.12 presents an example of this in the evaluation of the Payment resource. This resource depends on the Purchase PC Products task within both PC User instances, having values of denied and satisfied respectively. Given our evaluation guidelines, these values would be combined using an And relationship, producing denied, and resulting in the denial of Sell PC Products for Profit within Technology Provider. However, as we were attempting to evaluate the situation where only *some* of the Technology Users Obtained PC Products from the Data Pirate, we chose to give this element a value of partially satisfied, representing an effect from piracy which is harmful, but not sufficient enough to deny Profit in the Technology Provider. It is likely that further consideration of this situation could result in changes to the model that may produce the desired evaluation result without having to ignore the propagation guidelines. However, as the model is already quite large, and as the required changes seem to be non-trivial, we have decided that the effort of model iteration is not worth the potential gain of clarity in this particular case.

7.5.4.2 How does the Hacker/Malicious User fit in the Situation?

Next, continuing with the shared view, we explored the threats to security via the actions of the Hacker/Malicious User. We first looked at the relationship between this role and the Technology Provider, modeling the elements of security which could be threatened by Hacker/Malicious User actions. These views are included in Figures D.6 and D.7 of Appendix D. We then included the Technology User in this view, finally creating a general view of the Hacker/Malicious User included in the interactions of all previously considered actors, shown in Figure 7.13.

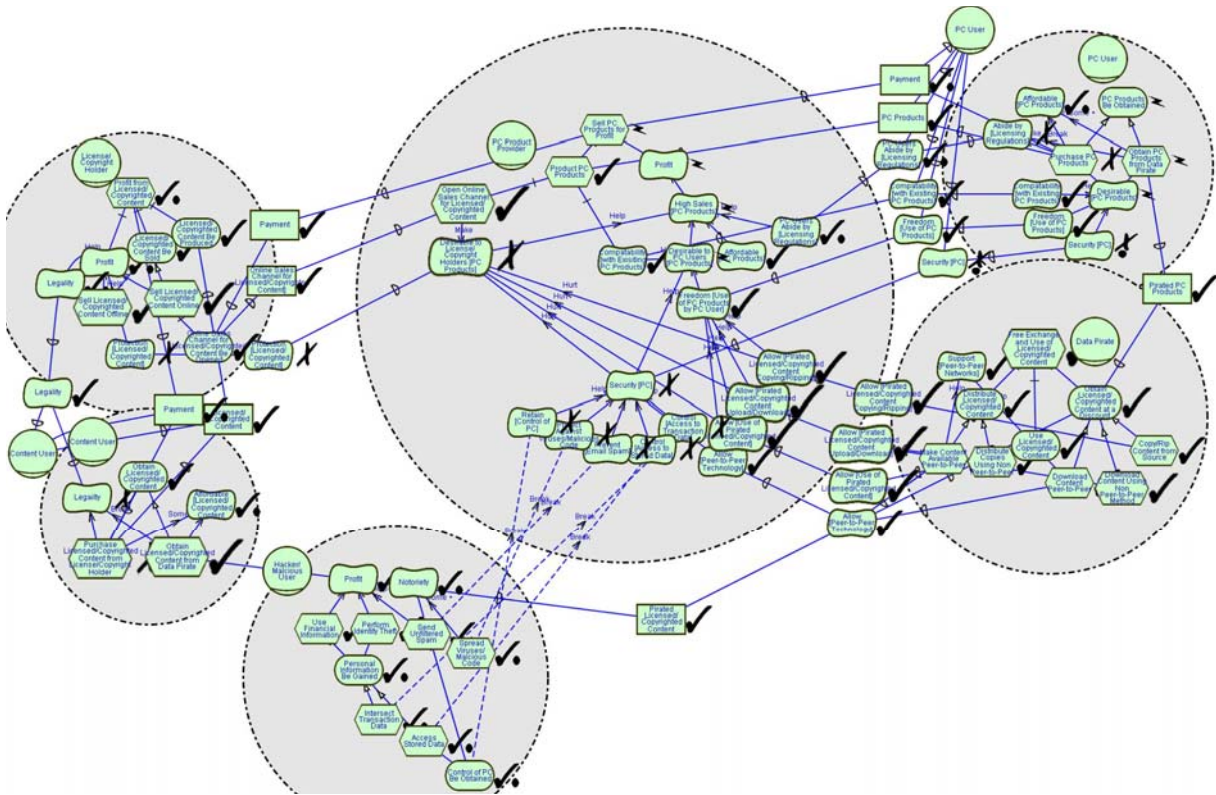


Figure 7.13: Trusted Computing Case Study Model Showing the Interaction of the Hacker/Malicious User with All Previously Considered Actors

At this point we have produced a sufficient exploration of the technology business situation before the introduction of Trusted Computing technology, including the motivations for such a technology from the point of view of the Technology Provider, and possibly the License/Copyright Holder, depending on the viewpoint. To explore the contrasting viewpoints, we built upon the model shown in Figure 7.13 in two branches, one for each viewpoint.

7.5.5 TC Proponent Viewpoint

First, we focus on the point of view of TC technology proponents, adding elements representing the proposed functionality of TC. As in the previous sections, we presented the addition of TC technology using a gradual combination of smaller views. First we examined the interactions between the Technology Provider providing TC technology and the Data Pirate. We then modeled the effects of TC technology on the actions of the Hacker/Malicious User, and extended this model to show how these effects

affect the Technology User. These three models are included in Figures D.8, D.9, and D.10 of Appendix D. Finally, we combine these elements into a “big picture” view for TC proponents, as shown in Figure 7.14. The general conclusions reached by evaluation were that, according to TC proponents, TC technology was at least partially effective in thwarting the malicious activities of the Hacker/Malicious User, helping to ensure Security and making technology more Desirable to users, resulting in a partial satisfaction of Profit for the Technology Provider. However, the effects on the Data Pirate according to the model were more difficult to determine, as the DRM components facilitated by TC were optional, resulting in a conflicted value for the Profit of the License/Copyright Holder. This result is somewhat contradictory to the proponents’ claims that the intention of TC is not to enforce DRM.

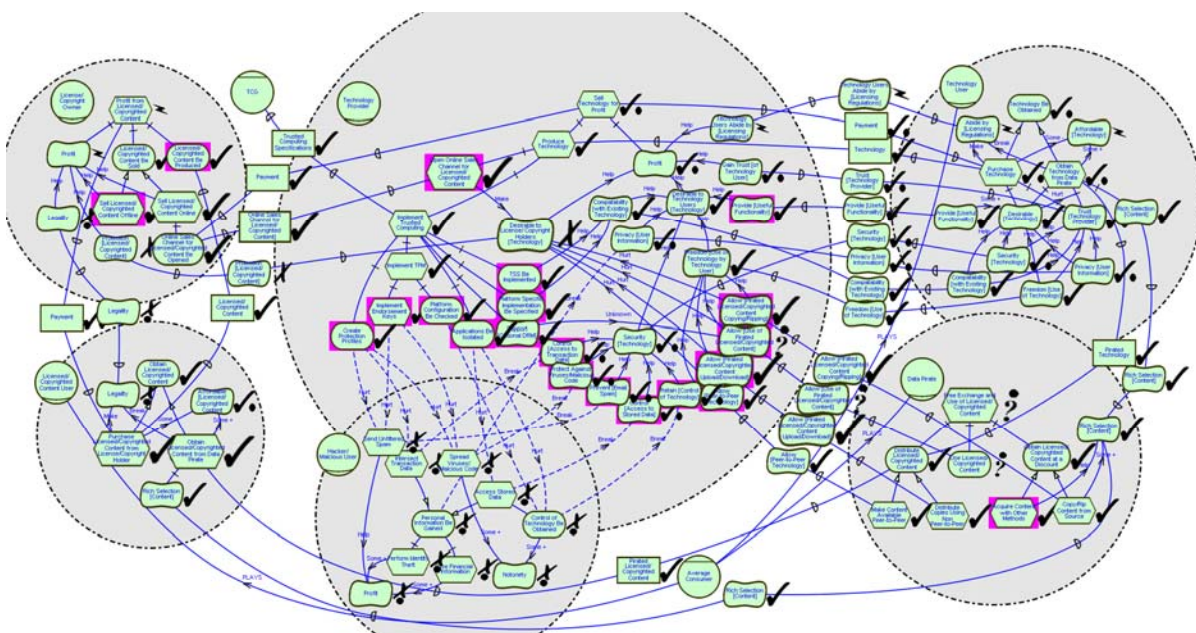


Figure 7.14: Trusted Computing Case Study Model Showing the Big Picture for TC Proponents

7.5.6 TC Opposition Viewpoint

Next, we turned to the point of view of the TC opposition. Using Figure 7.13 as a starting point, we created a second version of the technical components of TC from the point of the TC opposition. This version included such elements as Support [DRM],

Remote Censorship/Deletion Be Available, Lock-In [PC Users], and Backdoor Access to Authorities Be Provided. We showed the effects of these components on the relationship between the Technology Provider and the Hacker/Malicious User, which, according to TC opponents, is minimal. We then explored the relationship between the Technology Provider and the Data Pirate, showing the detrimental effect of TC technology on piracy. These models are included in Appendix D as Figures D.11 and D.12. Finally we presented the big picture view for the opposition, as shown in Figure 7.15. The general conclusions from this viewpoint include the ineffectiveness of TC technology on Security threats, and the effectiveness of this technology on preventing Data Piracy. Profit increases for both the Technology Provider and License/Copyright Owner. The opposition emphasizes the fact that Technology Providers and License/Copyright Owners are often played by the same agent, providing an explanation for the concern of the Technology Provider with the Profit of the License/Copyright Owner. The technology is no longer considered Desirable to the Technology User, yet they are forced to purchase it legally due to product Lock-In, the lack of pirated alternatives, and general ignorance of its true qualities.

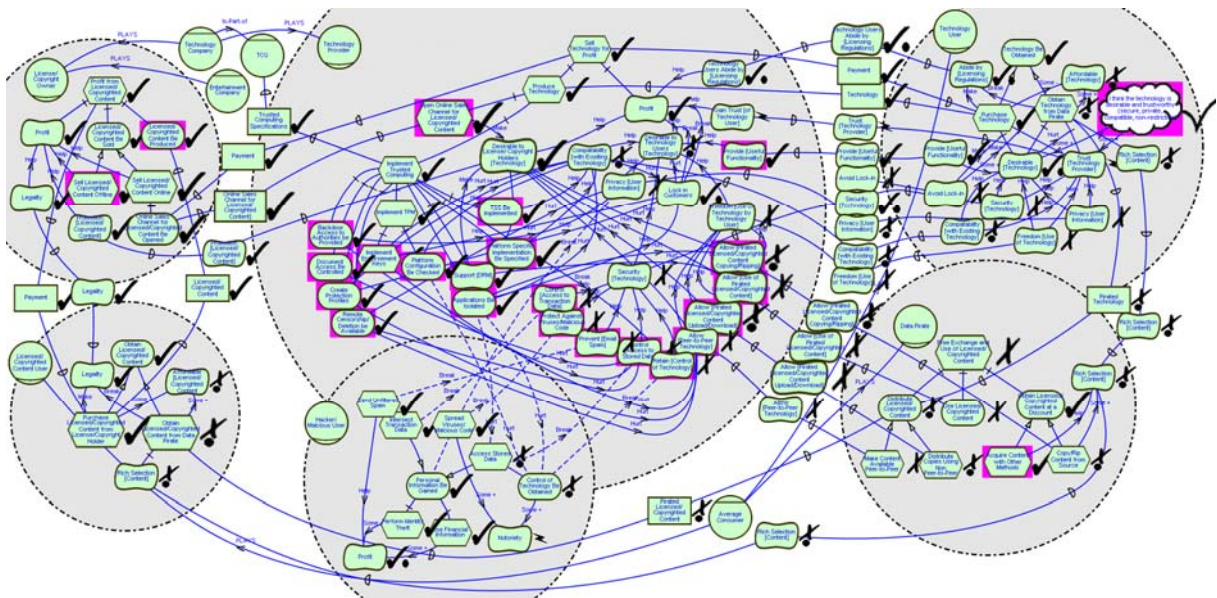


Figure 7.15: Trusted Computing Case Study Model Showing the Big Picture for TC Opponents

Semantic Improvement. The evaluation of the TC Opponent model in Figure 7.15 helped to solidify the idea concerning semantic improvement brought on by evaluation. The first version of this complex model seemed semantically correct before

evaluation; however, the evaluation revealed that the Technology User would not Purchase TC Technology, as it did not Desire it, and therefore the Technology Provider would not Profit. This result contradicted reality: Why would the Technology Provider implement and distribute TC Technology if it did not expect to make a Profit? This result prompted a deeper consideration of the model, specifically concerning the aspects of Lock-In and consumer Ignorance. The result was the iteration shown in Figure 7.15, where the Technology User Purchases Technology despite its undesirable status because of the effect of Lock-in and Ignorance, resulting in a positive situation for the Technology Provider. Of course, one could expand this model to consider the role of competition, a Technology Provider who does not implement TC Technology. However, results in this case would likely be similar due to the negative influence of Compatibility and Lock-In.

7.5.7 Discussion

In order to use the full potential of the i* Framework in this study, one would use the results to derive an alternative model representing a compromise between the goals of the major actors. However, due to the greedy nature of goals such as Profit and Affordable, a compromise that all actors can agree upon may not exist. Nevertheless, the results of this case study not only demonstrate the ability of i* evaluation to analyze complex situations, but also the capability for demonstrating the conflicts between viewpoints. Future work could be devoted to producing a systematic method which facilitates the comparison and potential resolution of conflicts in model structure and evaluation results.

7.6 Kids Help Phone

The Kids Help Phone study provided a valuable opportunity for testing the practical viability of i* modeling and evaluation, as it was grounded in interactions with real stakeholders, as opposed to document analysis. Kids Help Phone (KHP) is a charitable Canadian organization committed to providing counseling for Canadian Children (<http://www.kidshelpphone.ca/en/>). In the Strategic Requirements Analysis for Kids Help Phone project, researchers from Bell University Labs at the University of Toronto studied the KHP organization in order to assist them in determining and

retaining their core competencies in the face of increased usage of technology for counseling (<http://www.bul.utoronto.ca/labs-sra.html>). The progress of the project thus far can be divided into two major stages. The first stage involved general domain elicitation and modeling for the purpose of evaluating the potential effects of new counseling technologies and exploring the use of viewpoint modeling. The second stage involved a focus on elicitation specifically concerning the Ask a Counselor Online Service for the purpose of creating a specification for a replacement system and investigating the use of i* models in the prioritization of system features.

7.6.1 First Project Stage: General Elicitation, Technology for Counseling, and Viewpoint Modeling

In the first stage, we performed extensive domain elicitation through individual interviews, group meetings, and document analysis, the purpose of which was to gain a general understanding of all stakeholders in the organization, especially related to their role in web counseling. Through this process we were able to instigate an exploratory case study involving viewpoint modeling using the i* Framework (Easterbrook et al., 2005). Two separate modeling groups endeavored to model the KHP organization, one group creating many separate models from the viewpoints of various stakeholders, and the other group creating one conceptually large model of the entire organization by performing an implicit merging of viewpoints. The size of the models, especially in the second group, caused the models to be split into consistent, overlapping views, as was done in the TC and MJF study.

The large global models from the second group served an additional purpose in that they allowed the application of the evaluation procedure in order to evaluate the effectiveness of possible new technologies used for counseling. The models from the first group, representing individual viewpoints, seemed less suitable for evaluation, as they reflected only a portion of the domain and had a generally incomplete nature.

7.6.1.1 Model and Evaluation Challenges and Solutions

Model Size Hindering Evaluation. Evaluating the global model views was problematic due to their large size. For example, Figure 7.16 shows the high-level view of one of the largest model views, representing the services provided by KHP.

Model Views Hindering Evaluation. In addition, there was the issue of propagating evaluation values across views, in order to acquire a global assessment of a particular implementation option, as we mentioned in the context of the MJF Study. The global KHP model consisted of nine different model views, contained in separate physical files. Therefore, a single propagation could potentially have to be propagated across nine files. An SD model for the entire global model, shown in Figure 7.17, gives a further indication of model size.

Model Slices. In order to deal with propagation across views, as well as the large model size, we attempted to divide the models into “slices”, as described by Marcel Leica (2005). This involves choosing one or more model elements of interest, typically the leaf nodes in an evaluation of a particular counseling technology, then extracting a bottom-up “slice” of the model by recursively tracing all links from these elements. This simple algorithm can be described by the pseudocode in Figure 7.18.

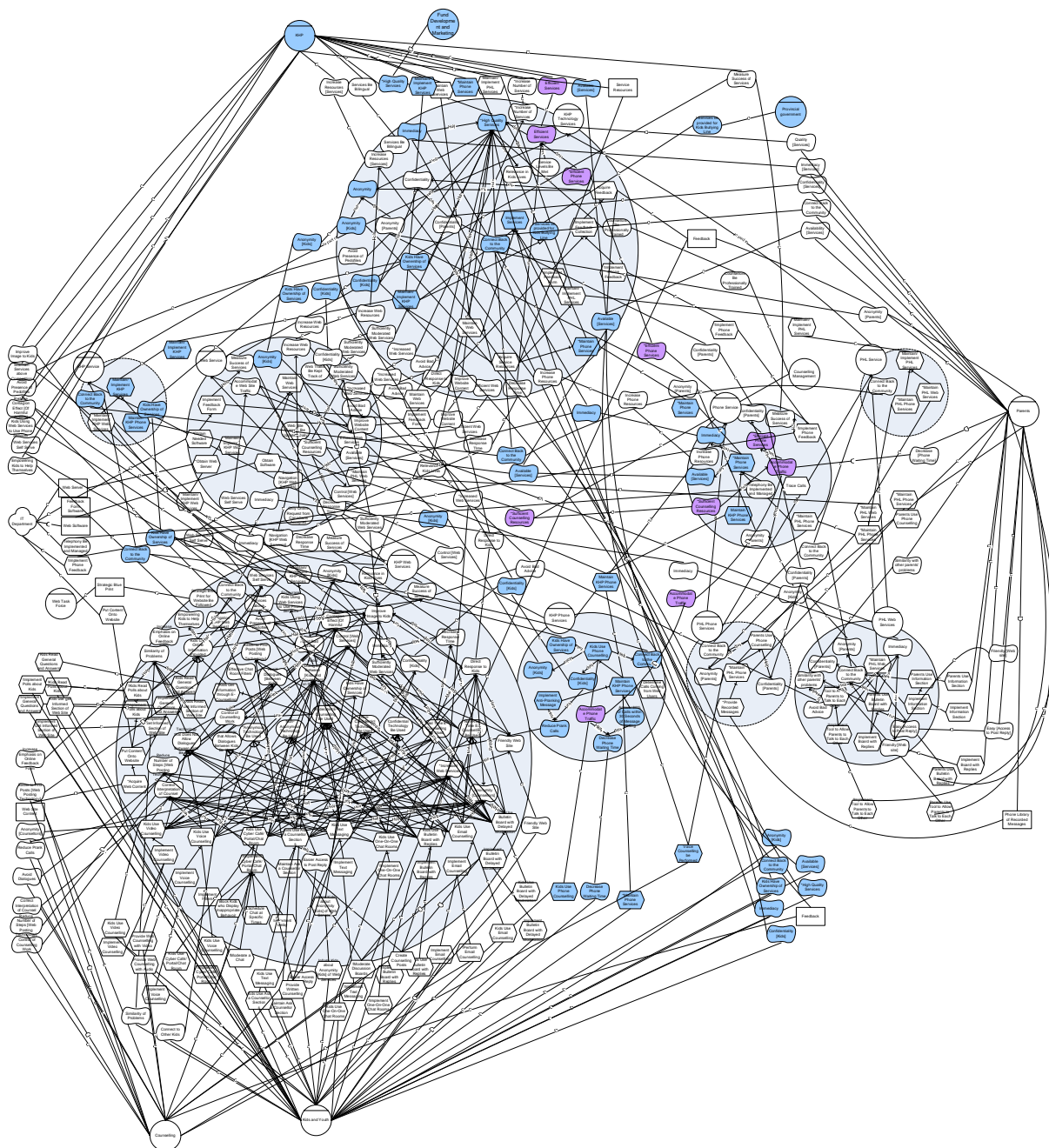


Figure 7.16: The Services Provided by Kids Help Phone
Note: The figure is intended to show the complexity and topology of a model produced in the case study; the text within is not meant to be legible.

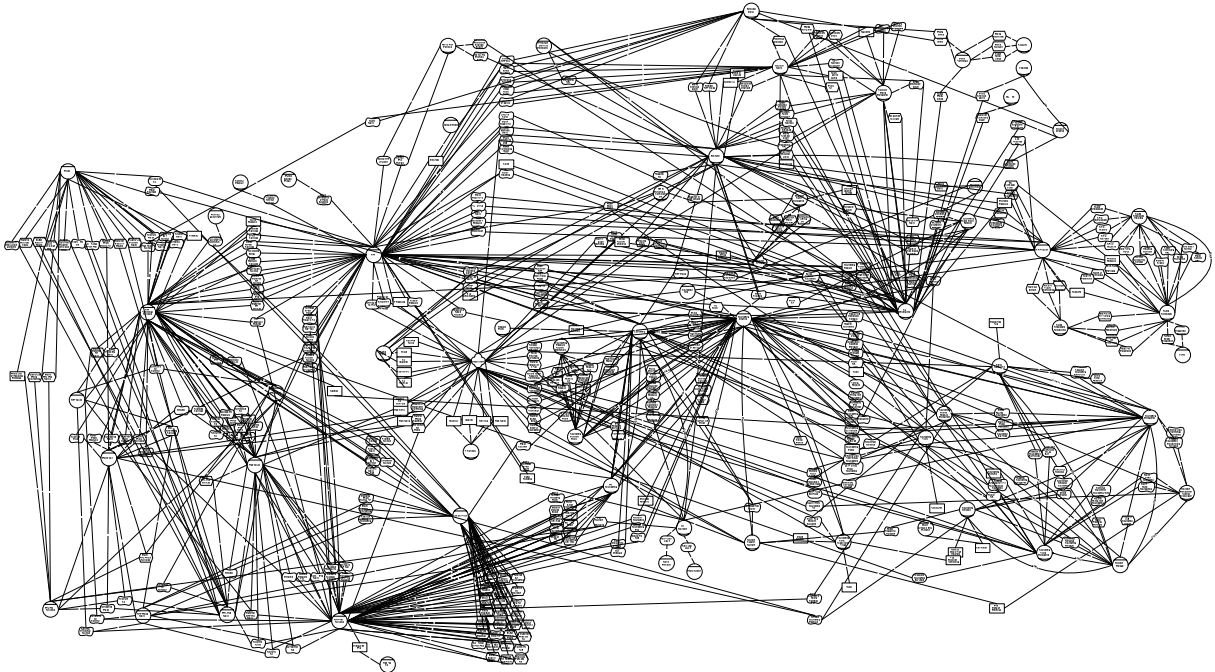


Figure 7.17: Kids Help Phone Global SD Model

Note: The figure is intended to show the complexity and topology of a model produced in the case study; the text within is not meant to be legible.

```

StartElements = all starting elements for slice
Slice = empty
For each element in StartElements
    GetSlice(element)

GetSlice (element) {
    Add element to slice
    For each link from element
        GetSlice(link.destinationElement)
}

```

Figure 7.18: Bottom-Up Slice Algorithm from (Leica, 2005)

Use of the slicing method across multiple views facilitated the creation of a subset of the global model which allowed global analysis of the effects of one technology alternative. However, manually creating such slices was difficult and time consuming, and the resulting slices were still large enough to make evaluation cumbersome. Figure 7.19 shows a slice of the global model for the purpose of evaluating the effects of using a bulletin board with delayed counselor moderation. Despite the difficulties with the

method, it shows promise in facilitating the evaluation of large models, especially if the creation of model slices were automated in an application such as OpenOME.

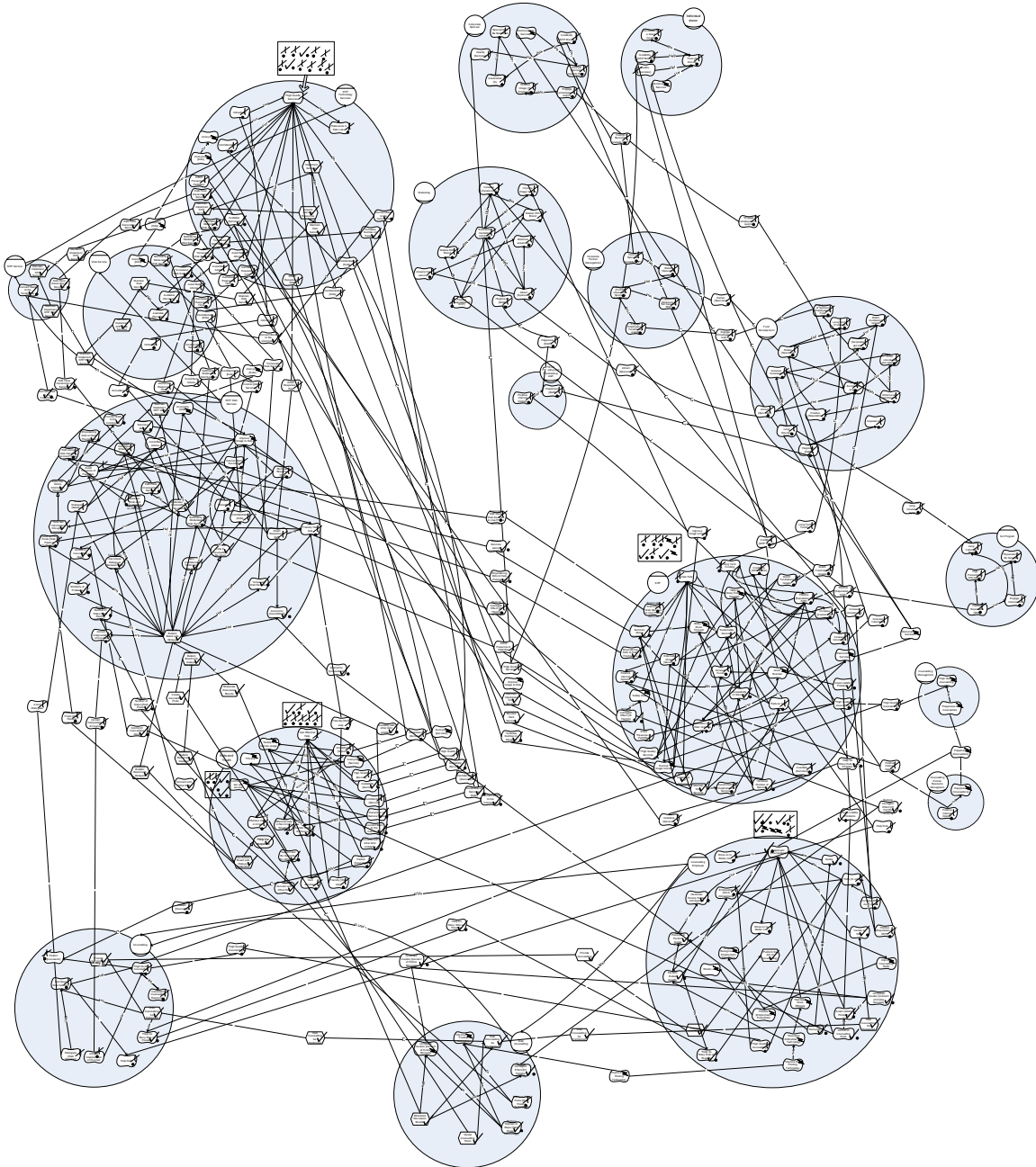


Figure 7.19: KHP Global Model Slice of the Figure 7.15 Model to Evaluate the Effects of a Delayed Moderation Bulletin Board

Note: The figure is intended to show the complexity and topology of a model produced in the case study; the text within is not meant to be legible.

Long Contribution Paths. Another issue discovered with the application of the evaluation procedure to very large models concerns the existence of very long contribution paths. In some cases, when a qualitative value is propagated along very long paths, the effects of the element originating this value on the elements that are towards the end of this path seem minimal, or non-existent. For example, by tracing through multiple evaluation links in the model slice shown in Figure 7.19, the implementation of a bulletin board with delayed moderation results in the partial denial of Increase Awareness (of KHP). This relationship is weak at best. However, there are cases where the effects of contributions propagated over long paths are valid. For example, implementing a bulletin board with delayed moderation has a negative effect on Retain Sponsors, as the lack of moderation may allow negative interaction between children, hurting the positive image of the KHP organization. These examples indicate that determining the relevance of evaluation values propagated through long paths should be included in the realm of human judgment, leaving it up to the modeler to determine whether or not the values are relevant. Of course, in order to do this effectively, some method of value traceability is needed. We will return to possibilities for traceability in *i** evaluation in Chapter 8.

Repetition of Softgoals across Actors. The models created in this stage of the KHP project provide extensive examples of the repetition of softgoals across actors. For example, in the view representing the Counseling components of the KHP organization the softgoals Help as Many Kids as Possible and High Quality Counseling are repeated four times, twice inside two different actors and twice as a dependum; The Improve Counseling Skills softgoal is also repeated four times, three times inside three different actors and once as a dependum. A high-level view of the entire model, with the repeated softgoals highlighted in pink, is shown in Figure 7.20, while Figure 7.21 contains a simplified version of this model showing only the repeated softgoals.

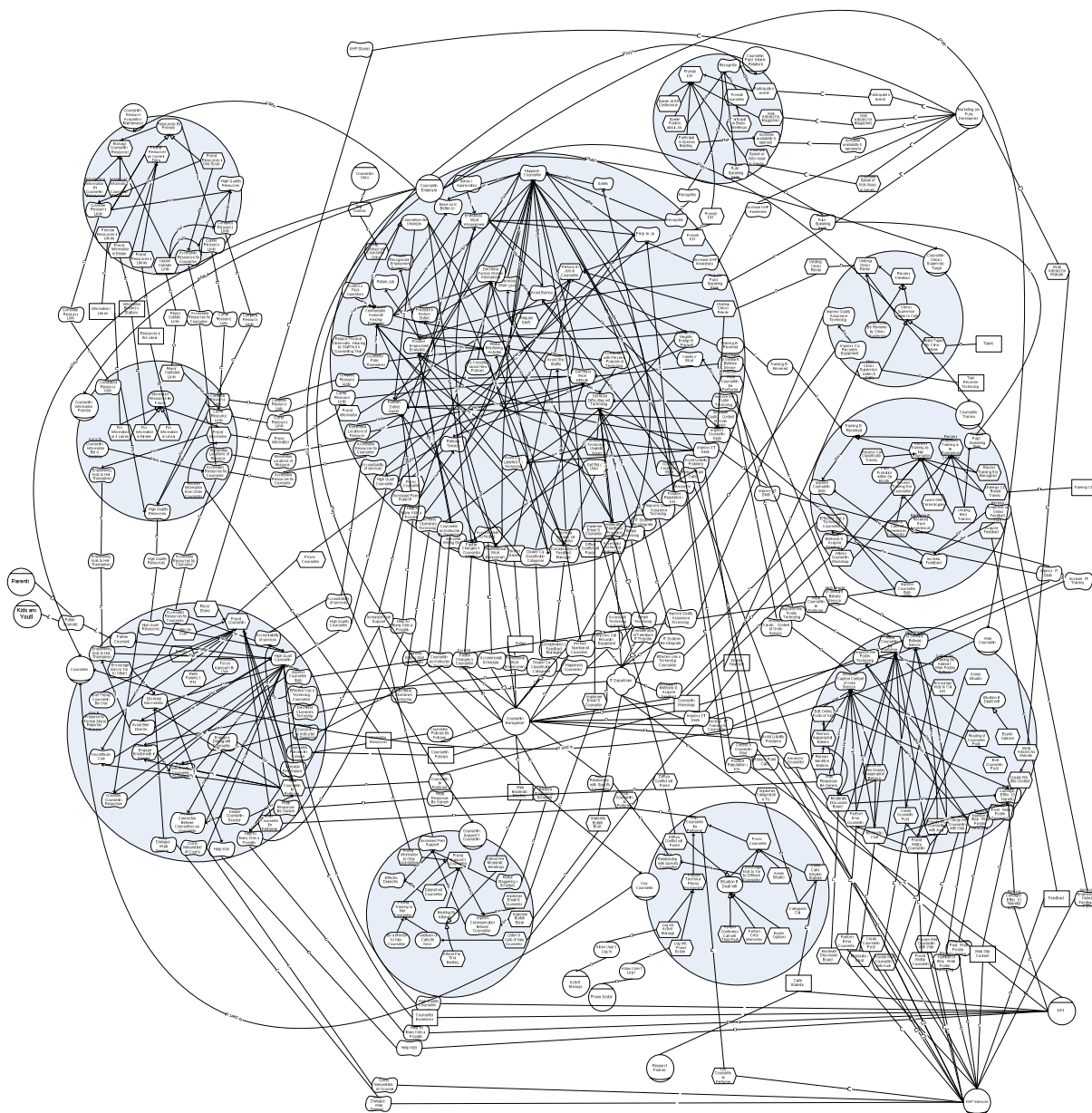


Figure 7.20: KHP Counseling Model

Note: The figure is intended to show the complexity and topology of a model produced in the case study; the text within is not meant to be legible.

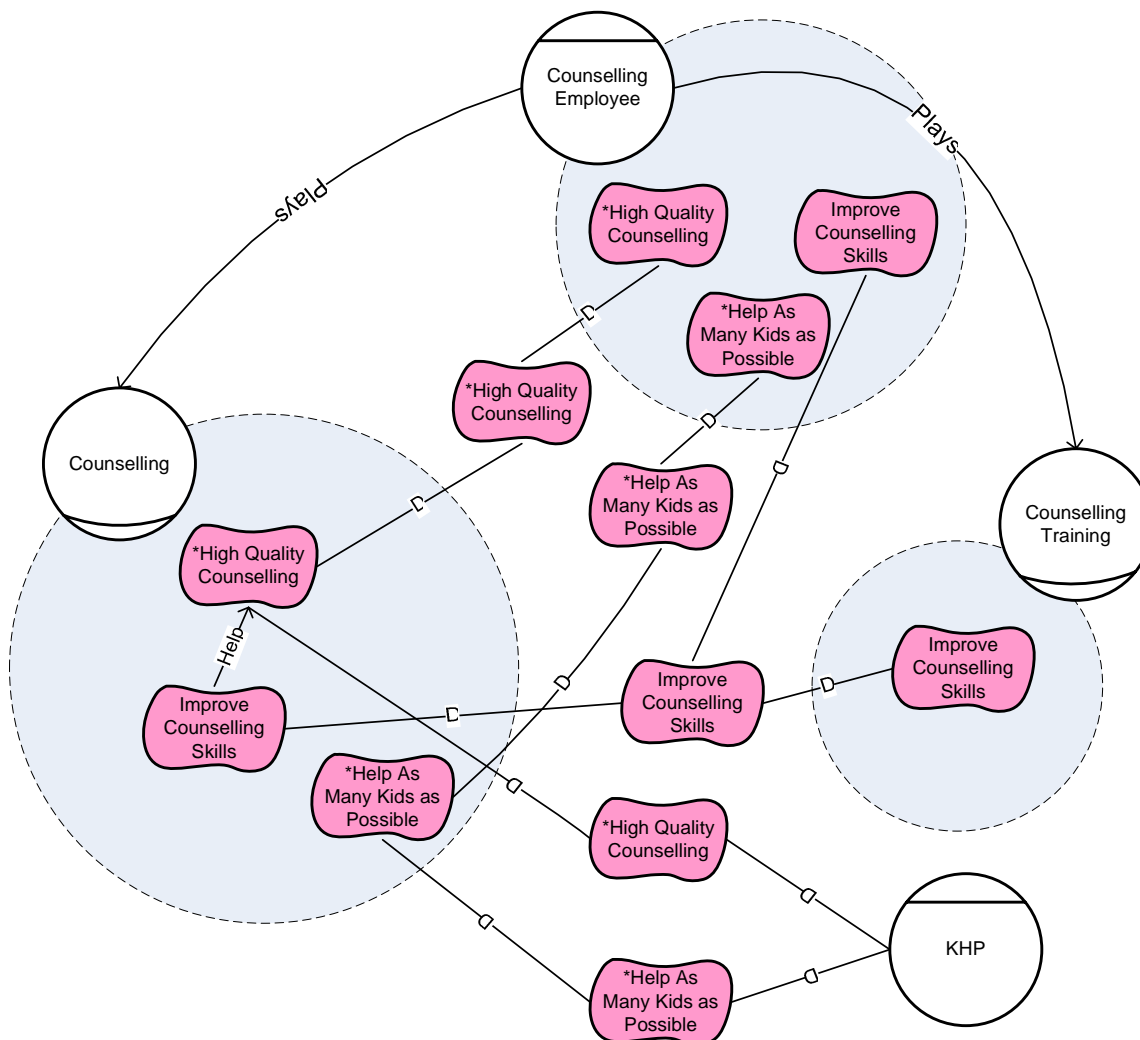


Figure 7.21: Reduced View of the KHP Counseling Model in Figure 7.19 Showing Repeated Softgoals

In the Figure 7.21 model, softgoals of the same name are included in multiple actors as it has been judged that each actor specifically desires the accomplishment of these goals. Although these goals are technically not identical, as they represent the accomplishment of a quality as per the criteria of a specific actor, they are still semantically linked. As in the MJF Example, the linking of such softgoals through dependencies aids in adding this semantic link to the model by propagating the effects of each softgoal to other softgoals of the same name. This situation was especially prevalent in the KHP study as we were modeling multiple, highly related roles within the same organization. These roles often shared the same high-level goals concerning the general mission of the organization. However, each individual role depended on or

affected these shared goals in unique ways. Future investigation into the representation of such shared goals using a clearly defined, systematic method requiring less graphical space would be beneficial.

7.6.2 Second Project Stage: Specification for Ask a Counselor Replacement using Prioritization

In the second stage of the Kids Help Phone Project, a single global *i** model focusing on the counseling process, especially on the process of web counseling through the Ask a Counselor System, was created. Our ultimate goal was to use the information contained within this model to create a specification for a replacement for the Ask a Counselor System, which was suffering from efficiency and user interface problems. In the pursuit of this goal, we created methods to extract scenarios and system features from this large model. It was our belief that our understanding of the processes involved in web counseling would benefit from explicit representation of these processes in scenarios. The potential for combination of the usage of goal models and scenarios has received particular attention in Requirements Engineering due to the complementary nature of the abstractness of goals and the concreteness of scenarios, see for instance (Rolland, Souveyet, & Ben Achour, 1998), (Antón, & Potts, 1998) and (Kavakli, Loucopoulos, & Filippidou, 1996).

7.6.2.1 Evaluation of Scenarios in *i**

It became apparent that the effectiveness of scenarios extracted from our *i** model could be evaluated in the model using the *i** evaluation procedure, and that the results of these evaluations could be used as a means of prioritization. Our intention was to use these prioritization results for scenarios derived from the model of the current Ask a Counselor System in order to drive the creation of a new model describing a potential future system. In Figure 7.22 we see the high-level view of the KHP counseling model created to represent the current phone and web counseling system. This model contains approximately 525 links and 350 elements, 230 of which represent quality criteria and

system goals, the rest of which represent specific tasks in the current system. In Figure 7.23 we see a closer view of the model showing a section of the detailed task structure for the Ask a Counselor System.

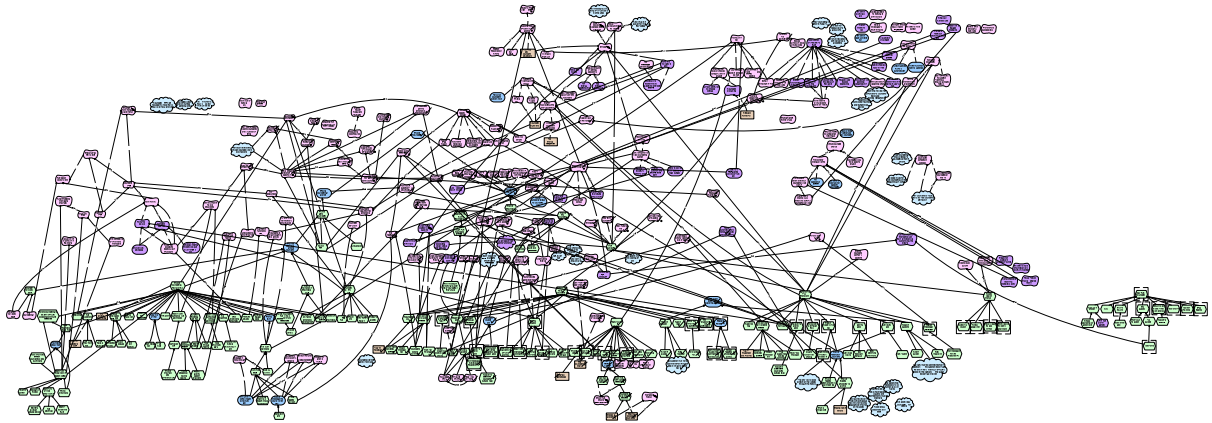


Figure 7.22: KHP Current System Counseling Model Created for the Second Stage of the KHP Project

Note: The figure is intended to show the complexity and topology of a model produced in the case study; the text within is not meant to be legible.

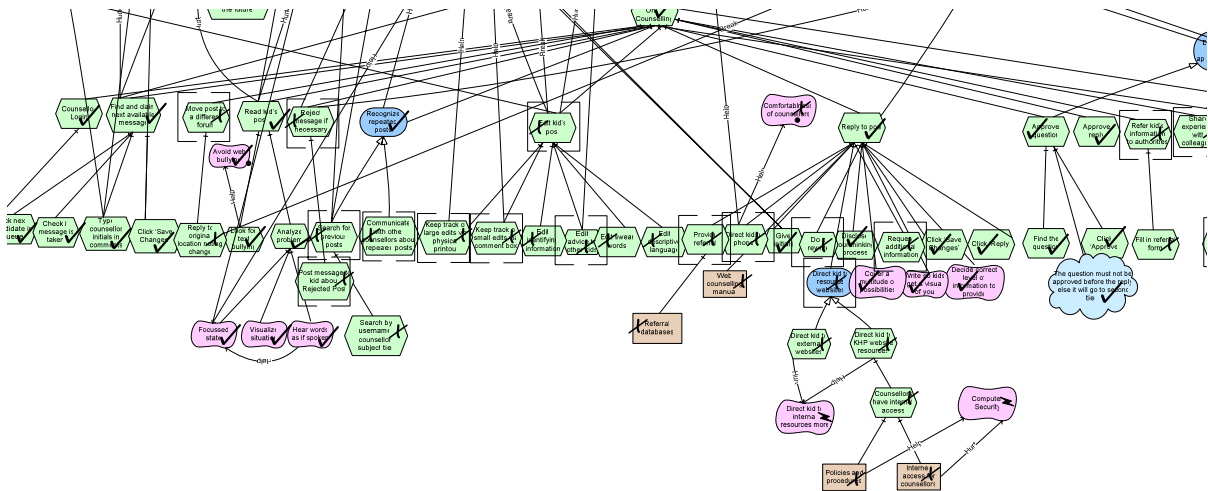


Figure 7.23: A Section of the KHP Counseling Model Created for the Second Stage of the KHP Project

Note: The figure is intended to show the complexity and topology of a model produced in the case study; the text within is not meant to be legible.

The rough temporal ordering of the detailed tasks describing the Ask a Counselor Processes facilitated our method of extracting scenarios from this model. We classified each of the tasks as either necessary or conditional, with the conditional tasks enclosed by parenthesis. All tasks for a particular subsection were then listed in textual form. This

list was treated as a “meta” scenario representing all possible paths for one overall purpose or Use Case. From this list multiple scenario instances could be derived by choosing whether or not certain optional tasks occurred. We see an example of this extraction in Figures 7.23 and 7.24. Figure 7.24 shows a small excerpt of the task structure from the counseling model in Figure 7.22 and 7.23. These tasks are converted into a text representation and marked with conditional information in Figure 7.25. In this particular list of tasks, containing three necessary tasks and four conditional tasks, six different scenario instances can be derived.

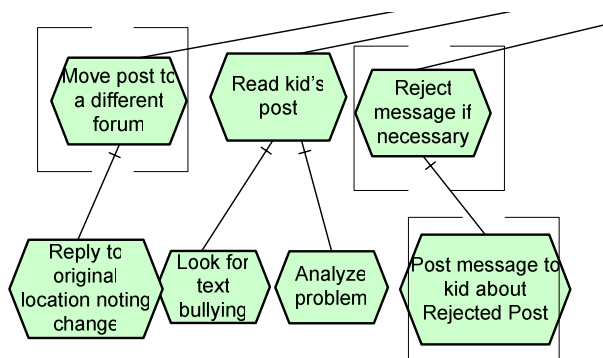


Figure 7.24: Part of the task structure in the i* model for the current system

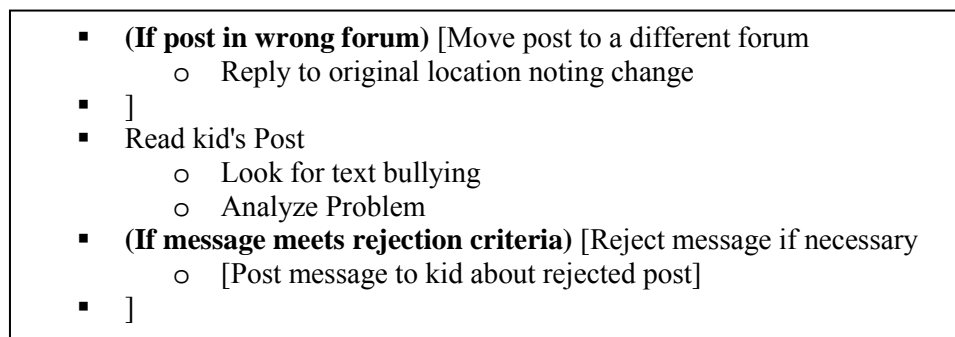


Figure 7.25: Part of the Meta- Scenario corresponding to Figure 7.22

In order to evaluate the effectiveness of individual scenarios in terms of the goals captured in Figure 7.22, we chose 25 scenario instances which we determined were controversial or interesting and evaluated these against the i* model using the evaluation procedure described in this work.

Evaluation of Scenarios. However, this evaluation was somewhat problematic. The evaluation of scenarios in a model representing an existing system had not been

attempted before and required some adjustments to the ideas involved in i^* evaluation. As we have explored, the i^* evaluation procedure is intended to evaluate the effects of potential design choices on the goals of system actors. In this project we were attempting to analyze the effectiveness of a current, not a potential, system. Therefore, we were trying to evaluate the effects of performing or not performing optional tasks. This turned out to be more complicated than originally envisioned, as the effect of a particular optional task can differ depending on the conditions. An example of this is the task Reject Message if Necessary, as shown in Figures 7.24 and 7.25. This task is considered optional, as messages are only rejected in some scenarios. However, the effects of rejecting or not rejecting messages will vary depending on the conditions of the rejection, divided into four cases: the message is rejected with the correct conditions, the message is rejected under incorrect conditions, the message is not rejected when the conditions were correct for rejection, and the message was not rejected when conditions for rejection did not apply. Each of the four situations could produce a different set of contribution links, for example the second and third case will likely cause problems for system supervisors, while the first and second cases will likely annoy users (kids), and the third case may have a negative effect on user Anonymity. In the context of this project, taking into account resource limits, the evaluation was performed focusing on the first and last cases, assuming that users followed the conditions correctly.

Evaluation Metric. The process of evaluating each of the 25 scenarios manually was extremely laborious, motivating the need for an automated evaluation procedure. Upon completion of the evaluation, it became apparent that, due to the large size of the model, a metric was needed in order to compare evaluation results. In typical i^* models of smaller sizes, a modeler can compare the evaluation results of two implementation options by visually comparing the results of certain key elements. However, when there are 300+ model elements, a manual comparison of evaluation results for key elements is difficult. To address this problem, we created a simple metric by converting each qualitative evaluation result into a number. This conversion is shown in Table 7.1. The numeric result for an evaluation then corresponded to the sum of these numbers for all quality criteria in the graph.

Table 7.1: Quantitative Values Assigned to Qualitative Evaluation Results

Evaluation Result	Satisfied	Partially Satisfied	Conflict	Partially Denied	Denied
Graphical Symbol	✓	✓•	✗	•✗	✗
Quantitative Value	1.0	0.5	0	-0.5	-1.0

The numerical evaluation results demonstrated by these numerical scores effectively treated each individual element as equally important. In reality, quality criteria within the model would have varying degrees of importance according to stakeholders. In a related project in the same domain, we collected prioritizations for the subset of the quality criteria relating to items on a new system wish list. We decided to incorporate this data into the i* model and the evaluation score by converting these relative measurements of importance into numerical weights. Elements that were not assigned a specific level of importance in the experiment were assigned a default importance value. The weights for each element were multiplied into the evaluation result before the summation to produce a new score. The overall score for each scenario is then given by:

$$\sum_{g=1}^n E_g \cdot P_g$$

where E_g is the numeric equivalent of the qualitative evaluation result for a goal g , P_g is the numeric value of the relative importance of goal g , and n is the number of goals. Table 7.2 contains the short names for each of the 25 scenario instances evaluated and their resulting evaluation score without (left column) and with individual goal priorities (right column).

Due to the accuracy issues in converting qualitative values directly into quantitative numbers, the application of a quantitative evaluation procedure would likely have been more appropriate. By applying such a procedure, we would have lost the ability to apply human judgment to the model; however, due to the very large size of the model, applying human judgment may not have been practical due to the potentially large

number of times human judgment is required, especially in automated evaluation. Therefore, in this case, it may have been reasonable to trade human intervention for full automation and increased numerical accuracy.

Table 7.2: Scenario Evaluation Scores for the Current KHP System

Scores without Individual Goal Importance Weights		Scores with Individual Goal Importance Weights	
Score	Scenario Name	Score	Scenario Name
-145.5	1.7 Edit Post	-140	1.7 Edit Post
-139.5	1.3 Move Post	-134	1.3 Move Post
-136.5	1.4 Reject Message	-131.5	3.1 All
-136.5	3.1 All	-131	1.4 Reject Message
-133.5	1.2 Min Actions	-128	1.2 Min Actions
-133.5	1.9 Authority Referral	-128	1.9 Authority Referral
-127.5	All of 4 but 4.5	-122.5	All of 4 but 4.5
-127.5	4.5 Post	-122.5	4.5 Post
-124.5	1.10 Share Experiences	-119	1.10 Share Experiences
-124.5	1.11 Feedback from Supervisors	-119	1.11 Feedback from Supervisors
-123	2.4 Move Post	-118	2.4 Move Post
-121.5	1.8 Reply with Extra Actions	-116	1.8 Reply with Extra Actions
-120	1.5 Search for Posts	-115	2.10 Reject
-120	1.6 Communicate about Repeat	-114.5	1.5 Search for Posts
-120	2.10 Reject	-114.5	1.6 Communicate about Repeat
-118.5	2.5 Edit Kid Post	-113.5	2.5 Edit Kid Post
-117	2.2 Min Actions	-112	2.2 Min Actions
-115.5	2.1 All Actions	-110.5	2.1 All Actions
-112.5	2.9 Edit Reply 2	-107.5	2.9 Edit Reply 2
-111	2.3 Search	-106	2.3 Search
-111	2.6 Return 1	-106	2.6 Return 1
-111	2.7 Return 2	-106	2.7 Return 2
-111	2.8 Edit Reply 1	-106	2.8 Edit Reply 1
-109.5	All Scenarios	-104	All Scenarios
-99	1.1: All Actions	-93.5	1.1: All Actions

Our intention was to use the results of the scenario evaluations for the current system to guide our design of the features of the new system. However, we found that it was easier to derive the functions of the new system by directly operationalizing the goals contained in the Figure 7.22 model. The process of considering each of the approximately 230 quality criteria for operationalization produced about 200 new or modified model elements, although some elements were repeated across more than one system section. From the existing 121 operationalizations in the current system model,

we removed 43, with the decision that one or more of the new operationalizations replaced their functionality. As a result of these additions and further changes, the final element count for the future system model was approximately 520 elements with 760 links. Our next step was to take the roughly 200 new elements with the remaining 78 operationalizations from the current system and place them in rough sequential order to form new system meta-scenarios. In this manner, we designed the new system using both a top-down and bottom-up method, using the scenario tasks remaining from the current system model as well as new tasks derived from quality criteria. The resulting counseling model is shown in Figure 7.26.

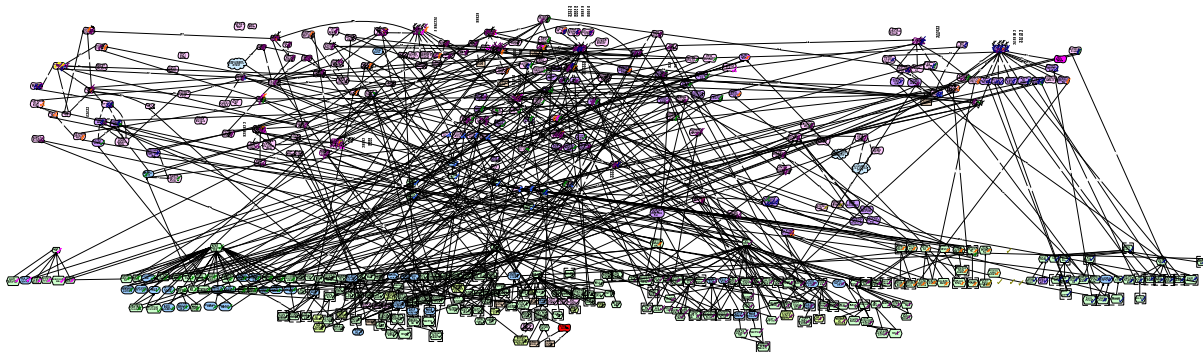


Figure 7.26: KHP Future System Counseling Model Created for the Second Stage of the KHP Project

Note: The figure is intended to show the complexity and topology of a model produced in the case study; the text within is not meant to be legible.

7.6.2.2 Prioritization of Features using i^*

At this point it became obvious that there were far too many functional elements to implement in the first version of the new system. As a result, in addition to the extraction of scenarios as had been done with the current system model, we grouped the functional elements into groups of related functionality, or features. A single task could be a feature in and of itself if it were considered to be at a high enough level of abstraction, with no other closely related tasks existing in the model. Specifically, we were looking for features whose implementation was not essential for system operation. From the Figure 7.26 we extracted 37 of these optional or non-essential features. Due to time limitations, we chose to evaluate the effectiveness of only 29 of these features. This

evaluation proved to be simpler than the evaluation of scenarios, as it involved the typical intentions of i* evaluation, to assess the effectiveness of a design choice, with the design choice being whether or not to implement a certain feature. Nevertheless, this process was still extremely time-consuming. We used the same system of calculating a numerical metric for feature evaluations as we had for scenarios in the current model. However, in order to produce scores with greater variance, we calculated the difference between the individual feature scores and a “base” score created by performing an evaluation with no optional features implemented. The numerical evaluation results for these features, with and without the incorporation of individual goal importance, are shown in Table 7.3.

Table 7.3: Priorities of future system optional features

Scores without Individual Goal Importance Weights		Scores with Individual Goal Importance Weights	
Score	Feature	Score	Feature
33.75	Link to Previous and Pending	27	Link to Previous and Pending
22.5	List of Logged in Moderators	26	Provide and Manage Best Answers
21.25	Provide and Manage Best Answers	21	List of Logged in Moderators
17.5	Feedback sections for each post	19.5	Personal Space for Kids
15	Counselor Space/Feedback	18	Question and Answer one entity
15	Automatic Message Assigning	17.5	Optional Public/Private Threads
13.75	Optional Public/Private Threads	16	Automatic Message Assigning
13.75	Question and Answer one entity	15	Sorting
13.75	Personal Space for Kids	14	Feedback sections for each post
12.5	Timeouts	13.5	Counselor Space/Feedback
11.25	Print	12	Automatically Save and View Edits
10	Sorting	10.5	Estimate Response Time
10	Automatic Moving Notice	10	Automatic Moving Notice
7.5	Automatically Save and View Edits	10	Timeouts
7.5	Auto save	9	Print
7.5	See Messages in Both Tiers	7	Spellchecking
6.25	Blank Forum	7	See Messages in Both Tiers
6.25	Spellchecking	6	Auto save
6.25	Language viewing control/filter	6	Post Archiving
6.25	Estimate Response Time	5	Language viewing control/filter
5	Post Locking	4	Post Locking
5	Post Archiving	4	Country Filtering
5	French moderator view	3	Blank Forum
3.75	Show Relevant Internal Links	3	Show Relevant Internal Links
3.75	Country Filtering	3	Record of Counselor Picks
3.75	Record of Counselor Picks	2	French moderator view
2.5	Prompt kid for updates	2	Prompt kid for updates
-2.5	Case Files	0	Case Files

These scores were used to produce a relative prioritization of feature importance. Modified versions of the scenarios and features from the new system were presented to various KHP counselors and supervisors as a form of validation for our high-level design. With the optional scenarios, we presented a conversion of our prioritization scores into ordinal categories of importance (Very High, Medium High, Medium, Medium Low and Low). Specific feedback from these interactions was used to modify details of the features and scenarios. In addition, we collected a survey concerning the accuracy of our prioritization of the optional system features from four KHP stakeholders. The survey results indicated whether or not the importance category of each optional feature was correct, adjusting our classification to the correct category according to the opinion of that particular stakeholder. From these results, we calculated the average of the number of features which were adjusted a certain distance by the stakeholders: no adjustment, off by one, off by two or off by three. We calculated the adjustment distance by comparing to our prioritization with and without individual goal importance, with the results shown in Table 7.4. These results were generally promising, with 50% of the categories judged to be correct by the stakeholders for our categorizations including individual goal importance.

Table 7.4: Stakeholder Priority Category Adjustment

Stakeholder Adjustment	With Individual Goal Importance		Without Individual Goal Importance	
	# Feat.	%	# Feat.	%
None	14	50.00	10.75	38.39
Off by 1	4.25	15.18	7.5	26.79
Off by 2	4.75	16.96	5	17.86
Off by 3	2.25	8.04	3.5	12.50
Off by 4	0.75	2.68	0.75	2.68

Finally, the modified scenarios and features, the prioritization of optional features as adjusted by the stakeholders, as well as the elements within the model were used in the production of a requirements specification for a new online counseling system. This system is currently being implemented by a team of undergraduate students as part of a project for course credit (<https://www.cgi.cdf.utoronto.ca/~cs494hf/cgi-bin/argon.cgi/helpphone/wiki>).

During the execution of this project, we determined that the method of extracting feature prioritizations from the information contained within *i** models using *i** evaluation constituted a valuable contribution to research in both goal modeling and requirement prioritization. Previous work had focused on effective methods of requirement or feature prioritization, see for example (Karlsson, & Ryan, 1997) and (Park, Port, Boehm, & In, 1999), but none of the methods introduced in these methods explicitly used the information contained within goal models. As a result, we abstracted our process to develop a general methodology for deriving feature priorities from *i** models. This methodology is described in greater detail in (Horkoff, Aranda, Easterbrook, & Yu, 2006). Figure 7.27 provides a graphical summary of the steps involved in this method.

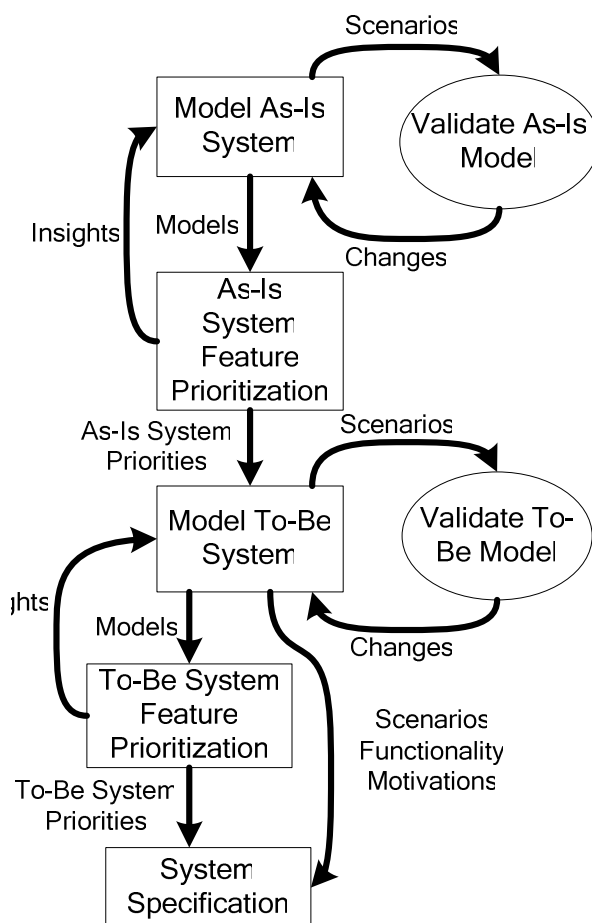


Figure 7.27: Prioritization Methodology

7.6.2.3 Model and Evaluation Challenges and Solutions

While completing the second stage of the KHP project, multiple issues concerning evaluation arose. We have already mentioned the confusion with evaluating a scenario or process as opposed to an implementation option. More investigation is needed into determining how the evaluation of scenarios in i^* models can be performed with some accuracy, producing useful results which can be interpreted in sensible ways.

Model Size Hindering Evaluation. In addition, the problems involved in creating and using models of large sizes have been mentioned extensively. One may notice that none of the i^* models in the second stage of the KHP project contained actors. This was due in part to the graphical difficulty of fitting 300+ model elements into the circles representing actor boundaries. The previously explored issues of repeating softgoals across actors also contributed to this difficulty, as much of the quality criteria would have had to have been repeated in more than one actor and linked via a dependency, further increasing the size of the model.

Model Layering. In order to help enable manual evaluation of the very large models, the elements and links within the model were divided into conceptually related layers. For example, the model shown in Figure 7.26 had nine element layers including, for example, a layer with the elements for Kids, a layer containing the quality criteria, and a layer for each of the two “tiers” in the system. An example view of one of these layers, the layer for tier two, is shown in Figure 7.28.

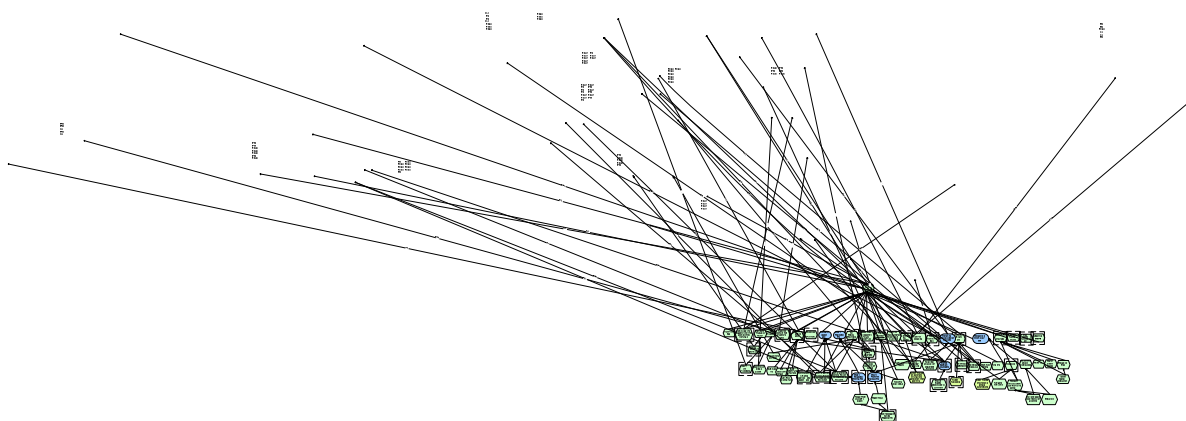


Figure 7.28: The Tier Two Layer of Figure 7.26

These layers were used in evaluation to make it easier to see and trace the individual contribution links. Evaluation values were propagated separately in each layer, with the evaluation values for each layer represented in a different color. Once the values for each layer had been propagated, the final values were combined together into one overall value. While this method did provide a means to manage the large size of the model, combining the results of each layer into an overall evaluation value was problematic. These results could not be treated as a regular bag of contributed values, such as in the evaluation algorithm described in Chapter 4, as multiple layers may contribute the same evaluation value through the same links. For example, in Figure 7.29 we see the combined evaluation results of all of the layers. Although the three softgoals at the top of the diagram have only one incoming link, they each have incoming evaluation values from multiple layers. Therefore, the number of links to an element must be closely considered when manually determining the final values (in this case partially denied for all three softgoals). Although the formation of model layers shows promise as a means to deal with large model sizes, a more effective method of layers that avoids these difficulties is needed. Perhaps the layers need to be created based on model structure instead of semantic relatedness, as is done when creating model slices based on contribution links.

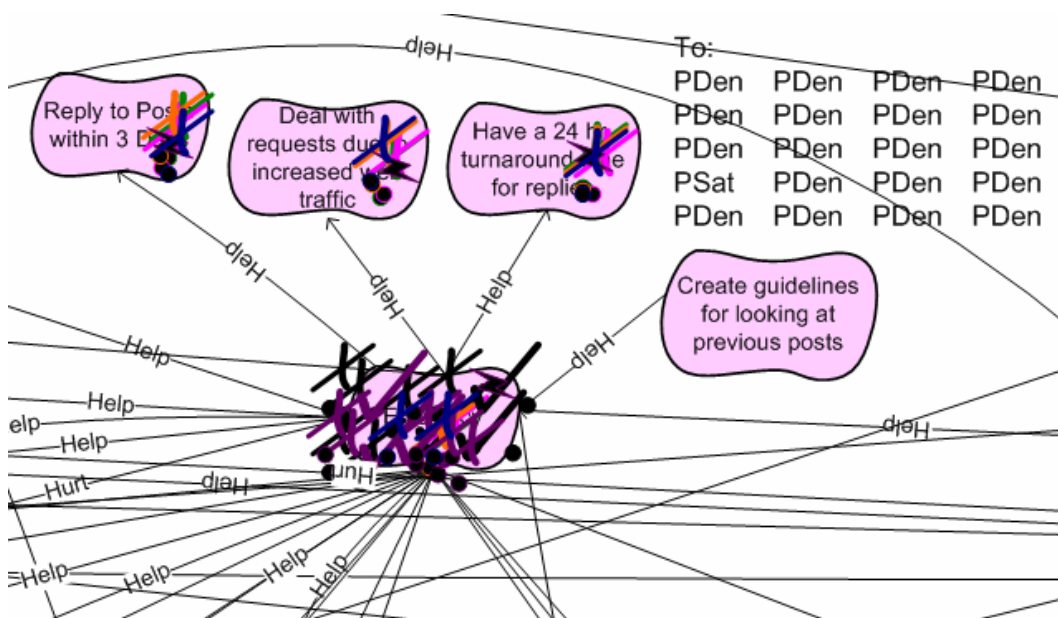


Figure 7.29: A Section of the KHP Future System Counseling Model shown in Figure 7.24

Placing Multiple Labels on a Single Element. In some cases, when the number of links to an element was large, such as for the Efficiency softgoal in Figure 7.29, (the element covered by evaluation values), it is difficult in this case to make a judgment by carefully considering the sources of the labels, as there are many incoming links (in this case 25). In cases where the number of incoming evaluation values are large, and evaluation is done by hand, instead of mentally considering the bag of incoming labels, these labels can be temporarily drawn on the element, then combined together into one label when the judgment for the final label has been finished. One can see this in Figure 7.29 for the Efficiency softgoal, as there is more than one evaluation value from each of the different layers, as indicated by the presence of more than one value of each color.

Tables to assist Human Judgment. In the case of Efficiency, the number of incoming contributions was so large that the mechanism of drawing multiple labels was not sufficient to support human judgment, and a table had to be created to keep track of all of the incoming evaluation values, such as is shown in the top right corner of this Figure. In the automated version of the evaluation procedure, as described in Chapter 6, the former problem would be solved, as a table would be created for the user containing a list of sources for the contributed evaluation values. However, even in this case, carefully considering each of the many sources in human judgment is difficult. As mentioned in Chapter 5, Section 5.2.2, this is likely a case where softgoal refinement is necessary, exploring the different types of Efficiency.

The Affects of Link Completeness. When evaluating large models it became clear that evaluation could serve as an indication of the connectivity of a model, illuminating areas of the model which were isolated. This idea was previously mentioned in Chapter 5, Section 5.3.1 when referring to semantic validation. In the current system model in Figure 7.22, when all operationalizations were evaluated as satisfied, only 62% of the quality criteria received evaluation values. As a result, we can see that if the intention of the modeler is to create a completely connected graph, with a sufficient completeness in links, application of the evaluation procedure can be used to indicate areas of the model with potential deficiencies in links. The possibility of such deficiencies raises questions concerning how link deficiencies would affect the accuracy of evaluation results. For instance, in large models such as Figure 7.22 and 7.26, it is

likely that many potential links have been overlooked. It is possible that the omission of such links has affected the overall evaluation results, but how significant is this effect? For elements which receive very few incoming values, such as Have a 24 hour turnaround for replies in Figure 7.29, the effect of a missing incoming link may be significant. Therefore, the omission of a link to an important element which has only a few incoming links could make a noticeable difference in evaluation results. However, for the Efficiency softgoal in the same Figure, the addition of one or two more incoming links to a collection of 25 incoming links is not likely to produce significant changes in the resulting evaluation value. In this case the addition of new links may be “drowned” by the existence of many pre-existing links. However, it is typically important elements, such as Efficiency, which have many incoming links, and are therefore not significantly affected by the omission of a few potential links. Future investigation involving a comparison of evaluation results with varying levels of link completion is needed to investigate the importance of this issue.

7.6.3 Future Project Directions

It is the intention of the KHP research team to carry on with studies in this domain. Specifically, we would like to reapply the methodology described in Figure 7.27 to a new system within Kids Help Phone providing for Knowledge Management needs. During this project, we intend to use the automated i* evaluation procedure, in order to continue to test the viability of this procedure and its implementation.

7.7 Conclusion

In this Chapter we have described the work involved in five case studies, focusing specifically on the impact these case studies had in forming the evaluation procedure as described in this work. By examining the role of evaluation in these case studies we can see many areas where further investigation would be beneficial, such as the comparison of conflicting evaluation results, the sharing of elements between actors, and the effects of link completeness on evaluation. In particular, it is clear that effective methods are

needed in order to deal with large models. These methods should address not only the evaluation of large models, but also creation, modification, and comprehension. Despite the issues concerning model scalability, the application of the evaluation procedure to these multiple complex studies have demonstrated its viability as a useful tool for domain analysis, including its effective Accuracy (iv) and Usefulness in Multiple Contexts (v) In addition, we have shown the potential for model improvement, as well as for increasing and sharing knowledge concerning the domain.

Chapter 8: i* Evaluation Procedure Assessment

8.1 Introduction

In this work we have illustrated the need for a systematic evaluation procedure for the i* Framework, we have described the desired qualities that such a procedure should possess, and we have outlined and implemented a procedure which is intended to effectively address these qualities. Based on the definition, implementation and application of the procedure described in this work, we will perform an assessment of how well the procedure addresses the desired qualities we have defined at the outset of this study.

When choosing a goal model evaluation procedure to adapt for use with the i* Framework, we selected the CNYM over other procedures for its Allowance for Human Intervention (iii) and support of an improvement in model quality (xiii, xiv). As a means of validating this selection and further assessing the procedure developed from this selection, we shall return to the other goal model evaluation procedures and expand them for i* application. The application of these procedures and the procedure produced in this work shall be compared in order to compare the relative suitability of each procedure for i*.

8.2 Achieving the Desired Qualities of an i* Evaluation Procedure

In Chapter 2, based on our application of i* to extensive case studies, as described in Chapter 7, we have described fourteen qualities and sub-qualities which should be effectively addressed by an i* evaluation procedure. We shall review how the procedure described in this work successfully balances these desiderata.

(i) Element Evaluation. Adopting the conventions of the CNYM procedure, the i* evaluation procedure uses a set of six qualitative labels to represent the achievement and denial of an element, given the structure and state of achievement of the rest of the

model. This includes the ability to represent intermediate evaluation results, storing multiple sources evidence until a resolution can be made.

(i.a) Evaluation Value Expressiveness. Although our chosen evaluation procedure does not provide a quantitative range of evaluation values, such as the GMNS-# procedure, we have determined that due to the frequent lack of concrete or numerical measures in the early, high-level analysis for which i^* is intended, the qualitative values offered by the procedures offer sufficient expressiveness. In fact, adding expressiveness through the introduction of a more fine-grained label system would greatly increase the complexity of the procedure, especially in regards to the propagation rules.

(i.b) Overall Evaluation Value. The Allowance for Human Intervention (iii) allows multiple sources of evidence, including conflicting evidence, to be resolved into an overall indication of element achievement, facilitating informative evaluation results.

(ii) Clear Procedural Guidelines. We have provided an in-depth description of the guidelines for the procedure, including the propagation of evaluation values throughout an i^* model. We have included guidelines for dealing with more complex syntactic constructions such as a mixture of links and links to other links. The guidelines include the situations where human intervention from the user is explicitly required.

(iii) Allowance for Human Intervention. The i^* evaluation procedure developed in this work includes a detailed description of the role of human judgment in this procedure. This judgment is explicitly needed in cases with partial or conflicting evidence. In addition, such judgment can be applied in exceptional cases to modify the propagation guidelines when the modeler has not been able to include tacit domain knowledge required for evaluation in the model, likely due to scalability concerns.

(iv) Accuracy. By applying the i^* evaluation procedure to multiple case studies in Chapter 7, we have begun to demonstrate its ability to produce results which are sufficiently accurate as per real phenomenon in the domain. In this case the “sufficiency” of the results is judged by their ability to match real-life experiences, such as the denial of consumer Privacy in the domain of E-Commerce, and by their ability to illuminate interesting facets of the domain, such as the effects of Lock-In on Technology Users decision to Purchase Technology. Future studies which further confirm the accuracy of this procedure are needed.

(v) Usefulness in Multiple Contexts. The application of the evaluation procedure to case studies also works towards confirming the ability of the procedure to be useful on multiple contexts. Our successful application of the procedure has covered the contexts of Requirements Engineering (KHP), Software Development (KHP), Business Process Analysis (KHP, MJF), Identity Management (E-Commerce), Security (TC), Trust and Privacy (Privacy in E-Commerce and TC).

(vi) Modes of Analysis. Due to tradeoffs involving our desire to Allow for Human Intervention (iii) and for Simplicity (xi), the evaluation procedure described in this work currently only provides functionality for a bottom-up form of analysis. This tradeoff is reasonable as we have indicated that the bottom-up direction is more appropriate for the early analysis facilitated by i^* , as it provides the capability to explore high-level design alternatives.

(vii) Traceability, (viii) Conflict Detection, (ix) Constraints on Values, and (x) Facilitating Cost Analysis. Our evaluation procedure does not explicitly address the need for traceability, conflict detection, constraints on values or costs analysis. However, we have indicated that these qualities are not essential for i^* analysis, although they may be beneficial for some users. The primary motivations for excluding these aspects are Simplicity (xi) and feasibility. It is clear that defining conventions to provide for these qualities would greatly increase the complexity of the procedure. Furthermore, the addition of some of these qualities to i^* evaluation are not straightforward. How do we provide for traceability when human intervention determines the overall values for some elements? How do we analyze cost when the cost of most high-level elements cannot be measured in the same scale? We shall further address some of these issues in the when we consider the future incorporation of these qualities in Chapter 9.

(xi) Simplicity. As we believe simplicity is essential for a wide-spread adoption of an evaluation procedure, we have placed a particular emphasis on this quality. In this light, we have limited the labels of goal achievement to six qualitative labels, we have endeavored to adopt the intuitive nature of the CNYM propagation rules, we have clearly defined the cases where human intervention is required, and we have avoided the addition of optional qualities such as top-down evaluation, traceability, conflict detection, constraints on values, and cost analysis. By making this effort and including these

tradeoffs we have successfully developed a procedure which is simple, and yet provides the essential qualities necessary for i^* evaluation.

(xii) Automation. Although we have achieved only partial automation due to our Allowance for Human Intervention (iii), our work in Chapter 6 has demonstrated that implementation of our evaluation procedure is viable. Furthermore, disregarding the time needed to Allow for Human Intervention (iii), which can vary highly, our implementation has a reasonable running time, making it viable for real use.

(xiii) Syntax Checking and (xiv) Semantic Improvement. In Chapter 5 we have explored the capability of the i^* evaluation procedure developed in this work to encourage model quality improvement through syntax checking and semantic improvement. We have shown that the careful consideration of the model prompted by the application of human judgment and the examination of evaluation results often leads to the discovery of syntax errors and semantic faults. Concerning syntax, the changes prompted by evaluation do not always involve errors, but instead an expansion of syntax which facilitates a clearer application of evaluation, but which may also increase the complexity of the model. Further investigation into the usefulness of this expanded syntax is needed. In terms of semantics, we have shown that the iteration prompted by evaluation not only improves the correspondence of the model to the domain, but causes a careful consideration of domain aspects which promotes learning and can guide further elicitation.

In general, we have developed an evaluation procedure which effectively balances all of the desired criteria, addressing those which are most critical, and laying the groundwork for addressing secondary beneficial qualities in future work. As a result, the procedure facilitates useful analysis, and has great potential for extensive application.

8.3 Assessing Goal Model Evaluation Procedures Adapted for i^*

In order to further assess and confirm the appropriateness of the evaluation procedure developed in this work for application to i^* , we shall compare this procedure

to procedures developed by adapting other goal model evaluation procedures for use with i^* .

8.3.1 Adaptation of Goal Model Evaluation Procedures for the i^* Framework

Just as the CNYM evaluation procedure for the NFR Framework was adapted for use with the i^* Framework in Chapter 4, other goal model evaluation procedures explored in Chapter 3 such as GMNS and GMNS-# from (Giorgini, Mylopoulos, Nicchiarelli, & Sebastiani, 2004) and SGM-TD from (Giorgini, Mylopoulos, & Sebastiani, 2004) could also be adapted. In fact, many of the propagation rules introduced in Chapter 4 can be reused in the adaptation of these methods. For instance, the treatment of decomposition links as an And relationship, the treatment of means-ends links as Or, and the direct propagation across dependency links can be retained as sensible rules for procedure expansions.

In terms of propagation across contribution links, the goal model procedures mentioned include rules for such links, although the exact types of these links are not identical. The goal model syntax used in the qualitative version of these methods includes the following types of links: *and*, *or*, *+*, *-*, *++*, *--*, *+s*, *-s*, *++s*, *--s*, *+d*, *-d*, *++d* and *--d*. In this case *+* and *-* indicate a partial positive and negative contribution, corresponding sufficiently to i^* 's notion of *Help* and *Hurt*. *++* and *--* indicate a full positive and negative contribution, corresponding sufficiently to Make and Break. As propagation in i^* treats *Some+/Help* and *Some-/Hurt* as having the same effect, we shall map both of these links to *+* and *-*, respectively. In these conventions *s* and *d* indicate a non-symmetric contribution, with only the positive evidence (*s*) or only the negative evidence (*d*) propagated. As i^* does not support non-symmetric links at this point, we ignore these conventions and use only symmetric contributions. Based on these adaptations, we can produce a table of contribution rules for the qualitative GMNS and SGM-TD procedures as applied to i^* models, as shown in Table 8.1.

Table 8.1: GMNS and SGM-TD Propagation Rules Adapted for the i^* Framework

	And/Decomposition $(G_2, G_3) \Rightarrow G_1$	Or/Means-Ends $(G_2, G_3) \Rightarrow G_1$	Decomposition $G_1 \text{ --D-- } G_2$	
Sat (G ₁)	$\min \{\text{Sat}(G_2), \text{Sat}(G_3)\}$	$\max \{\text{Sat}(G_2), \text{Sat}(G_3)\}$	$\text{Sat}(G_2)$	
Den (G ₁)	$\max \{\text{Den}(G_2), \text{Den}(G_3)\}$	$\min \{\text{Den}(G_2), \text{Den}(G_3)\}$	$\text{Den}(G_2)$	
	Help/Some+ $G_2 \Rightarrow G_1$	Hurt/Some- $G_2 \Rightarrow G_1$	Make $G_2 \Rightarrow G_1$	Break $G_2 \Rightarrow G_1$
Sat (G ₁)	$\min \{\text{Sat}(G_2), P\}$	$\min \{\text{Den}(G_2), P\}$	$\text{Sat}(G_2)$	$\text{Den}(G_2)$
Den (G ₁)	$\min \{\text{Den}(G_2), P\}$	$\min \{\text{Sat}(G_2), P\}$	$\text{Den}(G_2)$	$\text{Sat}(G_2)$

Concerning the quantitative propagation of the GMNS-# and SGM-TD-# procedures, we can adapt the quantitative propagation rules provided for these procedures using the same conventions as in Table 8.1. As the reader may recall, these rules make use of the \otimes and \oplus operator, with \otimes defined as multiplication and \oplus defined as:

$$x \oplus y = x + y - x \cdot y$$

These rules require contribution links to be labeled with a quantitative strength, as opposed to the typical i^* ordinal strengths (Make, Help, Hurt, Break), represented by the variable w . The resulting rules, adapted from (Giorgini, Mylopoulos, Nicchiarelli, & Sebastiani, 2004), are shown in Table 8.2.

Table 8.2: GMNS-# and SGM-TD-# Propagation Rules Adapted for the i^* Framework

	And/Decomposition $(G_2, G_3) \Rightarrow G_1$	Or/Means-Ends $(G_2, G_3) \Rightarrow G_1$	Decomposition $G_1 \text{ --D-- } G_2$	
Sat (G ₁)	$\text{Sat}(G_2) \otimes \text{Sat}(G_3)$	$\text{Sat}(G_2) \oplus \text{Sat}(G_3)$	$\text{Sat}(G_2)$	
Den (G ₁)	$\text{Den}(G_2) \oplus \text{Den}(G_3)$	$\text{Den}(G_2) \otimes \text{Den}(G_3)$	$\text{Den}(G_2)$	
	Help/Some+ $G_2 \text{ =(w)> } G_1$	Hurt/Some- $G_2 \text{ =(w)> } G_1$	Make $G_2 \text{ =(w)> } G_1$	Break $G_2 \text{ =(w)> } G_1$
Sat (G ₁)	$\text{Sat}(G_2) \otimes w$	$\text{Den}(G_2) \otimes w$	$\text{Sat}(G_2)$	$\text{Den}(G_2)$
Den (G ₁)	$\text{Den}(G_2) \otimes w$	$\text{Sat}(G_2) \otimes w$	$\text{Den}(G_2)$	$\text{Sat}(G_2)$

When adapting the CNYM procedure for i^* , consideration of additional syntactical constructs, such as a mixture of link types to a single node and links to other links, was needed. Similarly, a definition of how the goal model evaluation procedures will deal with these situations is required. In expanding CNYM to deal with a mixture of links, we used an And relationship for a mixture of links to a non-softgoal. For a mixture of links to a softgoal we added the results from the other links to the softgoal resolution bag. In the algorithms under consideration here, the lack of human judgment input means

that there is no softgoal bag. When multiple results arrive to a single softgoal, the minimum result is taken (the maximum for negative values). In this light, the results from all mixtures of links shall be combined using an And relationship, as is shown in the first three rows of the second column of Table 8.1 for qualitative values and the first three rows of the second column of Table 8.2 for quantitative values.

Regarding links to other links, as links in i^* are symmetrical, the results of the Sat and Den values propagated onto a link will have to be combined into one value, similar to the single value in the CNYM procedure. We consider the results of combining positive and negative evidence in the GMNS qualitative procedure in Table 8.3. These values can be applied to the recipient link in a manner similar to the rules outlined in Chapter 4, Section 4.3.3.6. In Table 8.4, we outline the effects of this combined qualitative evidence on the recipient i^* links, with the columns of Table 8.4 described in Figure 8.1.

Table 8.3: Combining Qualitative Evidence in the GMNS Procedure

Sat Value Sat (G_1)	Den Value Den(G_1)	Combined Result	Equivalent i^* Evaluation Value
F	F	Strong Conflict	Conflict
F	P	PS (Partially Satisfied)	Partially Satisfied
F	N	FS (Fully Satisfied)	Satisfied
P	F	PD (Partially Denied)	Partially Denied
P	P	Weak Conflict	Conflict
P	N	PS (Partially Satisfied)	Partially Satisfied
N	F	FD (Fully Denied)	Denied
N	P	PD (Partially Denied)	Partially Denied
N	N	N	No Label

Table 8.4: Propagation Rules for Links to Links in the GMNS and SGM-TD Adaptations

Contributing Label	First Link Contribution	Actual Contribution of First Link
FS or PS	Make	No Change
	Some+/Help	
	Some-/Hurt	
	Break	
N	Anything	No Change
Any Conflict	Anything	No Change
PD	Make	Some+
	Some+	Help
	Help/Hurt	None
	Some-	Hurt
	Break	Some-
FD	Anything	None
Anything	Unknown	Unknown

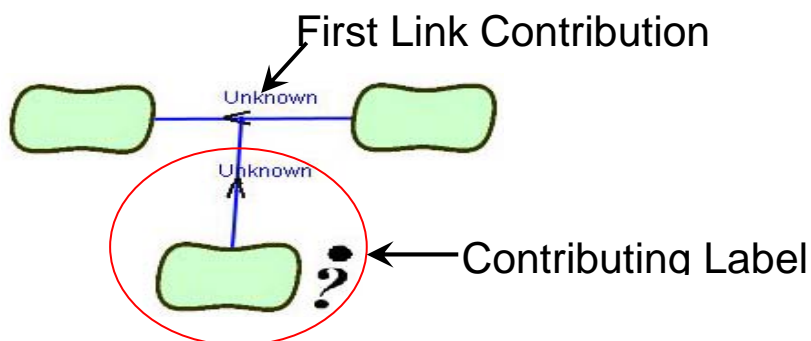


Figure 8.1: Description of Columns in Table 8.4

Regarding links to other links in the quantitative procedure, the combination of quantitative evidence can be acquired by taking the difference between the Sat and Den values, potentially producing negative numbers when the Den value is larger. To calculate the effect of this combined value on the recipient link, the \otimes relation can be used, allowing the contributing link to potentially reduce the strength of the target link, but not increase it, as was done in the qualitative procedure described in this work. The resolution of links to other links in the quantitative adaptation is summarized in Table 8.5.

Table 8.5: Propagation Rules for Links to Links in the GMNS-# and SGM-TD-# Adaptations

Contributing Value	First Link Contributing Value	Actual Contributing Value of First Link
w_1	w_2	$w_1 \otimes w_2$

We avoid the adaptation of the Jarvis Method, described in Chapter 3, to the evaluation of i^* models as this method is identical to the already adapted CNYM method, with the exception of the propagation of label sources. We shall consider the issues involved in adding traceability to the i^* evaluation procedure in Section 8.4.4 of this Chapter.

Chapter 3 contains a description of a method used for the evaluation of KAOS goal models, allowing for partial satisfaction in terms of probabilities (Letier et al., 2004). It may be possible to adapt such a procedure calculating satisfaction probabilities to i^* . This would require defining objective functions and quality variables for i^* elements as well as refinement equations for i^* links. Our exploration of the i^* Framework has indicated that acquiring this level of detailed information in a high-level model is very unlikely. In addition, one would have to consider if these conventions would have to be adjusted for softgoals, not present in KAOS. Taking these points into consideration, as well as the complexity of the KAOS method, we leave its potential adaptation for the i^* Framework to future investigations.

Ideally, the i^* adaptation of goal model evaluation methods would be implemented to allow automatic application. An implementation of the GMNS and SGM-TD qualitative and quantitative methods has been provided by the authors of the method (<http://sesa.dit.unitn.it/goaleditor/>). This implementation has been integrated into the OpenOME implementation in order to allow evaluation of i^* models. However, at this time, only the quantitative versions of the bottom-up and top-down procedures are operational in OpenOME. In addition, this implementation does not yet deal with links to other links, although it does deal with a mixture of links using the And relationship as suggested. Unfortunately, due to time limitations, we have not yet been able to adjust this implementation to process links to other links, or to expand the OpenOME implementation of these algorithms to include the qualitative versions. However, it is our

intention that such modifications and expansions be accomplished at some point in the future.

8.3.2 Comparison of Evaluation Procedures

We shall perform a comparison of the goal model evaluation procedure adapted for use with i^* to the i^* evaluation method described in this work by applying these methods to a large example model. We shall use an example model from the TC domain model with four actors and 52 elements. We avoid the use of a larger model, as some of the evaluation results must be obtained manually. The results for the evaluations on this model are presented in two forms, as a table in Table E.1 of Appendix E, and on the model itself in Figure 8.2. In the Figure, in an attempt to avoid cluttering, if no GMNS or GMNS-# evaluation value for an element is shown, it should be assumed that the element has a value of $\langle \text{Sat} = F, \text{Den} = N \rangle$ or $\langle \text{Sat} = 1.0, \text{Den} = 0.0 \rangle$, respectively. The results from the evaluation procedure described in this work come from the OpenOME implementation, as do the results of the GMNS-# procedure, adjusted to consider the presence of links to other links. In order to apply this procedure on i^* models, the implementers have assumed a simple conversion from i^* links to quantitative propagation values (the w variable), as is required by the GMNS-# procedure. Specifically, all links are treated as having the full 1.0 contribution value, except for Help/Some+ and Hurt/Some- links, which are given values of 0.5 and -0.5, respectively. Application of the OpenOME, qualitative, human judgment procedure required 16.7 seconds to complete, as measured by the application, prompting for human judgment five times.

This measure represents the time for one instance of application by a well-trained user. For users who are not as familiar with i^* , and who are likely to take much longer to make judgments concerning the final values of elements, the procedure may take much longer. However, this measure demonstrates that the computational component of the procedure completes in an amount of time which is feasible for real use. Comparatively, the fully-automated, quantitative evaluation took 0.4 seconds, as measured by the OpenOME application. This measure demonstrates that the time taken by the procedures is not significant, and is also feasible for real use. Therefore both procedures sufficiently

achieve the desired quality for Automation (xii). The GMNS procedure results have been calculated manually.

In order to facilitate a comparison of the evaluation results, we introduce a table of approximate equivalences for the evaluation values from each method in Table 8.6. This table is similar to Table 3.7 in Chapter 3, defining the equivalences between the goal model evaluation procedures. In Table E.1 of Appendix E, when the resulting evaluation values are not equivalent according to this table, the row is highlighted. We can see that out of the 52 elements in the model in Figure 8.2, 15, or 29%, of the element values disagree, according to the equivalences in Table 8.6, between the i^* evaluation procedure described in this work and the results of the GMNS and GMNS-# procedures (the GMNS and GMNS-# values agree with each other, as expected). By examining the structure of the example model, it becomes apparent that the differences between evaluation results are caused by the human judgment promotion for softgoals such as Desirable to Technology Users [Technology], Profit, Desirable [Technology], and Trust [Technology Provider]. This difference is propagated upwards to other elements such as Sell Technology for Profit and Purchase Technology. The differences are more pronounced between the Chapter 4 procedure and the GMNS-# procedure than between this procedure and the GMNS procedure. This difference is especially obvious in the Sell Technology for Profit softgoal, which is satisfied in the Chapter 4 procedure, but has a Sat value of only 0.125 in the GMNS-# procedure. This low value is due to our decision to treat the mixture of links to Sell Technology for Profit as an And relation, using the \otimes operator.

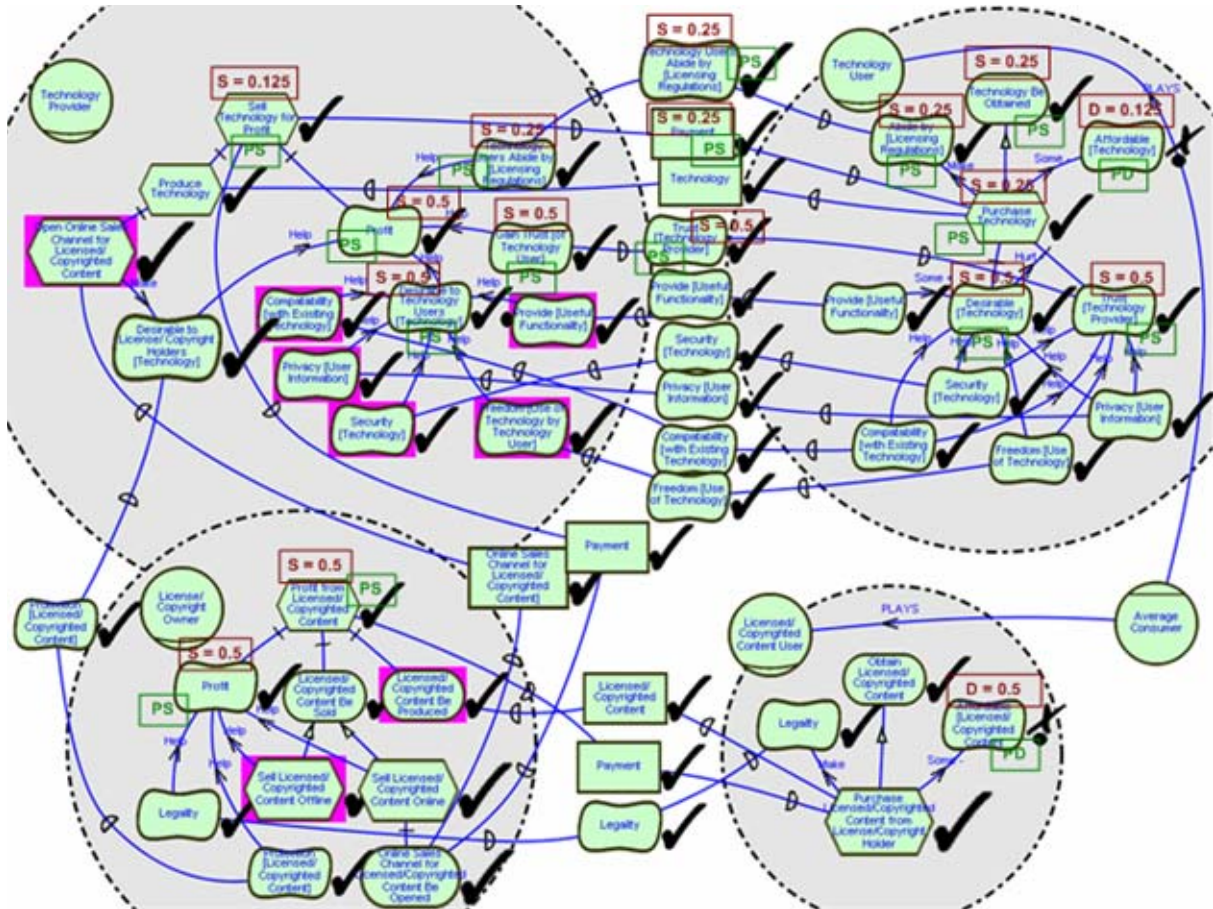


Figure 8.2: TC Example Model Used to Compare Evaluation Procedures

Table 8.6: Approximate Equivalencies of Evaluation Labels across Evaluation Procedures

i*	GMNS Values	GMNS-# Values
✓	<Sat = F, Den = N>	Sat = 1.0 Den = 0.0
✓.	<Sat = F, Den = P> <Sat = P, Den = N>	Sat + Den = (0.0 – 1.0)
✗	<Sat = F, Den = F> <Sat = F, Den = P> <Sat = P, Den = F> <Sat = P, Den = P>	Sat = 1.0 Den = 1.0
?	<Sat = N, Den = N>	Sat = 0.0 Den = 0.0
✗.	<Sat = P, Den = F> <Sat = N, Den = P>	Sat + Den = (0.0 – -1.0)
✗	<Sat = N, Den = F>	Sat = 0.0 Den = 1.0

However, examining the value differences between the procedures is not as significant as examining the differences in analysis, and resulting domain conclusions between the procedures, as a result of these value differences. As the reader may recall, in Chapter 7, Section 7.5, our evaluation of this model led us to the conclusion that the situation between these four actors was generally satisfactory and functional, with the exception of the Content User and Technology User's desire to have Affordable Products and Content. By examining the qualitative GMNS results, one might reach a different conclusion, that the relationships amongst these actors is only partially functional, as the Technology User is Purchasing Technology they do not fully Desire or Trust, and as the Technology Provider and License/Copyright Owner are only partially satisficing their Profit goals. The dissatisfaction with Affordability is also apparent in these results. In the quantitative GMNS-# procedure, the interpretation of the results may differ even more. Not only are the major concerns of the Technology Provider and the Technology User not fully satisfied, their satisfaction level is quite low, indicating that the relationship between these actors contains serious deficiencies.

The discoveries and success of the Trusted Computing Study demonstrate that the conclusions derived by the Chapter 4 procedure are Accurate (iv) for this model. This indicates that the GMNS and GMNS-# procedures are not appropriate for analysis of such high-level models. The inability to promote multiple instances of partial evidence manifests as a deficiency in the Accuracy (iv) of the evaluation results for these models, and therefore reduces the analysis capabilities of these methods for such models. However, it is still possible that a quantitative evaluation procedure which is appropriate for i^* models exists, as we shall discuss in Chapter 9.

In addition to the bottom-up results analyzed above, it would be interesting to examine the results of the SGM-TD-# procedure on this model. Unfortunately, running this procedure on the Figure 8.2 model, with the four top-level goals/tasks of each actor marked as satisfied, produces non-sensible results, indicating errors in the OpenOME integration. Correcting this implementation will be left to future work.

8.4 Conclusion

In this Chapter we have assessed the effectiveness of the evaluation procedure developed in this work in terms of the desired qualities for i^* evaluation defined in Chapter 2. We have adapted goal evaluation procedures for use with i^* and tested these methods on an i^* example, concluding that the propagation rules within these methods are not suitable for the high-level analysis required for the i^* Framework.

Chapter 9: Contributions, Limitations and Future Directions

9.1 Contributions

In this work we have introduced the notion and utility of system modeling as a method of abstraction and modularity, facilitating comprehension and aiding system design. We have pointed out that typical system modeling lacks the ability to provide an intentional and organizational ontology, explaining the motivations and organizations behind system structure and function, as emphasized in (Yu, 1997). The introduction of goal modeling addressed this deficiency by adding the underlying motivations for system constructs to systems modeling. The *i** framework builds upon this work, adding an emphasis on capturing intentions as ascribed to domain actors, stressing the organizational interactions which motivate systems. By explicitly capturing the role of a system in the organizational domain we can better understand how a system may solve real problems, including the consequences of integrating the system into the social environment. The explicit consideration of these elements, including the articulation of assumptions concerning the domain can assist in producing systems which are more likely to be successful.

Capturing the motivations behind the relationship between a system and its environment through modeling in the *i** framework, although useful, does not fully utilize the capabilities of organizational modeling. Once a model is created, iteration and analysis of the model can help to acquire a better understanding of the domain, can bring to light deficiencies in knowledge, and can help to answer strategic domain questions, leading to the choice of an optimal design alternative. However, the ad hoc manual application of such analysis is problematic due to the potential size and complexity of *i** models and the need for a standardized, transferable method of analysis. The definition of an evaluation procedure for *i** models would enable such analysis and allow the potential of *i** models to be more fully utilized. In this work we have defining such an evaluation procedure for *i** and made various contributions towards the research area as a whole.

- **Desired Qualities for i* Evaluation.** By examining the intended use of i* analysis, we have defined desired qualities that such a procedure should exhibit, including (i) Element Evaluation, (i.a) Evaluation Value Expressiveness, (i.b) Overall Evaluation Value, (ii) Clear Procedural Guidelines, (iii) Allowance for Human Intervention, (iv) Accuracy, (v) Usefulness in Multiple Contexts, (vi) Modes of Analysis, (vii) Traceability, (viii) Conflict Detection, (ix) Constraints on Values, (x) Facilitating Cost Analysis, (xi) Simplicity, (xii) Automation, (xiii) Syntax Checking, and (xiv) Semantic Improvement.
- **Assessment of Goal Model Evaluation.** By assessing the existing evaluation procedures for goal models by our desired criteria, we have chosen the qualitative CNYM method described in (Chung et al., 2000) as providing the most appropriate base for an i* evaluation procedure, due to its simplicity, potential to allow for human intervention, and general appropriateness for high-level analysis.
- **Elaboration of the CNYM Method.** In order to adapt the CNYM procedure for use with i*, we elaborated on and proposed solutions for aspects of the procedure that were not well-defined, such as initial goal labels, non-leaf nodes, and multiple judgments per goal.
- **Adaptation and Expansion of CNYM Method to i*.** After carefully considering the specifics of this method, we expanded and modified the procedure to be applicable to models in the i* Framework. This involved defining a means of Element Evaluation (i), defining the Clear Procedural Guidelines (ii) for syntax specific to i*, and thoroughly defining the role of Human Intervention (iii).
- **Exploration of Semantic and Syntactic Benefits and Results of i* Evaluation.** Through the exploration of examples we have described the benefits and results of applying the i* evaluation procedure. Evaluation prompts changes in both model syntax (xiii) and semantics (xiv). Application of evaluation can prompt an expansion of i* syntax, which might aid model comprehension, but also increase model size. Examination of the evaluation results highlights semantic model deficiencies, which can be corrected in successive iterations.
- **Exploration of Analysis Capabilities.** We have demonstrated that the application of the procedure can answer complex questions concerning the effects of

decisions on the goals of domain actors, allowing for an exploration of domain alternatives in order to discover the most favorable solutions to stakeholder problems.

➤ **Implementation.** The qualitative i^* evaluation procedure was implemented in OpenOME, a modeling tool embedded in the Eclipse IDE. The soundness of this implementation, including convergence, termination, and performance, was tested by employing large-scale example models. Methods attempting to reduce the need for human judgment were explored and tested.

➤ **Application to Multiple Case Studies.** The Accuracy (iv) and Usefulness of the i^* evaluation procedure in Multiple Contexts (v) of was confirmed by using the procedure in the context of five case studies, four of which were based on documentation, and one of which was based on continuing interactions with stakeholders in a real-life project.

➤ **Discovering Areas of Future Investigation.** The extensive use of i^* evaluation in the large scale models within these case studies revealed many issues and considerations for the procedure, some of which were used to create and refine the procedure itself, and others which were identified as areas of future investigation.

➤ **Assessment of Evaluation Procedure against Desired Qualities.** Through our achievement of the above contributions, we have shown that the procedure developed in this work sufficiently balances the desired qualities of an evaluation procedure.

Element Evaluation (i). It includes a notion of satisfaction and denial by covering six qualitative ordinal values ranging from full satisfaction to full denial. These values provide sufficient expressiveness for early analysis. Placing such values on the model allows for storage of intermediate evaluation values, during the propagation of multiple paths.

Clear Procedural Guidelines (ii). The meaning of each link and evaluation value combination has been defined through propagation guidelines.

Allowance for Human Intervention (iii). We have incorporated flexibility into the procedure by treating these conventions as guidelines and prompting for human judgment when resolving multiple sources of evidence in certain ambiguous cases.

Simplicity (xi). Our use of a single qualitative value per node and intuitive propagation rules keeps the procedure simple, allowing for a reasonably efficient manual application to small or medium models.

Automation

(xii). Through our implementation, we have automated the parts of the procedure which do not need human guidance, allowing for more efficient application, especially necessary when evaluating large models. **Syntax Checking (xiii) and Semantic Improvement (xiv).** Our examples have shown the ability of evaluation to improve the quality of models by improving their correspondence with real life phenomena, and to expand the syntax of models, potentially reducing ambiguity. **Accuracy (iv) and Usefulness in Multiple Contexts (v).** As shown by our case studies, we believe that the procedure facilitates useful and accurate domain analysis, helping to evaluate system alternatives in multiple contexts, promoting a deeper understanding of the domain.

➤ **Assessment of Evaluation Procedure by Comparison to Other Adaptations Procedures.** Alternative goal model evaluation methods were adapted for use with the i* Framework and the application of these procedures were compared to the procedure developed in this work. The results indicated these procedures confirmed our earlier assertion, that these methods were not highly suitable for the high-level, organizational analysis needed with i* models.

➤ **Outlining of Evaluation Features for Future Addition.** In this Chapter, we make suggestions concerning the adoption of various useful features into the i* evaluation procedure as optional functionality, including beneficial qualities such as top-down analysis (vi), Traceability (vii), Conflict Detection (viii), Constraints on Values (x), and Facilitating Cost Analysis (x).

9.2 *Limitations*

Despite our success in meeting the requirements for an effective i* evaluation procedure, some limitations remain. In spite of the strong organizational emphasis of i* modeling, our procedure does not explicitly take into account actor boundaries, or the position of elements within the intentions of a specific actor. Although these factors should be considered when making human judgment, there may be methods to incorporate this knowledge into the evaluation in a more predominant way, stressing the organizational context. Additionally, the implementation of our algorithm is limited in

that it does not completely minimize the need for human judgment, thus making the partially automated application potentially tedious for large models. This implementation, and its underlying application, should be more thoroughly tested and made more robust for general use.

9.3 Future Work

During the development of the evaluation procedure in this work, many additional areas have been identified as requiring future work.

9.3.1 Implementation Issues

Regarding the implementation of i^* and goal model evaluation, we can outline multiple relevant and beneficial tasks which should be completed and issues which should be explored in the future.

- The correct integration of the goal model evaluation procedures adapted for i^* in OpenOME, including the GMNS, GMNS-#, GMNS-TD and GMNS-#-TD procedures.
- The addition of evaluation value graphics to the Human Intervention pop-up screen in OpenOME.
- The exploration of further ways to reduce the number of times human intervention is required in the i^* evaluation algorithm.

9.3.2 Examination of i^* Syntax

Several of the areas of future work have involved aspects of i^* syntax.

- The instance vs. class nature of i^* actors, and how one can explicitly manipulate the default meaning in order to produce the desired evaluation results, if possible.
- Methods to effectively deal with the iteration of cyclical evaluation values in i^* models, especially when this iteration does not converge.

- i* syntax involving the repetition of softgoals across actors has appeared frequently in this work. A detailed investigation into semantically unambiguous and graphically efficient ways to represent this effect is desirable.

9.3.3 Scalability Concerns

A significant portion of the future work identified has focused on dealing with the evaluation of large i* models.

- We have identified the need to implement a method to link evaluation results between views of one conceptually large i* model, a feature which we anticipate will be added to the OpenOME implementation.
- We have used methods such as layering and slicing to manage evaluation on large models. Although our results have shown that the layering technique seems especially promising, a thorough investigation and implementation of these techniques will likely offer significant benefits to large-scale evaluation.
- It would be informative to determine the effect of link completeness on evaluation results in large models, determining to what degree the addition of more links affects such results, and whether there is an effective point of link “saturation”.

9.3.4 Investigation of Broader Approaches to Evaluation

Beyond the methods for goal model evaluation explored in Chapter 3, there exists broader approaches to graph evaluation and propagation. For example, the work of (Chiang, & Menzies, 2003) has applied a simulation tool for models in the NFR Framework using Monte Carlo simulations to produce “behaviors” which are summarized by a machine learning tool which produces recommendations for the system. Multi-value logic has also been used to deal with graph evaluation (Chechik, Devereux, Easterbrook, & Gurfinkel, 2003).

In the future, we would like to examine these broader approaches carefully to determine if these procedures offer useful features which could be used to modify or

extend the procedure described in this work. Additionally, if one of these procedures provides a balance of desired qualities that is as effective as the balance provided by the procedure in this work, this other procedure could be used as an alternative form of i^* evaluation.

9.3.5 The Need for Further Empirical Studies

In general, the claims within this work could benefit by the design and execution of experiments or detailed case studies to test the accuracy and utility of evaluation results. Such tests may help to further confirm Accuracy (iv) and Usefulness in Multiple Contexts (v). Work such as this could further confirm that the model changes prompted by evaluation are truly beneficial for domain analysis. The case studies included in this work, especially the Kids Help Phone study, are a positive step towards this goal.

9.3.6 Potential Additional Features for the i^* Evaluation Procedure

There are a number of desired qualities for i^* evaluation which we have not been able to include in the procedure. In addition, we can identify features which have not been explicitly included in the desired qualities for i^* evaluation, but which may be useful if i^* is to be used in contexts outside of the high-level early analysis which has been the focus of this work. Namely, quantitative analysis, and the avoidance of partial values could be useful if i^* is used in a lower-level, more design specific context. These features are discussed in this section.

Ideally, the additional evaluation features which are identified as being feasible would be added into the OpenOME implementation of the procedure. The implementation of such features would increase the viability of evaluation for i^* by making the procedure more flexible and catering to a larger group of potential users. However, due to time restrictions, we are unable to implement such features at this time. Nevertheless, using the knowledge gained from extensive application of i^* evaluation, we will provide a description of the behavior that the implementation of such qualities will have to exhibit, in order to be useful in i^* evaluation.

9.3.6.1 Top Down Evaluation (vi)

As we have discussed in Chapter 4, Section 4.1.2, performing a top-down, qualitative evaluation of i^* models using an interactive procedure would be difficult due to the nature of the questions posed to the user. Instead of determining a final value for an element based on contributing values, this would involve determining the set of contributing values that would permit the element in question to take on the value in question. Although the user may be able to come up with one set of possible values, asking the user to determine the set of all possible values which would result in the final label given would be unreasonable. Therefore, top-down i^* evaluation can likely only be performed if the procedure can be fully automated. Due to the arguments given for the need for human judgment, it is unlikely that a qualitative evaluation procedure for i^* can be fully automated.

In this light, it is worth investigating whether our proposal for an i^* quantitative evaluation procedure to appear in Section 1.3.5.6 may be adjusted to work in a top-down manner, as this method embeds the knowledge needed for human judgment into the model. It may be possible that the model and propagation rules can be converted into a boolean expression as input to a SAT solver as is done in (Giorgini, Mylopoulos, & Sebastiani, 2004). This possibility warrants further investigation in future studies.

9.3.6.2 Traceability (vii)

The current implementation of the qualitative i^* evaluation procedure in OpenOME provides a shallow traceability in human judgment decisions by listing the sources for all contributing labels. However, it may be useful to provide more thorough traceability by displaying the original source of such labels, either to aid in making a final label decision, or, at the end of the procedure, to aid in the interpretation of final evaluation results. Unfortunately, the origin of labels in i^* models is not always clear, due to the presence of human judgment as well as the combination of evidence in means-ends, decomposition, and multiple contribution labels. For example, in Figure 8.4, what is the source for the label of the Profit from Licensed/Copyrighted Content task? Certainly

Licensed/Copyrighted Content be Produced is a source, but also either Sell Licensed Copyrighted Content Offline or Sell Licensed Copyrighted Content Online, which is satisfied via Open Online Sales Channel for Licensed/Copyright Content are sources as well. In addition, the sources for the Profit softgoal are also a source for the Profit from Licensed/Copyrighted Content value. In general, due to the tree-like structure of i^* models, an overall evaluation value for an element which is the result of a combination of evidence will have multiple sources. Nevertheless, it may be possible to achieve traceability for an element by displaying a list of all elements with initial values that contribute to the value of this element. However, for values near the top of i^* models, this list may include nearly all of the initial values in the model. For instance, in Figure 8.4, the set of initial labels which contribute to Sell Technology for Profit includes all initial labels except for the two within the License/Copyright Content Holder. In addition, without the contextual knowledge of the graph structure to explain the effects that these initially labeled elements have on the affected element, the meaning of this list may not be clear.

An alternative method to provide traceability would be to use the slicing methodology, as described in Chapter 7, Section 7.6, to highlight the sub-tree of an element. This could be done when human judgment is being performed on an element, or when a user selects an element in a certain way. For example, in order to trace the evidence sources for the Abide by Licensing Regulations softgoal in Figure 8.4, upon selection of this element, the implementation could display something similar to what is shown in Figure 8.5. Such graphical methods could facilitate the tracing of evidence while avoiding the complexity of reporting traceability by trying to represent a portion of the graph structure in textual form.

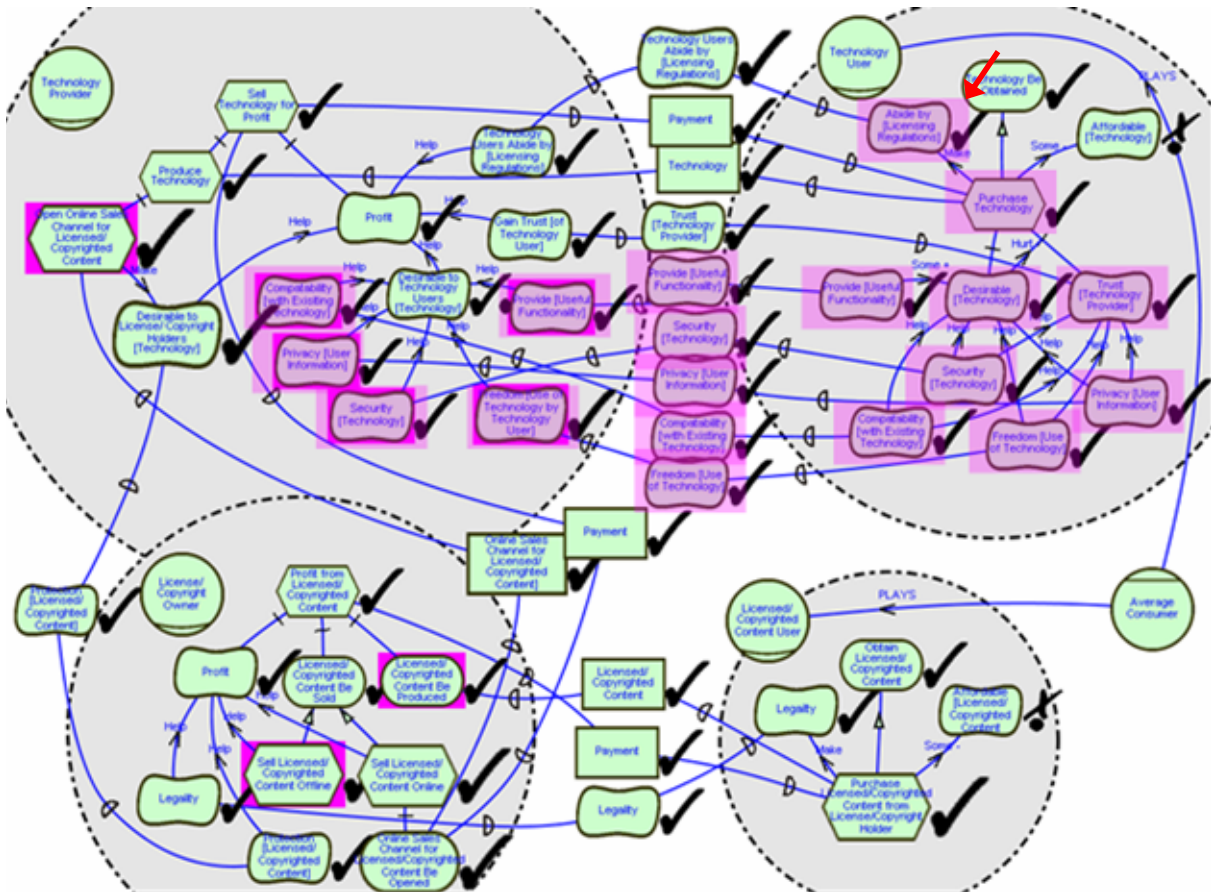


Figure 9.1: Example TC Model Showing a Possible Implementation of Traceability

9.3.6.3 Conflict Detection (viii)

In i^* evaluation, it may be useful to highlight the presence of conflicting evidence, in order to explicitly assess areas of the model which could be of particular interest in analysis. Often, due to the role of human intervention, conflicts in i^* are resolved by making a decision to merge positive and negative evidence into a positive or negative value, especially when the evidence of one polarity greatly outweighs the other. However, in other cases, when the strength of the negative and positive evidence is judged to be relatively equal, a conflict may be maintained, and a conflict label may be assigned to the element.

In terms of conflict detection in the algorithm described in Chapter 4, when adding initial values to the model, before propagation evaluation is performed, an algorithm could graphically highlight areas in the model that it can detect will have both

positive and negative evidence. Elements which receive evidence from these potential conflicts could also be highlighted, as they may also be areas of conflict, if human judgment determines that the contributing conflict is not resolved. Once a judgment has been made, the algorithm could recalculate the elements which will receive conflicting evidence, and adjust the conflict highlighting. By this method, the effects of initial values and human judgment in terms of causing conflicts will be immediately displayed, allowing evaluators to better understand the effects of their decisions.

We shall illustrate this potential method with a simple example. In Figure 9.2, repeated from Chapter 2 and 4, we see two initial evaluation values for Produce PC Products and Allow Peer-to-Peer Technology. Based on these values, an algorithm could propagate the values it is able to without intervention and detect the presence of conflicting evidence for Affordable PC Products, the Abide By Licensing Regulations softgoals, Profit and Sell PC Products for Profit. These conflicts could be pointed out with a red highlight, such as is shown in Figure 9.2. As the evaluation procedure progresses and human intervention is performed, the highlighted areas of potential conflict could be updated dynamically.

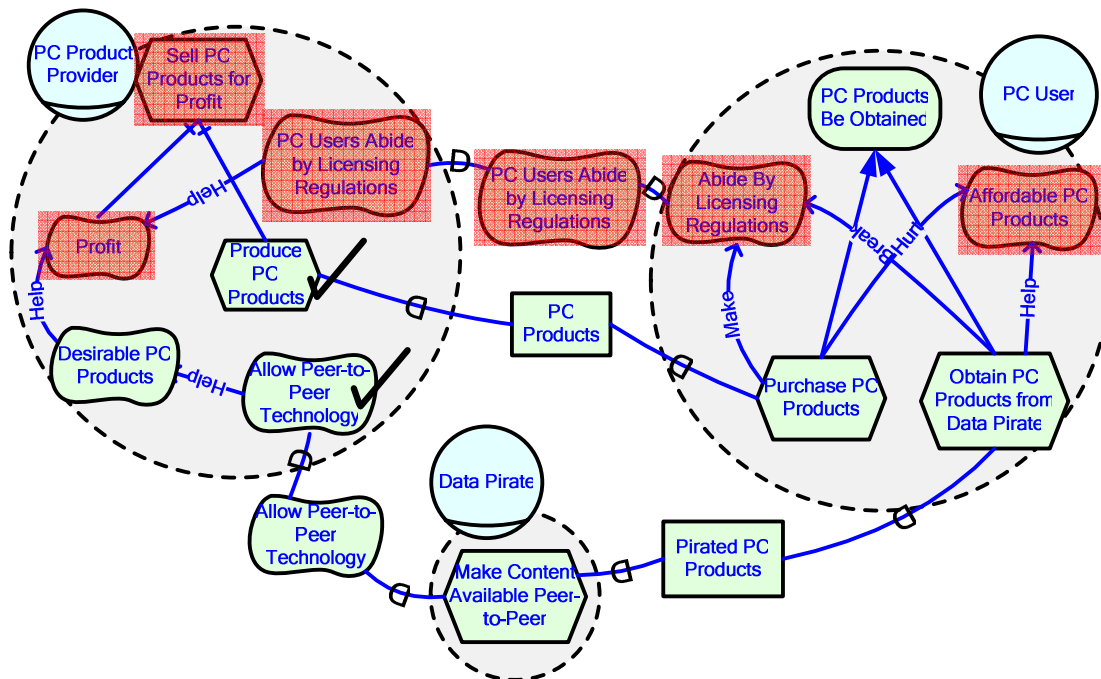


Figure 9.2: SR Model Showing Potential Conflict Detection

9.3.6.4 Constraints on Values(ix)

The SGM-TD procedure introduced us to the possibility of integrating constraints on evaluation values into a goal model evaluation procedure. This feature is particularly useful in a top-down propagation, as there are potentially many values that the lower-level elements can take in order to satisfy the inputs given for top-level elements. However, we can also envision how such constraints may be used in bottom-up propagation. Constraints could be added by specifying specific values or value ranges that a node must or must not have, or by indicating a general avoidance of “strong” or “weak” conflicts, as defined in the SGM-TD procedure. If an initial value or human decision produces a result which is contrary to one of the constraints, a warning could be given to the user, allowing the user to stop or continue propagation. Graphical symbols could be used to indicate the presence of constraints and the violation of constraints if the user chooses to continue propagation. For example, in the screenshot shown in Figure 8.3, the green circle may indicate the presence of a constraint concerning the Partners Follow Conditions softgoal, and the red circle may indicate the presence of a constraint which has been violated for the Put on ‘On’ Festival Events Task. The specifics of the constraint could be available in “tool-tip” form on mouse over of the circle. Further consideration is needed in order to determine how to graphically represent global constraints such as an avoidance of conflicts.

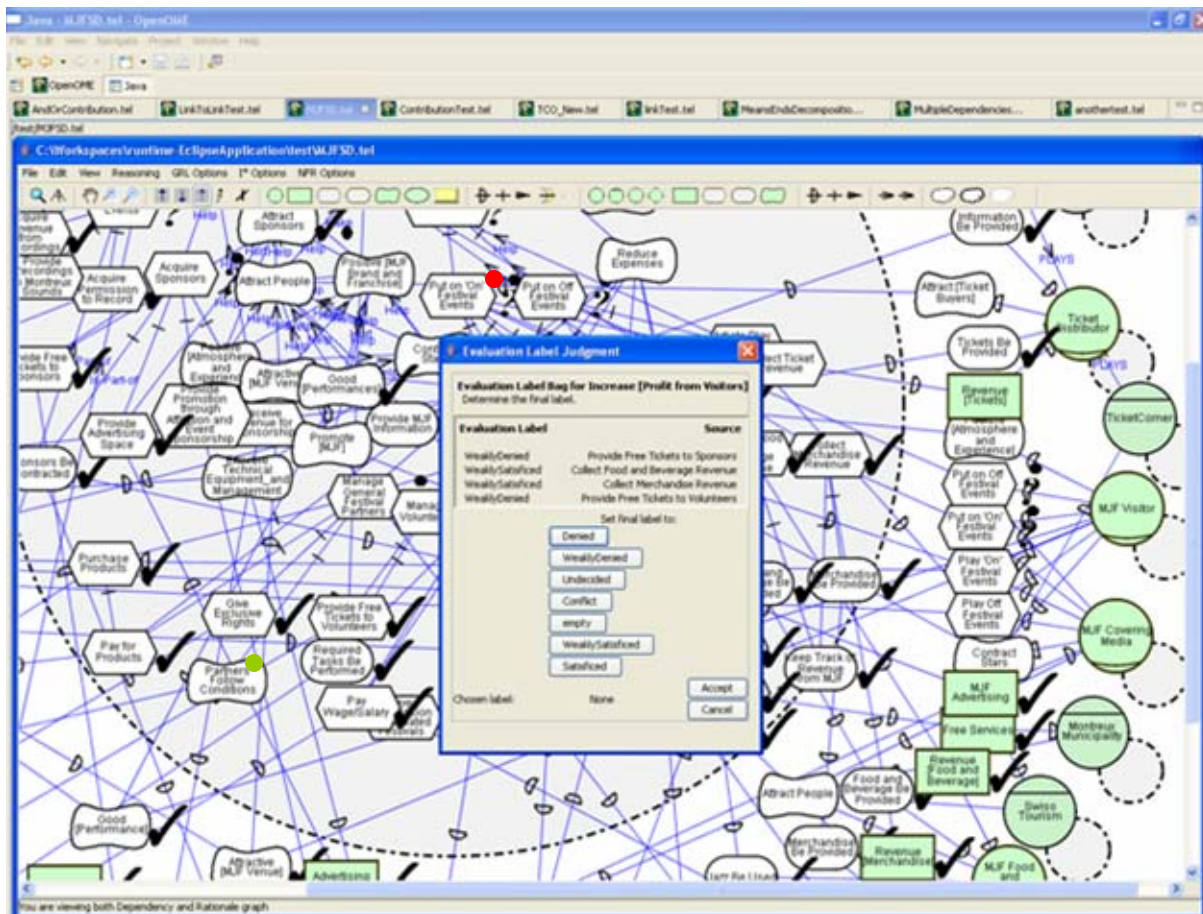


Figure 9.3: An Example Graphical Representation for Constraints

9.3.6.5 Facilitating Cost Analysis (x)

The incorporation of cost information into i^* evaluation requires considerations which are similar to the above section exploring quantitative analysis. Essentially, this type of analysis would use rules similar to what has been defined in this section, but with grounding in real-life financial evidence. It is possible that the procedure may have to use a range of quantitative results which differ from $[1.0 - -1.0]$. For example, the procedure may use an unbounded range of positive or negative monetary values. The quantitative values contained in contribution links would ideally reflect some real-life monetary phenomena, such as inflation or market cycles. Of course “cost” does not

necessarily always refer to financial aspects; scales could be derived to evaluate cost in terms of time, relative difficulty of implementation, or any other useful measures.

Although the use of financial information in i^* analysis is possible, unless all model elements refer to some element of cost implementing an analysis which accurately reflect financial values would be difficult. For example, although the Figure 8.4 model contains financial elements such as Profit, it also contains elements such as Trust [Technology Provider] whose satisfaction or denial cannot be represented as a monetary value. Therefore, it is problematic to try to propagate accurate financial information while using the same system of propagation for all elements in the model. Perhaps the solution is to use a separate system of propagation for financial elements, defining how their satisfaction contributes to non-financial elements, and vice-versa. Alternatively, we could avoid these types of definitions by attempt to implement something similar to the methods in (Giorgini, Mylopoulos, & Sebastiani, 2004), where the minimum cost solution is derived. In this case cost is represented using relative weights. Using a similar method, design alternatives could be ranked using two different comparison dimensions: their qualitative effects on model elements, and their total cost. The alternative which best balances these dimensions may be optimal. We leave the detailed exploration of these possibilities to future investigations.

9.3.6.6 Quantitative Values

Our example in Section 8.3 has revealed that the GMNS-# procedure may not be wholly appropriate for use with high-level i^* models; however, there may still be a place for quantitative evaluation in i^* . Although in the high-level, early analysis for which i^* is intended concrete measurement allowing for a numerical analysis is often not present, it is possible that i^* may be adapted and used in situations where such evidence is available, or where a more fine-grained analysis is needed. If we were to make adjustments to the propagation rules of the GMNS-# procedure to allow the effects of contribution to softgoals to be cumulative, this may produce a procedure which is lenient enough to facilitate analysis for i^* .

A quantitative evaluation procedure for i^* could use either two values, such as in GMNS-# or one numeric value per node, using a range of values where negative values indicate a level of denial, for instance a range of [1.0 - -1.0]. In our previous discussions concerning the appropriate qualities of an i^* evaluation in Chapter 4, Section 4.1.2 we determined that producing a single value per node, combining positive and negative evidence, is more appropriate for analysis in i^* , due to the need for overall decisions concerning the satisfaction or denial of elements. Therefore, we consider the one-value-per-element system as a default convention, keeping in mind the separation of satisfaction and denial values as a secondary option in the implementation.

Combining positive and negative quantitative evidence requires a definition of propagation rules which differ from the GMNS-# rules. These rules employed operators which created a Bayesian network where the values for each goal represented the conditional probability of its satisfaction or denial given the value for a contributing goal. The rules we shall define for i^* are based on the intuitive meanings of the i^* syntax, and as a result, do not retain the properties of a Bayesian network. However, our aim is to facilitate informal analysis given quantitative evidence, as opposed to a formal evaluation of probability. In addition, the quantitative values used for i^* evaluation will likely not always be based on real evidence, instead they will represent a more fine-grained judgment of affect according to the modeler; in essence, they are a means of adding more precise measures of human judgment to the construction of the model. As a result, the sacrifice of the probabilistic qualities in the propagation rules is justified. Because of the human judgment embedded in the choice of quantitative contribution strengths, such a procedure would not have to stop to request a human judgment from the user. The quantitative link values can be used as a general blueprint for human decisions, in the cases where human judgment would typically be required in the qualitative procedure. Given the above considerations, we define the proposed propagation rules for a quantitative i^* evaluation procedure in Table 8.6, with $Val(G_1)$ representing the quantitative value for an element.

Table 9.1: Proposed Propagation Rules for a Quantitative i* Evaluation Procedure

	And/Decomposition $(G_2, G_3) \Rightarrow G_1$	Or/Means-Ends $(G_2, G_3) \Rightarrow G_1$	Decomposition $G_1 \dashv\vdash G_2$	Contributions $G_2 \text{=(w)} > G_1$
Val(G₁)	min {Val(G ₂), Val(G ₃)}	max {Val(G ₂), Val(G ₃)}	Val(G ₂)	Val(G ₂) x w

When multiple contributions are made to the same softgoal, G_1 , or, in other words for each G_2 to G_n there exists a link, $G_i \text{=(w)}_i > G_1$, where $2 \leq i \leq n$, the quantitative result for G_1 can be acquired by:

$$Val(G_1) = \min\left(\sum_{i=2}^n Val(G_i) \cdot w_i, 1.0\right)$$

Given these definitions, we can repeat the quantitative evaluation of Figure 8.2, with results as shown in Figure 8.4. As in the previous example, we assume that all quantitative values not shown are fully satisfied, i.e. $Val(G) = 1.0$. We also retain the simple conversion of i* links to propagating quantitative values of either 1.0 or 0.5.

The results of this evaluation correspond to the interpretation derived from our original qualitative evaluation in the Trusted Computing Case Study. Namely, this model represents a generally non-problematic situation with the exception of Affordability. Although this is a sign that the proposed quantitative procedure is reasonable, this example does not make use of the elements of quantitative evaluation which add value to this form of analysis over qualitative equivalents. Specifically, as all of our partial contribution labels are given a value of 0.5, we are not demonstrating how human judgment can be embedded in the model. For example, in the determination of the Desirable [Technology] softgoal, perhaps the modeler determines that Provide [Useful Functionality] contributes at a level of 0.8, Compatibility [with Existing Technology] contributes 0.4, Security [Technology] contributes 0.6, Freedom [User of Technology] contributes 0.5, and Privacy [User Information] contributes 0.7. As the relative importance of each softgoal to Desirable is embedded within the model, the model becomes more easily transferable to other parties, as it can be evaluated without the need for human intervention based on domain knowledge.

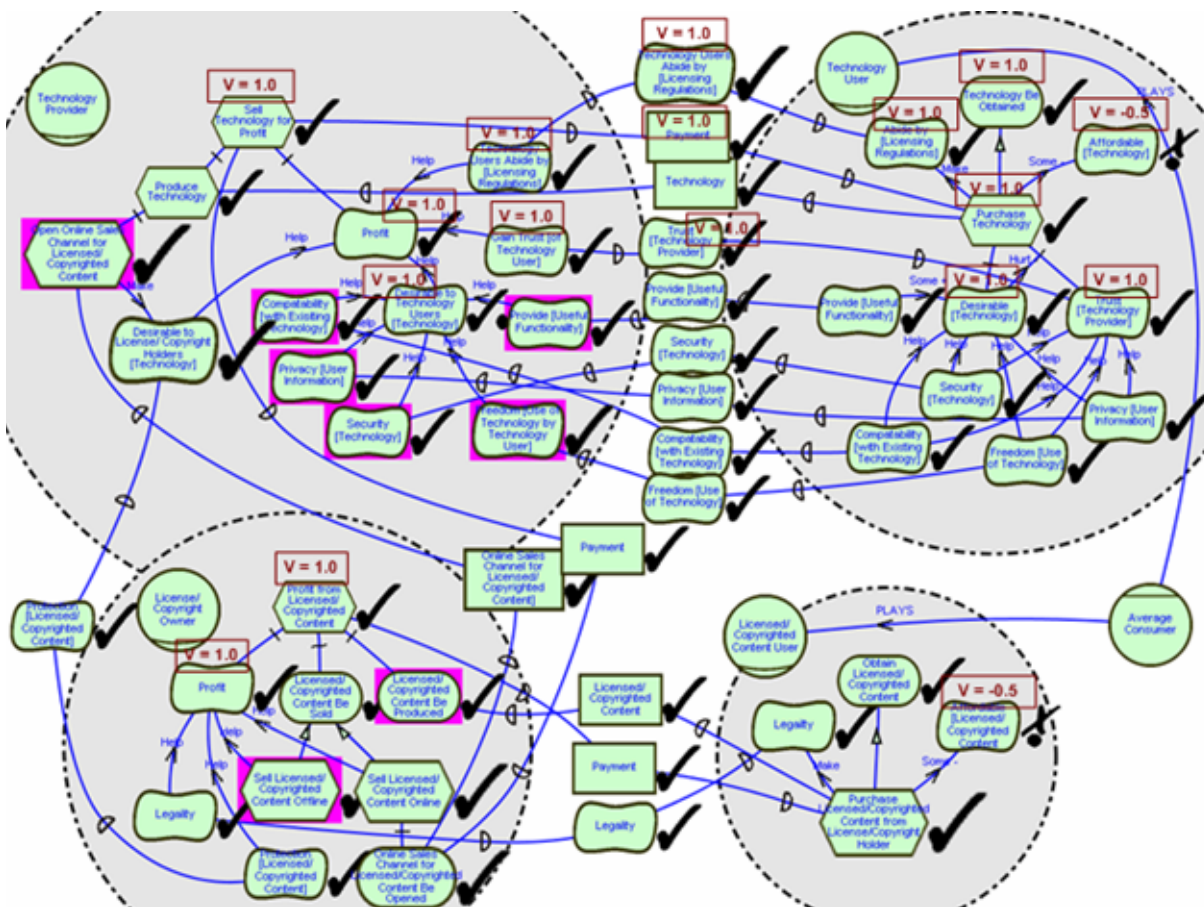


Figure 9.4: TC Example Model Used to Test Qualitative Evaluation for i^*

In summation, a quantitative version of the i^* evaluation procedure, such as described above, seems to show great promise in evaluation and model clarity. As a result, this procedure should be more thoroughly explored in successive investigations.

9.3.6.7 Optional Avoidance of Partial Values

As there may be situations where an evaluator wishes to perform an evaluation with stricter criteria on the satisfaction of elements, it would be sensible to implement an evaluation option where partial values are not allowed as final evaluation values. This corresponds closely to the propagation first described in the CNYM procedure. In this case, stricter forms of propagation rules would be in effect, where the results of human judgment must correspond to a full value (or conflict). In this case, “hard” elements

would not receive partial values, as partial values are not propagated from softgoals. It would be feasible to implement this type of evaluation as an option to the currently implemented version in OpenOME.

9.4 Conclusion

Since the introduction of the i* Framework, various areas of research have adopted and adapted the framework to suit a variety of purposes. It would be interesting to see the i* evaluation procedure introduced in this work adopted and adapted in the same manner. It is our hope that the description and examples in this work will act as a suitable guideline for evaluation, assisting i* users in using the framework to its full analysis potential, and adding incentive for new use of i*. Through the increased usage of this framework, focusing on the satisfaction of stakeholder goals in an organizational context, the success rate of system development could be improved.

References

- Anderson, R. (2001). Why Information Security is Hard-An Economic Perspective. In *Proceedings of the 17th Annual Computer Security Applications Conference, December 10 – 14*, (pp. 358-365). New Orleans: IEEE Computer Society.
- Anderson, R. (2003, August). 'Trusted Computing' Frequently Asked Questions. Retrieved July 2004 from <http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html>
- Antón, A. I., & Potts, C. (1998). The use of goals to surface requirements for evolving systems. In *Proceedings of the 20th international Conference on Software Engineering, April 19 - 25*, (pp. 157). Kyoto: IEEE Computer Society.
- Bell Kids Help Phone. (n.d.) Retrieved January 2006 from <https://www.cgi.cdf.utoronto.ca/~cs494hf/cgi-bin/argon.cgi/helpphone/wiki>
- Bell University Laboratories. (n.d.) Retrieved February 2006 from <http://www.bul.utoronto.ca/labs-sra.html>
- Bertolini, D., Busetta, P., Nori, M., & Perin, A. (2002). Peer-to-Peer and Multi-Agent Systems technologies for Knowledge Management Applications. An Agent-Oriented analysis. In *Workshop Dagli Oggetti agli Agenti: Dall'Informazione alla Conoscenza, November 18 – 19*, (pp 1-6). Milano: WOA 2002.
- Booch, G., Rumbaugh, J., and Jacobson, I. (1999). *The Unified Modeling Language User Guide*. Addison Wesley Longman Publishing Co., Inc.
- Chechik, M., Devereux, B., Easterbrook, S. M., & Gurfinkel, A. (2003, October). Multi-Valued Symbolic Model-Checking. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 12(4), 371-408.
- Chiang, E., Menzies, T. (2003). Simulations for very early lifecycle quality evaluations. *Software Process: Improvement and Practice*, 7(3-4), 141-159.

- Chung, L., Nixon, B.A., Yu, E., & Mylopoulos, J. (2000). Non-Functional Requirements in Software Engineering (Monograph). *Kluwer Academic Publishers*.
- Dardenne, A., Fickas, S., & van Lamsweerde, A. (1991). Goal-Directed Concept Acquisition in Requirements Elicitation. In *Proceedings of IWSSD-6, 6th International Workshop on Software Specification and Design, October*, (pp. 14-21). Como: IEEE Computer Society.
- DeMarco, T. (1978). *Structured Analysis and System Specification*. Englewood Cliff, New Jersey: Yourdon, Inc.
- Easterbrook, S.M., Yu, E., Aranda, J., Fan, Y., Horkoff, J., Leica, M., & Oadir, R.A. (2005). Do Viewpoints lead to Better Conceptual Models?: An Exploratory Case Study. In *Proceedings 13th IEEE International Requirements Engineering Conference (RE'05), August 29-September 2*, (pp. 199-208). Paris: IEEE Computer Society.
- Giorgini, P., Mylopoulos, J., Nicchiarelli, E., & Sebastiani, R. (2002). Reasoning with Goal Models. In *Proceedings of 21st International Conference on Conceptual Modeling (ER 2002), October 7-11*, (pp. 167-181). Tampere: Springer-Verlag.
- Giorgini, P., Mylopoulos, J., Nicchiarelli, E., & Sebastiani, R. (2004) "Formal Reasoning Techniques for Goal Models", *Journal of Data Semantic*. LNCS 2800 - Journal Subline, Springer-Verlag.
- Giorgini, P., Mylopoulos, J., & Sebastiani, R. (2004). Simple and Minimum-Cost Satisfiability for Goal Models. In *Proceedings On Advanced Information Systems Engineering (CAiSE*04), June 7-11*, (pp. 20-35). Riga: Springer-Verlag.

- Giorgini, P., Massacci, F., Mylopoulos, J., & Zannone, N. (2004) Requirements Engineering meets Trust Management: Model, Methodology, and Reasoning. In *Proceedings of the Second International Conference on Trust Management (iTrust), March 29-April 1*, (pp. 20-35). Oxford: Springer-Verlag.
- Giorgini, P., Massacci, F., & Mylopoulos, J. (2003) Requirement Engineering meets Security: A Case Study on Modelling Secure Electronic Transactions by VISA and Mastercard. In *Proceedings of the 22nd International Conference on Conceptual Modeling (ER 2003), October 13-16*, (pp. 263-276). Chicago: LNCS, Springer-Verlag.
- GRL - Goal-oriented Requirement Language. (n.d.) Retrieved February 2006 from <http://www.cs.toronto.edu/km/GRL/>
- GR-Tool. (n.d.) Retrieved March 2006 from <http://sesa.dit.unitn.it/goaleditor/>
- Horkoff, J., Aranda, J., Easterbrook, S., & Yu, E. (2006). *Feature Prioritization and Analysis Using i* Models*. Unpublished manuscript.
- Jackson, M. (1997). The Meaning of Requirements. *Annals of Software Engineering*, 3, 5-21.
- Jarvis, R. (1992). Modeling and Analysis of Strategic Business Objectives. *Master of Science Thesis*, Department of Computer Science, University of Toronto.
- Karlsson, J., & Ryan, K. (1997, September/October). A Cost-Value Approach for Prioritizing Requirements. *IEEE Software*, 14(5), 67-74.
- Katzenstein, G.J. and Lerch, F.J. (2001) Beneath the Surface of Organizational Processes: A Social Context Framework for Business Process Redesign. *ACM Transactions on Information Systems*, 18(4), 383-422.

- Kavakli, E., Loucopoulos, P., & Filippidou, D. (1996) Using Scenarios to Systematically Support Goal-Directed Elaboration for Information System Requirements. In *Proceedings of the IEEE International Symposium and Workshop on Engineering of Computer Based Systems (ECBS'96), March 11-15*, (pp. 308-314). Freiderichshafen: IEEE Computer Society.
- Kitchenham, B., Pickard, L., & Pfleeger, S.L. (1995, July). Case Studies for Method and Tool Evaluation. *IEEE Software*, 12(4), 52-62.
- Leica, M. (2005). *Scalability Concepts for i* Modelling Analysis*. Master of Science Thesis, Department of Computer Science, University of Toronto.
- Letier, E., & van Lamsweerde, A. (2004). Reasoning about Partial Goal Satisfaction for Requirements and Design Engineering. In *Proceedings of FSE'04, 12th ACM International Symposium on the Foundations of Software Engineering, October 31-November 5*, (pp 53-62). Newport Beach: ACM SIGSOFT Engineering Notes.
- Liu, L., & Yu, E. (2001). From Requirements to Architectural Design-Using Goals and Scenarios. In *Proceedings of ICSE-2001 Workshop: From Software Requirements to Architectures (STRAW 2001), May 14*, (pp. 22-30). Toronto, Canada.
- Liu, L., Yu, E., & Mylopoulos, J.(2003). Security and Privacy Requirements Analysis within a Social Setting. In *Proceedings for International Conference on Requirements Engineering (RE'03), September 8-12*, (pp151-161). Monterey: IEEE Computer Society.
- Liu, L., & Yu, E. (2004). Designing Information Systems in Social Context: A Goal and Scenario Modelling Approach Information Systems. 29(2), 187-203.

- Liu, L., & Yu, E. (2004). Intentional Modeling to Support Identity Management. In *Proceedings of 23rd International Conference on Conceptual Modeling (ER 2004), November 8-12*, (pp. 555-556). Shanghai: LNCS 3288, Springer-Verlag.
- Maiden, N.A.M, Jones, S.V., Manning, S., Greenwood, J., & Renou., L. (2004). Model-Driven Requirements Engineering: Synchronising Models in an Air Traffic Management Case Study. In *Proceedings of 16th International Conference (CAiSE 2004), June 7-11*, (pp. 368-383). Riga: LNCS, Springer-Verlag.
- Martínez, A., Pastor, O., & Estrada, H. (2004, January). Isolating and Specifying the Relevant Information of an Organizational Model: A Process Oriented Towards Information System Generation. *Lecture Notes in Computer Science, 3046*, 783-790.
- Mayer, N., Rifaut, A., & Dubois, E. (2005). Towards a Risk-Based Security Requirements Engineering Framework. In *Proceedings 11th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'05), in conjunction with CAiSE'05, June 13—14*,. Porto, Portugal
- Microsoft Next-Generation Secure Computing Base - Technical FAQ. (n.d.) Retrieved July 2004 from <http://www.microsoft.com/technet/Security/news/ngscb.msp>
- Molani, A., Perini, A., Yu, E., & Bresciani, P. (2003). Analysing the Requirements for Knowledge Management using Intentional Analysis. In *Proceedings of AAAI Spring Symposium on Agent-Mediated Knowledge Management (AMKM-03), March 24-26*. Stanford University
- Montreux Jazz Festival. (n.d.) Retrieved December 2005 from http://www.montreuxjazz.com/index_en.aspx

- Mouratidis, H., Giorgini, P., & Manson, G. A. (2003). Integrating Security and Systems Engineering : Towards the Modelling of Secure Information Systems. In *Proceedings of 15th International Conference CAiSE, June 16-18*, (pp. 63-78). Klagenfurt: LNCS 2681(Springer).
- Mylopoulos, J. (1998, May). Information Modeling in the Time of the Revolution. *Information Systems* 23(3-4), 127-156.
- Mylopoulos, J., Chung, L., & Nixon, B. (1992, June). Representing and Using Nonfunctional Requirements: A Process-Oriented Approach. *IEEE Transaction on Software Engineering*, 18(6), 483-497.
- Mylopoulos, J., Chung, L., & Yu, E. (1999, January). From Object-Oriented to Goal-Oriented Requirements Analysis. *Communications of the ACM*, 42(1), 31-37.
- Mylopoulos, J., & Castro, J. (2000). Tropos: A Framework for Requirements-Driven Software Development. In J. Brinkkemper & A. Solvberg (Eds), *Information Systems Engineering: State of the Art and Research Themes, Lecture Notes in Computer Science* (pp 261-273). Springer-Verlag.
- OpenOME, an open-source requirements engineering tool. (n.d.) Retrieved February 2006 from <http://www.cs.toronto.edu/km/ome/index.html>.
- Organization Modelling Environment. (n.d.) Retrieved February 2006 from <http://www.cs.toronto.edu/km/ome/index.html>.
- Osterwalder, A. (2004). The Business Model Ontology-a proposition in a design science approach (PhD Dissertation, Ecole des Hautes Etudes Commerciales (HEC), University of Lausanne, Switzerland).

- Park, J., Port, D., & Boehm, H. (1999). Supporting Distributed Collaborative Prioritization for WinWin Requirements Capture and Negotiations. In *Proceedings of the International 3rd World Multiconference on Systemics, Cybernetics and Informatics (SCI'99), July 31-August 4, Vol 2.* (pp 578-584). Orlando: IEEE Computer Society.
- Rolland, C., Souveyet, C., Ben Achour, C. (1998, December) Guiding Goal Modeling Using Scenarios. *IEEE Transactions on Software Engineering*, 24(12), 1055-1071.
- Ross, D. & Schoman, K. (1977, January). Structured Analysis for Requirements Definition. *IEEE Transactions on Software Engineering*, 3(1).
- Santander, V. F., & Castro, J. (2002). Deriving Use Cases from Organizational Modeling. In *Proceedings of the 10th Anniversary IEEE Joint International Conference on Requirements Engineering, September 9 -13*, (pp. 32-42). Washington: IEEE Computer Society.
- Software Chaos, The Standish Group.(n.d.) Retrieved February 2006 from <http://www.standishgroup.com/chaos.html>.
- Spiekermann, S., Dickinson, I., Gunther, O., & Reynolds, D. (2003). User Agents in E-commerce Environments: Industry vs. Consumer Perspectives on Data Exchange. In *Proceedings Advanced Information Systems Engineering, 15th International Conference, CAiSE 2003, June 16-18*, (pp. 696-710). Berlin: LNCS, Springer-Verlag.
- Sutcliffe, G. A., & Minocha, S. (1999). Linking Business Modelling to Socio-technical System Design. In *Proceedings CAiSE'99, June 14 – 18*, (pp. 73-87). Heidelberg: Springer-Verlag.

Trusted Computing Group. (n.d.) Retrieved July 2004 from

<https://www.trustedcomputinggroup.org/>

van Lamsweerde, A. (2000), Requirements engineering in the year 00: a research perspective. In *Proceedings of 22nd International Conference on Software Engineering (ICSE 00), June 5-9*, (pp. 5-19). Limerick: IEEE Computer Society Press.

White, S. (2004). Introduction to BPMN. IBM. Retrieved April 2006 from

<http://www.bpmn.org/Documents/Introduction%20to%20BPMN.pdf>

Yu, E. S.K. (1993). Modelling Organizations for Information Systems Requirements Engineering. In *Proceedings 1st IEEE International Symposium on Requirements Engineering, January 4-6*, (pp, 34-41). San Diego: IEEE Computer Society Press.

Yu, E. S. K., & Mylopoulos, J. (1994). Understanding ‘Why’ in Software Process Modelling, Analysis, and Design. In *Proceedings of 16th International Conference on Software Engineering, May 16-21*, (pp. 159-168). Sorrento: IEEE Computer Society Press.

Yu, E. (1995) *Modelling Strategic Relationships for Process Reengineering*. Phd Thesis, Department of Computer Science, University of Toronto. Retrieved March 2006 from <http://ww.cs.toronto.edu/pub/eric/DKBS-TR-94-6.pdf>.

Yu, E. S.K. (1997), Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering. In *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering, January 6-8*, (pp. 226-235). Washington: IEEE Computer Society Press.

- Yu, E., Liu, L. (2000). Modelling Trust in the i* Strategic Actors Framework. In *Proceedings of the 3rd Workshop in Deception, Fraud and Trust in Agent Societies (Agents 2000), June 3-4*, (pp. 587-607). Barcelona: Elsevier Science Ltd.
- Yu, E. (2001, April). Agent Orientation as a Modelling Paradigm. *Wirtschaftsinformatik*, 43(2), 123-132.
- Yu, E., & Liu, L.(2001). Modelling Trust for System Design Using the i* Strategic Actors Framework. In R. Falcone, M. Singh & Y.H. Tan (Eds.), *Trust in Cyber-Societies – Integrating the Human and Artificial Perspectives* (pp.175-194). LNAI-2246: Springer-Verlag.
- Yu, E., Liu, L., & Li, Y. (2001). Modelling Strategic Actor Relationships to Support Intellectual Property Management. In *Proceedings of 20th International Conference on Conceptual Modeling (ER 2001), November 27-30*, (pp. 164-178). Yokohama: LNCS 2224, Spring-Verlag.
- Yu, E., & Cysneiros, L. M. (2002). Designing for Privacy in the Presence of Other Requirements. In *Proceedings of 4th Workshop on Deception, Fraud and Trust in Agent Societies, July*. Bologna, Italy.
- Yu, E., & Cysneiros, L. (2002). Designing for Privacy and Other Competing Requirements. In *Proceedings 2nd Symposium on Requirements Engineering for Information Security (SREIS'02), November 15-16*. Raleigh, North Carolina.
- Yu, E., & Cysneiros, L. M. (2003). Designing for Privacy in a Multi-Agent World. In R. Falcone, S. Barber, & L. Korba (Eds.), *Trust, Reputation and Security: Theories and Practice*. LNAI 2631, Springer-Verlag.

Yu, E. S. K., & Mylopoulos, J. (2003, September). Using Goals, Rules and Methods to Support Reasoning in Business Process Reengineering. *International Journal of Intelligent Systems in Accounting, Finance, and Management*, 5(1), 1-13.

Appendix A Evaluation Algorithm Pseudocode

Table A.1: CNYM Algorithm without Partial Final Values

```

1.  //*****
2.  //Defining a goal “object”
3.  //*****
4.
5.  //The current evaluation label for the goal, can be satisfied, denied, fully
6.  //satisfied, fully denied, conflict, or unknown.
7.  label = unknown
8.
9.  //Store the initial label of the element separately
10. initialLabel = empty
11.
12. //We need to know which goal the label came from, so we store them as tuples
13. //(label, sourceGoal)
14. labelBag = empty
15.
16. //The name of the goal
17. Name
18.
19. //A list of links that start from this goal
20. linksFrom = empty
21.
22. //Boolean to determine if goal is a leaf goal
23. isLeaf = true
24.
25. //*****
26. //Defining a link “object”. For And and Or links, all links are treated as a single
27. //link
28. //*****
29.
30. //The goal the links points to
31. ToGoal
32. //The goal(s) the link is from
33. FromGoals
34. //The type of link: Make, Break, Some+, Some-, Hurt, Help, Unknown, And, or
    Or
35. Type
36.
37. //*****
38. //The evaluation procedure algorithm
39. //*****
40.
41. //A hash of all goals, initialized in some undefined way, keys are goal names

```



```

42. GoalList.initializeGoals(some parameters)
43.
44. //The list of all links, initialized in some undefined way, the correct references
45. //between links and goals are created
46. LinkList.initializeGoals(goalList, some parameters)
47.
48. //Get initial labels via human input, stored in a list of (goal, label name) tuples
49. //This list of labels is used to hold the labels to be propagated
50. labels = PromptUserforInitialLabels()
51.
52. //Puts the correct labels into the goal objects
53. setInitialLabels(labels)
54.
55.
56. //While there are still labels in the labels list left to be propagated
57. While labels.size > 0
58. {
59. //Propagate all labels in step 1
60. for each (goal, label) in labels {
61.     //Assign the correct label to the goal
62.     for each link in goal.linksFrom {
63.         //If the link is an And or Or link process it differently
64.         if link is And or Or {
65.             resultLabel = processAndOr(label, link)
66.             //Remove previous labels from the same link in the
67.             //label bag
68.             removeLinkLabels(Link)
69.
70.             link.toGoal.labelBag += (resultLabel, goal)
71.         } else {
72.             //get the resulting labels for contribution links
73.             resultLabel = getResultingLabel(label, link.type)
74.             removeLinkLabels(Link)
75.             link.toGoal.labelBag += (resultLabel, goal)
76.         }
77.     }
78. //Remove the goal, label tuple from the labels list, because it's
79. // already been propagated
80. labels.remove(goal, label)
81. }
82.
83. //Process step 2 by resolving label bag either automatically or through
84. //human input
85. for each goal in goalList {
86.     if goal.labelBag is not empty {
87.         result = empty

```

```

88.         result = testforAutomaticCases(goal.labelBag)
89.         if result is not empty {
90.             goal.label = result
91.
92.             //Add this label to the list of labels to be propagated
93.             labels += (goal, label)
94.         }
95.         //Automatic cases did not apply, human input is needed
96.         else {
97.             display: "Input needed to determine the label for"
98.                 goal.name
99.             for each (label, origGoal) in goal.labelBag {
100.                 if label is partiallySatisficed or
101.                    partiallyDenied {
102.                     if origGoal is goal
103.                         display "Initial label of " label
104.                            " has been Placed"
105.                     else
106.                         display "Label " label "from
107.                            goal" origGoal "has been
108.                            propagated."
109.                     display: Promote or demote this label
110. to: "
111.                 if label is partiallySatisficed
112.                     display "Satisficed,
113.                            Conflict or Unknown"
114.                 if label is partiallyDenied
115.                     display "Denied, Conflict,
116.                            or Unknown"
117. //We assume the user inputs one of the
118. //choices, replace bag label with input
119. goal.labelBag.remove(label, origGoal)
120. goal.labelBag += (getUserInput(),
121. origGoal)
122.                 }
123.             }
124. //At this point no partial values are left so one of the
125. //automatic cases must apply
126. goal.label = testforAutomaticCases(goal.labelBag)
127.
128. //Add this label to the list of labels to be propagated
129. labels += (goal, goal.label)
130.         }
131.     }
132. }
133. }

```

```

134.
135. //*****
136. // Helper Methods
137. //*****
138.
139. /***
140. // Method that sets initial labels
141. /***
142. setInitialLabels(initialLabels) {
143.
144.   for each (goal, label) in initialLabels {
145.     goal.setLabel(label)
146.     goal.setInitialLabel(label)
147.
148.     //Put the label that was given to the goal as an initial value
149.     //in the label bag. If the origin of the label is itself
150.     //we know it is an initial input
151.     if link.toGoal.initialLabel is not empty
152.       Goal.labelBag += (goal.initialLabel, goal)
153.   }
154. }
155.
156. /***
157. // Method to process And and Or links, And returns the "min" Or returns the
158. // "max"
159. //**
159. ProcessAndOr(label, link) {
160.   //Store min and max label
161.   maxLabel = unknown //a minimum label
162.   minLabel = satisfied //a maximum label
163.   //Look at the labels for each of the goals that the link is from
164.   for each goal in link.fromGoals {
165.     if goal.label is empty then goal.label = unknown
166.     if goal.label is greater than maxLabel
167.       maxLabel = goal.label
168.     if goal.label is less than minLabel
169.       minLabel = goal.label
170.   }
171.   If label.type is And return minLabel
172.   else return maxLabel
173. }
174.
175. /***
176. // Method to remove old labels from a link in the label bag for the parent link
177. /***
178. removeLinkLabels(link) {

```

```

179.         for each goal in link.fromGoals
180.             if (*, goal) is in link.toGoal.labelBag
181.                 link.toGoal.labelBag.remove(*, goal)
182.     }
183.
184.     /***
185.     // Method to get the correct label according to the rules in Table 4.1. Takes in
186.     // originating label and the contribution link type
187.     /***
188.     getResultingLabel(origLabel, linkType) {
189.         if linkType is unknown
190.             return unknown
191.         Case origLabel {
192.             is satisfied:
193.                 Case linkType:
194.                     is make:
195.                         return satisfied
196.                     is help or some+:
197.                         return partiallySatisfied
198.                     is break:
199.                         return denied
200.                     is hurt or some-:
201.                         return partiallyDenied
202.             is partiallySatisfied:
203.                 Case linkType:
204.                     is make or help or some+:
205.                         return partiallySatisfied
206.                     is break or hurt or some-:
207.                         return partiallyDenied
208.             is conflict:
209.                 return conflict
210.             is unknown:
211.                 return unknown
212.             is partiallyDenied:
213.                 Case linkType:
214.                     is make or help or some+:
215.                         return partiallyDenied
216.                     is break or hurt or some-:
217.                         return partiallySatisfied
218.             is denied:
219.                 Case linkType:
220.                     is make:
221.                         return denied
222.                     is break hurt or some-:
223.                         return partiallySatisfied

```

```

224.             is help or some+:
225.                 return partiallyDenied
226.         }
227.     }
228.
229.     /**
230.     // Method to test for cases for automatic label resolution in step 2
231.     /**
232.     TestforAutomaticCases(labelBag) {
233.         //Case 1
234.         if labelBag contains (unknown, *)
235.             return unknown
236.         //Case 2
237.         if labelBag contains (conflict, *)
238.             return conflict
239.
240.         //Case 3
241.         if labelBag contains (satisficed,*) and (denied, *)
242.             return conflict
243.         //Case 4
244.         if labelBag.size is 1 and
245.             labelBag does not contain (partiallySatisficed, *) or (partiallyDenied, *)
246.             return = goal.labelBag.getFirstLabel()
247.
248.         //Case 5
249.         allFullySatisficed = true
250.         AllFullyDenied = true
251.         for each (label, goal) in labelBag {
252.             if label.type is not satisficed
253.                 allFullySatisficed = false
254.             if label.type is not denied
255.                 allFullyDenied = false
256.         }
257.         if allFullySatisficed
258.             return = satisficed
259.         if allFullyDenied
260.             return = denied
261.
262.         //Case 6
263.         allPositive = true
264.         allNegative = true
265.         HasFullySatisficed = false
266.         hasFullyDenied = false
267.         for each (label, goal) in labelBag {
268.             Case label:
269.                 is satisficed:

```

```

270.             hasFullySatisfied = true
271.             allNegative = false
272.             is partiallySatisfied:
273.                 allNegative = false
274.             is denied:
275.                 hasFullyDenied = true
276.                 allPositive = false
277.             is partiallyDenied:
278.                 allPositive = false
279.             if allNegative and allPositive = false
280.                 Break
281.         }
282.     if allNegative and hasFullyDenied
283.         return denied
284.     if allPositive and hasFullySatisfied
285.         return satisfied
286.
287.     //none of the cases apply
288.     return empty
289. }

```

Table A.2: Changes to CNYM Algorithm in Figure 1 to Allow Partial Final Values

Lines 99-126 to be replaced with:

```

99   for each (label, element) in element.labelBag {
100       //Display all of the labels in the bag, and their
101       //sources
102       if origGoal is goal
103           display "Initial label of " label
104           " has been Placed"
105       Else
106           display "Label " label "from element"
107   element "has been propagated."
108       }
109       display "What is the final label for " element.name "?"
110       element.label = getUserInput()
111       ⋮
112       ⋮
126

```

Lines 243-246 to be replaced with:

```

243     //Case 4
244     if labelBag.size is 1
245
246         return = goal.labelBag.getFirstLabel()

```

Table A.3: The i* Evaluation Algorithm Pseudo code

```

1  //*****
2  //Defining an element "object"
3  //*****
4
5  //The current evaluation label for the element, can be satisfied, denied, fully
6  //satisfied, fully denied, conflict, or unknown.
7  Label = unknown
8
9  //Store the initial label of the element separately
10 initialLabel = empty
11
12 //The previous label propagated by this element, stored for termination purposes
13 previousLabel = empty
14
15 //The bag of intermediate labels collected after step 1 of the procedure
16 //We need to know which element the label came from, so we store them as tuples
17 //(label, sourceElement)
18 labelBag = empty
19
20 //Keep track of whether or not a label bag for an element has been resolved
21 //This value is reset when the contents of a bag change
22 decided = false
23
24 //The type of element, can be goal, softgoal, task, resource, or belief
25 Type
26
27 //The name of the Element
28 Name
29
30 //A list of links that start from this element
31 linksFrom = empty
32
33 //Boolean to determine if element is a leaf element
34 isLeaf = true
35
36 //*****
37 //Defining a link "object". For And, Or, Decomposition, and Means-ends links,
38 // all links are treated as a single link with multiple "from" elements
39 //*****
40
41 //The element the links points to
42 ToElement
43 //The element(s) the link is from
44 FromElement
45 //The type of link: Make, Break, Some+, Some-, Hurt, Help, Unknown, And, Or,

```

```

46 //Means-Ends, Decomposition, Dependency
47 Type
48 //The Evaluation label assigned to this link when there is a link to a link
49 EvalLabel
50 //The type of link which produced the eval result
51 EvalResultLinkType
52 //*****
53 //The evaluation procedure algorithm
54 //*****
55
56 //A hash of all elements, initialized in some undefined way, keys are element
names
57 elementList.initializeElements(some parameters)
58
59 //The list of all links, initialized in some undefined way, the correct references
60 //between links and elements are created
61 linkList.initializeElements(elementList, some parameters)
62
63 //Get initial labels via human input, stored in a list of (element, label name) tuples
64 //This list of labels is used to hold the labels to be propagated
65 labels = PromptUserforInitialLabels()
66
67 //Puts the correct labels into the element objects
68 setInitialLabels(labels)
69
70
71 //While there are still labels in the labels list left to be propagated
72 While labels.size > 0
73 {
74 //Propagate all labels in step 1
75 for each (element, label) in labels {
76 //Store this label as the previous label propagated by this element
77 element.previousLabel = label
78 //Assign the correct label to the element
79 for each link in element.linksFrom {
80 //Deal with links to other links, assign recipient links eval
81 //labels, ignore non-contribution links to other links
82 If link.toElement is a link {
83 if link is a contribution link {
84 link.toElement.setEvalLabel(
85 getResultingLabel(label, link.type))
86 }
87 continue
88 }
89 //if the link has been assigned an evaluation label from a
link

```



```

90          //to it determine what the new strength of the links should
          be
91          String linkType = null
92          If link.getEvalLabel() is not null {
93              linkType = getResultingLink(link)
94              if linkType is "none"
95                  continue
96          }
97
98
99          if link is a contribution link {
100             //If the link is an And or Or process it differently
101             If link is And or Or {
102                 resultLabel = processAndOr(label, link)
103
104                 removeLinkLabels(Link)
105                 link.toElement.labelBag += (resultLabel,
106                 element)
107                 //Mark the bag as undecided, as the contents
108                 //have
109                 //changed
110                 link.toElement.decided = false;
111             //Link is a contribution link that is not And or Or, add
112             // to the bag.
113             else {
114                 //get the resulting labels for contribution links
115                 //check for a link adjusted from a link to it
116                 if (linkType is null)
117                     resultLabel = getResultingLabel(label,
118                     link.type)
119                 else
120                     resultLabel = getResultingLabel(label,
121                     linkType)
122                 removeLinkLabels(Link)
123                 link.toElement.labelBag +=(resultLabel,
124                 element)
125                 //Mark the bag as undecided, as the contents
126                 // have changed
127                 link.toElement.decided = false;
128             }
129         }
130     }
131     else {
132         //If the link is a dependency, look for mix of links
133         If link is Dependency {
134             if link.toElement is softgoal {
135                 removeLinkLabels(Link)

```

```

133         link.toElement.labelBag +=(label,
134         element)
135         //Mark the bag as undecided, as the
136         //contents have changed
137         link.toElement.labelBag.decided = false;
138     }
139     else
140         resultLabel = label
141     }
142     If link is Means-ends or Decomposition {
143         resultLabel = processAndOrLinks(label, link)
144     }
145     //To deal with a mixture of links, we must check to see
146     //If there is a previous value from a different type of
147     //node, and then take the min of the new and old
148     //values
149     If(link.toElement.getEvalResultLinkType() ==
150     link.type
151     or link.toElement has no evalLabel
152     or link.toElement.getEvalLabel() > resultLabel )
153     //Add this label to the list of labels to be
154     //propagated
155     labels += (link.toElement, link.toElement.label)
156     link.toElement.label = resultLabel
157     link.toElement.previousLabel = resultLabel
158 link.toElement.evalResultLinkType = link.type
159 }
160 }
161 //Remove the element, label tuple from the labels list, because it's
162 // already been propagated
163 labels.remove(element, label)
164 }
165 }
166 //Process step 2 by resolving label bag either automatically or through
167 // human input
168 for each element in elementList {
169     if element.labelBag is not empty and
170     element.labelBag.decided is false{
171         result = empty
172         result = testforAutomaticCases(element)
173     if result is not empty {
174         element.label = result
175         //Mark the bag as decided, as the bag has been
176         resolved

```

```

175             link.toElement.decided = true;
176     }
177     //Automatic cases did not apply, human input is needed
178     Else {
179     display: "Input is needed to determine the label for"
180             element.name
181     for each (label, element) in element.labelBag {
182             //Display all of the labels in the bag, and their
183             //sources
184             If origGoal is goal
185                 display "Initial label of " label
186                 " has been Placed"
187             else
188                 display "Label " label "from element"
189     element "has been propagated."
190     }
191     display "What is the final label for " element.name "?"
192     element.label = getUserInput()
193     //Mark the bag as decided, as the bag has been
194     resolved
195     link.toElement.decided = true;
196     }
197     //Check that the previous label propagated is not the
198     // same as the new label (for termination)
199     if element.previousLabel not = link.toElement.label
200     //Add this label to the list of labels to be propagated
201     labels += (element, element.label)
202     }
203     }
204
205     //*****
206     // Helper Methods
207     //*****
208
209     //***
210     // Method that sets initial labels
211     //***
212     SetInitialLabels(initialLabels) {
213
214         For each (element, label) in initialLabels {
215             element.setInitialLabel(label)
216
217             element.setLabel(label)
218
219             //Put the label that was given to the element as an initial value

```

```

220         //in the label bag. If the origin of the label is itself
221         //we know it is an initial input
222         element.labelBag += (element.initialLabel, element)
223     }
224 }
225
226 /**
227 // Method to process And and Or links, And returns the "min" Or returns the
    "max"
228 /**
229 processAndOr(label, link) {
230     //Store min and max label
231     minLabel = satisfied //a minimum label
232     maxLabel = denied    //a maximum label
233     //Look at the labels for each of the elements that the link is from
234     For each element in link.fromElements {
235         if element.label is empty then element.label = unknown
236         if element.label is greater than maxLabel
237             MaxLabel = element.label
238         if element.label is less than minLabel
239             minLabel = element.label
240         //Remove a value from this link in the elements bag, if it exists
241         Link.toElement.removeFromBag(element)
242     }
243     if label.type is And return minLabel
244     else return maxLabel
245 }
246
247 /**
248 // Method to remove old labels from a link in the label bag for the parent link
249 /**
250 removeLinkLabels(link) {
251     For each element in link.fromElement
252         if (*, element) is in link.toElement.l.labelBag
253             link.toElement.l.labelBag.remove(*, element)
254 }
255
256
257 /**
258 // Method to get the correct label according to the rules in Table 4.1. Takes in
    the
259 // originating label and the contribution link type
260 /**
261 getResultingLabel(origLabel, linkType) {
262     If linkType is unknown
263         return unknown

```

```

264     Case origLabel {
265         is satisfied:
266             Case linkType:
267                 Is make:
268                     return satisfied
269                 Is help or some+:
270                     return partiallySatisfied
271                 Is break:
272                     return denied
273                 Is hurt or some-:
274                     return partiallyDenied
275         is partiallySatisfied:
276             Case linkType:
277                 Is make or help or some+:
278                     return partiallySatisfied
279                 Is break or hurt or some-:
280                     return partiallyDenied
281         is conflict:
282             return conflict
283         is unknown:
284             return unknown
285         is partiallyDenied:
286             Case linkType:
287                 Is make or help or some+:
288                     return partiallyDenied
289                 Is break or hurt or some-:
290                     return partiallySatisfied
291         is denied:
292             Case linkType:
293                 Is make:
294                     return denied
295                 Is break hurt or some-:
296                     return partiallySatisfied
297                 Is help or some+:
298                     return partiallyDenied
299     }
300 }
301
302 ****
303 // Method to test for cases for automatic label resolution in step 2
304 ****
305 testforAutomaticCases(element) {
306     labelBag = element.labelBag
307
308     //Case 1
309     if labelBag.size is 1

```

```

310         return = element.labelBag.getFirstLabel()
311
312     //Case 2
313     allFullySatisficed = true
314     allFullyDenied = true
315     for each (label, element) in labelBag {
316         if label.type is not satisficed
317             allFullySatisficed = false
318         if label.type is not denied
319             allFullyDenied = false
320     }
321     if allFullySatisficed
322         return = satisficed
323     if allFullyDenied
324         return = denied
325
326     //Case 3
327     allPositive = true
328     allNegative = true
329     hasFullySatisficed = false
330     hasFullyDenied = false
331     for each (label, element) in labelBag {
332         Case label:
333             is satisficed:
334                 hasFullySatisficed = true
335                 allNegative = false
336             is partiallySatisficed:
337                 allNegative = false
338             is denied:
339                 hasFullyDenied = true
340                 allPositive = false
341             is partiallyDenied:
342                 allPositive = false
343         if allNegative and allPositive = false
344             Break
345     }
346     if allNegative and hasFullyDenied
347         return denied
348     if allPositive and hasFullySatisficed
349         return satisficed
350
351     //Case 4
352     if element.label is satisficed and allPositive
353         return satisficed
354     if element.label is denied and allNegative
355         return denied

```

```
356
357     //none of the cases apply
358     return empty
359 }
360
361 /**
362 // Method to determine the new link type given a link to a link
363 /**
364 String getResultingLink(link) {
365     label = link.getEvalLabel()
366
367     //links can't be promoted
368     if label is fullysatisficed or paritallysatisficed or conflict
369     return "NoChange"
370     //effect of the link is removed completely
371     if label is fully Denied
372     return "none"
373     //the link is unknown now
374     if label is unknown
375     return "unknown"
376     //At this point label must be partially satisficed, remove these types of links
377     if link is not a contribution or link is an And or Or contribution
378     return "none"
379     else { //link is a regular contribution link, value is partially denied
380     if link is help or hurt
381     return "none"
382     if link is Some-
383     return "hurt"
384     if link is Some+
385     return "help"
386     if link is Make
387     return "Some+"
388     if link is Break
389     return "Some-"
390 }
```

Appendix B Privacy in E-Commerce Study Additional Models

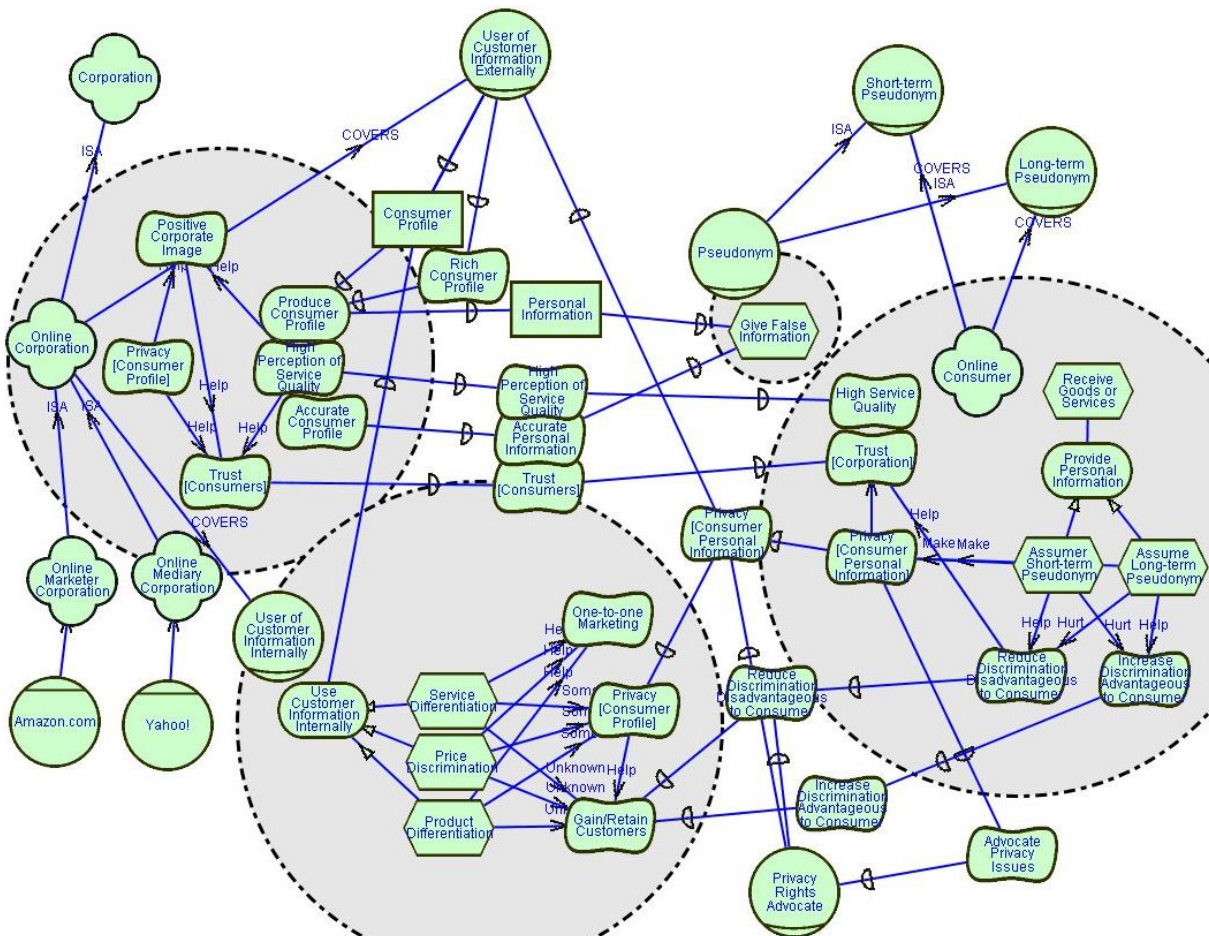


Figure B.1: The Potential Solution of Using Pseudonyms to Protect Personal Privacy

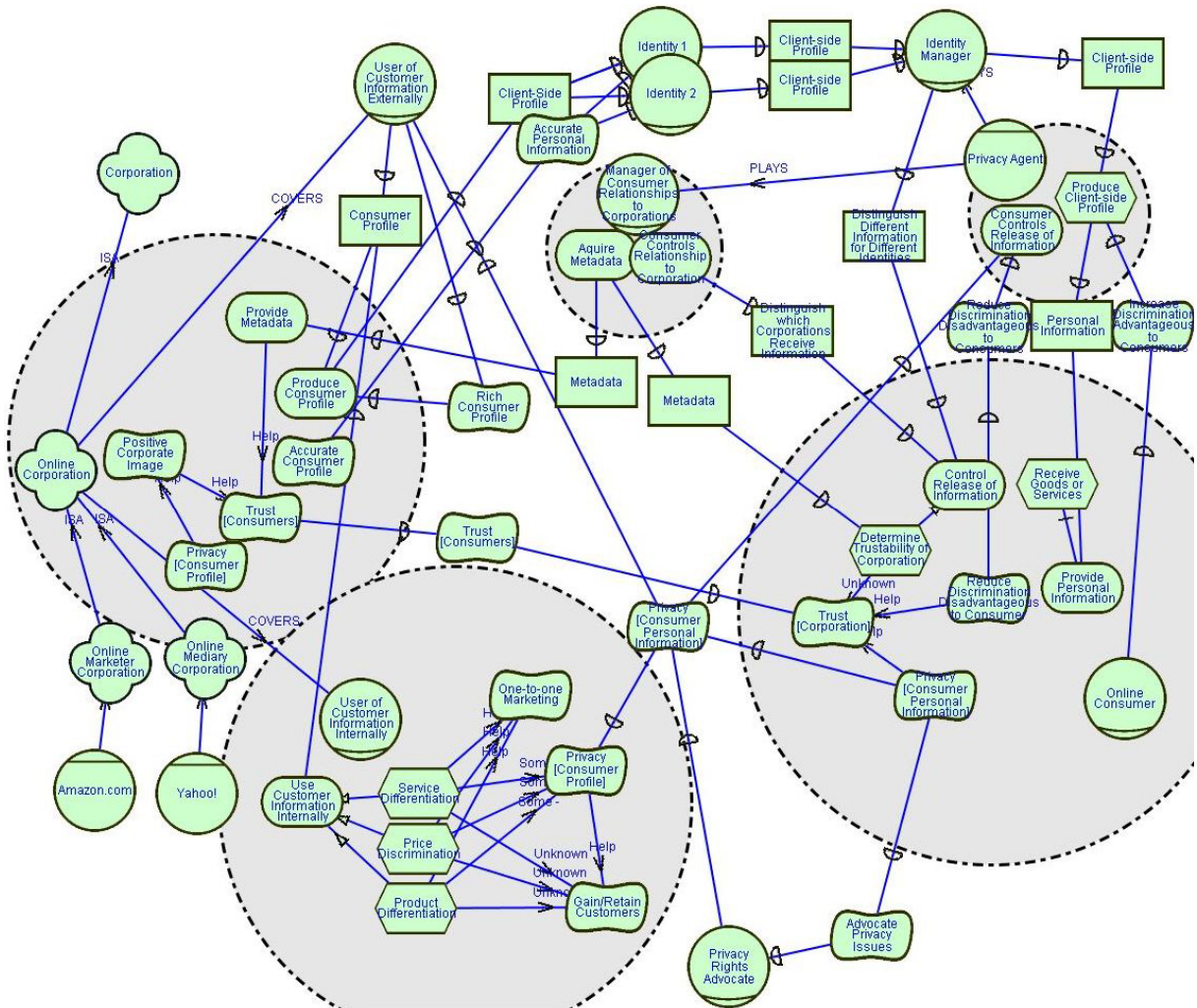


Figure B.2: The Potential Solution of Using Agents to Protect Personal Privacy

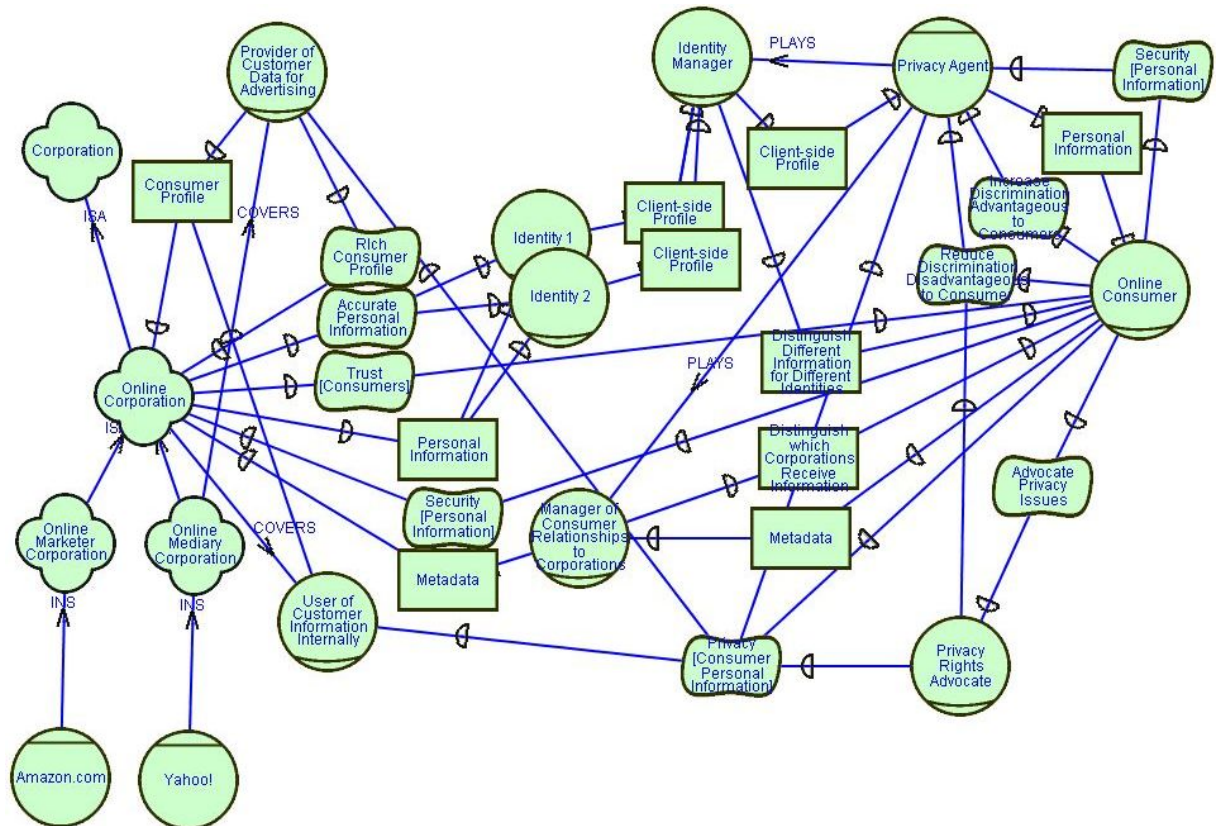


Figure B.3: An SD Model Representing Security Concerns in the Potential Solution of using Agents to Protect Personal Privacy

Appendix C Economic Information Security Study Additional Models

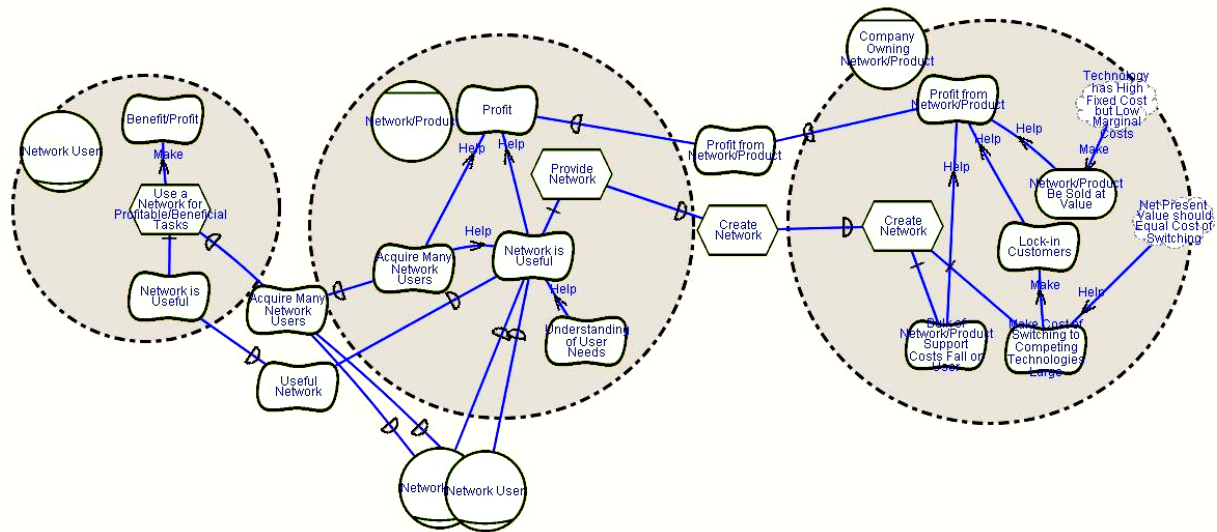


Figure C.1: A Specific Example of Network Externality Involving the Network User, Network/Product and Company Owning Network/Product

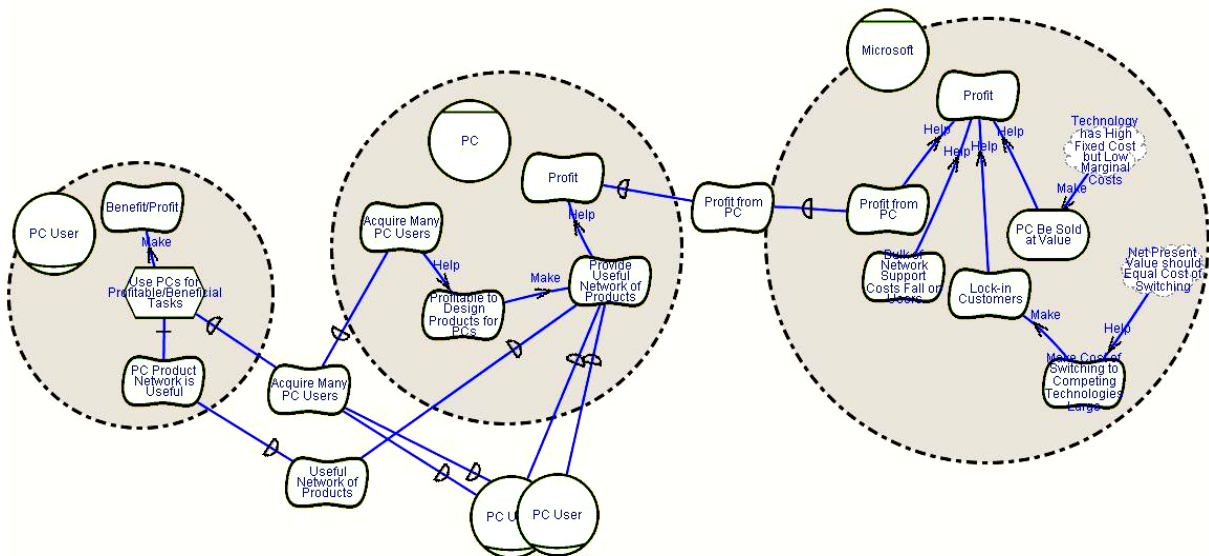


Figure C.2: A Specific Example of Network Externality Involving Personal Computers

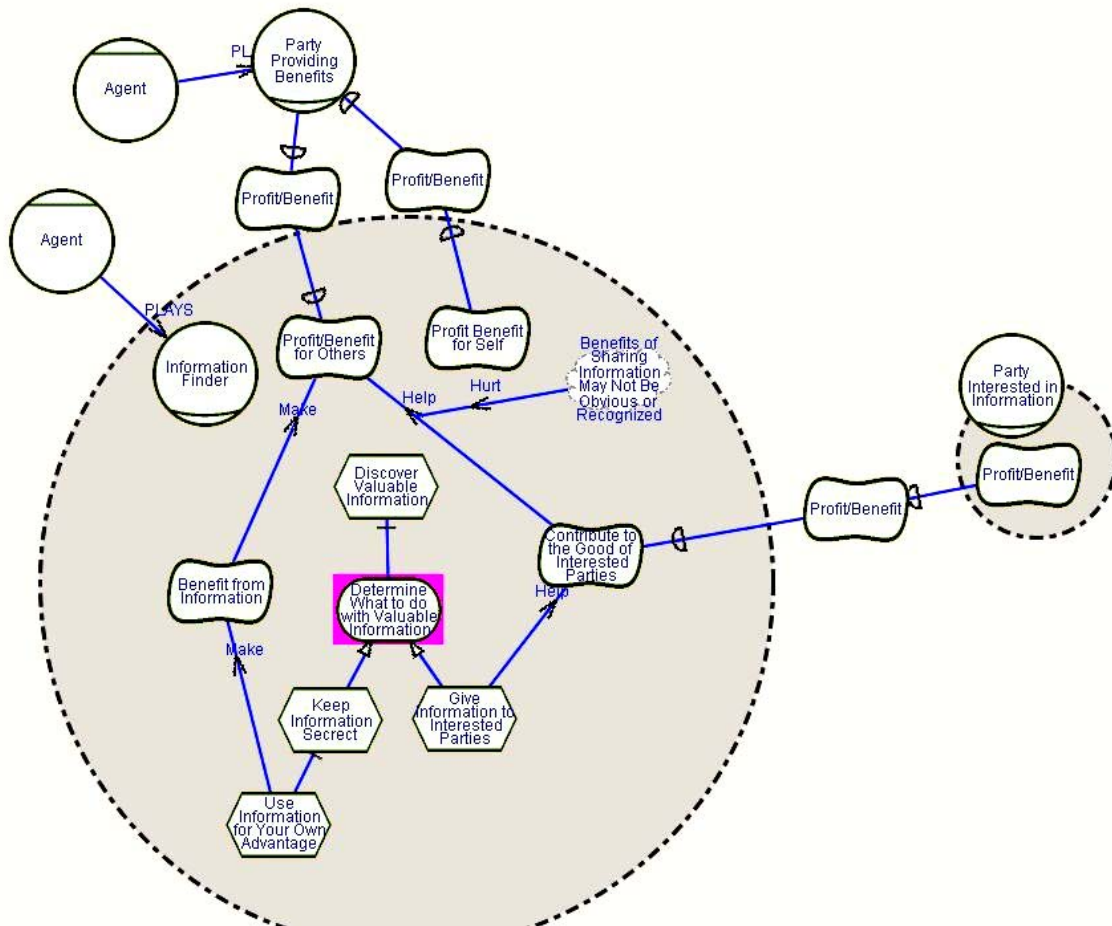


Figure C.3: The General Situation for Asymmetric Information

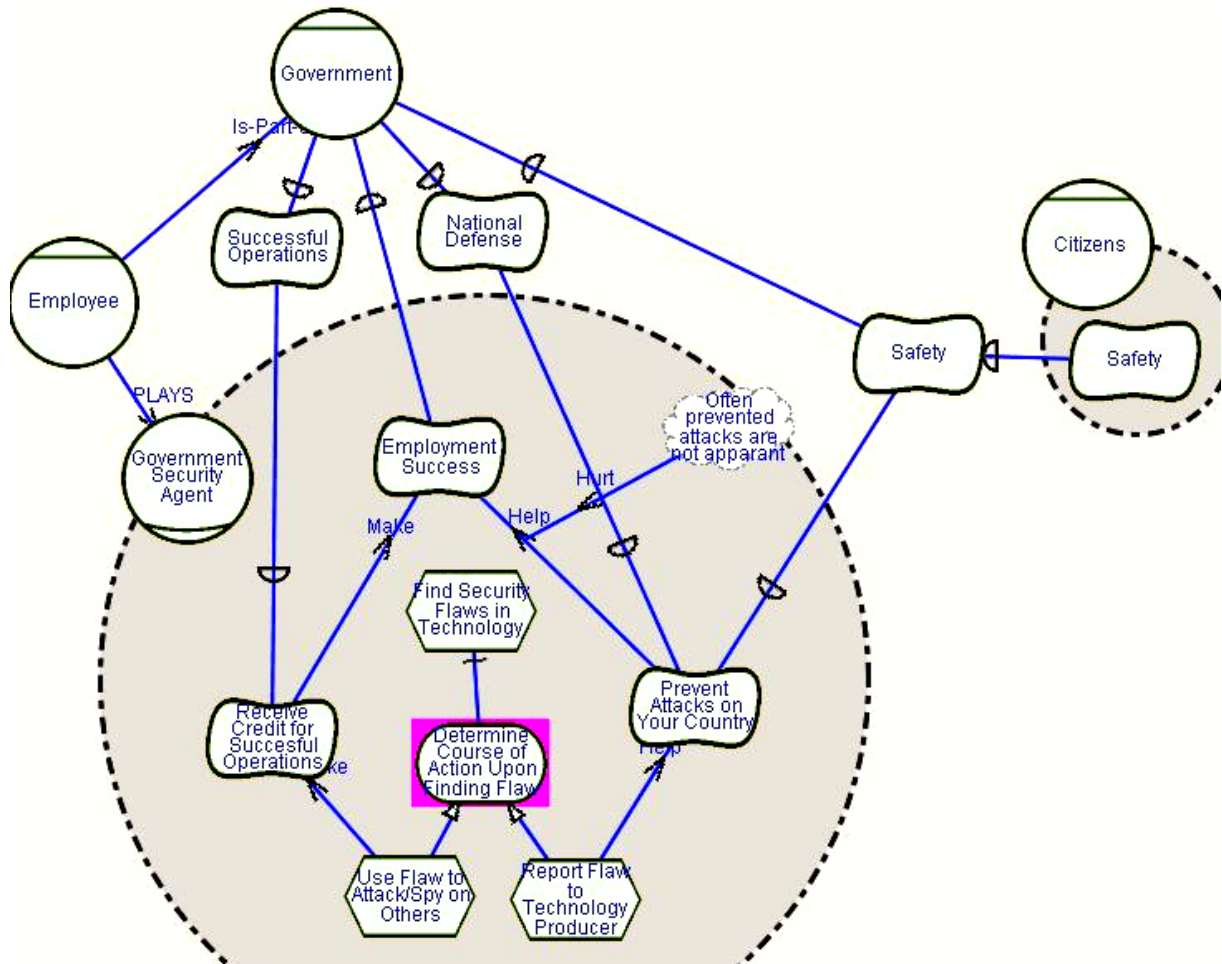


Figure C.4: The Government Asymmetric Information

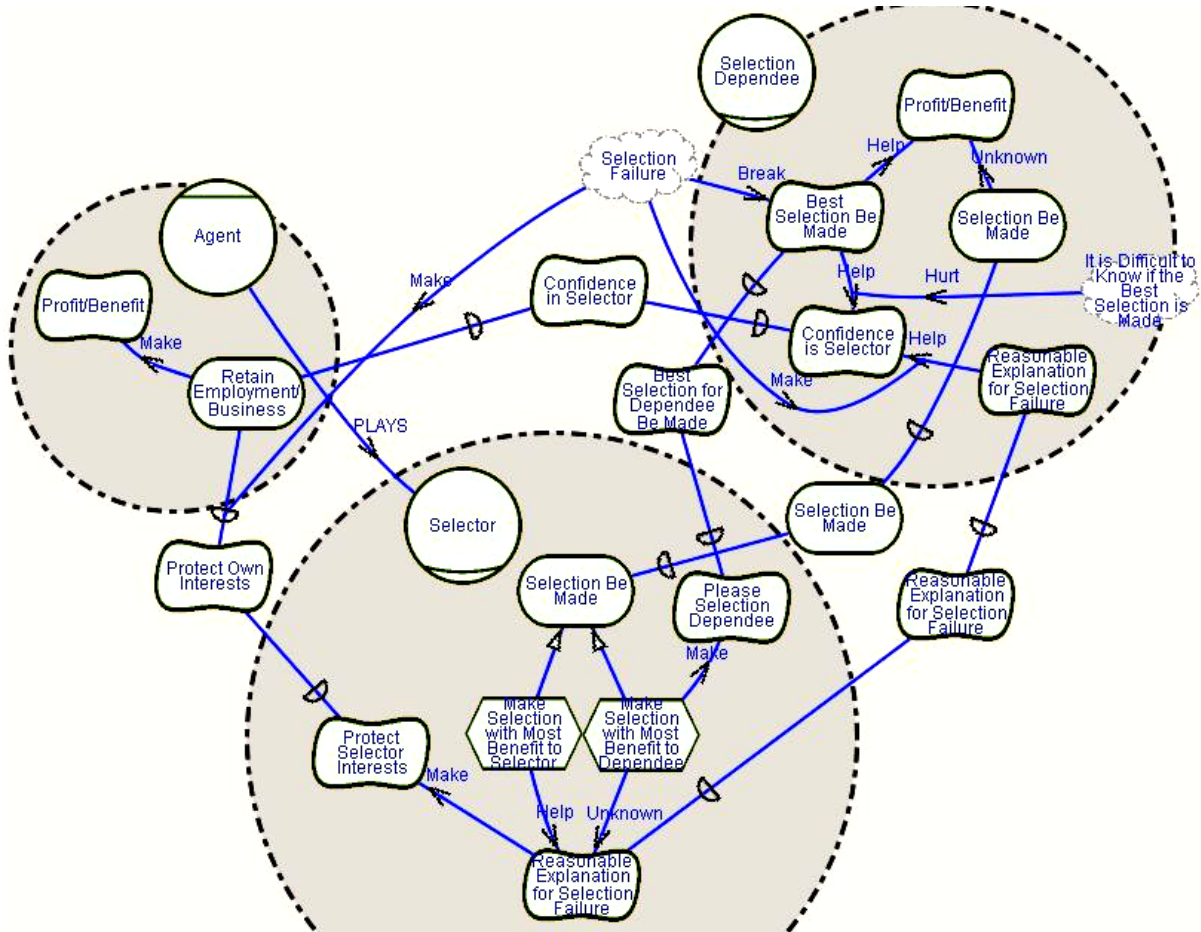


Figure C.6: General Adverse Selection

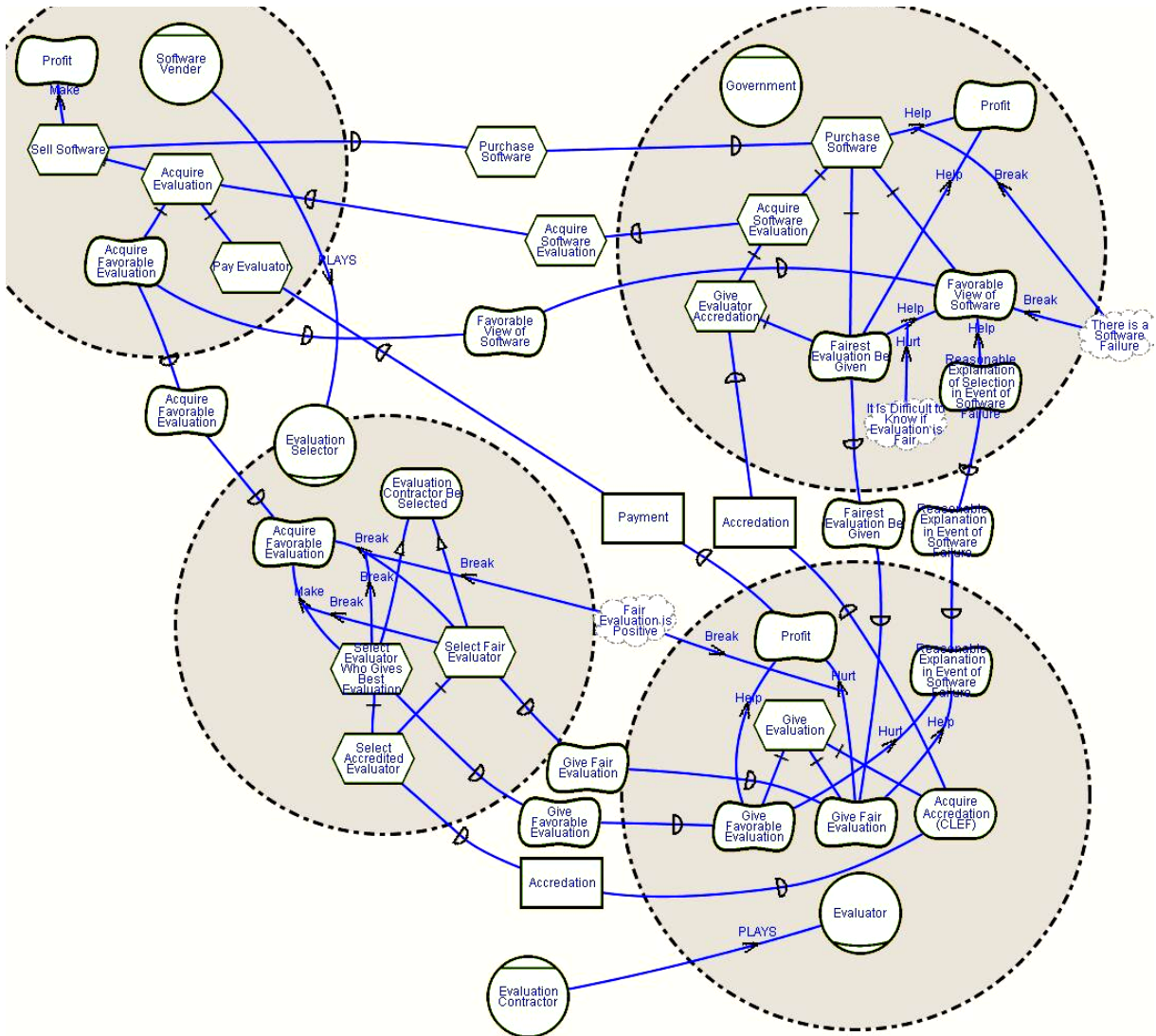


Figure C.7: Software Adverse Selection

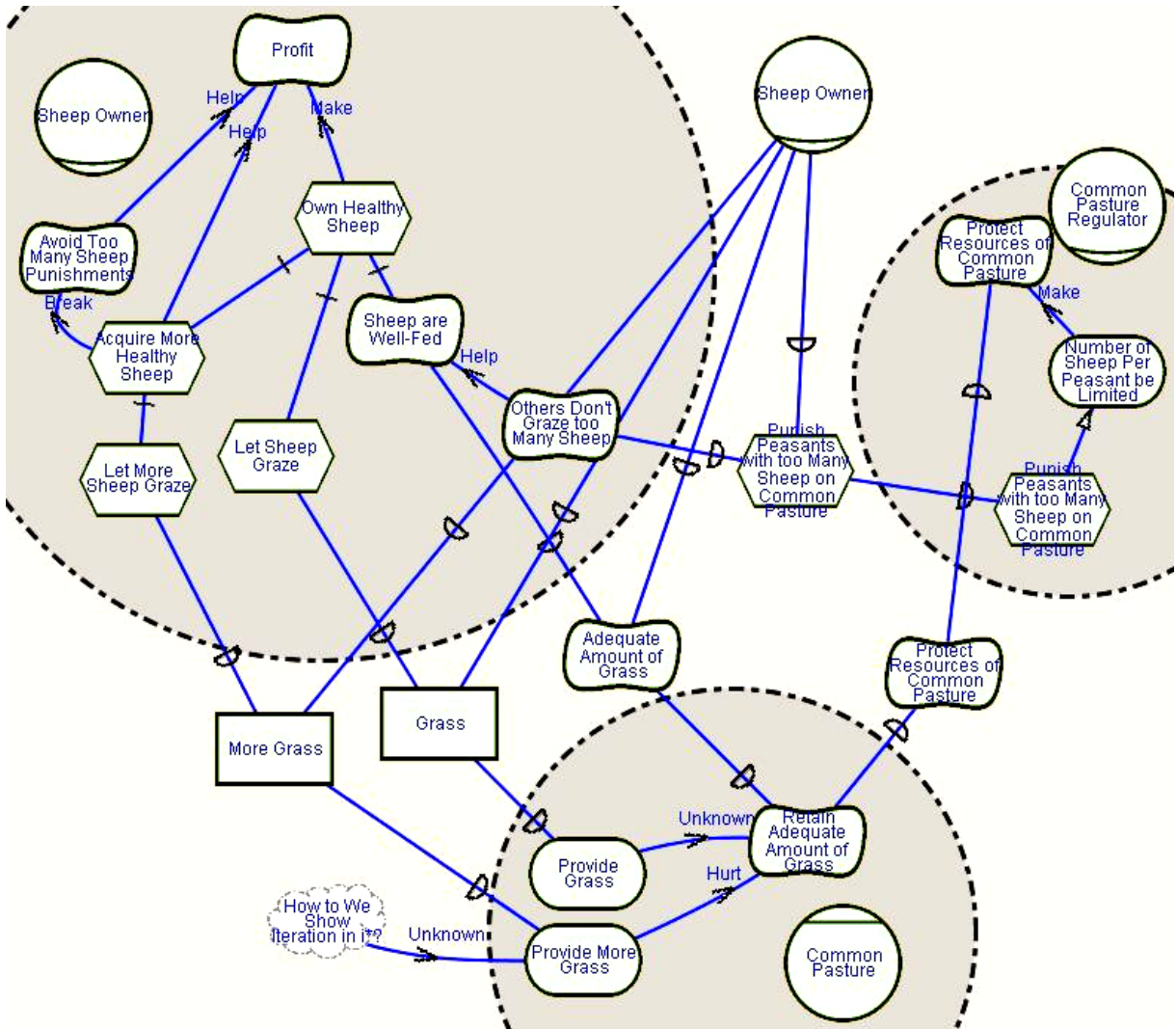


Figure C.8: Tragedy of the Commons Example for Sheep Grazing

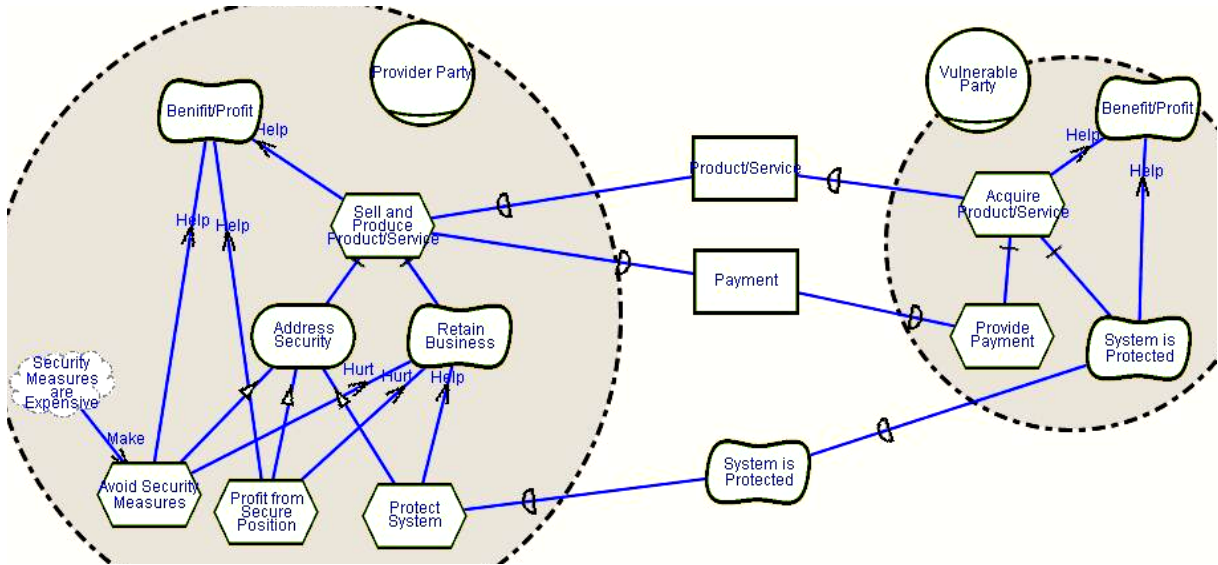


Figure C.9: Liability Dumping General Model

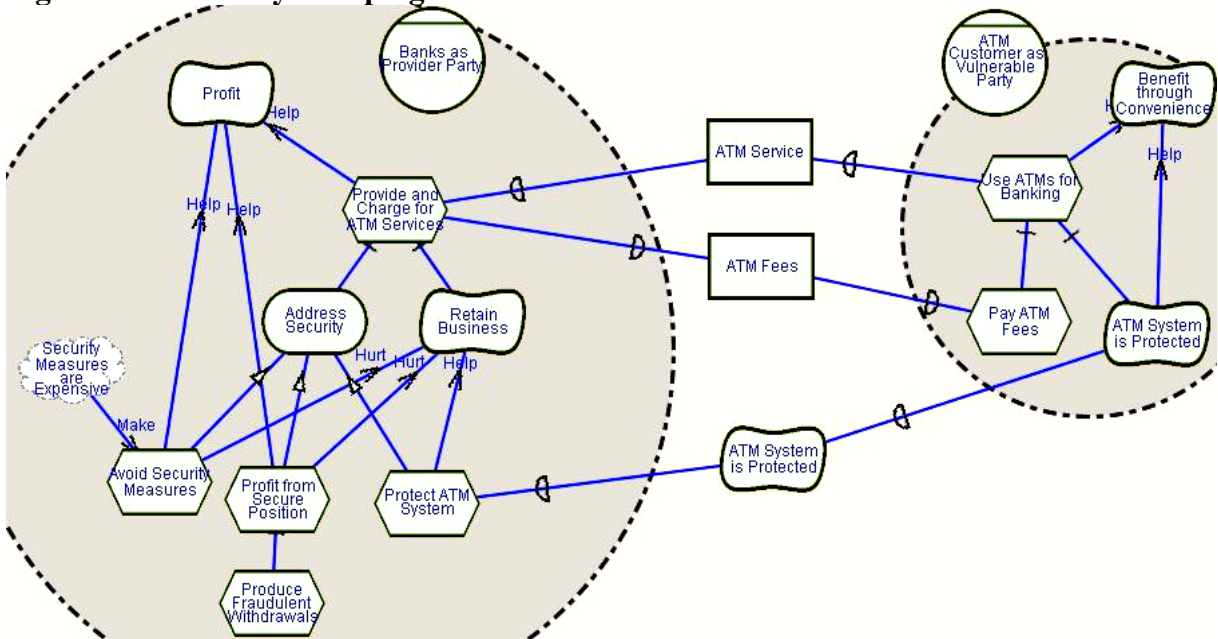


Figure C.10: Liability Dumping ATM Example

Appendix D Trusted Computing Study Additional Models

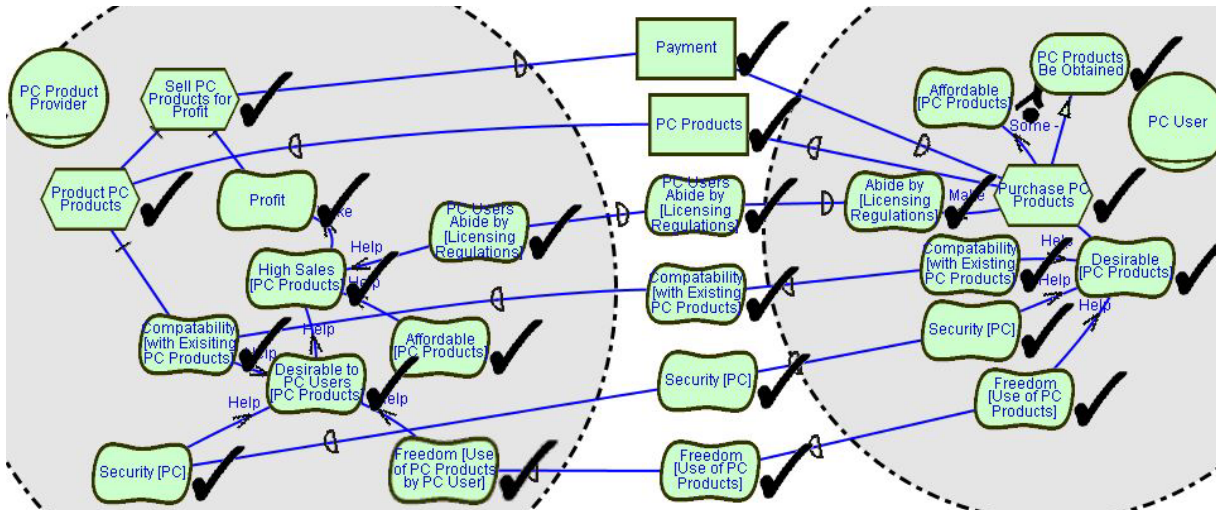


Figure D.1: Technology Provider and Technology User

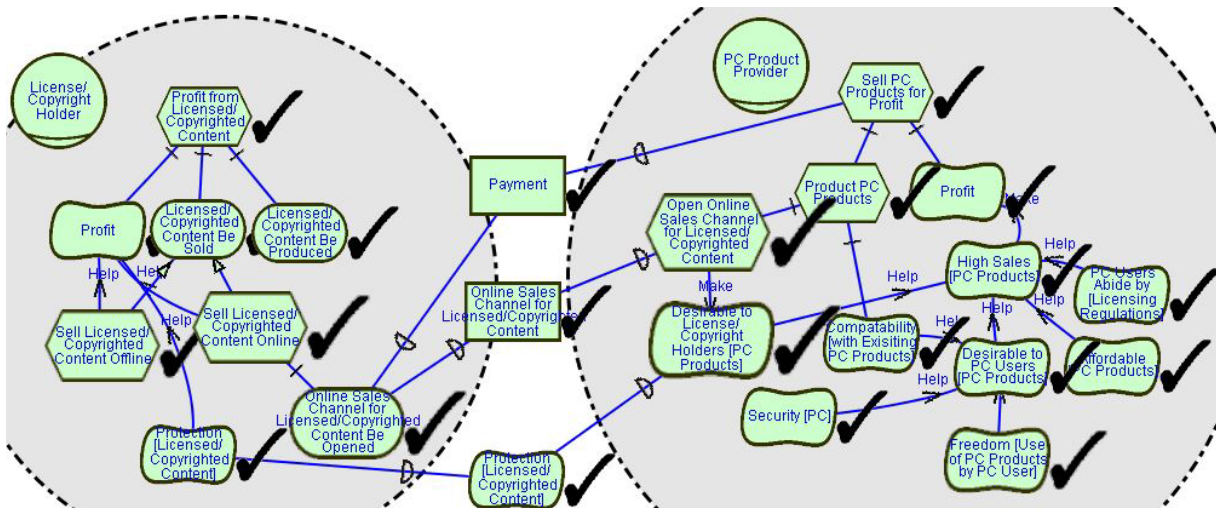


Figure D.2: Technology Provider and Technology User

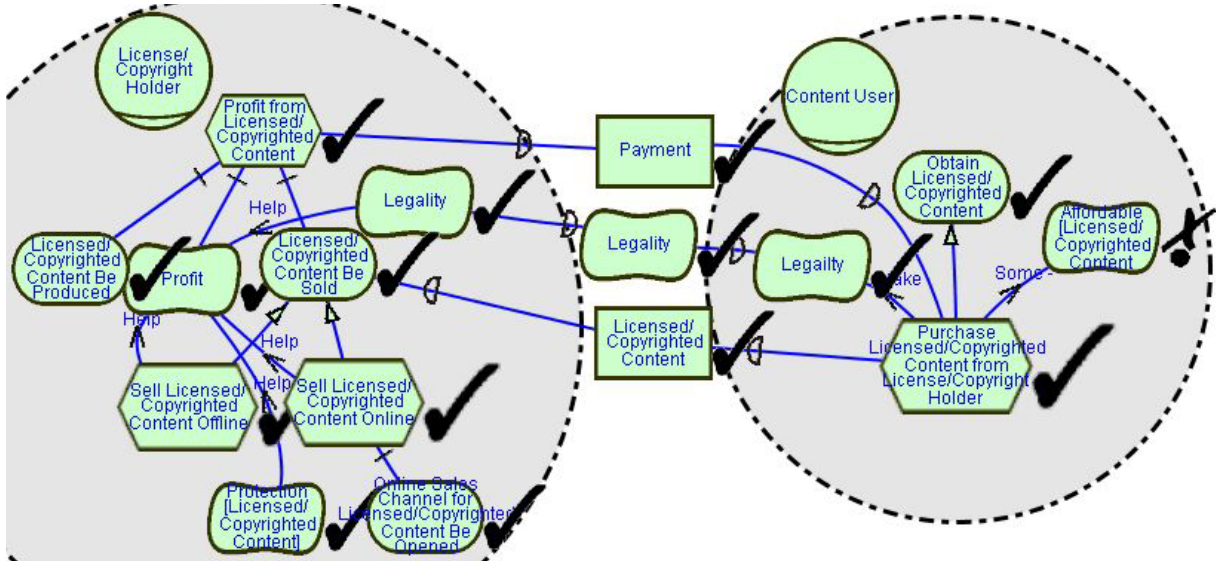


Figure D.3: Technology Provider and Technology User

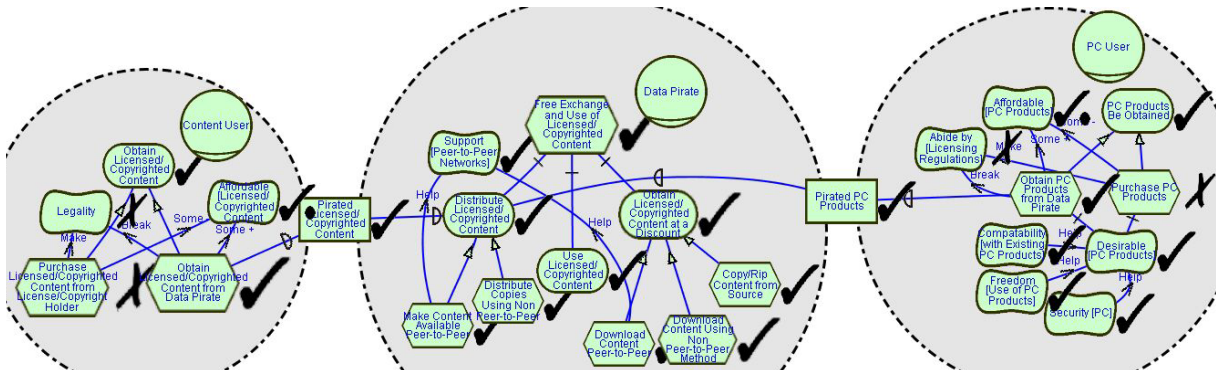


Figure D.4: Content User, Data Pirate and Technology User

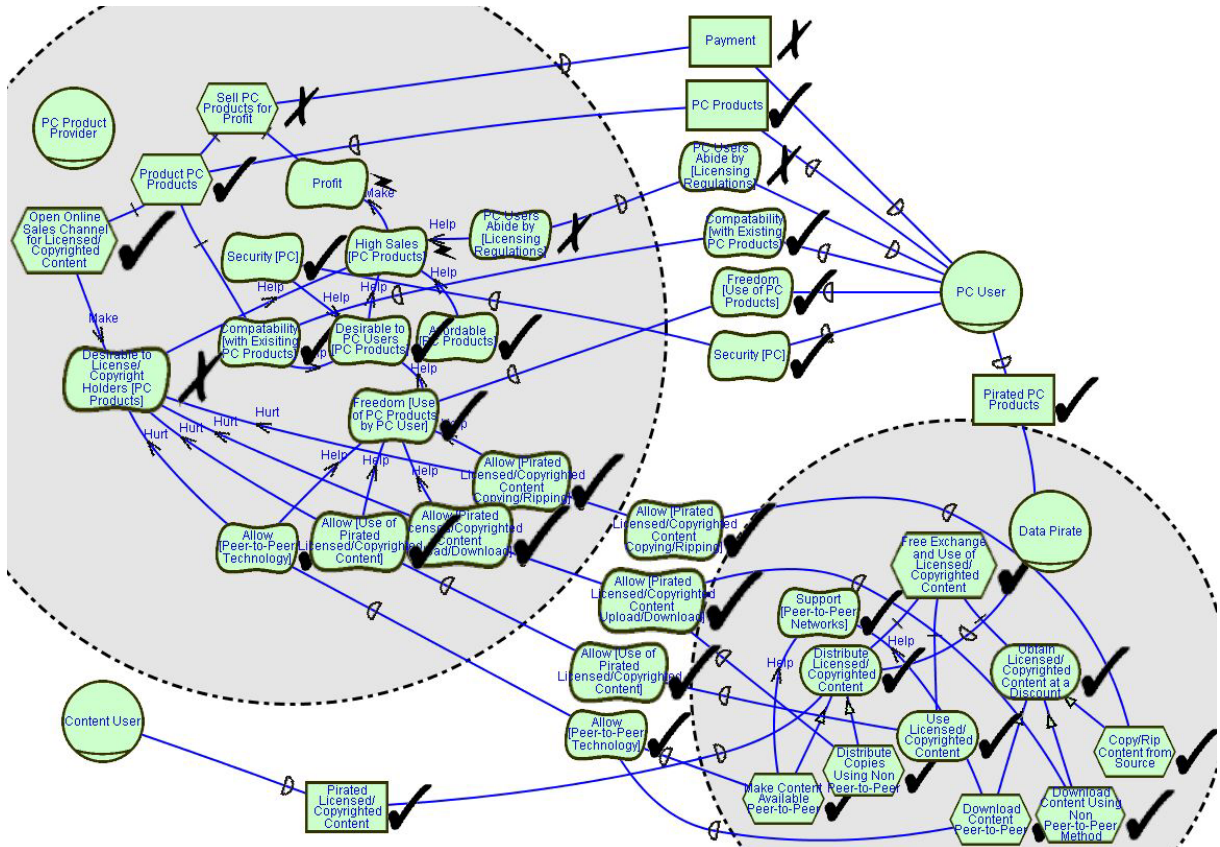


Figure D.5: Technology Provider and the Data Pirate

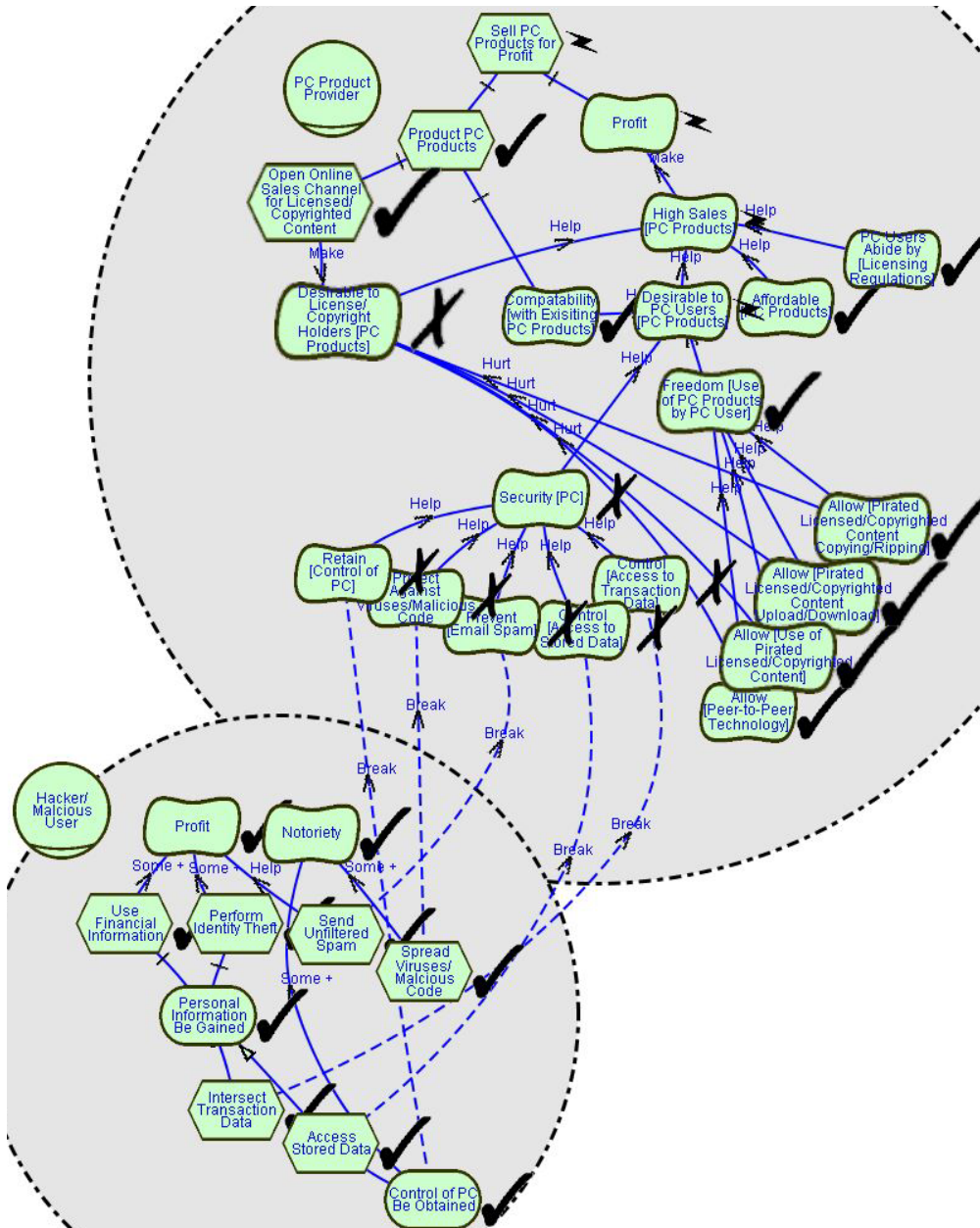


Figure D.6: Technology Provider and the Data Pirate

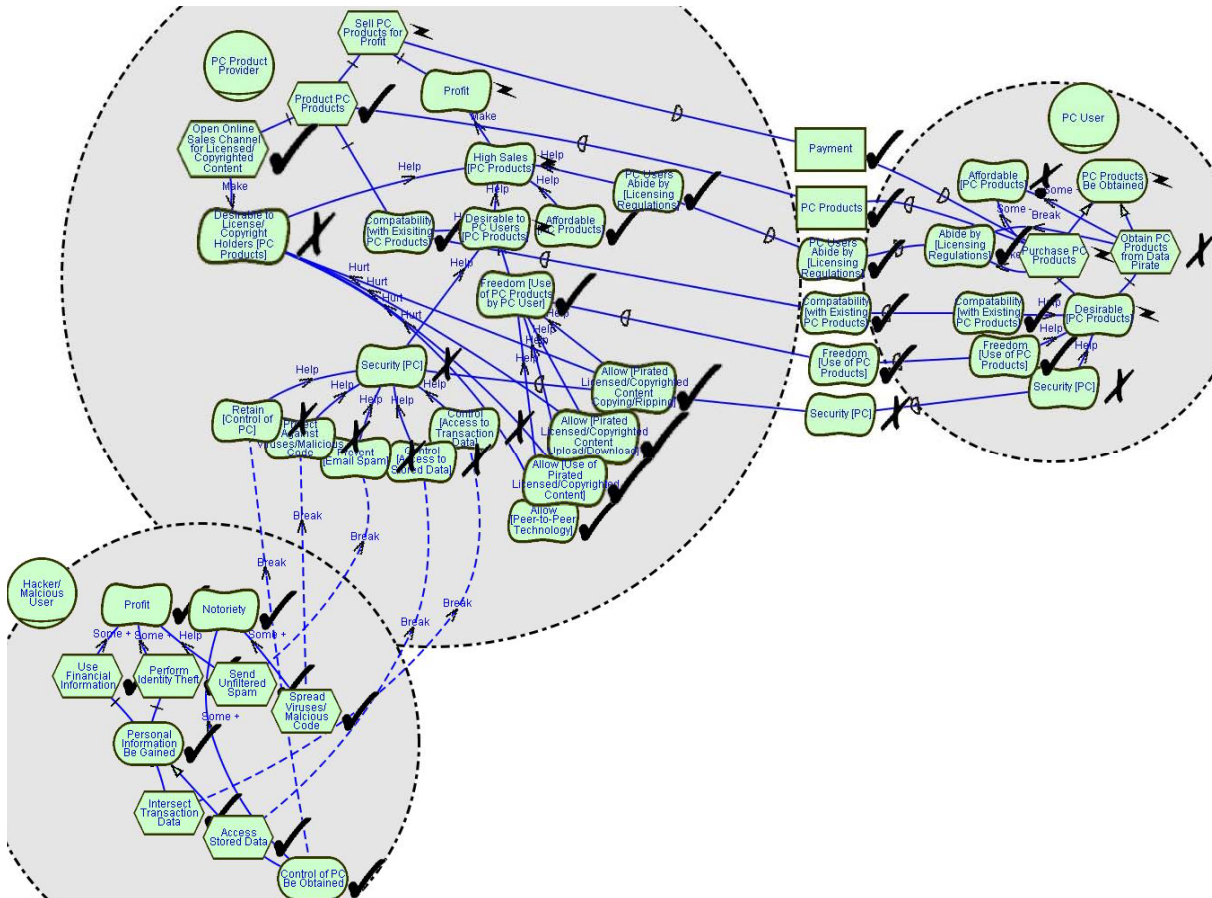


Figure D.7: Technology Provider, Hacker/Malicious User and the Technology User

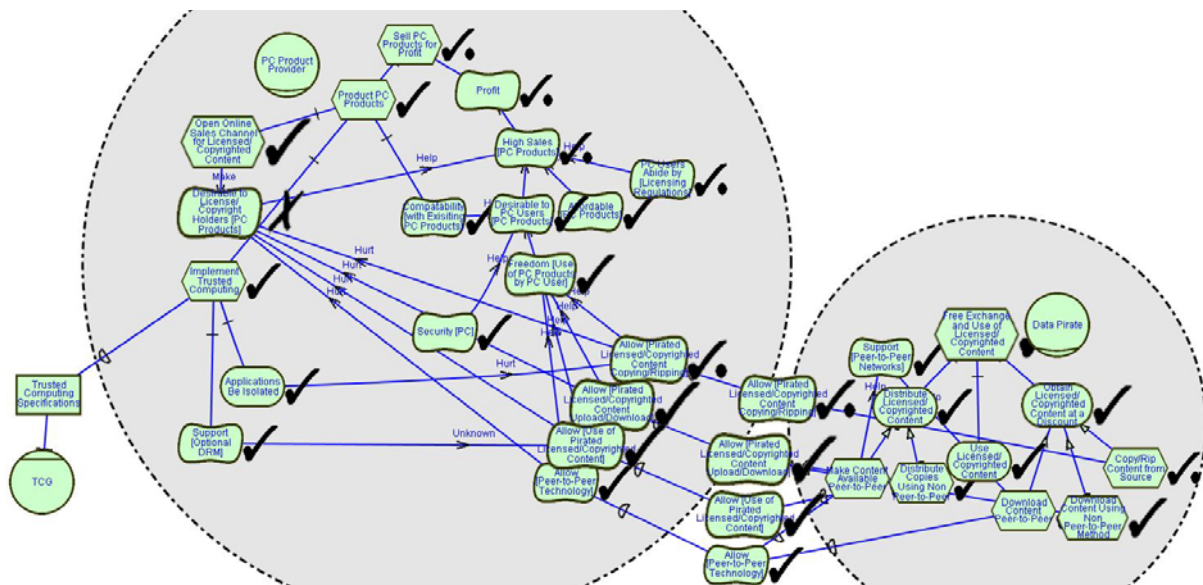


Figure D.8: Proponent point of View showing the Technology Provider with TC Technology and the Data Pirate

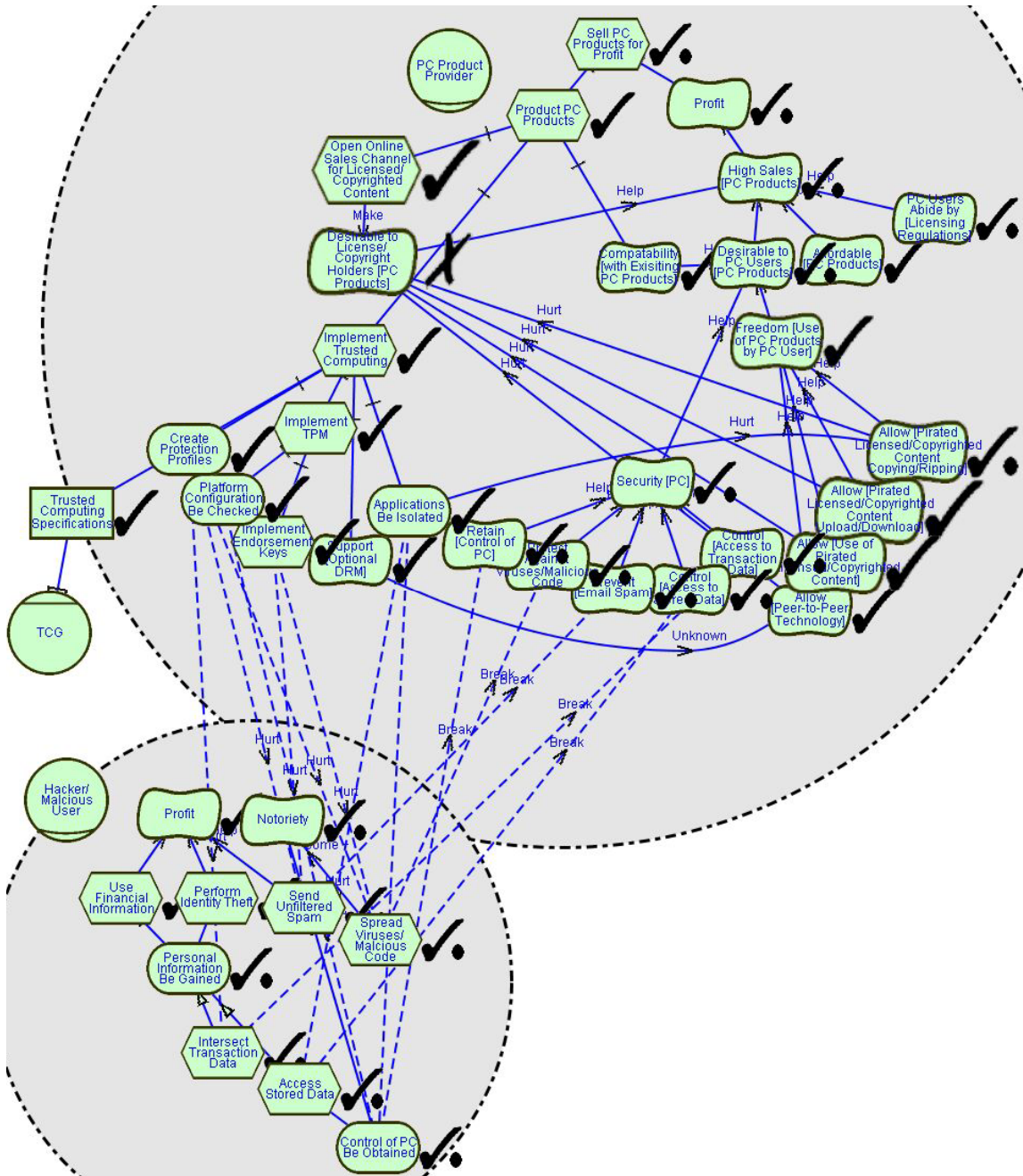


Figure D.9: Proponent point of View showing the Technology Provider with TC Technology and the Hacker/Malicious User

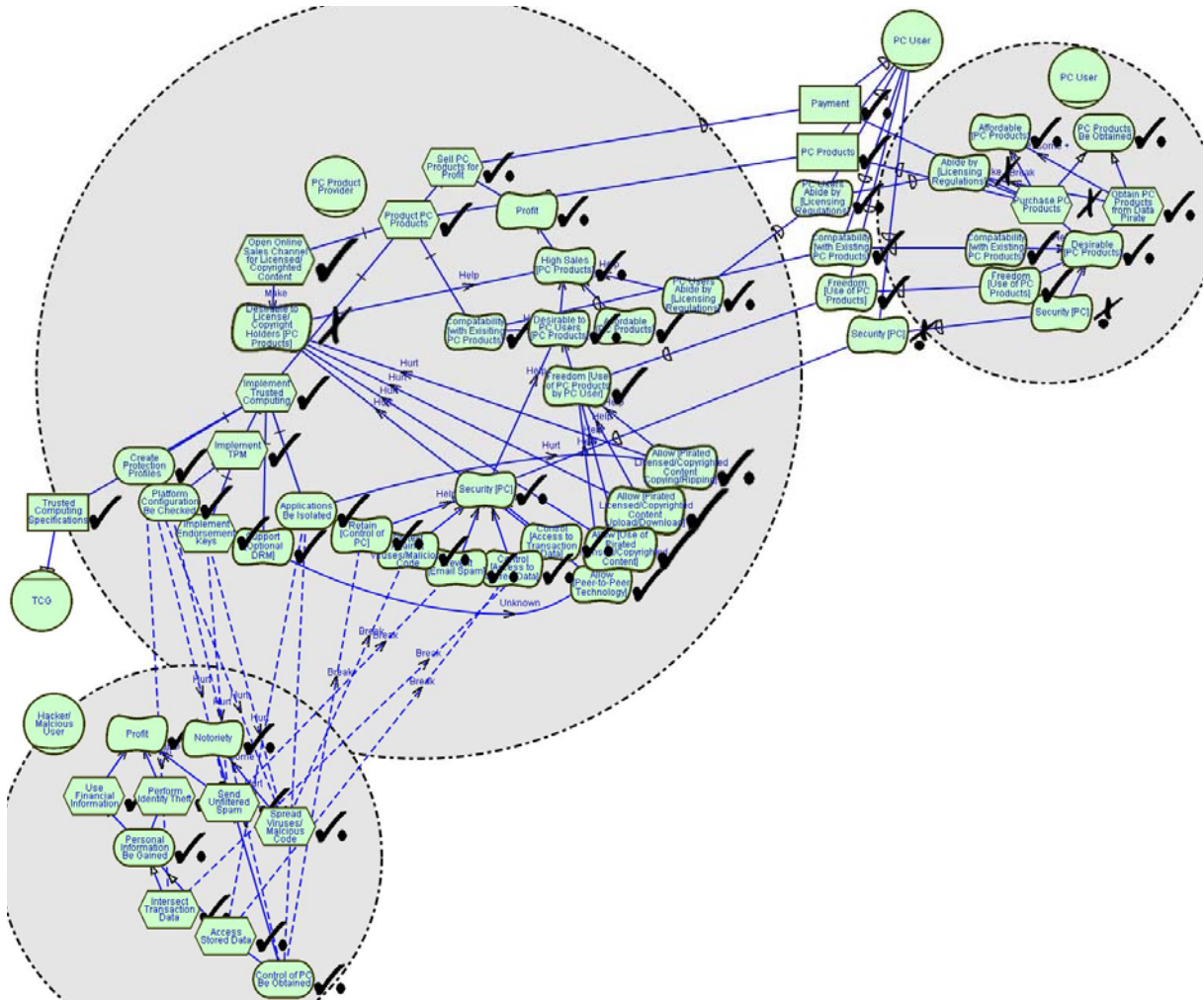


Figure D.10: Proponent Point of View Showing the Technology Provider with TC Technology, the Hacker/Malicious User and the Technology User

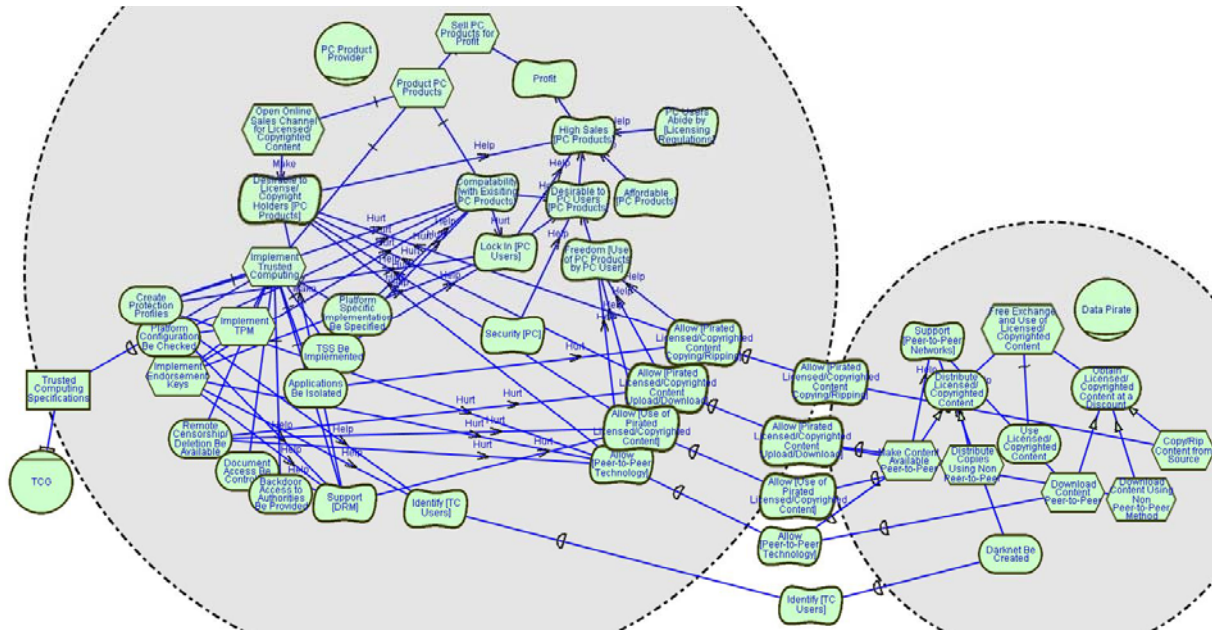


Figure D.12: Opponent Point of View Showing the Technology Provider with TC Technology and the Data Pirate






Appendix E Comparison of Evaluation Procedure Results

Table E.1: Results of Three Evaluation Procedures on the TC Model in Figure 8.2

Actor in/ to- from/ from-to	Element Name	Human Judgment Evaluation Result	GMNS Results		GMNS-# Results	
			Sat	Den	Sat	Den
License/Copyright Holder	Online Sales Channel for Licensed/ Copyrighted Content Be Opened	✓	F	N	1.0	0.0
	Protection [Licensed/ Copyrighted Content]	✓	F	N	1.0	0.0
	Sell Licensed/ Copyrighted Content Offline	✓	F	N	1.0	0.0
	Sell Licensed/ Copyrighted Content Online	✓	F	N	1.0	0.0
	Legality	✓	F	N	1.0	0.0
	Profit	✓	P	N	0.5	0.0
	Licensed/ Copyrighted Content Be Sold	✓	F	N	1.0	0.0
	Licensed/ Copyrighted Content Be Produced	✓	F	N	1.0	0.0
	Profit from Licensed/ Copyrighted Content	✓	P	N	0.5	0.0
License/Copyright Holder and Technology Provider	Payment	✓	F	N	1.0	0.0
	Online Sales Channel for Licensed/ Copyrighted Content]	✓	F	N	1.0	0.0

	Protection [Licensed/ Copyrighted Content]	✓	F	N	1.0	0.0
Technology Provider	Freedom [Use of Technology by Technology User]	✓	F	N	1.0	0.0
	Security [Technology]	✓	F	N	1.0	0.0
	Desirable to Technology Users [Technology]	✓	P	N	0.5	0.0
	Provide [Useful Functionality]	✓	F	N	1.0	0.0
	Technology Users Abide by [Licensing Regulations]	✓	P	N	0.25	0.0
	Privacy [User Information]	✓	F	N	1.0	0.0
	Desirable to License/ Copyright Holders [Technology]	✓	F	N	1.0	0.0
	Compatibility [with Existing Technology]	✓	F	N	1.0	0.0
	Profit	✓	P	N	0.5	0.0
	Produce Technology	✓	F	N	1.0	0.0
	Open Online Sales Channel for Licensed/ Copyrighted Content	✓	F	N	1.0	0.0
	Sell Technology for Profit	✓	P	N	0.125	0.0
	Gain Trust [of Technology Users]	✓	P	N	0.5	0.0
	Technology Provider and Technology User	Technology Users Abide by [Licensing Regulations]	✓	P	N	0.25
Payment		✓	P	N	0.25	0.0
Technology		✓	F	N	1.0	0.0

	Trust [Technology Provider]	✓	P	N	0.5	0.0
	Provide [Useful Functionality]	✓	F	N	1.0	0.0
	Privacy [User Information]	✓	F	N	1.0	0.0
	Freedom [Use of Technology]	✓	F	N	1.0	0.0
	Compatibility [with Existing Technology]	✓	F	N	1.0	0.0
	Security [Technology]	✓	F	N	1.0	0.0
Technology User	Technology Be Obtained	✓	P	N	0.25	0.0
	Affordable [Technology]	✗	N	P	0.0	0.125
	Purchase Technology	✓	P	N	0.25	0.0
	Desirable [Technology]	✓	P	N	0.5	0.0
	Abide by [Licensing Regulations]	✓	P	N	0.25	0.0
	Trust [Technology Provider]	✓	P	N	0.5	0.0
	Provide [Useful Functionality]	✓	F	N	1.0	0.0
	Privacy [User Information]	✓	F	N	1.0	0.0
	Freedom [Use of Technology]	✓	F	N	1.0	0.0
	Compatibility [with Existing Technology]	✓	F	N	1.0	0.0
	Security [Technology]	✓	F	N	1.0	0.0
Licensed/ Copyrighted Content User	Purchase Licensed/ Copyrighted Content from License/ Copyright Holder	✓	F	N	1.0	0.0
	Legality	✓	F	N	1.0	0.0

	Affordable [Licensed/ Copyrighted Content]		N	P	0.0	0.5
	Obtain Licensed/ Copyrighted Content		F	N	1.0	0.0
License/ Copyright Holder and Licensed/ Copyrighted Content User	Legality		F	N	1.0	0.0
	Payment		F	N	1.0	0.0
	Licensed/ Copyrighted Content		F	N	1.0	0.0